LREC-COLING 2024

The Seventh Workshop on e-Commerce and NLP (ECNLP 7)

Workshop Proceedings

Editors
Shervin Malmasi, Besnik Fetahu, Nicola Ueffing, Oleg Rokhlenko, Eugene Agichtein and Ido Guy

21 May, 2024
Torino, Italia

Proceedings of the Seventh Workshop on e-Commerce and NLP (ECNLP 7)

ISBN 978-2-493814-42-5
ISSN 2951-2093 (COLING); 2522-2686 (LREC)

Jointly organized by the ELRA Language Resources Association
and the International Committee on Computational Linguistics

Introduction

It is our great pleasure to welcome you to the Seventh Workshop on e-Commerce and NLP (ECNLP).

This workshop focuses on intersection of Natural Language Processing (NLP) and e-Commerce. NLP and information retrieval (IR) have been powering e-Commerce applications since the early days of the fields. Today, NLP and IR already play a significant role in e-Commerce tasks, including product search, recommender systems, product question answering, machine translation, sentiment analysis, product description and review summarization, and customer review processing. With the exploding popularity of chatbots and shopping assistants -– both text- and voice-based -– NLP, IR, question answering, and dialogue systems research is poised to transform e-Commerce once again.

The ECNLP workshop series was designed to provide a venue for the dissemination of late-breaking research results and ideas related to e-commerce and online shopping, as well as a forum where new and unfinished ideas could be discussed. This is the seventh edition of the workshop since its inception in 2019.

We have received a larger number of submissions than we could accept for presentation, (55% acceptance rate). The selection process was competitive and we believe it resulted in a balanced and varied program that is appealing to audiences from the various sub-areas of e-Commerce.

We would like to thank everyone who submitted a paper to the workshop. We would also like to express our gratitude to the members of the Program Committee for their timely reviews, and for supporting the tight schedule by providing reviews at short notice.

We hope that you enjoy the workshop!

The ECNLP Organizers April 2024

Organizing Committee

Shervin Malmasi (Amazon)
Besnik Fetahu (Amazon)
Nicola Ueffing (eBay Inc)
Oleg Rokhlenko (Amazon)
Eugene Agichtein (Emory University)
Ido Guy (Meta AI)

Table of Contents

Learning Reasons for Product Returns on E-Commerce
Miriam Farber, Slava Novgorodov and Ido Guy 1

Towards Multi-Modal Co-Reference Resolution in Conversational Shopping Agents
Samuel Osebe, Prashan Wanigasekara, Thomas Gueudre, Thanh Tran, Rahul Sharma, Fan Yang, Qian Hu, Weitong Ruan, Emre Barut and Chengwei Su 8

Efficient and Interpretable Information Retrieval for Product Question Answering with Heterogeneous Data
Biplob Biswas and Rajiv Ramnath 19

Hallucination Detection in LLM-enriched Product Listings
Ling Jiang, Keer Jiang, Xiaoyu Chu, Saaransh Gulati and Pulkit Garg 29

Self-Improving Customer Review Response Generation Based on LLMs
Guy Azov, Tatiana Pelc, Adi Fledel Alon and Gila Kamhi 40

Don't Just Translate, Summarize Too: Cross-lingual Product Title Generation in E-commerce
Bryan Zhang, Taichi Nakatani, Daniel Vidal Hussey, Stephan Walter and Liling Tan 58

Turkish Typo Correction for E-Commerce Search Engines
Elif Oral, Koray Mancuhan, Hüseyin Varol Erdem and Pınar Ece Hatipoglu 65

Detecting AI-enhanced Opinion Spambots: a study on LLM-generated Hotel Reviews
Davide Buscaldi and Vijini Liyanage 74

Assessing Image-Captioning Models: A Novel Framework Integrating Statistical Analysis and Metric Patterns
Qiaomu Li, Ying Xie, Nina Grundlingh, Varsha Rani Chawan and Cody Wang 79

Frogs into princes: A generative model to understand the success of product descriptions
Takehiro Takayanagi, Bruno Charron and Marco Visentini-Scarzanella 88

STA: Self-controlled Text Augmentation for Improving Text Classifications
Congcong Wang, Gonzalo Fiz Pontiveros, Steven Derby and Tri Kurniawan Wijaya 97

Multi-word Term Embeddings Improve Lexical Product Retrieval
Viktor Shcherbakov and Fedor Krasnov 115

Explicit Attribute Extraction in e-Commerce Search
Robyn Loughnane, Jiaxin Liu, Zhilin Chen, Zhiqi Wang, Joseph Giroux, Tianchuan Du, Benjamin Schroeder and Weiyi Sun 125

TAAL: Target-Aware Active Learning
Kunal Kotian, Indranil Bhattacharya, Shikhar Gupta, Kaushik Pavani, Naval Bhandari and Sunny Dasgupta 136

Cluster Language Model for Improved E-Commerce Retrieval and Ranking: Leveraging Query Similarity and Fine-Tuning for Personalized Results
Duleep Rathgamage Don, Ying Xie, Le Yu, Simon Hughes and Yun Zhu 145

Learning Reasons for Product Returns on E-Commerce

Miriam Farber, Slava Novgorodov, Ido Guy
Meta
mfarber@meta.com, slavanov@meta.com, idoguy@acm.org

Abstract

In the rapidly evolving landscape of e-commerce, product returns have become a significant economic burden for businesses, where the reasons for returns may vary from wrong sizing and defective products to simply no longer needing the purchased product. This paper presents, to the best of our knowledge, the first comprehensive study of the complexities of product returns across a variety of e-commerce domains, focusing on the task of predicting the return reason. We propose a supervised approach for predicting return likelihood and the underlying return reason. We test our approach over a real-world dataset from a large e-commerce platform.

Keywords: e-commerce, product return, return reason prediction

1. Introduction

Due to the rapid growth of the e-commerce industry in the past years, online selling has become very trending. E-commerce platforms deal with many technological problems such as recommendations and personalization, search, product categorization, content generation, and various logistic aspects such as inventory optimization and delivery. The e-commerce supply chain is becoming more complex as organisations are both expanding their businesses geographically and increasing their supplier base to continue their growth. Consumers are frequently ordering and returning items (the return rate may vary from 5% to up to 60% (Zhu et al., 2018; Cullinane et al., 2019; Li et al., 2018), dependent on product category, returns policy and other reasons). Moreover, some buyers will not make a purchase if there is no return policy and will prefer sellers that provide comfortable and fair return policy (Hjort and Lantz, 2016).

Managing product returns in e-commerce is an important problem in the past years due to several main factors. The first is financial impact, since high return rates can significantly impact a retailer's bottom line. Returns lead to additional costs in terms of restocking, and potential loss of saleable inventory, which can erode profit margins. Second factor is customer satisfaction. A smooth return process is crucial for maintaining customer satisfaction and trust. Negative experiences with returns can lead to loss of customer loyalty and negative word-of-mouth, impacting future sales. This factor includes resource allocation and inventory management. Handling returns requires time, labor, and infrastructure, diverting resources from other essential business operations. An efficient return management system is needed to minimize these resource demands. Moreover, high return rates can disrupt inventory management and forecasting, making it more challenging for retailers to maintain optimal stock levels and meet customer demand. Last but not the least factor is the environmental impact. Frequent returns contribute to higher carbon emissions due to increased transportation needs for reverse logistics. Additionally, returned items may end up in landfills if they cannot be resold, contributing to waste and pollution. Hence, by investing effort in dealing with the problem of product returns, e-commerce businesses can improve their financial performance, enhance customer satisfaction, optimize resource allocation, maintain better inventory management and even reduce their environmental impact.

The problem of products returns can be seen as part of a wider field of reverse logistics. In general, reverse logistics is the process of managing the flow of goods from the point of consumption back to the point of origin for various purposes such as returns, repairs, recycling, or disposal. In the context of e-commerce, reverse logistics primarily deals with the management of product returns. Managing reverse logistics effectively in e-commerce requires a combination of efficient processes, technology, and partnerships. By addressing these challenges, retailers can minimize the financial and environmental impact of returns, improve customer satisfaction, and maintain optimal inventory levels.

In this paper we present a deep-dive study of the complexities of product returns across a variety of e-commerce domains, focusing on the task of predicting the return reasons. To the best of our knowledge we are the first to extensively study this problem in general e-commerce setting, in opposite to previous works that focus on specific domains or specific reasons. We propose an ensemble-based machine learning approach for predicting return likelihood and the underlying return reason. We showcase the performance of our proposed approach over real-world dataset of product transaction from a large e-commerce platform.

The Seventh Workshop on e-Commerce and NLP (ECNLP 7), pages 1–7

2. Related Work

Many works studied product returns in e-commerce, however, in general, the problem of product return prediction in e-commerce has not attracted much attention from the data mining community, despite the large amount of data available from historical purchase and return records (Li et al., 2018). A line of papers that is most related to our work, focus on predictive analytics using machine learning methods (e.g., (Fuchs and Lutz, 2021; Ma and Kim, 2016; Urbanke et al., 2015)). These works apply advanced data mining and machine learning techniques to predict the likelihood of product returns and in some cases try to predict the return reason. These predictive models can help businesses identify high-risk customers or products, allowing for proactive interventions to reduce return rates. For example, Urbanke et al. (Urbanke et al., 2015) use feature extraction to generate a large set of features that are originated from various categorical variables such as return history, preferred payment method and device information from which the returned product was originally ordered. Some information is available only after the customer finishes the transaction, hence this methods limits the ability to take proactive actions. Other works focus on improvements in product information, images, and descriptions, which can reduce return rates by ensuring that customers have a clear understanding of what they are purchasing. The works mentioned above focus on prediction of the return event (binary classification). In our paper we focus on the more fine-grained task of predicting a return reason out of large list of possible reasons.

Moreover, while in our paper we work on various e-commerce domains, some of the papers (e.g. (Seewald et al., 2019; Nestler et al., 2021; Kedia et al., 2019)) focus on fashion, where the return rate may reach up to 60% (Zhu et al., 2018; Cullinane et al., 2019; Li et al., 2018). One of the most popular reason for returns in fashion is wrong size(Nestler et al., 2021). To deal with the size-related returns, some works propose methods that unify sizes across different platforms (e.g., (Du et al., 2019)) and help users to choose the correct size on any platform. Other works (e.g., (Abdulla and Borar, 2017)) proposed personalized size recommendations, or other innovative tools to prevent the return event (Castelblanco DÃaz, 2021). Many works try to proactively predict the return event, e.g. Kedia et al. (Kedia et al., 2019) that proposes a method to predict the chance that the customer will return the product even before the order is completed. It uses deep neural network model that uses latent size and fit features of the product and the customer. As mentioned above, in this work we do not focus on any specific domain, but provide a solution for various domains that predicts the return event and the return reason for a specific product, given only information about the product.

Other line of works (e.g., (Hjort and Lantz, 2016; El Kihal et al., 2021; Ambilkar et al., 2022)) study the connection between the returns and the return policies: These studies focused on understanding the impact of different return policies on consumer behavior, sales, and returns. They observe that factors such as return time windows, restocking fees, and return shipping costs, can affect the customer satisfaction and minimizing return rates.

Finally, returns management in e-commerce can be viewed as part of a larger problem of reverse logistics management. The reverse logistics optimization research (e.g., (EL HACHIMI et al., 2018; Sandhya and Kumara, 2020)) focuses on improving the efficiency and cost-effectiveness of reverse logistics processes, such as transportation, inspection, refurbishment, and disposition of returned products. The goal is to minimize the financial and environmental impact of returns.

3. Dataset and characteristics

We start by describing the dataset utilized in this work. Our dataset is obtained from one of the largest e-commerce platforms, covering 618240 products across 2928 categories from 26 domains. Each entry in the dataset is associated with a transaction, with an indication regarding whether it was resulted in return. The data is split 50/50, with 309120 of the entries resulted in return and 309120 did not result in return. In case of a return, the customer can choose one of predefined 13 options as the return reason, whose distribution is presented in Table 1. Customers can also include free-form text elaborating on the return reason, and $\sim 10\%$ chose to do so. In addition, the following information regarding the products and transactions is provided: Textual features (product name, category and description), numerical features (product price and quantity) and categorical features (product size, transaction country, transaction platform, coarse platform category).

4. Returns prediction

In this paper we deal with two types of prediction tasks: Binary prediction - whether the product is going to be returned, and Multiclass prediction - Predicting a return reason for products that were returned. Here we analyze two types of return reasons lists - an extensive list consisting of all available return reasons (see Section 3), and a concise list consisted of common 5 reasons. To examine the contribution of the different features, for each task we train several models, some of them utilize

Table 1: Distribution of return reason across our dataset

Return reason	Percentage
Too small	21.0%
Too large	15.4%
Item quality not as expected	10.3%
Not needed anymore	9.9%
Inaccurate description	9.2%
Did not like the style	8.7%
Bought by mistake	7.7%
Defective item	5.7%
Damaged item	4.1%
Wrong item received	3.9%
Did not like the color	2.2%
Found better price	1.2%
Item not compatible	0.6%

only textual features while others utilize the entire range of features. For the latter type of tasks, an analysis of the features importance is provided.

4.1. Methods

As described in Section 3, our dataset consists of several types of features: textual, categorical and numerical. To train a range of models on a tabular data utilizing those different types, we use AutoGluon (Erickson et al., 2020). This is an AutoML package that trains common types of classification models (including tree-based, neural networks and transformers), and performs model selection and hyperparameters tuning. The models are then combined to produce an ensemble model that provides the final predictions (Shi et al., 2021).

We use *TabularPredictor* with multimodal support [1] to train models that utilize numerical, categorical and textual features. Models that utilize only textual features are trained using *TextPredictor* [2]. For both TabularPredictor and TextPredictor, transformers-based model ELECTRA (Clark et al., 2020) with the hyperparameters specified in footnote 1 is used to train classification task on the textual features. Textual features are concatenated with a separator between each pair. In TabularPredictor, the categorical and numerical features are fed into tree based models like XGBoost as well as well as neural networks (see Figure 1 on page 3 in (Erickson et al., 2020) for details), and the final model is formed from an ensemble of these models together with the Electra model mentioned above.

In our experiments, the data is split randomly into training (70%), validation (15%) and test (15%), with a distinct set of products belonging to each. Results are reported on the test set.

[1] https://auto.gluon.ai/0.4.0/tutorials/text_prediction/multimodal_text.html

[2] https://auto.gluon.ai/0.0.15/tutorials/text_prediction/beginner.html

Table 2: Multi-class performance (full set of reasons).

Model features	Accuracy	Macro F1
Product Category	0.255	0.119
Product name	0.319	0.202
Product description	0.322	0.220
All textual features	0.337	0.223
All features (ensemble)	0.352	0.249

4.2. Experiments and Results

The binary model for predicting a return, based on ensemble with all features, reached 0.942 ROC AUC on the test set, with 0.876 F1 score and 0.877 accuracy. This accuracy reduces a bit to 0.866 when utilizing textual features only. This is on par or above with previously reported results for this task (Zhu et al., 2018; Urbanke et al., 2015; Li et al., 2018).

As the binary classification task is already well studied (see above and Section 2) and achieves high accuracy, we move to the more challenging task of predicting the specific return reason. For this purpose, we limit the data only to entries that resulted in a return, and build a classifier to predict the return reason. First we perform the experiments on the full list of 13 return reasons using the models described in Section 4.1. Table 2 summarizes the results over different facets of the products as features. When limiting to a single textual feature, performance is higher when using the product description, compared to using its name only and, in turn, category. It makes sense as the description contains richer data compared to the other 2 fields, and is directly tied to some of the return reasons (e.g "inaccurate description"). Using all the 3 textual features yields further performance boost, and using all available features via an ensemble reaches the highest performance at 0.352 accuracy.

To gain a deeper understanding of the roles the different features play, we display feature importance in Table 3. The importance of each feature is measured via the impact on model's accuracy when fixing the rest of the features and permuting the entries of the given feature[3]. We can see that product description (which intuitively contains the most rich information about the product) is the most important feature, followed by product category. In fact, all features except of country and quantity have significant importance (p-value smaller than 0.01).

[3] https://auto.gluon.ai/stable/api/autogluon.tabular.TabularPredictor.feature_importance.html

Table 3: Feature importance (full set of reasons).

Feature	Importance	p-value
Description	0.061	0.000007
Inferred Model Category	0.035	0.0002
Size	0.021	0.00008
Page category name	0.017	0.001
Checkout product	0.014	0.00003
Name	0.011	0.002
Price	0.007	0.0004
Country	0.001	0.15
Pack Quantity	0.001	0.08

Since our data is based mostly in the US and the majority of product quantities is 1, these features become somewhat redundant, which explains their low importance. The high importance of the textual features is also reflected in the fact that the model that was trained on textual features only is not significantly inferior compared to the best performing model.

The relatively low accuracy, even of the best performing model, can be explained by the following factors: 1) The data is highly unbalanced, with some of the return reasons having very few entries in the dataset (see Table 1), making it much harder to infer those. 2) Some of the return reasons require much deeper familiarity with the customer or the product journey which is not present in the data we have ("item not needed anymore", "bought by mistake", "wrong item received", etc). 3) The predefined list of return reasons provided by the e-commerce platform includes many subjective and also not so well defined/overlapping reasons (e.g "item not compatible" vs "item quality not as expected", and also "defective item" vs "damaged item", etc). Customer's confusion is also demonstrated in the free text responses that they provide, which sometimes are not aligned with the reason they picked from the list. To demonstrate the model's difficulty to distinguish between "overlapping" reasons, we examined the confusion matrix. Consider the following two return reasons: "found better price" and "not needed anymore". These reasons overlap, since if the customer found the same item in a better price, then they don't need this item anymore. In our dataset the latter reason is 8 times more common than the former. Thus, unsurprisingly, 24% of the test samples who belong to "found better price" category were labeled as "not needed anymore" by the model (most common label for this category). Similar phenomenon occurs for the classes "item not compatible" and "defective item". The latter is 9 times more common that the former, and is labeled as such by the model in 40% of the cases that belong to "item not compatible".

Table 4: Distribution of return reason across our dataset, when limiting to 5 common return reasons

Wrong size	Quality	No need	Description	Defective
50.9%	14.4%	13.9%	12.8%	8.0%

Table 5: Multi-class performance (5 common reasons).

Model features	Accuracy	Macro F1
All textual features	0.648	0.464
All features (ensemble)	0.656	0.463

4.2.1. Predicting common return reasons

To alleviate some of the issues above, we filtered a more concise list of 5 common return reasons: *Wrong size* - union of too small and too large, item quality not as expected (*Quality*), not needed anymore (*No need*), inaccurate description (*Description*), and defective item (*Defective*). Their distribution is presented in Table 4.

Table 5 shows the performance of the prediction model when limiting the data to these reasons. It is substantially higher than in the previous task, indicating they could be distinguished more effectively, with accuracy reaching 0.656 using the ensemble with all features. Note that this is significantly better than the baseline of choosing the most common class (wrong size), which according to Table 4 would have reached an accuracy of 0.509. In Table 6 we detail the precision and recall over each of the 5 reasons.

To provide more insights into the ensemble model , we depict in Figure 1 the components of the ensemble (trained on all features), showing the score of each one on the validation set, as well as the score of the ensemble on the validation set. The weights of each component within the ensemble are as follows: XGBoost: 0.26, NeuralNetTorch: 0.05, LightGBMLarge: 0.32, TextPredictor: 0.37 (see https://auto.gluon.ai/ for details about these models). This demonstrates the high

Table 6: Precision and recall of the ensemble classifier across each of the 5 reasons.

Reason	Precision	Recall
Wrong size	0.776	0.978
Quality	0.396	0.253
No need	0.421	0.339
Description	0.463	0.374
Defective	0.531	0.346

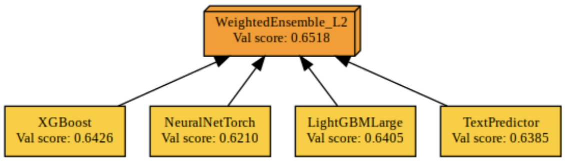


Figure 1: Composition of the top performer ensemble model, predicting 5 return reasons

importance of the textual features, with the TextPredictor model receiving the highest weight among the list.

Next we display in Table 9 the confusion matrix between the classes. Naturally as "wrong size" is the largest class (and significantly larger compared to the others), some of the data that belongs to the other classes is predicted as "wrong size". Other than that, the diagonal elements are the largest in each row, meaning that for each class, the largest bucket predicted is indeed the class itself.

Table 7: Distribution of return reason across the 5 most common domains.

Reason	Clothing	Home	J&W	H&B	Electronics
Wrong size	78.1%	0.7%	43.1%	4.7%	9.5%
Quality	8.9%	29.7%	21.6%	35.9%	12.7%
No need	6.9%	30.2%	16.3%	30.8%	26.7%
Description	4.4%	27.3%	12.1%	15.3%	17.9%
Defective	1.7%	12.2%	6.8%	13.3%	33.2%

4.2.2. Cross-domain analysis

As mentioned in Section 3, our dataset spans a variety of e-commerce domains. We set out to compare product return behavior and predictability across different domains. To this end, we considered the 5 most common domains in our dataset, which account overall for 85% of the return instances. As Table 7 shows, the distribution of return reasons varies substantially over these domains. Particularly, the distribution within the Clothing domain, where most of the previous work has focused, as mentioned in Section 2, is largely different than within other domains, reinforcing the need to study product return behavior across multiple domains. As might be expected, the majority of returns in the Clothing domain (nearly 80%), are due to wrong size. The only other domain where “wrong size” is the most common reason is Jewelry & Watches, but to a lesser extent than in Clothing. In the Home and Health & Beauty domains, wrong size is a rare return reason. In Home, “quality not expected”, “not needed anymore” and “inaccurate description” are the most common reasons. In Electronics, “defective item” is the most common.

After observing the notable differences in return reasons across e-commerce domains, we set out

Table 8: Performance of the ensemble model across the 5 most common domains.

Reason	Accuracy	Macro F1
Clothing	0.790	0.243
Home	0.410	0.334
Jewelry & Watches	0.472	0.314
Health & Beauty	0.456	0.363
Electronics	0.505	0.432

to examine the performance differences of our ensemble classifier across domains. Table 8 summarizes these results. The performance in the Clothing domain is noticeably different than in all other domains. Accuracy reaches 0.79, higher than any other domain, whereas macro F1 is the lowest among all domains. This is due to particularly strong performance of the classifier for the “wrong size” reason, at the expense of the performance for other reasons. In fact, on the Clothing domain, the accuracy of the ensemble model yield an uplift of only 1% compared to a majority baselines always deeming the reason as “wrong size” (see Table 7). Yet, the uplift in Macro F1 is more substantial, and, as discussed, the overall performance of the model across all categories is substantially higher than the majority baseline.

For the other four domains, results are more similar across reasons, which yields a more balanced trade-off between the accuracy and macro F1 metrics. For the Electronics domain, macro F1 is the highest, while accuracy is second best among the 5 domains. Table 10 demonstrates the precision and recall across the 5 reasons for Electronics. It can be seen that precision and recall are fairly high for three of the reasons: “wrong size”, “defective item”, and “not needed anymore”. It is especially interesting to observe the performance for “wrong size” which account for only 9.5% of the Electronics returns (Table 7). This may indicate that the model learns to generalize this reason from other categories, where it is more frequent (e.g., Clothing). We leave further exploration of cross-domain transfer learning for future work.

Table 10: Precision and recall of the ensemble classifier over the Electronics domain across each of the 5 reasons.

Reason	Precision	Recall
Wrong size	0.554	0.864
Quality	0.293	0.119
No need	0.518	0.587
Description	0.238	0.099
Defective	0.552	0.716

Table 9: Confusion matrix for the 5 classes prediction model. Rows represent GT label and columns represent model's prediction. Rows are normalized.

	Wrong size	Quality	No need	Description	Defective
Wrong size	0.978	0.008	0.007	0.007	0.000
Quality	0.391	0.252	0.168	0.139	0.048
No need	0.309	0.158	0.338	0.128	0.064
Description	0.245	0.147	0.166	0.373	0.066
Defective	0.156	0.118	0.176	0.202	0.345

5. Conclusions

In this work, we study the problem of product returns in e-commerce. To the best of our knowledge, we are the first to systematically investigate the underlying reasons for returns and aims to predict in e-commerce in general, as opposed to focusing on specific domains. In this paper we proposed an ensemble-based machine learning approach for predicting return likelihood and the underlying return reason. The proposed method was tested over real-world dataset of product transactions from a large e-commerce platform.

6. References

G Mohammed Abdulla and Sumit Borar. 2017. Size recommendation system for fashion e-commerce. In *KDD workshop on machine learning meets fashion*.

Priya Ambilkar, Vishwas Dohale, Angappa Gunasekaran, and Vijay Bilolikar. 2022. Product returns management: a comprehensive review and future research agenda. *International Journal of Production Research*, 60(12):3920–3944.

Tatiana Alexandra Castelblanco DÃaz. 2021. Innovative tools for the prevention of product returns in e-commerce.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Sharon Cullinane, Michael Browne, Elisabeth Karlsson, and Yingli Wang. 2019. Retail clothing returns: A review of key issues. *Contemporary operations and logistics: Achieving excellence in turbulent times*, pages 301–322.

Eddie SJ Du, Chang Liu, and David H Wayne. 2019. Automated fashion size normalization. *arXiv preprint arXiv:1908.09980*.

Hajar EL HACHIMI, Mourad OUBRICH, and Omar SOUISSI. 2018. The optimization of reverse logistics activities: a literature review and future directions. In *2018 IEEE International Conference on Technology Management, Operations and Decisions (ICTMOD)*, pages 18–24. IEEE.

Siham El Kihal, Namig Nurullayev, Christian Schulze, and Bernd Skiera. 2021. A comparison of return rate calculation methods: Evidence from 16 retailers. *Journal of Retailing*, 97(4):676–696.

Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. 2020. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*.

Kevin Fuchs and Oliver Lutz. 2021. A stitch in time saves nine-a meta-model for real-time prediction of product returns in erp systems. In *ECIS*.

Klas Hjort and Björn Lantz. 2016. The impact of returns policies on profitability: A fashion e-commerce case. *Journal of Business Research*, 69(11):4980–4985.

Sajan Kedia, Manchit Madan, and Sumit Borar. 2019. Early bird catches the worm: Predicting returns even before purchase in fashion e-commerce. *arXiv preprint arXiv:1906.12128*.

Jianbo Li, Jingrui He, and Yada Zhu. 2018. E-tail product return prediction via hypergraph-based local graph cut. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 519–527.

Jungmok Ma and Harrison M Kim. 2016. Predictive model selection for forecasting product returns. *Journal of Mechanical Design*, 138(5):054501.

Andrea Nestler, Nour Karessli, Karl Hajjar, Rodrigo Weffer, and Reza Shirvany. 2021. Sizeflags: reducing size and fit related returns in fashion e-commerce. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3432–3440.

S Sandhya and SA Vasantha Kumara. 2020. System and method to predict the number of returns in a supply chain: A reverse logistics case study. *IUP Journal of Operations Management*, 19(3):16–24.

Alexander K Seewald, Thomas Wernbacher, Alex Pfeiffer, Natalie Denk, Mario Platzer, Martin Berger, and Thomas Winter. 2019. Towards minimizing e-commerce returns for clothing. In *ICAART (2)*, pages 801–808.

Xingjian Shi, Jonas Mueller, Nick Erickson, Mu Li, and Alex Smola. 2021. Multimodal automl on structured tables with text fields. In *8th ICML Workshop on Automated Machine Learning (AutoML)*.

Patrick Urbanke, Johann Kranz, and Lutz Kolbe. 2015. Predicting product returns in e-commerce: the contribution of mahalanobis feature extraction.

Yada Zhu, Jianbo Li, Jingrui He, Brian Leo Quanz, and Ajay A Deshpande. 2018. A local algorithm for product return prediction in e-commerce. In *IJCAI*, pages 3718–3724.

Towards Multi-Modal Co-Reference Resolution in Conversational Shopping Agents

Samuel Osebe*[1], Prashan Wanigasekara*[2], Thomas Gueudre[2], Thanh Tran[2],
Rahul Sharma[2], Fan Yang[2], Qian Hu[2], Weitong Ruan[2], Emre Barut[2], Chengwei Su[2]
[1]University of Massachusetts Amherst, [2] Amazon AGI Foundations
sosebe@umass.edu,
{wprasha, tgueudre, tdt, zarahuls, fyaamz, huqia, weiton, ebarut, chengwes}@amazon.com

Abstract

The context of modern smart voice assistants is often multi-modal, where images, audio and video content are consumed by users simultaneously. In such a setup, co-reference resolution is especially challenging, and runs across modalities and dialogue turns. We explore the problem of multi-modal co-reference resolution in multi-turn dialogues and quantify the performance of multi-modal LLMs on a specially curated dataset of long, image-interleaved conversations between a voice assistant and human in a shopping use case. We propose a custom architecture for multi-modal embedding alignment using a novel parameter augmentation technique. Our proposed Parameter Augmented LLM approach shows a 4.9% absolute F1 improvement above a cross-attention baseline while reducing the number of parameters being trained by $4\times$.

Keywords: multi-modality, co-referencing, parameter-augmentation

1. Introduction

Recent advancements in multi-modal large language models (MLLMS) have pushed the capabilities of conversational agents, extending beyond processing and generating human-like text to include understanding and integrating multiple modalities such as images and audio. These advancements have led to substantial progress in tasks like image captioning (Lin et al., 2014), image classification (Russakovsky et al., 2015) and visual question answering (Goyal et al., 2017). However, these tasks often have a clear division between the text and images, not fully reflecting the complex, interwoven nature of the inputs encountered by conversational agents. This intertwining of visual and textual inputs is more pronounced in the environments like online shopping, where users seamlessly shift between textual and visual references.

Addressing this gap, Multi-modal Co-reference Resolution (MCR) emerges as a critical challenge, aiming to connect language and visual content by mapping textual references to their corresponding spatial regions in images. In this work we focus on MCR within the context of conversational agents in the shopping domain where the challenge is amplified by the vast diversity of products and the ambiguity of natural language descriptions, marking a stark contrast to areas like visual question answering (VQA) and image captioning. Efforts to address MCR for conversational agents have been relatively limited, further compounded by the fact that most multi-modal dialogue datasets (Zang et al., 2021; Kottur et al., 2021), contain very few images among the dialogues. In contrast, a typical dialogue in the shopping context can involve $5-66$ utterances, with an average of 32 images, highlighting the need for specialized attention to MCR in this domain. Figure 1 shows a sample dialogue for the shopping use case.

Figure 1: An example of a multi-turn dialogue with multi-modal co-referencing. The co-references are color coded and shown by arrows.

*equal contribution

The Seventh Workshop on e-Commerce and NLP (ECNLP 7), pages 8–18

Among the recent efforts (Lee et al., 2022; Guo et al., 2022; Chen et al., 2023) to address MCR for conversational agents, an encompassing strategy has been the end-to-end training of multi-modal transformer architectures. While effective, this strategy demands significant computational resources, manifesting in both a high number of parameters and extensive training time. To mitigate these challenges, we propose a novel technique that leverages existing unimodal large language models (LLMs) and adapt them for multi-modal inputs and outputs. Our approach augments the weights of pre-trained unimodal LLMs to learn an alignment with the pre-trained visual encoder's embeddings, thereby converting them into multi-modal system. This method significantly reduces the number of parameters to be trained along with a notable improvement in the MCR performance.

To evaluate the effectiveness of our methodology in practical conversational tasks within the shopping domain, we focus on two key areas: i) image selection and ii) image retrieval. The image selection task leverages textual and visual attributes to identify and select the most relevant product from a list of options; e.g., the utterance "show me the bag with brown handles" in Figure 1. For this, we employ the Multi-Modal Context Carryover (MMCC) (Wanigasekara et al., 2022) dataset to assess our model's performance in accurately selecting the correct product based on the given criteria. In the image retrieval task, the objective is to identify relevant products at the final turn of a multi-turn, multi-modal dialogue.

We use the Multi-Modal Domain Aware (MMDA) dataset (Saha et al., 2018), which is rich in image-inclusive dialogues, for evaluating our model's performance for an image retrieval task. Our results demonstrate a significant improvement over existing models, including the pretrained multi-modal cross-attention model, OpenFlamingo (Awadalla et al., 2023). We achieve an increase of approximately 5 points in F1 score, while training 4x fewer parameters. This underscores the efficiency of our proposed parameter augmentation methodology for multi-modal co-reference resolution.

2. Related Work

There have been several elaborate image-text models over the years, such as CLIP (Radford et al., 2021) and BLIP models (Li et al., 2022, 2023). The goal of this work is aligning the embeddings of such visual models with embeddings of pretrained language models efficiently. Alignment can be classified as either natural language alignment or embedding alignment.

2.1. Multi-Modal Alignment Approaches

Natural language alignment between vision and language foundation models consists of first representing the vision input as text using an image-text model (such as CLIP, BLIP, and BLIP-2) then processing the unified text using a language model (Guo et al., 2023; Wu et al., 2023). This has shown to have zero-shot capabilities, but can be limited because of its discrete nature. To overcome this, Visual ChatGPT (Wu et al., 2023) combines 22 vision foundation models for different vision tasks and a prompt manager that determines how the vision foundation models are used. This is a complex and resource-intensive setup.

Embedding alignment employs neural approaches to translate the embeddings of the vision foundation model to the embedding space of the language model. This approach can be robust, but does not have zero-shot capabilities unless pretrained on a multi-modal dataset first. To achieve such an alignment, Flamingo(Alayrac et al., 2022), Open-Flamingo (Awadalla et al., 2023) and BLIP (Li et al., 2022) use cross attention and contrastive learning objectives. BLIP2 (Li et al., 2023) proposes a querying transformer to learn queries for the visual embeddings. Mini-GPT4 (Zhu et al., 2023) proposes to only train a linear projection layer to project visual embeddings to text space. Alternatively, one can use convolution and linear layer (Koh et al., 2023a; Lyu et al., 2023) with or without a separate modality encoder (Lyu et al., 2023; Moon et al., 2023; Koh et al., 2023b) for a similar projection. Most recently, GILL(Koh et al., 2023a) uses a linear projection and a learnable query embeddings module.

Currently. there are closed-source pipelines such as GPT-4 (OpenAI, 2023; Yang et al., 2023) and GEMINI (Team et al., 2023) that perform a similar multi-modal co-reference resolution task as ours. Given that they are closed-source and have the possibility of using a multi-modal mixture of experts setup, we do not compare our work with them.

2.2. Multi-Modal Co-Reference Resolution

MCR bridges the gap between language and images by mapping the text to spatial regions being referred. A closely related field, Visual Grounding (VG) seeks to align text queries with their corresponding locations in images. In the VG domain, JR-net (Jain and Gandhi, 2022) is one of the SOTA methods; it separately encodes images and queries and then employs a sophisticated joint reasoning and fusion method to generate results. VLT (Ding et al., 2023) is another method that transforms the image data into the same space as language token embeddings and uses a masked decoder to locate targets. Several datasets have also been introduced to support the research in the direction of MCR which includes CIN (Goel et al., 2022) which is rich in co-reference chains and grounding annotations, and others (Parcalabescu et al., 2022; Ramanathan et al., 2014; Cui et al., 2021; Hong et al., 2023) that link textual mentions of people with their images.

More recently, SIMMC2.0 and SIMMC2.1 datasets are introduced as challenges in DSTC. These datasets encompass 11,000 task-oriented dialogues for shopping scenarios with photorealistic scenes, spurring the development of numerous multi-modal methods tailored to conversational agents. In (Lee et al., 2022) proposes a multi-modal encoder-decoder model that offers a unified solution for various tasks associated with situated conversational agents, including MCR. GraVL (Guo et al., 2022) introduce an innovative approach to merge Graph Neural Networks with VL BERT capturing visual relationships alongside dialogue and metadata for nuanced understanding. SHIKRA (Chen et al., 2023) stands out by proposing a multi-modal model capable of engaging in referential dialogue, enabling users to input specific image regions and responding by referencing the pertinent areas if required.

3. Our Approach

3.1. Motivation

In the techniques discussed previously (Alayrac et al., 2022; Awadalla et al., 2023; Zhu et al., 2023; Lyu et al., 2023; Koh et al., 2023a), there is a logical separation of input based on modalities, even though the model may accept interleaved multimodal inputs. For instance, cross attention uses one modality as attention query and another modality as attention key, whereas the querying transformer learns queries from one modality then feeds it through self and cross attention layers to the other modality. We argue that such a logical separation, though sufficient for types of tasks where modalities are separate e.g. VQA, Image Captioning etc., is suboptimal for a multi-modal co-reference resolution. This is in line with findings from (Koh et al., 2023b) who observe poor performance when performing an image retrieval task over multiple co-referenced images. We test this hypothesis using our proposed approach, which preserves the sequence of the multi-modal information during processing. We use OpenFlamingo as baseline for our experiments.

3.2. Problem Formulation

In our setup, a multi-modal dialogue $D := \{(U_i, S_i, I_i)\}_{i=1}^{s}$ contains s turns, each of them composed of a user textual utterance U_i, the system answer S_i, and the images I_i.

Due to the nature of the chosen datasets (i.e. shopping context), at each turn, the images I_i are interleaved within the system utterance, while the user utterance is fully uni-modal. Note however that both the user and the system can reference textual or image entities from past turns, requiring multi-modal co-reference resolution. An example of such an interaction is shown in Figure 1.

In what follows, we refer to token and image embeddings as $\mathbf{x}^t$ and $\mathbf{x}^v$. Our approach relies on augmenting a pre-trained LLM $h_\theta(\mathbf{x}^t)$ with frozen parameters θ and hidden dimension d_{llm}. We denote their augmented counterparts with a hat superscript: the augmented LLM is noted $h_{\theta,\hat{\theta}}(\mathbf{x}^t)$, with the set of additional parameters $\hat{\theta}$ and the final augmented hidden dimension $\hat{d}_{llm}$. The difference $\Delta\hat{d} = \hat{d}_{llm} - d_{llm} > 0$ measures the amount of parameters augmentation.

3.3. Architecture

3.3.1. Prompting

To aid the LLM to perform multi-modal co-referencing, we introduce special tokens to delineate the beginning and end of dialogues, as well as the beginning and end of images. We also introduce a special token ($< im >$) to mark the positions of images in the text. This will then be used by the Multi-Modal Interleaver shown in Figure 2 to insert the image embeddings into the text embeddings at the same position the image was in the input, i.e, fusing the special token embeddings with the respective image embeddings.

```
<dialogue>
    ...
    <image><im></image>
    ...
</dialogue>
```

3.3.2. Linear Layer

Image embeddings are obtained from a frozen image encoder v_ϕ that maps the collection of p images to vectors $\mathbf{x}^{v_m} \in \mathbb{R}^{p \times d_{v_m}}$ (e.g., CLIP (Radford et al., 2021)). These visual embeddings need to be aligned with text embeddings coming from the LLM $h_\theta(x^t) \in \mathbb{R}^{d_{llm}}$ (which also includes the placeholder $< im >$ tokens).

To achieve this, we simply apply a linear transformation by multipling with $\mathrm{W}_l \in \mathbb{R}^{d_{v_m} \times d_{llm}}$ similar to (Lyu et al., 2023; Koh et al., 2023b):

$$\mathbf{x}^v = W_l^T \mathbf{x}^{v_m} \quad , \quad \mathbf{x}^v \in \mathbb{R}^{p \times d_{llm}}. \tag{1}$$

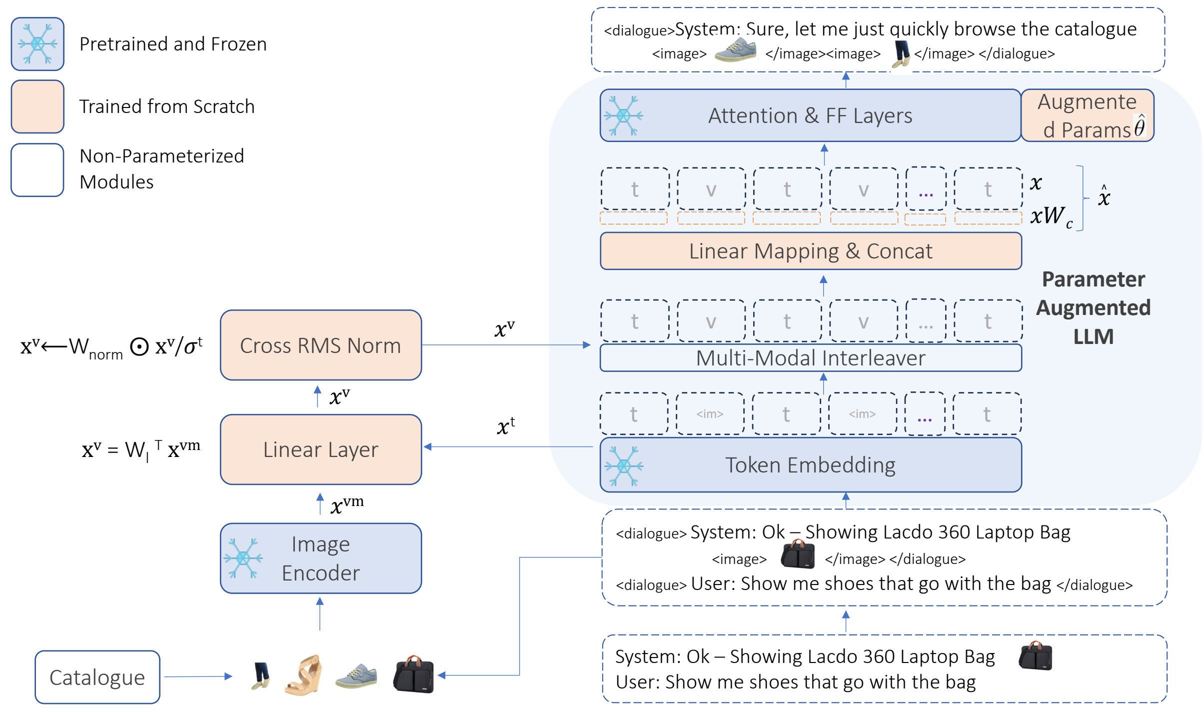


Figure 2: LLM-Agnostic Architecture for Parameter Augmentation. Boxes with t and v refers to text and image embeddings, respectively. We optimize the linear layer, cross RMS normalization module, and the augmented parameters, the rest of the LLM remains frozen. The Multi-Modal Interleaver looks up the position of the images in the input sequence and inserts the image embeddings in their respective positions.

3.3.3. Cross-modality Normalization

Previous works have demonstrated neural architectures to be especially sensitive to the statistics of their activations, exemplified by popular layer normalization blocks such as LayerNorm or RMS (Zhang and Sennrich, 2019) used in LLM architectures. This problem is accentuated in a multi-modal setup; indeed, differences in activations distributions for visual and textual inputs require different normalizations (BatchNorm in Vision vs LayerNorm in NLP) (Shen et al., 2020). As we wish to fuse both the image embeddings $\mathbf{x}^v$ onto LLM token representations $\mathbf{x}^t$, we compute the magnitude of $\mathbf{x}^t$, averaged across the interleaved sequence, and use them to rescale $\mathbf{x}^v$ component-wise. More precisely, considering a sequence of n textual embeddings $\mathbf{x}^t \in \mathbf{R}^{n \times d_{llm}}$:

$$\sigma^t = \sqrt{\frac{1}{d_{llm} \times n} \sum_{i,j} \left(\mathbf{x}^t_{ij} - \mu(\mathbf{x}^t)\right)^2}, \tag{2}$$

$$\mathbf{x}^v \leftarrow \mathbf{x}^v / \sigma^t, \tag{3}$$

with $0 < i < n$ indexing the tokens sequence, $0 < j < d_{llm}$ indexing the features and $\mu(\cdot)$ the mean over both sequence and feature dimensions.

3.3.4. Multi-Modal Interleaver

The role of the Multi-Modal Interleaver (shown in Figure 2, right-hand side) is to preserve the integrity of the sequence of multi-modal input. Since the images are separated from the text so that they can be processed by the image encoder, it is possible to lose the original order of the multi-modal input. Recent works (Lyu et al., 2023) concatenate the multi-modal aligned embeddings, but this changes the sequence of the inputs that will be processed by the model. We replace the removed images with the special token $< im >$ which marks the position of the images. These special tokens will be replaced with cross-modalities normalized embeddings by the Multi-Modal Interleaver. We fuse the embeddings of the special token $< im >$ with the respective aligned image embeddings by a simple elementwise addition operation. The resulting multi-modal embeddings are then passed on to the rest of the LLM Layers as interleaved text and image embeddings, as seen in Figure 2. This has the advantage of performing the essential cross attention operation as shown in Figure 3 without the use of a separate module. It also preserves the distances between tokens and images in the dialogue, which is likely helpful for co-reference resolution.

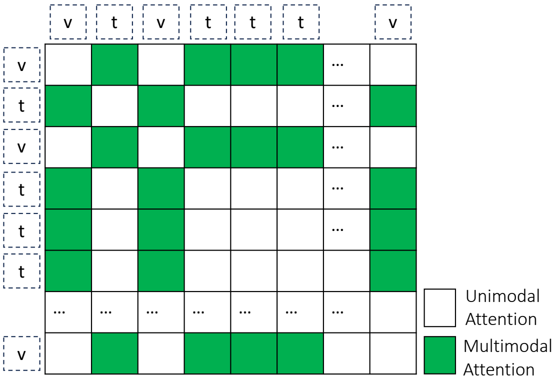


Figure 3: Multi-modal Attention has the advantage of implying both self and cross attention using the same parameters, while preserving the original order of the interleaved image-text sequence.

3.3.5. Parameter Augmentation

LLMs have been shown to exhibit the catastrophic forgetting phenomena after being fine-tuned on data with a different underlying distribution (Zhai et al., 2023). A straightforward mitigation is to freeze the LLM (Zhai et al., 2023). This is the foundation principle behind Parameter Augmentation, i.e., we freeze the uni-modal LLM parameters θ and introduce separate parameters $\hat{\theta}$ to map separate modalities together as seen in Figures 2 and 4. We argue that this preserves the robustness of LLMs, allowing the transfer of their high-quality representations to other modalities. To upcycle the LLM $h_\theta(\cdot)$

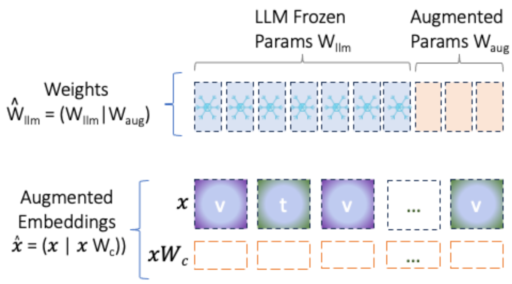


Figure 4: LLM parameters are depicted with the ice icon, showing they are frozen. The Parameter Augmented LLM weights are obtained by concatenating the frozen weights of the LLM and the augmented parameters.

to $h_{\theta,\hat{\theta}}(\cdot)$, we augment the modules at each layer by extending the hidden dimension d_{llm} through concatenation of additional weights: for each existing LLM weight matrix $W_{llm} \in \mathbf{R}^{r \times d_{llm}}$, we create $\hat{W}_{llm} = (W_{llm}|W_{aug})$, with trainable weights where $W_{aug} \in \mathbf{R}^{r \times \Delta\hat{d}}$. All subsequent operations (attention, normalization, feed forward) are therefore between inputs and augmented weights $\hat{W}_{llm}$. We demonstrate that even a small increase $\Delta\hat{d}$ along the hidden dimension is sufficient for the augmented LLM to learn complex relationships such as those in Figure 1. By only optimizing W_{aug} (and freezing W_{llm}), our approach reaps computation and memory benefits. As a comparison, the baseline (Awadalla et al., 2023) increases the LLM parameters by 18.7% while our approach increases it by 5.3%.

After the multi-model interleaver, the sequence of n fused image and token vectors $\mathbf{x} \in \mathbb{R}^{n \times d_{llm}}$ are still of the dimension of the original LLM d_{llm}. To map them to $\hat{\mathbf{x}} \in \mathbb{R}^{n \times \hat{d}_{llm}}$, we add an additional linear adapter $W_c \in \mathbb{R}^{d_{llm} \times \Delta\hat{d}}$:

$$\hat{\mathbf{x}} = (\mathbf{x}|\mathbf{x}W_c) \quad , \quad \hat{\mathbf{x}} \in \mathbb{R}^{n \times \hat{d}_{llm}}. \tag{4}$$

By concatenating the augmented dimensions with the original embedding themselves, we hope to keep intact the spatial information encoded in pretrained LLM embeddings (also see illustration in Figure 4).

We can now optimize the negative log likelihood $\mathcal{L}_{\hat{\theta}}$ of the augmented LLM, with respect to $\hat{\theta}$. Element x_i at any position i below can be either image or text, their order determined by the interleaved sequence:

$$\mathcal{L}_{\hat{\theta}} = -\frac{1}{B}\sum_{j=0}^{B}\sum_{i=1}^{n} \log \left(h_{\theta,\hat{\theta}}(x_i|x_0, \cdots, x_{i-1}) \right), \tag{5}$$

where j indexes the dataset of size B.

4. Experiment Set Up

We experiment on an image selection (Wanigasekara et al., 2022, 2023) and a specially curated image retrieval dataset adapted from (Saha et al., 2018). We measure performance for image selection using accuracy while we use classification metrics (accuracy, precision, recall, F1) for the image retrieval task. For both datasets, we fine-tune the models for only 1 epoch.

4.1. Datasets

The Multi-Modal Context Carryover (MMCC) dataset (Wanigasekara et al., 2022, 2023) is similar to datasets used in VQA and Image Captioning, i.e. images can be logically separated from text. The Multi-Modal Domain Aware (MMDA) (Saha et al., 2018) contains an average of 32 images per dialogue, logically separating the images from text can impede performance. Also, in the MMDA dataset multiple images can be correct, unlike the MMCC dataset which has only one correct image.

The **Image Selection** task is performed on the Multi-Modal Context Carryover (MMCC) dataset (Wanigasekara et al., 2022, 2023). This dataset

	Train	Valid	Test
# Dialogues	38,843	8,373	8,478
Avg # Tokens	717.5	713.7	707.9
Avg # Images	32.2	32.2	31.9
Avg # Utterances	13.3	13.1	13.1
Ratio P:N	1:6	1:6	1:6

Table 1: Dataset Statistics for the curated MMDA dataset. The ratio P:N is the ratio of the positively annotated images against negatively annotated images at the terminal utterance. A label is considered positive if it is relevant to the user's query that involves co-reference resolution.

has $33k$ entries, each containing 3 product images, their descriptions and selection criteria. Given the list of products images, product descriptions and selection criteria, the task is to select the product which has the highest probability to match the criteria. We model this as a generation rather than a classification task, where the LLM generates the index of the product image. We prompt this as shown below:

```
Image <position><image><im><image>
<description>
Action: Given the list of images,
determine the position of the image
that satisfies the criteria
Criteria: <criteria>
Position: <MASK>
```

The **Image Retrieval** task is performed on the Multi-Modal Domain-Aware (MMDA) (Saha et al., 2018) dataset. We require that contexts have at least 1 multi-modal utterance and that the last utterance (where inference happens) have both positive and negative labelled data. We discard all dialogues that do not meet this criteria. During inference, we shuffle the list of positive and negative images and predict whether each one belongs to the last utterance or not.

```
User: ...
System: ...
Question: Is <image><im></image>
a good match?
Answer: <MASK>
...
```

4.2. Pretrained vision encoders and multi-modal LLMs

Pretrained vision encoders: We are able to directly use pretrained vision encoders like CLIP and BLIP as simple baselines for the image selection task in a zero-shot manner. We extract product image and product description text embeddings separately. The image with the highest cosine similarity with the textual referring utterance is chosen as the selected image. For the image retrieval task in a dialogue setting, the dialogue contexts are too long for the direct use of CLIP and BLIP encoders (717 ± 410 tokens) so we use OpenFlamingo as our baseline.

Pretrained multi-modal LLMs: OpenFlamingo (Awadalla et al., 2023) is the publicly available version of the Flamingo (Alayrac et al., 2022) LLM. The 9B variant of OpenFlamingo is made up of a 7B MPT LLM (Team, 2023) with CLIP as the image encoder. It is pretrained on the LAION multi-modal dataset and so has some zero-shot capabilities. In the image selection task, we prompt the model to generate the index of the relevant image while for the image retrieval task, we prompt the model to generate a binary answer (Yes / No) for each image in the candidates.

4.3. Augmented LLM

The parameter augmentation technique we propose is LLM-agnostic. In our experiments, we augment the parameters of the Open LLaMA (Touvron et al., 2023a) 7B model. This model has a hidden dimension size of 4096, we explore augmentations in the range $\Delta\hat{d} = 0$ to $\Delta\hat{d} = 256$.

We prompt the augmented LLaMA similarly as OpenFlamingo and perform ablation experiments for both image selection and retrieval tasks, treating $\Delta\hat{d}$ as a hyperparameter. This augmented LLaMA model is at a disadvantage when compared to the OpenFlamingo model because the OpenFlamingo model has been further fine-tuned on multi-modal tasks using 2B image-text pairs from the LAION (Schuhmann et al., 2022) dataset. Thus, augmented LLaMA is not directly comparable with OpenFlamingo in zero-shot or in-context learning and is disadvantaged for fine-tuning. However, we see that it out-performs pretrained OpenFlamingo as shown in table 3

5. Results

The multi-modal multi-turn setting adds complexity to the co-referencing problem since each user utterance can reference *any* system utterance in the previous turns as seen in Figures 6 and 7. In this paper, we use image retrieval metrics as a proxy to measure multi-modal co-referencing.

5.1. Image Selection Results

Figure 5 shows ablation experiment results on the MMCC dataset (Wanigasekara et al., 2022, 2023). The “linear layer" only includes the linear module shown in Figure 2 and the “linear layer & norm" has the linear module and the cross RMS norm module. We sweep the hyperparameter $\Delta\hat{d}$ from 0 to 256 where 0 indicates no augmentation. We

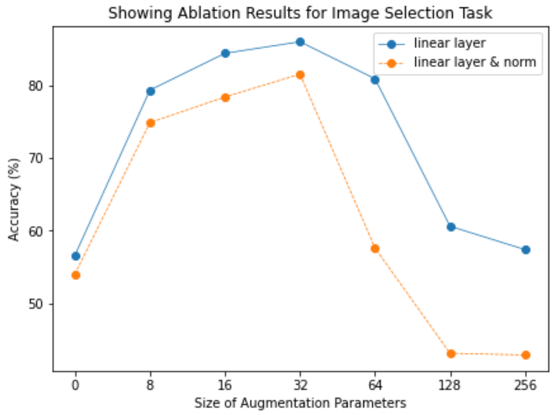


Figure 5: Ablation results of Parameter-Augmented LLM on image selection task, showing accuracy as a function of $\Delta\hat{d}$.

Figure 6: Sample Image Retrieval result for Parameter Augmented LLaMA on a dialogue with 5 utterances and 5 images before the final utterance. In this case, F1=1.0. The green box indicates that the image is relevant, and the red box indicates that the image is not relevant to the user query.

observe parameter augmentation range $8-64$ to be the best setting for both “linear layer" and “linear layer & norm".

Table 2 shows the results using different Visual Language models on MMCC dataset. We obtain the LSTM results from previous SOTA (Wanigasekara et al., 2022) for the image selection task. For the image encoders, we observe that CLIP (Radford et al., 2021) has better performance compared to BLIP (Li et al., 2022). Prompting and fine-tuning OpenFlamingo resulted in the best performance overall. Parameter Augmented LLaMA with $\Delta\hat{d}=32$ performed better than OpenFlamingo in zero-shot but was outperformed in in-context and fine-tuned settings. We see that a uni-modal LLM such as LLaMA can transfer its capabilities to the multi-modal setting through parameter augmentation and approach the performance of a pretrained model.

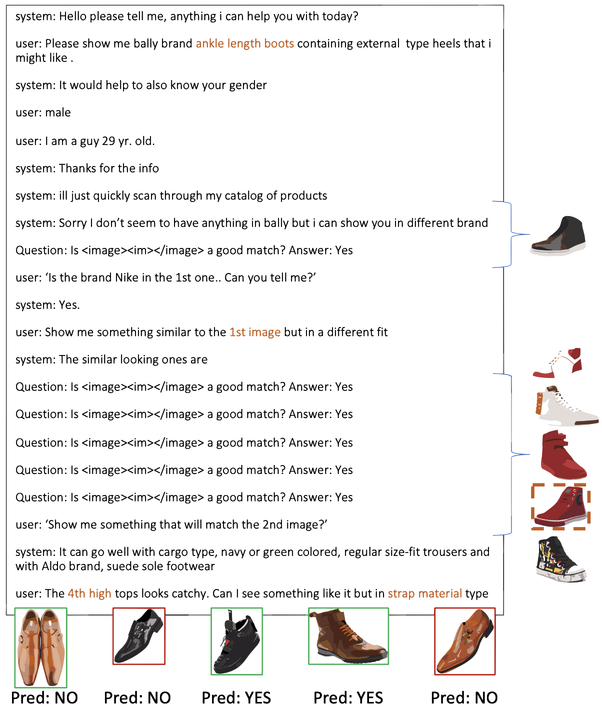


Figure 7: Sample Image Retrieval result for Parameter Augmented LLaMA on a dialogue with 16 utterances and 6 images before the final utterance. In this case, F1=0.8. The green box indicates that the image is relevant, and the red box indicates that the image is not relevant to the user query. The orange dashed box refers to the image the user is currently using as an example.

Model	Set Up	Accuracy
BLIP	zero-shot	44.17%
CLIP	zero-shot	77.40%
LSTM + CLIP	Fine-Tuning	84.84%
LSTM + ALBEF	Fine-Tuning	86.17%
OpenFlamingo	zero-shot	32.40%
	In Context	38.48%
	Fine-Tuning	**90.12%**
Parameter Augmented LLaMA $\Delta\hat{d}=32$	zero-shot	34.51%
	In Context	34.92%
	Fine-Tuning	85.95%

Table 2: Showing results of image-text models, ensemble, OpenFlamingo and Parameter-Augmented LLM on image selection task. LSTM results are from previous state of the art (Wanigasekara et al., 2022)

.

5.2. Image Retrieval Results

In Table 3, we show the performance of multi-modal LLMs on the MMDA dataset. We observe poor zero-shot and in-context performance using Open

Model	Experiment Set Up	Accuracy	Precision	Recall	F1
OpenFlamingo	zero-shot	35.67%	0.3472	0.9635	0.4621
	In-Context	36.14%	0.3486	0.9651	0.4648
	Fine-Tuning	76.70%	0.6953	0.8235	0.7240
Linear Layer	Fine-Tuning	66.77%	0.5583	0.8334	0.5995
Parameter Augmented LLaMA Fine-Tuning	Linear Layer $\Delta\hat{d} = 64$	77.89%	**0.7118**	**0.9334**	**0.7727**
	Linear Layer & Norm $\Delta\hat{d} = 64$	70.93%	0.6519	0.9096	0.7135
	Linear Layer $\Delta\hat{d} = 128$	77.84%	0.6702	0.6510	0.6228
	Linear Layer & Norm $\Delta\hat{d} = 128$	70.64%	0.5997	0.9213	0.6851
	Linear Layer $\Delta\hat{d} = 256$	78.73%	0.6373	0.5090	0.5323
	Linear Layer & Norm $\Delta\hat{d} = 256$	**79.05%**	0.6595	0.8437	0.7122

Table 3: Showing results of a OpenFlamingo and our Parameter-Augmented LLM for image retrieval task applied on the MMDA dataset.

Flamingo, highlighting the difficulty of the task. After fine-tuning, the Parameter Augmented LLaMA ($\Delta\hat{d} = 64$) outperforms fine-tuned OpenFlamingo. This highlights the robustness of parameter augmentation over cross-attention.

5.3. Qualitative Analysis

Figure 6 and 7 show sample result of Parameter Augmented LLaMA on the image retrieval MMDA dataset with F1 score of 1.0 and 0.8 respectively. A red box around an image refers to a negatively labelled image, while a green box refers to a positively labelled image. The dashed box refers to the image the user is currently using as an example. The models then predict Yes/No given a list of images.

Our approach is able to differentiate between styles of similar images as we show in Figure 6 where the candidate products are both sandles but different styles, this is more granular than differentiating unrelated objects e.g., sandles vs chair. In Figure 7, we see similar capabilities over more utterances. We attribute the false negative result (prediction No but the box is green) in Figure 7 as a mis-annotation because the shoe is not similar to co-referenced shoe (shoe with orange dashed border) and is not made of strap material nor a high top as specified by user utterance.

In the OpenFlamingo architecture, $1.3B$ of the $9B$ parameters are optimized, this accounts for 18.6% with respect to its uni-modal LLM ($7B$). In the parameter augmented setting with $\Delta\hat{d} = 64$, we optimize $370M$ parameters (5.3% of uni-modal LLM) which is a more resource efficient setup. Thus, our model optimizes 13.3% fewer parameters with respect to the uni-modal LLM (in both cases, the uni-modal LLM is 7B).

In the image selection results in Figure 5, we see a significant drop in performance for $\Delta\hat{d} = 128$ and $\Delta\hat{d} = 256$. This is because it introduces more than $1.5\times$ more parameters compared to the other augmentations. The image selection dataset is comparatively smaller and has a total of approximately $3M$ tokens, and training on one epoch is insufficient given the higher number of parameters. For image retrieval results (Table 3), the dataset is comparatively larger and has approximately $30M$ tokens, and so we see steady performance improvements with higher augmentations.

The augmented LLM variant also resulted in the best performance for the image retrieval dataset, exceeding that of a model with $1.2\times$ parameters, trained on $20k\times$ more data while optimizing $3.5\times$ fewer parameters. We see more gains for the MMDA dataset than the MMCC dataset, where the co-reference is simpler. This is in line with our hypothesis - we reap more benefits from using parameter augmentation when the degree of multi-modal co-referencing increases.

For the image selection task based on Figure 5, for augmentation $\Delta\hat{d} = 128$ and $\Delta\hat{d} = 256$, the Cross Normalization is significantly outperformed by the normalization ablation. Overall, variants with cross normalization are outperformed by the variants without normalization. We observe a different trend for image retrieval in that the $\Delta\hat{d} = 256$ augmented LLaMA, with normalization performing better than without normalization, setting the best accuracy result (see Table 3). However, we observe more over-fitting to the data when normalization is used, creating the need for a better design for multi-modal normalization. We will explore this in our future work.

6. Conclusion

We explore the possibility to leverage existing pre-trained LLM capabilities and offer a simple and robust parameter augmentation technique that does not require additional multi-modal pre-training tasks.

We demonstrate competitive results in image selection and best results in the image retrieval dataset compared to a cross-attention baseline pre-trained on billions of multi-modal examples.

7. References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. 2022. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736.

Anas Awadalla, Irena Gao, Josh Gardner, Jack Hessel, Yusuf Hanafy, Wanrong Zhu, Kalyani Marathe, Yonatan Bitton, Samir Gadre, Shiori Sagawa, et al. 2023. Openflamingo: An open-source framework for training large autoregressive vision-language models. *arXiv preprint arXiv:2308.01390*.

Keqin Chen, Zhao Zhang, Weili Zeng, Richong Zhang, Feng Zhu, and Rui Zhao. 2023. Shikra: Unleashing multimodal llm's referential dialogue magic. *arXiv preprint arXiv:2306.15195*.

Alexandra Chronopoulou, Christos Baziotis, and Alexandros Potamianos. 2019. An embarrassingly simple approach for transfer learning from pretrained language models. *arXiv preprint arXiv:1902.10547*.

Claire Yuqing Cui, Apoorv Khandelwal, Yoav Artzi, Noah Snavely, and Hadar Averbuch-Elor. 2021. Who's waldo? linking people across text and images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1374–1384.

Henghui Ding, Chang Liu, Suchen Wang, and Xudong Jiang. 2023. Vlt: Vision-language transformer and query generation for referring segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):7900–7916.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. 2023. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*.

Arushi Goel, Basura Fernando, Frank Keller, and Hakan Bilen. 2022. Who are you referring to? weakly supervised coreference resolution with multimodal grounding. *arXiv preprint arXiv:2211.14563*.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913.

Danfeng Guo, Arpit Gupta, Sanchit Agarwal, Jiun-Yu Kao, Shuyang Gao, Arijit Biswas, Chien-Wei Lin, Tagyoung Chung, and Mohit Bansal. 2022. Gravl-bert: graphical visual-linguistic representations for multimodal coreference resolution. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 285–297.

Jiaxian Guo, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Boyang Li, Dacheng Tao, and Steven Hoi. 2023. From images to textual prompts: Zero-shot visual question answering with frozen large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10867–10877.

Xudong Hong, Asad Sayeed, Khushboo Mehra, Vera Demberg, and Bernt Schiele. 2023. Visual writing prompts: Character-grounded story generation with curated image sequences. *Transactions of the Association for Computational Linguistics*, 11:565–581.

Kanishk Jain and Vineet Gandhi. 2022. Comprehensive multi-modal interactions for referring image segmentation. In *Findings of the Association for Computational Linguistics: ACL 2022*. Association for Computational Linguistics.

Jing Yu Koh, Daniel Fried, and Ruslan Salakhutdinov. 2023a. Generating images with multimodal language models. *arXiv preprint arXiv:2305.17216*.

Jing Yu Koh, Ruslan Salakhutdinov, and Daniel Fried. 2023b. Grounding language models to images for multimodal generation. *arXiv preprint arXiv:2301.13823*.

Satwik Kottur, Seungwhan Moon, Alborz Geramifard, and Babak Damavandi. 2021. Simmc 2.0: A task-oriented dialog dataset for immersive multimodal conversations. *arXiv preprint arXiv:2104.08667*.

Hung Le, Nancy F Chen, and Steven CH Hoi. 2022. Multimodal dialogue state tracking. *arXiv preprint arXiv:2206.07898*.

Haeju Lee, Oh Joon Kwon, Yunseon Choi, Minho Park, Ran Han, Yoonhyung Kim, Jinhyeon Kim, Youngjune Lee, Haebin Shin, Kangwook Lee, et al. 2022. Learning to embed multi-modal contexts for situated conversational agents. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 813–830.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, pages 12888–12900. PMLR.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Yunyao Li and Angeliki Lazaridou. 2022. Proceedings of the 2022 conference on empirical methods in natural language processing: Industry track. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer.

Chenyang Lyu, Minghao Wu, Longyue Wang, Xinting Huang, Bingshuai Liu, Zefeng Du, Shuming Shi, and Zhaopeng Tu. 2023. Macaw-llm: Multi-modal language modeling with image, audio, video, and text integration. *arXiv preprint arXiv:2306.09093*.

Seungwhan Moon, Andrea Madotto, Zhaojiang Lin, Tushar Nagarajan, Matt Smith, Shashank Jain, Chun-Fu Yeh, Prakash Murugesan, Peyman Heidari, Yue Liu, et al. 2023. Anymal: An efficient and scalable any-modality augmented language model. *arXiv preprint arXiv:2309.16058*.

OpenAI. 2023. Gpt-4 technical report.

Letitia Parcalabescu, Michele Cafagna, Lilitta Muradjan, Anette Frank, Iacer Calixto, and Albert Gatt. 2022. VALSE: A task-independent benchmark for vision and language models centered on linguistic phenomena. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8253–8280, Dublin, Ireland. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Vignesh Ramanathan, Armand Joulin, Percy Liang, and Li Fei-Fei. 2014. Linking people in videos with "their" names using coreference resolution. In *Computer Vision – ECCV 2014*, pages 95–110, Cham. Springer International Publishing.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Amrita Saha, Mitesh Khapra, and Karthik Sankaranarayanan. 2018. Towards building large scale multimodal domain-aware conversation systems. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. 2022. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294.

Sheng Shen, Zhewei Yao, Amir Gholami, Michael Mahoney, and Kurt Keutzer. 2020. Powernorm: Rethinking batch normalization in transformers. In *International Conference on Machine Learning*, pages 8741–8751. PMLR.

Krishna Srinivasan, Karthik Raman, Jiecao Chen, Michael Bendersky, and Marc Najork. 2021. Wit: Wikipedia-based image text dataset for multimodal multilingual machine learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2443–2449.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

MosaicML NLP Team. 2023. Introducing mpt-7b: A new standard for open-source, commercially usable llms.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Prashan Wanigasekara, Rafid Al-Humaimidi, Turan Gojayev, Niloofar Gheissari, Achal Dave, Stephen Rawls, Fan Yang, Kechen Qin, Nalin Gupta, Spurthi Sandiri, et al. 2023. Visual item selection with voice assistants: A systems perspective. In *Companion Proceedings of the ACM Web Conference 2023*, pages 500–507.

Prashan Wanigasekara, Nalin Gupta, Fan Yang, Emre Barut, Zeynab Raeesy, Kechen Qin, Stephen Rawls, Xinyue Liu, Chengwei Su, and Spurthi Sandiri. 2022. Multimodal context carryover. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 417–428.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. 2023. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*.

Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. 2023. The dawn of lmms: Preliminary explorations with gpt-4v(ision).

Xiaoxue Zang, Lijuan Liu, Maria Wang, Yang Song, Hao Zhang, and Jindong Chen. 2021. Photochat: A human-human dialogue dataset with photo sharing behavior for joint image-text modeling. *arXiv preprint arXiv:2108.01453*.

Yuexiang Zhai, Shengbang Tong, Xiao Li, Mu Cai, Qing Qu, Yong Jae Lee, and Yi Ma. 2023. Investigating the catastrophic forgetting in multimodal large language models. *arXiv preprint arXiv:2309.10313*.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*.

Efficient and Interpretable Information Retrieval for Product Question Answering with Heterogeneous Data

Biplob Biswas, Rajiv Ramnath
The Ohio State University
Columbus OH 43210, USA
{biswas.102, ramnath.6}@osu.edu

Abstract

Expansion-enhanced sparse lexical representation improves information retrieval (IR) by minimizing vocabulary mismatch problems during lexical matching. In this paper, we explore the potential of jointly learning dense semantic representation and combining it with the lexical one for ranking candidate information. We present a hybrid information retrieval mechanism that maximizes lexical and semantic matching while minimizing their shortcomings. Our architecture consists of dual hybrid encoders that independently encode queries and information elements. Each encoder jointly learns a dense semantic representation and a sparse lexical representation augmented by a learnable term expansion of the corresponding text through contrastive learning. We demonstrate the efficacy of our model in single-stage ranking of a benchmark product question-answering dataset containing the typical heterogeneous information available on online product pages. Our evaluation demonstrates that our hybrid approach outperforms independently trained retrievers by 10.95% (sparse) and 2.7% (dense) in MRR@5 score. Moreover, our model offers better interpretability and performs comparably to state-of-the-art cross encoders while reducing response time by 30% (latency) and cutting computational load by approximately 38% (FLOPs).

Keywords: Hybrid Information Retrieval, Interpretability, Heterogeneous Product Question-Answering

1. Introduction

In the field of natural language processing, ranked information retrieval (IR), refers to retrieving information ordered by relevance from a large collection, in response to a query. Ranked IR remains important even with the emergence of advanced large language models (LLMs) as a means of greatly enriching their outputs.

Existing retrieval approaches can be categorized into two groups - sparse and dense. Sparse retrieval uses a token-based sparse representation of the query and the information, such as bag-of-words (BoW) obtained via TF-IDF (Sparck Jones, 1988) or BM25 (Robertson and Walker, 1994), and an inverted index for query processing. Although these BoW models facilitate faster retrieval, they rely on exact matches, and hence cannot identify semantically relevant information having a different set of tokens than the query. Dense retrieval, on the other hand, retrieves by comparing dense representations often computed by neural networks such as BERT (Devlin et al., 2019). While these models can perform semantic-level matching, their computational complexity renders them impractical for online real-time ranking when the corpus becomes large.

In an effort to balance the quality-cost trade-off, a two-stage pipeline is proposed where a quicker retriever first retrieves a smaller set of candidates and then a dense retriever re-ranks them in a second stage. Unfortunately, this approach suffers from two major problems. First, any semantically relevant information pruned due to lack of exact word matches in the first stage is not considered for further ranking. Second, the neural ranker in the last stage lacks interpretability because, for scoring, it uses the inner product of the latent representation of the text which is difficult to explain in human understandable terms. Recently proposed transformer (Vaswani et al., 2017) encoders have the potential to tackle these issues. By utilizing a pre-trained masked language model (MLM), SparTerm (Bai et al., 2020) and SPLADE (Formal et al., 2021) progressively improved the use of expansion-aware sparse lexical representation learners in mitigating vocabulary mismatch problems, while enhancing interpretability. SparseEmbed (Kong et al., 2023) further extended this concept by learning contextual embeddings of the top-k tokens in the lexical representation. However, these models ignore the text-level dense representation (i.e. [CLS] token encoding) which captures the summarized expression of a text. Furthermore, being a byproduct of the BERT with MLM head, it can be obtained without additional computation and stored as a single vector. Finally, jointly learning lexical and semantic representations can pave the way for a single-stage ranking, especially in product-question-answering tasks (Shen et al., 2022) where information from an online product page can be pre-computed offline and then ranked at query time.

In this work, we investigate these possibilities and present a hybrid information ranker that balances the quality, cost, and interpretability by incorporating both lexical and semantic matching in ranking.

The Seventh Workshop on e-Commerce and NLP (ECNLP 7), pages 19–28

The contribution of our work is in two areas:

- We present a hybrid ranking model that jointly learns semantic and lexical representations and combines them for efficient information retrieval.
- We evaluate our model on a heterogeneous product question-answering dataset and show that our approach provides better performance and interpretability with a reasonable computational complexity and memory footprint. Our code is available online[1].

2. Related Works

Our hybrid model brings together ideas from both dense retrieval and sparse retrieval. Based on the scoring process, we find three variants of dense retrievers (as shown in Figure 1) related to our work; all of them employ pre-trained language models to learn dense semantic representations. Nogueira et al. (Nogueira and Cho, 2019) used BERT as a *cross-encoder*(Figure 1(c)) where concatenated query-information sequence is processed simultaneously through all-to-all interactions and a binary classifier maps the resultant representation to relevance probability. In DPR (Karpukhin et al., 2020), Karpukhin et al. employed two *independent dense encoders* (Figure 1(a)) that separately map query and information into their single-vector dense representations and the information score is computed by their inner product. To improve model expressiveness, Khattab et al. proposed ColBERT (Khattab and Zaharia, 2020), a *late-interaction*(Figure 1(b)) model, to utilize a multi-vector representation from dual encoders that allow deferred cross interaction among contextual token encodings. However, ColBERT suffers from scalability issues as it requires storing and indexing all the token encodings in a sequence.

Term-based BM25 (Robertson and Walker, 1994) has been long used as a baseline for sparse retrieval. In order to capture semantic relationships in sparse representations, SNRM (Zamani et al., 2018) uses high-dimensional vectors of latent terms. However, it loses the interpretability provided by actual vocabulary terms. SparTerm (Bai et al., 2020) addresses this (interpretability) issue by mapping text to a sparse term-importance distribution in BERT vocabulary space. In SPLADE (Formal et al., 2021), Formal et al. extended this idea by introducing a log-saturation effect in term-importance estimation and sparsity regularization in training loss. Following this, SparseEmbed (Kong et al., 2023) learns and uses contextual embeddings of the sparse lexical representation

[1] https://github.com/biplob1ly/HybridPQA

Evidence Ranking			
Items	**Train**	**Validation**	**Test**
Total records	24295	2731	309347
Unique query	4528	509	2773
Mean candidates per query	5.37	5.37	111.56
Mean +ve candidate ratio	0.25	0.24	0.06
Mean question words	11.23	11.73	6.98
Mean candidate words	17.19	18.49	12.59
Mean sources per query	1.09	1.10	5.12
Answer Generation			
Items	**Train**	**Validation**	**Test**
Total records	3693	398	2289
Unique query	3356	395	1340
Mean evidences per query	1.1	1.01	1.71
Mean answer words	8.22	8.27	7.24

Table 1: The summary of the hetPQA (Shen et al., 2022) dataset.

to improve model expressiveness. Our approach closely follows this direction of research. However, instead of only comparing lexical representation, we also consider summarized semantic matching without increasing encoding complexity, by leveraging the fact that BERT computes the [CLS] token encoding anyway. Moreover, unlike prior hybrid models (Karpukhin et al., 2020; Ma et al., 2021; Gao et al., 2021; Luan et al., 2021), our model *jointly* learns semantic representations and expandable lexical representations, enabling interpretability with expanded tokens.

3. Dataset

We apply our model to hetPQA (Shen et al., 2022), a large-scale benchmark dataset for product question-answering systems, that provides various information from product web pages as candidate evidence to answer a product-specific query. In production, after ranking the candidate evidence elements for a query, the higher-ranked ones are utilized for answer generation. The information (evidence) is extracted from heterogeneous sources that include: 1. product attributes in JSON format, 2. bullet points from product summary, 3. community answers to product questions (CQA), 4. product descriptions, 5. on-site publications (OSP) about products, and 6. user reviews on the product page. The collection has separate sets of data for evidence ranking and answer generation, and each dataset comprises train, validation, and test split. The details of the splits are reported in Table 1. Further, our manual inspection of the BM25-driven evidence ranking result on the test set revealed 1377 incorrect annotations; these were corrected. We have disclosed our correction in the

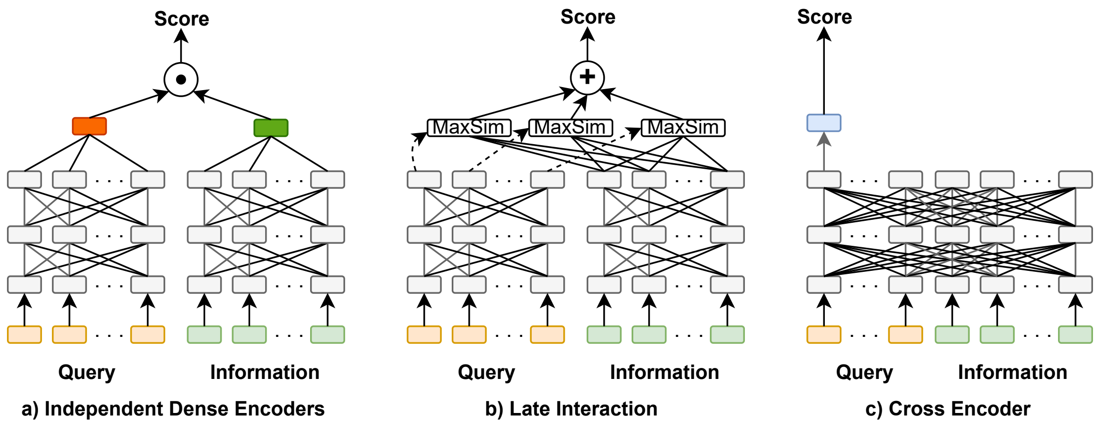


Figure 1: Existing neural rankers with different interaction schemes.

Items	Attribute	Bullet	CQA	Desc	OSP	Review
Ranking	4.0%	4.4%	44.1%	12.8%	2.6%	32.1%
Generation	11.4%	16.6%	21.8%	17.8%	8.6%	23.8%
Mean #words	5.8	12.6	13.3	12.9	17.8	18.4

Table 2: The distribution of sources and mean word count in the hetPQA (Shen et al., 2022) dataset.

repository shared above and also conducted all our experiments with the amended test set. Altogether, the evidence ranking set has 7585 unique questions and 149283 unique pieces of information distributed over the aforementioned 6 sources. The answer generation set contains a total of 5037 unique questions and 5229 unique evidence elements. The overall source distribution and average word counts are given in Table 2. More details can be found in hetPQA (Shen et al., 2022) paper.

Data Preparation To begin with, the text was normalized to a canonical representation. All non-English characters were replaced by their equivalents. Symbols and short forms of dimensions (e.g. $3''\ l \times 4''\ w$) were substituted by the corresponding English words (*length 3 inches* $\times$ *width 4 inches*). We also flattened JSON-formatted attributes to comma-separated strings.

4. Framework

Our framework comprises two major components: a ranker and a generator. Given a query and a set of candidate information, the ranker sorts the information in descending order of relevance. The generator then produces a coherent and informative response from the top-ranked results. We elaborate on this in the subsections below.

4.1. Ranker

The key function of a ranker is to measure the relevance of each candidate information element with respect to the query. Figure 2 depicts the architecture of our proposed hybrid ranker. It consists of two separate modules that can independently compute the representations of the queries and information elements. Given a query $Q = t_{1\cdots|Q|}$ where token $t_i \in V$ for vocabulary V, and a candidate information element $C = t_{1\cdots|C|}$ of the same vocabulary, Our ranker first obtains lexical(l) and semantic(d) representations of the query and the candidate information as l_Q and d_Q, l_C and d_C, respectively following the process described in the next subsections. Then the relevance score of the information is computed by the linear interpolation of their semantic and lexical matching:

$$r(Q,C) = \alpha \times f(d_Q, d_C) + (1-\alpha) \times f(l_Q, l_C) \quad (1)$$

Where $f(Q,C) = Q \cdot C$ and $\alpha \in (0,1)$ is a hyperparameter indicating importance given to the semantic match.

4.2. Representation Learning

The representation learning procedure for query and information has independent yet similar pipelines as shown in Figure 2. The query-encoder is a pre-trained masked language model (MLM) such as BERT (Devlin et al., 2019) and it maps the query token sequence to their contextual embeddings $\boldsymbol{H_Q} \in \mathbb{R}^{|Q|\times h}$ (h: hidden size) and also outputs a summarised representation of entire query in the form of `[CLS]` token embedding $h_{CLS} \in \mathbb{R}^h$. While the sequence encodings can also be pooled to obtain the summarized vector, it requires additional computation. Instead, we use the pre-trained h_{CLS} as the query's dense semantic representation: $d_Q = h_{CLS}$.

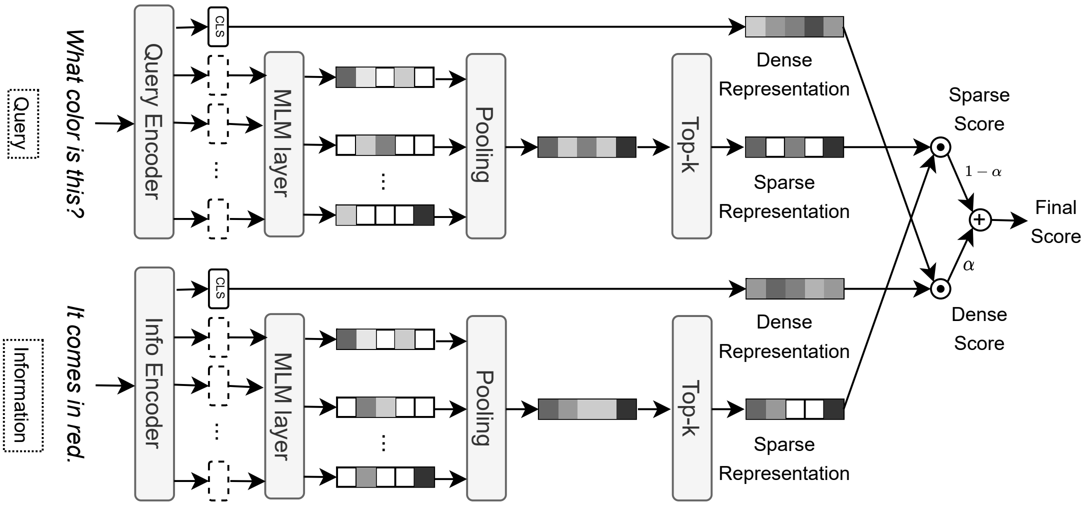


Figure 2: The proposed hybrid information ranker.

We build on the SPLADE (Formal et al., 2021) and SparseEmbed (Kong et al., 2023) methods to compute the lexical representation. In these methods, and as illustrated in Figure 2, the sequence encodings $\boldsymbol{H_Q}$ are fed to the BERT's pre-trained MLM head which maps them to MLM logits, $\boldsymbol{M_Q} \in \mathbb{R}^{|Q|\times|V|}$. Logit value $m_{i,j}$ in $\boldsymbol{M_Q}$ can be considered as an importance indicator of the vocabulary term $v_j \in V$ for the query token $t_i \in Q$. ReLU is applied to the raw logit values to ensure positivity and is followed by a log operation to reduce the dominance of fewer terms. Then the resultant logits are aggregated (using max-pooling or summation) along query token sequences to obtain the combined importance w_j of a term $v_j \in V$ using the following formula:

$$w_j = \max_{i=1\dots|Q|} log(1 + ReLU(m_{i,j})) \tag{2}$$

We collect the aggregated importance over the lexical terms, $W = w_{1\dots|V|}$ through the max pooling layer. To reduce computational complexity during score calculation, we enforce sparsity in W by retaining only the top-k weights in it and zeroing out the rests as shown in Figure 2. This leaves us with an expansion-aware sparse lexical representation $l_Q = W$ of the query. Following a similar approach for the candidate information element, we obtain its dense semantic representation d_C and its sparse lexical representation l_C.

4.3. Loss Function

For the training of the hybrid model, we combine ranking loss due to both semantic and lexical representation. Given a dataset $S_{i\dots|S|}$, where a training instance S_i comprises a query Q_i, a piece of positive information C_i^+ and b negative candidates $(C_{i,1}^-, C_{i,2}^-, \dots, C_{i,b}^-)$, our model is trained to minimize the following contrastive loss for each kind of representation:

$$\mathcal{L}_{rank} = -\log \frac{e^{f(Q_i, C_i^+)/\tau}}{e^{f(Q_i, C_i^+)/\tau} + \sum_{j=1}^{b} e^{f(Q_i, C_{i,j}^-)/\tau}} \tag{3}$$

Here, τ is a temperature hyperparameter. To have an efficient ranking system in terms of computational complexity and memory footprint, it is beneficial to enforce sparsity in the high-dimensional (size: $|V|$) lexical representation. Following SPLADE (Formal et al., 2021), we also use FLOPS loss for this regularization:

$$\mathcal{L}_{reg}^{C} = \sum_{j\in V} \left(\frac{1}{N} \sum_{i=1}^{N} w_j^{(C_i)} \right)^2 \tag{4}$$

where C_i is a candidate information element in a batch of size N and w_j is the importance weight of a vocabulary token computed from Equation 2. Collectively, the training procedure minimizes the following loss function:

$$\mathcal{L} = \mathcal{L}_{rank}^{d} + \mathcal{L}_{rank}^{l} + \lambda^{Q}\mathcal{L}_{reg}^{Q} + \lambda^{C}\mathcal{L}_{reg}^{C} \tag{5}$$

where λ^Q and λ^C are hyperparameters to introduce higher sparsity in query than information for less scoring cost.

4.4. Generator

Given a query Q and n number of potential information elements $C_{1\dots n}$, we aim to generate an answer A. To effectively combine multiple information elements for a query, we employ a fusion-in-decoder (Izacard and Grave, 2021) model for answer generation. It uses a pre-trained sequence-to-sequence network such as T5 (Raffel et al., 2020)

#	Model	MAP	R-Prec	MRR@5	NDCG	Hit Rate@5	P@1
a	**BM25** (Robertson and Walker, 1994)	0.435	0.388	0.622	0.658	0.796	0.510
b	**Cross Encoder** (Nogueira and Cho, 2019)	**0.604**^{acdefghi}	**0.540**^{acdefghi}	**0.795**^{acdefgh}	**0.780**^{acdefghi}	**0.930**^{acde}	**0.703**^{acdefgh}
c	**Independent Dense** (Karpukhin et al., 2020)	0.552^{ade}	0.488^{ae}	0.761^{ade}	0.752^{ade}	0.918^{ade}	0.659^{ade}
d	**Late Interaction**(Khattab and Zaharia, 2020)	0.544^{ae}	0.481^{ae}	0.734^{ae}	0.741^{ae}	0.902^{ae}	0.618^{ae}
e	**Sparse Lexical** (Formal et al., 2021), k=128	0.505^{a}	0.449^{a}	0.694^{a}	0.713^{a}	0.873^{a}	0.572^{a}
f	**Hybrid, k=128**	0.563^{acde}	0.498^{acde}	0.770^{ade}	0.757^{acde}	0.924^{ade}	0.665^{ade}
g	**Hybrid, k=256**	0.572^{acdef}	0.505^{acdef}	0.780^{acdef}	0.763^{acdef}	0.925^{ade}	0.679^{acdef}
h	**Hybrid, k=512**	0.573^{acdef}	0.507^{acdef}	0.782^{acdef}	0.764^{acdef}	0.924^{ade}	0.679^{acdef}
i	**Hybrid, k=512, source-aware**	0.575^{acdef}	0.508^{acdef}	0.792^{acdefgh}	0.766^{acdef}	0.927^{acde}	0.697^{acdefgh}

Table 3: The overall effectiveness of the experimented rankers on the hetPQA (Shen et al., 2022) dataset. The best results are highlighted in boldface. Our hybrid model scores are obtained with $\alpha = 0.5$. Superscripts denote significant differences in both Fisher's randomization test and paired Student's t-test with $p \leq 0.05$.

that first encodes pairs of question and information $< (Q, C_1), (Q, C_2), \cdots, (Q, C_n) >$ independently and then joins the resultant representations in decoder before performing attention. Finally, we use greedy decoding to generate a natural language answer. As this method processes candidate information elements independently, it allows the aggregation of the elements at a relatively lower latency.

5. Experimental Environment

We use BERT-base-uncased (Devlin et al., 2019) (110M parameters) and T5-base (Raffel et al., 2020) (220M parameters) provided by Huggingface (Wolf et al., 2020) as the core model for evidence ranking and answer generation respectively. We set the following hyperparameters to the relevant models: {Max token length (each of question, evidence, answer): 128, Warm-up steps: 200, Batch size: 8, Gradient Accumulation Steps: 8, Learning rate: $1e-5$, λ^Q: $3e-4$, λ^C: $1e-4$}. The evidence rankers and generator are trained for 1,500 and 1000 steps respectively and the best checkpoints are considered for evaluation. All the experiments were conducted using a 5-core CPU node at 2.40 GHz, equipped with a single NVIDIA Tesla P100 16GB GPU core and 25 GB of memory. For preprocessing and evaluation, we use NLTK (Bird et al., 2009), calflops (xiaoju ye, 2023), and ranx (Bassani, 2022). Our baseline methods are listed in the first row of Table 3. We use Okapi BM25 implementation from rank_25 (Brown, 2020). For cross-encoder, independent dense encoders, and late-interaction method, we follow the implementation as described in §2. The only difference between the sparse-lexical method and our hybrid model is that the former does not incorporate semantic matching in computing the loss and the score. For fairness of comparison, none of our dual-encoders use any additional projection layer on top of BERT's layer and for ranking, we sorted all the candidate information based on Equation 1 instead of using any indexer.

6. Evaluation

In this section, we evaluate the performance of the two modules of our framework, viz., evidence ranking and answer generation.

6.1. Evidence ranking

We assess the impact of our proposed method along three dimensions: 1. ranking quality, 2. computational cost and memory footprint, and 3. interpretability. Table 3 lists the experiment results and provides a comparison of our proposed ranking method to the baselines specified in §5. Evaluation of all methods was conducted on the amended held-out test set and on the same environment as mentioned in §5. There are 2583 unique queries in the test set having at least one positive evidence and we consider only those queries for our evaluation.

Ranking Quality To report ranking quality, we utilize six commonly-used evaluation metrics- MAP: mean average precision, R-Prec: precision at the top-R retrieved information elements, MRR@5: mean reciprocal rank within top-5 candidates, NDCG: normalized discounted cumulative gain, Hit rate@5: fraction of queries with at least one positive evidence in top-5 ranked candidates and P@1: precision of the top-ranked evidence. As shown in Table 3, the hybrid approach outperformed all other methods except cross-encoder in all metrics. Although the hybrid model with $k = 128$ (top token count in lexical representation) bests the independent dense encoder model by a slim margin across the metrics, the difference in their effectiveness becomes statistically significant when more tokens ($k \geq 256$) are considered for lexical matching. The

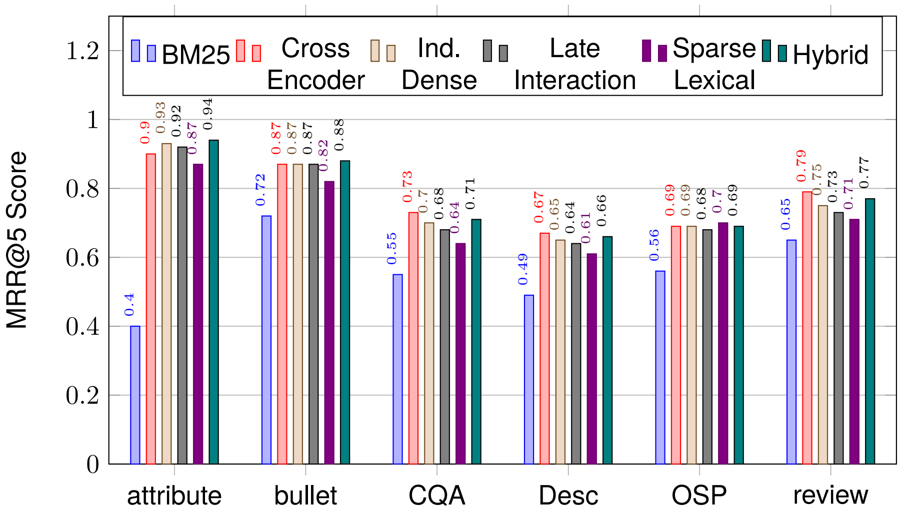


Figure 3: Ranking results of our hybrid ranker on heterogeneous evidence sources.

Resource Requirements

Metric		**BM25**	**Cross Encoder**	**Independent Dense**	**Late Interaction**	**Sparse Lexical**	**Hybrid**
Params		-	109.48M	2x109.48M	2x109.48M	2x109.51M	2x109.51M
Inference FLOPs	Encoding	-	45.94G	22.36G	22.36G	28.51G	28.51G
	Interaction	-	-	$2h$	$2n^2 \cdot h + n$	$2k$	$2(h+k)$
Latency (ms)	Per Query	0.75	475.24	229.93	275.05	296.66	331.83
	Per Info.	0.007	4.2	2.04	2.43	2.63	2.93
Offline Storage (Per Evidence)		-	-	h	$n \cdot h$	$2k$	$h + 2k$

Table 4: Resource requirements of the experimented rankers on the hetPQA (Shen et al., 2022) dataset. Here dense representation size $h = 768$, max sequence length $n = 128$, Count of top tokens considered in lexical representation $k = 128$.

hit-rate@5 indicates the model positions at least one relevant piece of information among the top five in 92.7% of the queries.

Figure 3 illustrates a comparative performance of our proposed method with others across the six different sources of evidence. It shows that our hybrid model dominates existing methods in ranking evidence belonging to the same source. The contrasting score differences between BM25 and neural rankers in attribute and bullet sources not only show the struggle of the pure lexical method with less expressive data but also corroborate the advantage of semantic matching in handling heterogeneous data. In contrast to attribute or bullet evidence which stores clear and concise information, user-driven sources such as CQA and review come with inherent noise including misspellings, presumptive opinions, and so on. According to our manual inspection, these noises contributed to the models' relatively poor performance in these sources.

Resource Requirements Table 4 summarizes the resource requirements of our experimented ranking methods. All the methods employ the identical configuration of BERT (Devlin et al., 2019). Consequently, the parameter size listed in the first row is roughly proportional to that of a single BERT model except for the sparse-lexical and hybrid models where we have $2 \times 0.03M$ additional parameters for MLM layers. The second row provides the number of floating point operations (FLOPs) needed to be done in the inference stage which includes computation for encoding (measured in Giga-scale: 10^9) and interaction. Expectedly, the highly performing cross-encoder costs almost double the GFLOPs incurred by the independent dense encoder as the latter only performs query encoding in live and pre-computes the information representation offline. Late-interaction method, on the other hand, is subject to a quadratic interaction cost $(2n^2 \cdot h + n)$ due to its cross term-alignment. In contrast, our hybrid model outperforms all other two-tower rankers (Independent dense, Late-interaction, Sparse lexical) with a moderate 21% increase in encoding GFLOPs

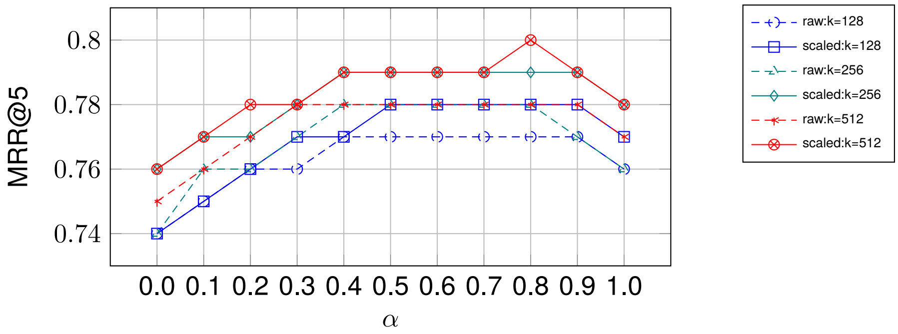


Figure 4: MRR@5 with regular (dashed) and source-scaled (solid) interaction scores at different semantic and lexical matching combinations.

and has a linear interaction cost as it only sums the product of the matching query tokens. For latency measurement, we consider the mean combined time elapsed for encoding, interaction, and score-sorting per query as well as per information. Each test set query has an average of 112 information elements. The inference latency is aligned with the inference FLOPs and the latency of our model is halfway between that of the independent dense model and cross encoder. In terms of offline storage required for each evidence representation, the hybrid approach demands space for a dense vector ($O(h)$) as well as key-value (key: vocabulary token index) pairs corresponding to non-zero elements ($O(k)$) of sparse lexical representation. This memory requirement ($O(h + k)$) is much smaller than that of the late-interaction method ($O(n \cdot h)$) as the latter stores all the token encodings.

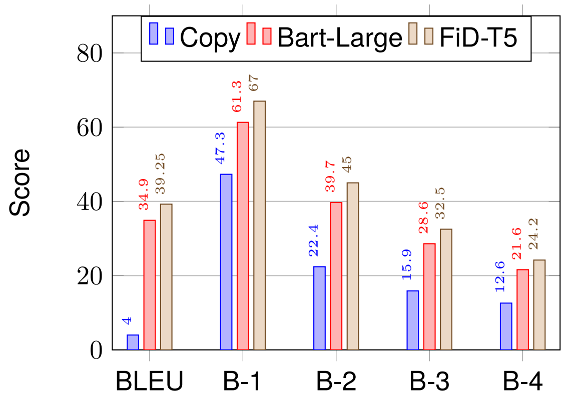


Figure 5: Results of answer generation.

6.2. Ablation Study

A comparison of evaluation results between our model and models using a subset of components reveals the contribution of additional components in our model. While results in all metrics show a similar trend, we use the standard MRR@5 for our ablation study. To begin with, our hybrid model is of identical architecture as in the sparse lexical model and differs from independent dense models only by the MLM layers. However, our model outperforms the sparse lexical model by 10.95% and the dense retriever by 0.7%-2.7% (for $128 \leq k \leq 512$) in MRR@5 (Table 3). This indicates the benefit of joint learning instead of maximizing only lexical or semantic matching independently.

While our semantic matching captures the underlying summarized meaning, explicit token matching compliments it by allowing us to interpret it. Figure 4 illustrates the effect of their contribution on MRR@5 by varying α in Equation 1 and changing sparsity i.e. the number of top-k tokens (represented by color) considered in sparse representation (see §4.2). The differences in area under curves indicate that a higher number of token considerations results in better ranking. On the other hand, the sub-optimal results with lexical-only ($\alpha = 0$) or semantic-only ($\alpha = 1$) matching and the consistently superior results with α in the range of $0.5 - 0.8$ further support our hybrid approach.

Furthermore, our analysis reveals that the distribution of scores across heterogeneous sources differs significantly and favors sources with high mean scores even if they obtain relatively low hit rates at top-5 positions. To counter this, we utilize the source-specific hit-rate@5 obtained from the regular ranking as prior confidence in those sources and multiply it with the combination of normalized scores obtained from Equation 1. The resulting ranking score, as shown by the solid lines in Figure 4, outperforms that of the regular ranking in dashed lines by 1%-3% across α and k values.

6.3. Generation Quality

Figure 5 illustrates the answer quality generated by three approaches: 1. simply copying the top evidence as an answer, 2. Bart-Large (406M params) and 3. Fusion-in-Decoder with T5 (FiD-T5: 220M params). We utilize the results of the copy-based

Id	Source	Text	Expansion
#1 Desc	**Query**	how fast does the car go?	speed, time
	Evidence	maximum speed: 12 mph	fast, time, go
	Answer	the maximum speed is 12 mph.	
#2 Review	**Query**	how long do they stay lit?	time, last, light
	Evidence	the glow only lasts for on average of 30 minutes.	time, long, light
	Answer	they last under an hour.	
#3 CQA	**Query**	how do you hook it up to a television?	tv, power, plug
	Evidence	you just plug it directly to your tv.	power
	Answer	plug it into your television.	
#4 attribute	**Query**	how tall is the castle??	height, size
	Evidence	item dimensions width: $15.75''$, length: $30.5''$, height:$23''$	tall, size
	Answer	the castle is 23 inches tall.	

Table 5: Sample evidence prediction and answer generation.

approach and Bart-Large model from (Shen et al., 2022). Despite having a smaller number of parameters, the responses generated by FiD-T5 result in a higher BLEU score than that of other approaches. Examples of sample answer generation can be found in Table 5.

6.4. Interpretability Analysis - Examples and Discussion

A desired quality of a model is to have a simple and human-understandable mechanism to explain its decision-making process. Expanded tokens selected by our model's lexical representations can be interpreted as visualizable faces of underlying thoughts captured in jointly learned semantic representation. Further, the dot product of a matched token importance can be considered as its alignment strength. Table 5 illustrates this idea by highlighting matching tokens of query and predicted evidence. The importance of a token is depicted by its highlighting intensity. The examples demonstrate that our model can match relevant tokens through expansion even if they are not present in the original text. More interestingly, the matching expansion (e.g. *time* in ex#1, *light* in ex#2, *power* in ex#3 and *size* in ex#4) reveals the shared implicit impression that connects the query and the evidence.

There are a few shortcomings to the model which we leave as future work. First, it treats different forms (e.g. *lasts* and *lasting*) of a root token (e.g. *last*) as separate tokens causing redundant expansion. It can be avoided by merging them with their normalized value. Second, although we reduce the memory footprint of sparse lexical representation by keeping only token index-value pairs, further analysis is required to check its compatibility and efficiency with an indexer such as FAISS (Johnson et al., 2019). Without using such an indexer, despite having lower FLOPs, the ranking latency may rise dramatically if we compute token interaction in a loop. Furthermore, differential studies on domain-specific signals such as rate of product sale, count of repeating questions, customers' feedback, and engagement can be measured to quantify the effectiveness of the generator as well as the retriever.

7. Conclusion

The study presents a hybrid information ranker that ranks information for a query by comparing their jointly learned dense semantic representations and sparse lexical representations. Our evaluation found that our approach outperformed widely popular sparse or dense retrievers while incurring only a linear cost for both computation and offline storage. Also, our expansion-enhanced lexical matching demonstrates signs of interpretability. In the future, we plan to extend the framework to an end-to-end system with extensive evaluation using a larger dataset.

8. Bibliographical References

Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. 2020. Sparterm: Learning term-based sparse representation for fast text retrieval. *arXiv preprint arXiv:2010.00768*.

Elias Bassani. 2022. ranx: A blazing-fast python library for ranking evaluation and comparison. In *ECIR (2)*, volume 13186 of *Lecture Notes in Computer Science*, pages 259–264. Springer.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: an-*

alyzing text with the natural language toolkit. " O'Reilly Media, Inc.".

Dorian Brown. 2020. Rank-BM25: A Collection of BM25 Algorithms in Python.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. Splade: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 2288–2292, New York, NY, USA. Association for Computing Machinery.

Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021. Complement lexical retrieval model with semantic residual embeddings. In *Advances in Information Retrieval*, pages 146–160, Cham. Springer International Publishing.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 39–48, New York, NY, USA. Association for Computing Machinery.

Weize Kong, Jeffrey M. Dudek, Cheng Li, Mingyang Zhang, and Michael Bendersky. 2023. Sparseembed: Learning sparse lexical representations with contextual embeddings for retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 2399–2403, New York, NY, USA. Association for Computing Machinery.

Bing Liu and Ian Lane. 2016. Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling. In *Proc. Interspeech 2016*, pages 685–689.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345.

Ji Ma, Ivan Korotkov, Yinfei Yang, Keith B. Hall, and Ryan T. McDonald. 2021. Zero-shot neural passage retrieval via domain-targeted synthetic question generation. In *Conference of the European Chapter of the Association for Computational Linguistics*.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *ArXiv*, abs/1901.04085.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Stephen E. Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*, pages 232–241. ACM/Springer.

Xiaoyu Shen, Gianni Barlacchi, Marco Del Tredici, Weiwei Cheng, Bill Byrne, and Adrià Gispert. 2022. Product answer generation from heterogeneous sources: A new benchmark and best practices. In *Proceedings of the Fifth Workshop on e-Commerce and NLP (ECNLP 5)*, pages 99–110, Dublin, Ireland. Association for Computational Linguistics.

Karen Sparck Jones. 1988. *A Statistical Interpretation of Term Specificity and Its Application in Retrieval*, page 132–142. Taylor Graham Publishing, GBR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in NeurIPS*, pages 5998–6008.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

xiaoju ye. 2023. calflops: a flops and params calculate tool for neural networks in pytorch framework.

Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. 2018. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 497–506, New York, NY, USA. Association for Computing Machinery.

Hallucination Detection in LLM-enriched Product Listings

Ling Jiang, Keer Jiang, Xiaoyu Chu, Saaransh Gulati, Pulkit Garg
Amazon
Sunnyvale, CA, USA
{jiangll, kjiang, xiaoychu, saaransg, pulkitg}@amazon.com

Abstract

E-commerce faces persistent challenges with data quality issue of product listings. Recent advances in Large Language Models (LLMs) offer a promising avenue for automated product listing enrichment. However, LLMs are prone to hallucinations, which we define as the generation of content that is unfaithful to the source input. This poses significant risks in customer-facing applications. Hallucination detection is particularly challenging in the vast e-commerce domain, where billions of products are sold. In this paper, we propose a two-phase approach for detecting hallucinations in LLM-enriched product listings. The first phase prioritizes recall through cost-effective unsupervised techniques. The second phase maximizes precision by leveraging LLMs to validate candidate hallucinations detected in phase one. The first phase significantly reduces the inference space and enables the resource-intensive methods in the second phase to scale effectively. Experiments on two real-world datasets demonstrated that our approach achieved satisfactory recall on unstructured product attributes with suboptimal precision, primarily due to the inherent ambiguity of unstructured attributes and the presence of common sense reasoning. This highlights the necessity for a refined approach to distinguish between common sense and hallucination. On structured attributes with clearly defined hallucinations, our approach effectively detected hallucinations with precision and recall surpassing targeted level.

Keywords: large language model, hallucination, e-commerce, product listing enrichment

1. Introduction

In e-commerce, the significance of comprehensive product listings cannot be overstated, as it plays a pivotal role in facilitating informed purchase decisions by customers. However, real-world product listings often suffer from diverse quality issues, such as data incompleteness, information redundancy, and misinformation. These challenges impact customers' shopping experiences. Therefore, product listing enrichment is a critical task in e-commerce to generate compelling product listings.

The product listing enrichment task aims to create concise yet informative product listings given the source product data. Figure 1 illustrates this process. In the initial listing, several essential product attribute values are missing, and the product name contains redundant details. After enrichment, the product name is more succinct and user-friendly, and a correct value was populated for the attribute *Material*. Such enriched listings can help customer reduce cognitive load during product evaluation and improve sales conversion (Purnomo, 2023).

Product listing enrichment involves generating structured data and free text from the source input, which is an essential task in various natural language generation applications, such as summarization (Nenkova et al., 2011) and data-to-text generation (Wiseman et al., 2017). Traditional template-based approaches (Gatt and Reiter, 2009; Reiter et al., 2005) rely on manually crafted rules and lack scalability. Transformers and language models (Vaswani et al., 2017; Devlin et al., 2018; Yang et al., 2019; Liu et al., 2019; Radford et al., 2019) have demonstrated exceptional capabilities of generating fluent text. Recently, Large Language Models (LLMs) (Brown et al., 2020; Ouyang et al., 2022; Achiam et al., 2023; Touvron et al., 2023; Chowdhery et al., 2023) have pushed the boundaries of natural language generation to new heights (Bubeck et al., 2023). The remarkable language comprehension of LLMs offers an opportunity for automating the generation and enhancement of product listings (Westmoreland, 2023).

However, a concerning drawback of LLMs is its tendency to hallucinate, when LLMs generate texts that appear fluent and coherent but are nonfactual or unsupported by the input data (Varshney et al.,

Source product listing	
Name	Leather Couch –Leather Sofa with Tufted Back – Leather Couch with Feather Down Topper On Seating Surfaces – Italian Leather
Material	NA
Color	NA
Size	NA

LLM-enriched product listing	
Name	Italian Leather Sofa with Tufted Back and Feather Down Topper
Material	Leather
Color	Black
Size	88.5 Inch

Figure 1: Example of product listing enrichment and hallucination.

The Seventh Workshop on e-Commerce and NLP (ECNLP 7), pages 29–39

2023). In Figure 1, the LLM successfully improved the product *Name* and accurately generated value for *Material*. Conversely, the reliability of the generated values for *Color* and *Size* is questionable when examining the source data alone without additional information. Such hallucinations pose risks by potentially leading to negative user experiences and, more critically, misinformation-induced purchases. In safety-critical scenarios, such as the failure to generate warnings on toy choking hazards, hallucinations may result in legal consequences.

In this work, we address the hallucination problem in LLM-enriched product listings. We define hallucination as the generation of text that is unfaithful to the provided source input (Ji et al., 2023). Some works also consider factual inaccuracies in their definition (Varshney et al., 2023; Zhang et al., 2023). The primary function of product listings is to communicate descriptive details about the items. For example, the value of *Material* is product-specific, inherently contingent upon the source product information. Here, factual accuracy depends on faithfully reflecting the source input for each unique product, assuming the provided product information accurately describes the products. The source input data serves as the definitive reference for truth in this context.

Previous studies have employed the hidden layer activations or logit values of LLMs to detect hallucinated content (Azaria and Mitchell, 2023; Varshney et al., 2023). Yet, such methods require access to the internal states of LLMs, which is typically not available in state-of-the-art black-box LLMs (e.g. ChatGPT). Some have integrated external knowledge bases with LLMs (Guo et al., 2022; Martino et al., 2023; Peng et al., 2023a; Lee et al., 2022), but this introduces additional cost and complexities. Alternatively, LLMs have been used to autonomously verify their outputs (Wang et al., 2023; Manakul et al., 2023) or have been fine-tuned for specific tasks (Cao et al., 2021; Yu et al., 2023), though LLM-based methods can be costly without in-house models. In-house LLMs, while circumventing some expenses, still demand extensive training data and substantial resources for model development.

While akin to detecting hallucinations in summarization (Cao et al., 2021) or data-to-text generation (Tian et al., 2019), our task involves unique challenges due to the mixture of free text and structured data in both source input and generated text. The former is often noisy and poor-formatted, particularly when sourced from third-party sellers in e-commerce. Furthermore, LLM-generated values may be embedded in various product attribute fields, necessitating a comprehensive examination of all product information for potential evidence. Given that an e-commerce website can sell billions of products, addressing hallucination detection at such a scale requires careful consideration of both performance and cost factors.

In this paper, we present an approach for hallucination detection in LLM-enriched product listings without accessing internal LLM states or relying on external data sources. We propose to detect hallucinations in a two-phase fashion, prioritizing recall in the initial phase and enhancing precision in the subsequent phase. The motivation stems from the high cost associated with LLM-only approaches in massive-scale e-commerce applications. In the initial phase, which we call Lexical and Semantic Screening (LSS), we apply cost-effective unsupervised techniques to detect a broad range of hallucinations. While these methods are efficient, their accuracy may be compromised due to limitations in text comprehension. The second phase, LLM validation, utilizes LLMs to confirm potential hallucinations detected in the first phase. Leveraging the robust language understanding capabilities of LLMs, we optimize precision in the second phase. The initial LSS phase significantly reduces the inference space, allowing the more resource-intensive LLM validation to scale effectively. Experiments on two real-world e-commerce datasets demonstrate the effectiveness of our proposed approach. In addition, we discovered that our approach performs better on structured attributes with concise, deterministic values, as opposed to unstructured attributes presented in long-form free text. We identified directions for future work through analysis of the experimental results.

2. Related Work

2.1. Definition of Hallucination

Varshney et al. (2023) defined hallucination as the generation of text or responses that seem syntactically sound, fluent, and natural but are factually incorrect, nonsensical, or unfaithful to the provided source input. This definition aligns with the taxonomy proposed by Zhang et al. (2023). However, studies on hallucination in various natural language generation tasks (Tian et al., 2019; Maynez et al., 2020; Weng et al., 2023) may emphasize distinct aspects of the phenomenon. Consequently, the definition of hallucination may exhibit variability across tasks. In the product listing enrichment task, product listings primarily convey descriptive information about the products, and factual accuracy is contingent on faithfully representing the source input for each individual product. In this sense, our perception of hallucination aligns more closely with Ji et al. (2023)'s definition, which refers to the generation of text that is nonsensical, or unfaithful to the provided source input. It is further categorized into intrinsic and extrinsic hallucination. Intrinsic hallucination involves output contradicting the source, while extrinsic hallucination refers to output unverifiable against the source. We are concerned with both types, encompassing content that either contradicts or lacks support in the source input, emphasizing the unfaithfulness aspect. After we identify unfaithful hallucinations, we need to further determine the factual accuracy to serve the end business goal. However, we focus this work on the faithfulness aspect and leave the factual part for future work.

2.2. Hallucination Detection

Many recent studies have been focused on mitigating hallucinations in LLMs. Depending on the accessibility of LLM models, there are white-box, grey-box and black-box approaches. A white-box method (Azaria and Mitchell, 2023) used the LLM's hidden layer activations to train a classifier that predicts the probability of a statement being true. Grey-box approaches detect the parts of the output sequence that the LLM is least confident about by examining the logit output values in the response (Varshney et al., 2023). Both white-box and grey-box approaches require access to internal states or token probabilities that may not necessarily be available, e.g. when LLMs are accessed through limited API calls. Black-box approaches (Manakul et al., 2023) are suitable for a wider range of applications when only LLMs responses are available.

Approaches for mitigating hallucination can also be grouped into zero-resource and external knowledge based approaches depending on if an external knowledge base is involved. External knowledge based approaches try to mitigate hallucination through information augmentation from external knowledge sources (Guo et al., 2022; Moiseev et al., 2022; Martino et al., 2023; Peng et al., 2023a). However, knowledge augmented approaches usually come with the cost of additional complexity and resource overhead (Lee et al., 2022).

Zero-resource approaches do not rely on external knowledge to detect hallucinated responses. One line of studies leverage unsupervised metrics scores (Celikyilmaz et al., 2020; Forbes et al., 2023) such as ROUGE (Lin, 2004), BLEU (Papineni et al., 2002), and METEOR (Banerjee and Lavie, 2005) to measure the consistency between the generated text and the source text. While these metrics offer simplicity, they frequently fall short in accurately aligning texts. This leads to sub-optimal performance when semantically relevant text diverges from the reference's surface form. Additionally, these metrics struggle to capture distant dependencies and tend to penalize changes in semantic ordering. BERTScore (Zhang et al., 2019) leverages the pre-trained contextual embeddings from BERT (Devlin et al., 2018) and matches words in candidate and reference sentences by cosine similarity. BERTScore has been shown to correlate well with human judgments. Another approach is converting hallucination detection to classification problem (Chen et al., 2023). Recent studies showed that LLMs can be good evaluators themselves (Wang et al., 2023) and many studies leverage LLMs to detect hallucinations (Mündler et al., 2023; Manakul et al., 2023; Weng et al., 2023; Fu et al., 2023). Another way to mitigate hallucination is fine-tuning LLMs on task-specific data (Cao et al., 2021; Yu et al., 2023). Some of these approaches can complement each other. For example, Guan et al. (2023) combined LLM verification, instruction tuning and retrieval augmentation to verify facts for LLMs outputs. LLM-based approaches can entail significant expenses when utilizing commercially available LLMs, especially on large-scale e-commerce applications. Alternatively, the development of proprietary LLMs within an organization introduces a different set of costs.

In this work, we present a method for detecting hallucinations in LLMs-enriched product listings. Our approach utilizes zero-resource black-box hallucination detection techniques, eliminating the need for external knowledge base or access to LLMs' internal states. This independence allows our system to be agnostic of upstream LLMs and be generalizable to a wider range of LLMs applications. Moreover, our method enhances scalability compared to LLM-only approaches by markedly reducing the inference space prior to LLM validation.

3. Methodology

We propose a two-phase approach for hallucination detection (Figure 2), emphasizing recall in the initial phase and enhancing precision in the subsequent phase. In the first phase, termed Lexical and Semantic Screening (LSS), cost-effective unsupervised techniques are applied to detect a broad spectrum of hallucinations. Although these methods are efficient, their performance may be compromised by text comprehension limitations. In the second phase, LLM validation, we utilize LLMs to validate the candidate hallucinations identified in the first phase. We optimize precision in the second phase by leveraging the robust language understanding capabilities of LLMs. Given the critical need for scalability in hallucination detection, particularly in the context of e-commerce with a vast product inventory, the initial LSS phase significantly reduces the inference space. This reduction enables the more resource-intensive approach to scale effectively in the second phase, addressing the challenge of processing billions of products for product listing enrichment in e-commerce. Figure 3 depicts LLM-generated attribute values, *Brand: Lilly Pulitzer* and *Size: 12inchx10inch*, for the given source product listing. Using LSS, we confirmed the legitimacy of *Brand: Lilly Pulitzer* by cross-referencing it with information in the source product name. Subsequently, we evaluated the *Size: 12inchx10inch* attribute and determined it to be hallucinated content employing LLM validation.

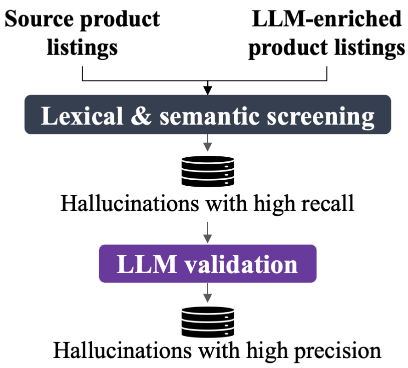


Figure 2: Two-phase hallucination detection.

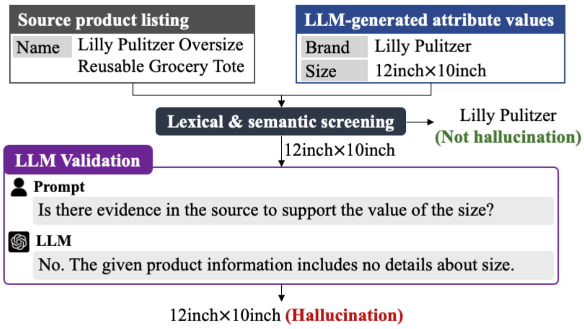


Figure 3: Example of hallucination detection.

3.1. Lexical and Semantic Screening (LSS)

In the initial phase, we employ unsupervised methods to flag all potential hallucinations by detecting information lacking supporting evidence in the source input. This support is traced through exact keywords or similar content, examined at either the token or the entire generated value level. We investigated techniques for locating supporting evidence from the source input:

3.1.1. Token-level LSS

Rebuffel et al. (2022) advocated addressing hallucinations at the word level rather than the instance level. They employed word-level alignment between candidate and inference text to control hallucinations. Their experiments demonstrated that word-level signals improved the fluency, factual accuracy, and relevance of LLM outputs. In our study, we similarly employ token-level alignment between the source input and LLM-enriched values to detect hallucinations.

Exact match. A direct method involves examining the presence of generated content in the source input. In Figure 2, the explicit mention of *Lilly Pulitzer* in the source input rules out hallucination. This method, denoted as T_{exact}, exhibits high recall but low precision in identifying hallucinations.

Exact matching often results in a large number false positive hallucinations, as LLMs may produce semantically similar but distinct words in enriched product listings. Fuzzy matching provides more flexibility, allowing LLMs to generate product information with enhanced fluency and coherence, leveraging advanced vocabulary. We assessed three fuzzy matching techniques:

Edit-distance. Token-level edit-distance (Levenshtein et al., 1966) between the source text and hallucinated text was used in generating synthetic data for hallucination detection and it was found that this approach provided sufficiently high quality training data in practice (Zhou et al., 2020). In our work, we adopt a similar approach by calculating the token-level edit distance between each token in the generated and source texts to pinpoint potential supporting evidence within the source data. This method is denoted as T_{edit}.

N-gram overlap metrics. N-gram matching metrics are commonly used for evaluating text generation by counting the number of n-grams that occur in the reference and candidate text. ROUGE (Lin, 2004) is often used for summarization evaluation, while BLEU is the most widely used metric in machine translation (Papineni et al., 2002). METEOR (Banerjee and Lavie, 2005) introduces flexibility by permitting a transition from strict unigram matching to encompassing word stems, synonyms, and paraphrases. These metrics provide a way to find the evidence for supporting the LLM-generated content from the source input. Therefore, we utilize three metrics: T_{rouge}, T_{bleu}, and T_{meteor}.

Embedding similarity. Token embeddings capture nuanced semantic and syntactic word relationships (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2016), and allow for a soft measure of similarity instead of strict string matching between the generated and source text. In this work, we experimented with three embedding models provided in Gensim (Rehurek and Sojka, 2011): *word2vec-google-news-300* (Mikolov et al., 2013) ($T_{word2vec}$), *glove-wiki-gigaword-300* (Pennington et al., 2014) (T_{glove}), and *fasttext-wiki-news-subwords-300* (Bojanowski et al., 2016) ($T_{fasttext}$).

3.1.2. Value-level LSS

Instead of checking evidence at the token level, we can evaluate the semantic similarity between the generated text and the source input as a whole. Significant deviations from the source text in the generated content can signal hallucination.

Sentence embedding. Sentence transformers convert sentences into semantically meaningful embeddings that can be compared using cosine-similarity (Reimers and Gurevych, 2019). In this work, we used four sentence-transformers (Reimers and Gurevych, 2019) models from HuggingFace (Wolf et al., 2019): *all-MiniLM-L6-v2*, *all-mpnet-base-v2*, *gtr-t5-large* (Ni et al., 2021), and *multi-qa-mpnet-base-dot-v1*, denoted as V_{miniLM}, V_{mpnet}, V_{gtr}, and V_{qa} respectively.

BERTScore. BERTScore has been shown to correlate well with human judgments for evaluating natural language generation tasks (Zhang et al., 2019). BERTScore calculates a similarity score between the candidate and reference text by aggregating the cosine similarities of their token embeddings. Unlike traditional metrics such as ROUGE, BLEU and METEOR that rely on string matching or heuristics, BERTScore uses contextualized token embeddings. It demonstrates a stronger capability of accommodating instances where semantically correct phrases deviate from the surface form of the reference. We abbreviate this approach as V_{bert}.

ALIGNSCORE. ALIGNSCORE (Zha et al., 2023) evaluates the factual consistency of generated text against a model input. It is applicable to various factual inconsistency scenarios, as it employs a unified training framework of the alignment function by integrating diverse data sources from seven well-established tasks. We denote this as V_{align}.

3.2. LLM Validation

Recent studies have explored the use of LLMs for evaluating their own generated text (Varshney et al., 2023), showing promising results. However, LLM validation typically incurs a significant expense due to associated API fees, presenting a challenge that impedes the scalability of LLM validation for e-commerce product listing enrichment. This challenge underscores the need to initially filter out a substantial portion of non-hallucinated content in the LSS step, which allows the subsequent LLM validation step to focus on a more manageable number of candidates. In this work, we used Claude 2 (Anthropic, 2023) from Anthropic for LLM validation.

Zero-shot. LLMs have shown great potential in evaluating the factual consistency between a document and its summary in the zero-shot setting (Luo et al., 2023). Thus, we directly prompt an LLM to verify the hallucinations detected in the LSS step.

Chain-of-thought (CoT). Chain-of-thought (CoT) prompting significantly enhances LLMs' complex reasoning abilities (Wei et al., 2022). CoT prompting, involving the presentation of intermediate reasoning steps, prove effective in both zero-shot (Kojima et al., 2022) or in-context learning (Wei et al., 2022) settings. Kojima et al. showed that LLMs demonstrated decent zero-shot reasoning capability by instructing them to think step by step (Kojima et al., 2022). In this work, we adopt a similar approach, instructing LLMs to provide step-by-step reasoning and identify supporting evidence when available. We only use the final decision from LLM as the prediction. Nonetheless, prompting the LLM to seek evidence initiates an underlying reasoning process, which can potentially improve the overall performance (Kojima et al., 2022).

In-context learning. LLMs demonstrate impressive ability to do in-context learning (Brown et al., 2020). They can generalize to unseen data by leveraging a limited set of training examples in the prompt, without explicit pre-training for the specific task (Xie et al., 2021). We asked domain experts to select examples of product listings with and without hallucinations and include them in prompts. By augmenting the context with these selected examples, it is anticipated that the LLM will discern the underlying pattern present in the demonstrations, thus enabling accurate predictions.

Many studies have investigated instruction-tuning for LLMs to enhance alignment with specific tasks (Ouyang et al., 2022; Peng et al., 2023b). In contrast to in-context learning, wherein examples are presented during inference without updating the LLMs' parameters, instruction tuning involves utilizing a set of examples to adjust the parameters during training. However, no demonstrations are employed during inference in instruction tuning (Duan et al., 2023). Although instruction tuning has shown promising results, it requires human-annotated prompts and feedback on a specific task. In our work, we do not experiment with instruction-tuning but consider it a potential future direction.

4. Experiments

In this section, we first introduce the dataset employed for evaluating hallucination detection in LLM-enriched product listings. We then compare the performance of different approaches, demonstrating the effectiveness of our proposed approach.

Finally, we discuss our findings, providing insights that guide our future research.

4.1. Dataset

We conducted experiments on two sets of human-annotated LLM-enriched product listings as described in Table 1. D_S exclusively contains structured product attributes, whereas D_U comprises only unstructured attributes. Structured attributes typically encompass product features characterized by enumerated, categorical, numerical, or keyword values, whereas unstructured attributes comprise long-form free-text values. In Table 1, we provide a summary of word counts for both structured and unstructured attributes. Given that the majority of product attributes are structured, we consolidate these into a single total count rather than listing each attribute individually. Conversely, we detail the three most prevalent unstructured attributes. Structured attributes normally contain 1-2 words, while unstructured attributes can include more than one hundred words. Additionally, attributes in D_S exhibits deterministic values, enabling annotators to identify hallucinations through a direct examination of the source input. In contrast, D_U introduces greater ambiguity. For instance, D_S primarily includes attributes such as *Color*, making it straightforward to assess the faithfulness of generated values to the source. On the contrary, D_U contains descriptive attributes, where values are less deterministic. Annotators' perceptions play a critical role in human judgments, contributing to increased uncertainty in hallucination detection.

Dataset	Attribute type	#Listings	#Entries
D_S	Structured	200	2,765
D_U	Unstructured	4,042	12,126

Table 1: Dataset description.

Type	Attribute	#Words		
		$Avg.$	$50p$	$75p$
Structured	-	2	1	2
Unstructured	U-Attribute 1	17	16	21
	U-Attribute 2	91	81	102
	U-Attribute 3	138	130	160

Table 2: Number of words by attribute type.

The datasets include original product listing alongside LLM-enriched values for one or multiple attributes in that listing. Our approach focuses on detecting hallucination at the attribute level. For instance, Figure 3 displays a listing with 2 LLM-enriched attributes: *Brand* and *Size*. We make independent decisions for each attribute. Domain experts audited the dataset to pinpoint hallucinations in LLM-generated values, and we use these human labels as the gold standard to evaluate our proposed approaches.

4.2. Experimental results

We utilize precision, recall, and F1 score to assess model performance. In e-commerce, distributing hallucinated product listings may lead to negative user experiences or legal issues in critical scenarios. Nevertheless, rejecting LLM-enriched product listings based on mistakenly identified hallucinations carries substantial costs. The precision-recall trade-off allows optimizing the balance between the consequences of false positive and false negative predictions in practice.

4.2.1. LSS

Table 3 and 4 detail the performance of various LSS models on D_S and D_U. In the results, P, R, and $F1$ represent precision, recall, and F1 score, respectively. Precision and recall targets, established by domain experts, are denoted as p and r. The target $f1$ is calculated with p and r. We present the performance of various approaches compared against the targets. As we optimize recall in the LSS phase to maximize the coverage of hallucinations, we choose the optimal model from each method with $recall >= min(recall_{max}, r)$.

LSS Model	P/p	R/r	$F1/f1$	$\%Inf$
T_{exact}	1.05	1.09	**1.07**	7.4
T_{edit}	1.05	1.09	**1.07**	7.4
T_{rouge}	0.88	1.06	0.95	9.0
T_{bleu}	1.05	1.07	1.06	6.8
T_{meteor}	0.68	1.06	0.82	11.6
$T_{word2vec}$	1.09	1.04	1.06	6.4
T_{glove}	1.09	1.03	1.06	6.4
$T_{fasttext}$	1.09	1.04	**1.07**	6.4
V_{miniLM}	0.85	1.06	0.94	9.8
V_{mpnet}	0.81	1.07	0.91	116
V_{gtr}	0.84	1.06	0.93	9.5
V_{qa}	0.81	1.06	0.91	10.6
V_{bert}	0.40	1.08	0.57	48.5
V_{align}	0.80	1.07	0.91	10.6

Table 3: Performances of LSS models on D_S.

The results indicate that LSS models exhibited promising abilities in detecting hallucinations within D_S. T_{exact}, T_{edit} and $T_{fasttext}$ yielded the highest F1 score at 1.07$f1$. While other LSS models demonstrated slightly lower performance, the majority maintained recall rates above 1.05r and precision within the range of 0.8p-1.05p. In contrast, these models exhibited a significant drop in performance when applied to D_U. While most maintained over 1.05r recall, precision struggled, falling below 0.25p. In e-commerce, ensuring high recall in hallucination detection is crucial, as false negatives

LSS Model	P/p	R/r	$F1/f1$	$\%Inf$
T_{exact}	0.23	1.11	0.36	12.5
T_{edit}	0.23	1.11	0.36	12.5
T_{rouge}	0.23	1.11	0.37	13.2
T_{bleu}	0.23	1.11	0.36	12.5
T_{meteor}	0.26	1.08	**0.41**	7.2
$T_{word2vec}$	0.23	1.11	0.36	12.4
T_{glove}	0.23	1.11	0.36	12.4
$T_{fasttext}$	0.23	1.11	0.36	12.4
V_{miniLM}	0.27	0.96	**0.41**	2.9
V_{mpnet}	0.20	0.90	0.31	2.5
V_{gtr}	0.17	0.74	0.27	1.7
V_{qa}	0.24	1.06	0.37	6.0
V_{bert}	0.25	0.34	0.29	0.3
V_{align}	0.22	1.05	0.35	6.8

Table 4: Performances of LSS models on D_U.

directly affect customer experiences and can hurt brand reputation.

The $\%Inf$ column denotes the portion of the inference space identified by LSS as hallucination candidates, serving as input for LLM validation. LSS models effectively reduced the inference space to less than 12.5%, and some models further narrowed this down to below 10% for LLM validation.

V_{bert} is an anomalous case among the models for D_S, with a performance of only 0.4p. Our observations indicate that V_{bert} tends to give lower scores to generated values significantly shorter than the reference text, causing an increase in false positives. Conversely, it assigns high similarity scores to unstructured attribute values compared to the source text, which substantially reduces recall.

The primary difference between the datasets is their attribute types. The unstructured attributes in D_U contain substantial free-text content, differing significantly from the concise nature of structured attribute values in D_S. Detecting hallucination from unstructured attribute values poses a greater challenge compared to structured ones. Also, the difference between token-level and value-level approaches is larger on structured attributes than that on unstructured attributes. This indicates that word-to-word comparison approaches are more suitable when the LLM-generated values are a bag of keywords, rather than coherent paragraphs.

Combining LSS models improves performance on D_S through an AND operation on their predictions. We explored every possible pairing of two models, and Table 5 displays the top three combined LSS models for D_S, demonstrating notable enhancements. Intuitively, the combined models generated enhance precision but decreased recall. All the optimal combined models consist of a token-level and a text-level model. ALIGNSCORE significantly contributed to the top combined models D_S. On the contrary, combined LSS models demonstrated inferior performance compared to individual models on dataset D_U, as evidenced in Table 5. Notably, this combination led to a significant reduction in recall without improving precision, resulting in a decreased F1 score.

Dataset	Combined model	P/p	R/r	$F1/f1$
D_S	T_{exact}+V_{align}	1.12	1.06	1.09
	T_{edit}+V_{align}	1.12	1.06	1.09
	T_{bleu}+V_{align}	1.12	1.05	1.09
D_U	T_{bleu}+V_{bert}	0.23	0.31	0.26
	T_{edit}+V_{bert}	0.22	0.31	0.26
	T_{exact}+V_{bert}	0.22	0.31	0.26

Table 5: Combination of LSS models.

4.2.2. LLM validation

Table 6 presents the F1 scores of applying LLM validation to hallucination candidates identified by LSS models. Overall, LLM validation enhanced the performance on D_S, while yielding marginal improvement on D_U. On D_S, solely employing zero-shot LLM validation improved the performance for each LSS model. The CoT approach generally achieved higher F1 scores, with the exceptions of T_{bleu}, $T_{fasttext}$ and V_{bert}, where the zero-shot approach outperformed. For D_U, LLM validation with in-context examples consistently outperformed zero-shot and CoT. Optimal performance was achieved by combining LSS models with subsequent LLM validation for D_S, all models demonstrated comparable F1 scores for D_U post-LLM validation. Selection of models can be tailored based on specific business requirements for precision and recall.

It is known that LLM in-context learning faces a robustness challenge (Liu et al., 2021), with outcomes highly depend on the chosen in-context examples. We observed this dependence in our experiments. We asked domain experts to select examples of product listings with and without hallucinations. The selected examples aim to represent different situations where hallucinations may occur. Despite efforts to cover various scenarios, it remains challenging to encompass all possibilities within a limited set of examples. Our observation indicates that the LLM model tends to replicate the behavior of the provided examples during validation response generation. Consequently, it predominantly identifies semantically-close samples as hallucinations. A future direction would be strategically select examples based on their similarity to the query instance.

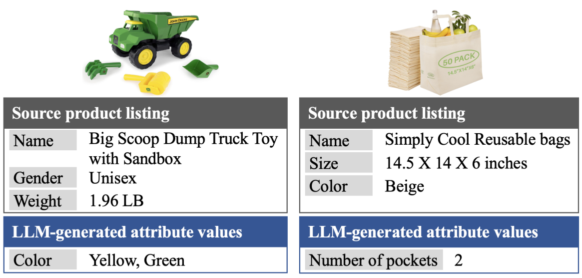
4.3. Discussions

Next, we discuss some key findings during the experiments and talk about a few open questions not covered by this work. This sheds lights on directions for future work.

Model	D_S			D_U		
	Z	C	I	Z	C	I
T_{exact}	1.08	1.09	1.08	0.37	0.36	0.37
T_{edit}	1.08	1.09	1.08	0.37	0.36	0.37
T_{rouge}	1.01	1.05	1.00	0.36	0.36	0.37
T_{bleu}	1.08	1.07	1.05	0.36	0.36	0.37
T_{meteor}	0.97	0.99	0.99	0.36	0.35	0.39
$T_{word2vec}$	1.07	1.07	1.01	0.37	0.37	0.37
T_{glove}	1.07	1.07	1.01	0.37	0.37	0.37
$T_{fasttext}$	1.08	1.07	1.03	0.37	0.37	0.37
V_{miniLM}	1.01	1.01	1.02	0.37	0.37	0.37
V_{mpnet}	1.01	1.04	1.00	0.37	0.36	0.37
V_{gtr}	1.01	1.04	1.00	0.37	0.36	0.37
V_{qa}	1.00	1.04	1.00	0.37	0.36	0.37
V_{bert}	1.02	1.00	1.00	0.32	0.33	0.34
V_{align}	1.07	1.04	1.01	0.37	0.36	0.36
T_{exact}+V_{align}	1.10	1.07	1.04	-	-	-

Table 6: LLM validation performance. Z denotes zero-shot, C denotes CoT, and I denotes in-context learning.

Factual hallucination. This study focuses on identifying hallucinated product information that lack support from the source input. However, not all detected hallucinations are necessarily incorrect; some may align with factual information (Cao et al., 2021). As illustrated in Figure 4(a), the LLM-generated *Color* value lacks support in the source input, but aligns with the product image. It is worth noting that the product image was not part of the source input but included here for illustrative purposes. Conversely, the generated value of *Number of pocket* for the tote bag in Figure 4(b) is both unsupported and non-factual. In this study, we aim to identify hallucinated content based on the given source product listings. Distinguishing between factual and non-factual hallucinations could facilitate taking follow-up actions on the detected hallucinations. However, verifying the correctness of hallucinations necessitates external knowledge sources, like supplementary product details or images. We leave this for future work.



(a) factual hallucination (b) non-factual hallucination

Figure 4: Factual and non-factual hallucination.

Common sense. We observed that annotators relied on common sense to evaluate hallucination in some cases. In Figure 5, LLM suggested *Walking* as the *Recommended use* for the sandal. Human annotators considered this non-hallucinatory, given the common understanding that sandals are suitable for walking rather than activities like running or jumping. However, our method flagged this as hallucination because there was no corresponding information in the source input supporting *Walking*, and the LLM validation step failed to capture it. Unlike the factual hallucination in Figure 4(a), which is clearly unfaithful to the source input even if factual, determining hallucination becomes challenging when common sense is a factor.

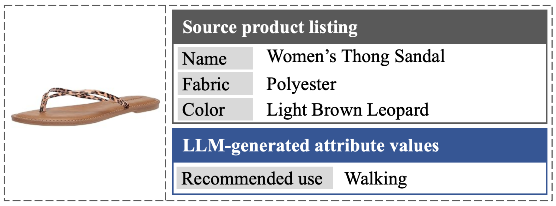


Figure 5: Common sense.

Common sense is a subjective and evolving concept, varying among individuals based on their experiences and knowledge. For instance, what is common knowledge today, such as Apple being the manufacturer of the iPhone, may not have been widely known several years ago. To distinguish common sense from hallucination in LLMs, we can leverage their hidden knowledge or external sources. However, a precise definition of common sense versus hallucination for different use cases is essential for effective hallucination detection.

Hallucination in LLM validation. The LLM validation phase, like other LLM applications, is prone to hallucinations. Instead of developing a new hallucination detection solution, a potential strategy to address this issue is to utilize multiple responses from one or more models. However, cost is a crucial consideration in real-world industrial applications, particularly in large-scale e-commerce settings. Another alternative is fine-tuning a task-specific LLM, but this necessitates high-quality training labels.

5. Conclusions

This paper introduces an effective approach for identifying hallucinations from LLM-enriched product listings. We proposed a two-phase approach, prioritizing recall in the initial phase and enhancing precision in the subsequent phase. Our experiments on two real-world e-commerce datasets demonstrate the efficacy of our proposed approach, with better performance observed on structured attributes compared to unstructured ones. We also highlight the challenge introduced by common sense when human annotators label the data, providing valuable insights for future work.

6. Bibliographical References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2023. *Claude 2*. https://www.anthropic.com/index/claude-2 [Accessed: 2024-01-08].

Amos Azaria and Tom Mitchell. 2023. The internal state of an llm knows when its lying. *arXiv preprint arXiv:2304.13734*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Meng Cao, Yue Dong, and Jackie Chi Kit Cheung. 2021. Hallucinated but factual! inspecting the factuality of hallucinations in abstractive summarization. *arXiv preprint arXiv:2109.09784*.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799*.

Yuyan Chen, Qiang Fu, Yichen Yuan, Zhihao Wen, Ge Fan, Dayiheng Liu, Dongmei Zhang, Zhixu Li, and Yanghua Xiao. 2023. Hallucination detection: Robustly discerning reliable answers in large language models. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 245–255.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Hanyu Duan, Yixuan Tang, Yi Yang, Ahmed Abbasi, and Kar Yan Tam. 2023. Exploring the relationship between in-context learning and instruction tuning. *arXiv preprint arXiv:2311.10367*.

Grant C. Forbes, Parth Katlana, and Zeydy Ortiz. 2023. Metric ensembles for hallucination detection.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*.

Albert Gatt and Ehud Reiter. 2009. Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European workshop on natural language generation (ENLG 2009)*, pages 90–93.

Jian Guan, Jesse Dodge, David Wadden, Minlie Huang, and Hao Peng. 2023. Language models hallucinate, but may excel at fact verification. *arXiv preprint arXiv:2310.14564*.

Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. 2022. A survey on automated fact-checking. *Transactions of the Association for Computational Linguistics*, 10:178–206.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *URL https://arxiv. org/abs/2205.11916*.

Nayeon Lee, Wei Ping, Peng Xu, Mostofa Patwary, Pascale N Fung, Mohammad Shoeybi, and Bryan Catanzaro. 2022. Factuality enhanced language models for open-ended text generation. *Advances in Neural Information Processing Systems*, 35:34586–34599.

Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Zheheng Luo, Qianqian Xie, and Sophia Ananiadou. 2023. Chatgpt as a factual inconsistency evaluator for abstractive text summarization. *arXiv preprint arXiv:2303.15621*.

Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*.

Ariana Martino, Michael Iannelli, and Coleen Truong. 2023. Knowledge injection to counter large language model (llm) hallucination. In *European Semantic Web Conference*, pages 182–185. Springer.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. *arXiv preprint arXiv:2005.00661*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Fedor Moiseev, Zhe Dong, Enrique Alfonseca, and Martin Jaggi. 2022. Skill: structured knowledge infusion for large language models. *arXiv preprint arXiv:2205.08184*.

Niels Mündler, Jingxuan He, Slobodan Jenko, and Martin Vechev. 2023. Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation. *arXiv preprint arXiv:2305.15852*.

Ani Nenkova, Kathleen McKeown, et al. 2011. Automatic summarization. *Foundations and Trends® in Information Retrieval*, 5(2–3):103–233.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y Zhao, Yi Luan, Keith B Hall, Ming-Wei Chang, et al. 2021. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023a. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023b. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Yudiyanto Joko Purnomo. 2023. Digital marketing strategy to increase sales conversion on e-commerce platforms. *Journal of Contemporary Administration and Management (ADMAN)*, 1(2):54–62.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. *OpenAI*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Clément Rebuffel, Marco Roberti, Laure Soulier, Geoffrey Scoutheeten, Rossella Cancelliere, and Patrick Gallinari. 2022. Controlling hallucinations at word level in data-to-text generation. *Data Mining and Knowledge Discovery*, pages 1–37.

Radim Rehurek and Petr Sojka. 2011. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2).

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167(1-2):137–169.

Ran Tian, Shashi Narayan, Thibault Sellam, and Ankur P Parikh. 2019. Sticking to the facts: Confident decoding for faithful data-to-text generation. *arXiv preprint arXiv:1910.08684*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. 2023. A stitch in time saves nine: Detecting and mitigating hallucinations of llms by validating low-confidence generation. *arXiv preprint arXiv:2307.03987*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jiaan Wang, Yunlong Liang, Fandong Meng, Haoxiang Shi, Zhixu Li, Jinan Xu, Jianfeng Qu, and Jie Zhou. 2023. Is chatgpt a good nlg evaluator? a preliminary study. *arXiv preprint arXiv:2303.04048*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification.

Mary Beth Westmoreland. 2023. *Amazon launches generative AI to help sellers write product descriptions*. https://www.aboutamazon.com/news/small-business/amazon-sellers-generative-ai-tool [Accessed: 2023-12-07].

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. *arXiv preprint arXiv:1707.08052*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Zhang Ze Yu, Lau Jia Jaw, Wong Qin Jiang, and Zhang Hui. 2023. Fine-tuning language models with generative adversarial feedback. *arXiv preprint arXiv:2305.06176*.

Yuheng Zha, Yichi Yang, Ruichen Li, and Zhiting Hu. 2023. Alignscore: Evaluating factual consistency with a unified alignment function. *arXiv preprint arXiv:2305.16739*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren's song in the ai ocean: A survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

Chunting Zhou, Graham Neubig, Jiatao Gu, Mona Diab, Paco Guzman, Luke Zettlemoyer, and Marjan Ghazvininejad. 2020. Detecting hallucinated content in conditional neural sequence generation. *arXiv preprint arXiv:2011.02593*.

Self-Improving Customer Review Response Generation Based on LLMs

Guy Azov*, Tatiana Pelc*, Adi Fledel Alon†, Gila Kamhi†
Intel Corporation

Abstract

Previous studies have demonstrated that proactive interaction with user reviews has a positive impact on the perception of app users and encourages them to submit revised ratings. Nevertheless, developers encounter challenges in managing a high volume of reviews, particularly in the case of popular apps with a substantial influx of daily reviews. Consequently, there is a demand for automated solutions aimed at streamlining the process of responding to user reviews. To address this, we have developed a new system for generating automatic responses by leveraging user-contributed documents with the help of retrieval-augmented generation (RAG) and advanced Large Language Models (LLMs). Our solution, named SC**R**ABLE, represents an adaptive customer review response automation that enhances itself with self-optimizing prompts and a judging mechanism based on LLMs. Additionally, we introduce an automatic scoring mechanism that mimics the role of a human evaluator to assess the quality of responses generated in customer review domains. Extensive experiments and analyses conducted on real-world datasets reveal that our method is effective in producing high-quality responses, yielding improvement of more than 8.5% compared to the baseline. Further validation through manual examination of the generated responses underscores the efficacy our proposed system.

Keywords: Review Response Generation, LLM-as-a-Judge, Prompt Optimization, Self Improving System

1. Introduction

Large language models (LLMs) have demonstrated remarkable performance in a wide range of tasks related to comprehending and generating natural language, text, and code (Devlin et al., 2019; Raffel et al., 2019; Brown et al., 2020), (Zhang et al., 2022; et al, 2022; Chung et al., 2022). The most notable advancement is that these tasks are executed using few-shot or in-context learning(Xie et al., 2022; Dong et al., 2023; Roberts et al., 2020), reducing the need for construction of traditional labeled datasets for supervised learning. Through their ability to efficiently store and apply knowledge, LLMs have shown outstanding capabilities in tasks involving information-seeking questions, where the question cannot be answered easily by the person asking it (Tunstall et al., 2022). Large Language Models (LLMs) are advanced AI systems designed to understand, generate, and manipulate human language. They are trained on vast amounts of text data, allowing them to perform a wide range of language-related tasks. In the contemporary digital age, customer reviews have become a cornerstone of consumer decision-making. Prospective buyers frequently rely on online reviews as a principal source for obtaining insights into various products and services. Empirical research indicates a robust and positive correlation between the numerical rating of a mobile application and the number of downloads it garners. Furthermore, it has been observed that users exhibit a tendency to modify their ratings following the reception of responses from developers. Consequently, the act of responding to user reviews is considered imperative in the realm of app development. However, crafting an appropriate response to an online review is a complex task that demands expertise to ensure it matches the customer's feedback in both content and tone. A response must cater to different audiences: the reviewer seeking resolution or acknowledgment, potential customers who use reviews to inform their buying choices, and search engines that use this content for search ranking purposes. The sheer volume and diversity of customer reviews across platforms like e-commerce sites, social media, and review websites present both a treasure trove of information and a daunting challenge for consumers seeking answers and for businesses. The latter often struggle with the resources and time to manage this feedback effectively, and they may not have staff skilled in crafting responses. In this paper, we introduce a scalable automatic end-to-end customer review response generation methodology based on LLMs. We aim to get high-quality responses leveraging an optimization strategy that relies on *LLM-as-a-Judge*, capable of iteratively scoring and proposing response improvements. Subsequently, these proposals are fed into a prompt generator that generates an improved prompt for each iteration in the response generation process. Our methodology, which deploys custom-tailored prompts for every customer support category, has demonstrated superior performance over the general prompt as per the research conducted by (Yuan et al., 2024).

Overall, we make the following contributions:

*Equal contribution
†Joint supervision

The Seventh Workshop on e-Commerce and NLP (ECNLP 7), pages 40–57
21 May, 2024.

- We propose (SC**R**ABLE - Self-Improving Customer Review Response Automation Based on LLMs), an LLM-based approach to automatically generate high quality responses to given user reviews. We demonstrate the power of customized prompt engineering to lead the LLM-based solutions to responses that raise customer satisfaction, engagement and delight. Furthermore, we employ automatic prompt engineering, using an LLM to improve a prompt, which is then evaluated against an objective function evaluator. We achieve an optimal review response prompt for inference via a two step method, 1) Review - Response generating LLM (calibrated by human evaluation) 2) Automatic prompt optimization using *LLM-as-a-Judge*.
- We conduct both manual and automatic evaluation on the performance of the proposed models and baselines. The experimental results indicate that our optimized prompt increased the human score of our test set response generations by more than 8.5% comapred to the generations obtained by using our initial base prompt.
- The results demonstrate that our proposed *LLM-as-a-Judge* approach achieves 3-5 times stronger correlation with human evaluation compared to (Yuan et al., 2024).

The rest of this paper is organized as follows. Section 2 surveys the related work. Section 3 introduces an overview of the proposed approach and the detailed design of the approach. Section 4 elaborates on the experimental results, including the results from the automatic LLM based evaluation and manual human evaluation. Sections 5 and 6 discuss conclude our work, summarizing the proposed future work.

2. Related Work

2.1. Customer Reviews Analysis

As noted in Pagano et al., user feedback and user involvement are crucial for modern software organizations (Pagano D, 2013). Data mining of user reviews has attracted significant research attention owing to the pivotal role reviews play in shaping consumer perceptions and decision-making regarding applications. Researchers have applied various techniques to analyze these reviews, ranging from fundamental structural features, such as review length and TF-IDF (Term Frequency-Inverse Document Frequency), which are frequently used to automatically classify user review emotions at a high level. Furthermore, more in-depth analyses have been pursued through the extraction of content features, including sentiment, topic, and keywords, often achieved through the application of machine learning methods(Guzman and Maalej, 2014; Martin et al., 2017; Gao et al., 2019; Palomba et al., 2017; Bharti and Babu, 2017). Other papers provide a unified summary of multiple customer reviews using machine learning models (Bražinskas et al., 2020; Brazinskas et al., 2022; Bhaskar et al., 2023).

2.2. Customer Reviews Response Generation

In addition to the process and analysis of the reviews, it is crucial to properly respond to the user. In addition to being informative, such response should be polite, address user's concerns, be empathic, leave a positive impression about the product, etc. It is important for developers to carefully respond to each and every customer review. Hassan et al. indicate that the chances that a user will revisit their review score are six times higher if the review gets a timely and to-the-point response from the product team (Hassan et al., 2018). However, some applications have so many users and reviews such that human responses are not always possible for all of the reviews. In recent years, efforts were made to automatically generate responses to customer reviews using machine learning techniques. Gao et al. suggest an RNN-based model named *RRGen* to encode the review with high level features such as occurrences of specific keywords, rating, review sentiment, review length and app category towards an automatic response generation (Gao et al., 2020). Zhang et al.(Zhang et al., 2023) propose a transformer (Vaswani et al., 2017) based model named *TRRGen* for automatic app review response generation. *TRRGen* fuses the features of app category and ratings and demonstrates that the fusion of app category feature and rating feature into token semantics is helpful for generating high-quality responses (competitive with human app expert responses). Gao et al. aim to address two limitations of the method they previously suggested, namely its lack of flexibility and generalization, which often leads to the generation of non-informative responses (Gao et al., 2021). Their proposed solution, named CoRe, leverages app details and responses from similar reviews. In addition, Farooq et al. train a seq2seq model with a retrieval component that merges user reviews with pertinent app descriptions and known user reviews, using specific app features to generate app-aware responses (Farooq et al., 2020). Cao et al. evaluate the performance of selected pre-trained language models against a transformer model trained from scratch in the context of automatic customer review response generation. They find that although pre-trained language models may score lower than baseline models in their experiments, they still prove effective in generating responses and show considerable robustness relative to the amount of training data used (Cao and Fard, 2022). Finally, Chen et al. propose a multi aspect attentive network to automatically attend different aspects of the review, ensuring most of the issues are being answered (Chen et al., 2022)

2.3. Response Evaluation

Assessing the quality of generated responses in the context of generative AI models involves multiple parameters such as relevance, coherence, and human-likeness. In the study by Katsiuba et al. (Katsiuba et al., 2023), an online experiment involving 502 participants was leveraged to determine the effectiveness of large language models (LLMs) in generating responses to customer feedback. The experiment's findings indicate that LLMs' responses were not only effective in achieving communicative goals but also held up well when compared to responses written by humans. One key methodology employed to evaluate the responses was the Turing test approach (Turing, 2009), which involves human evaluators to determine the human-like quality of an utterance generated by an AI.
Traditional automatic evaluation metrics such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), and METEOR (Banerjee and Lavie, 2005) often do not correlate well with human judgment due to their focus on lexical matching. Consequently, there is a pressing need for more advanced automatic evaluation techniques that better mirror human assessments. One approach is to employ semantic evaluation methods that measure the similarity between the ground truth and model-generated responses (Zhang et al., 2019; Zhao et al., 2019; Risch et al., 2021). Another emerging strategy is to utilize Large Language Models (LLMs) as evaluators to assess the quality of text and the overall performance of language language models, a practice known as *LLM-as-a-Judge* (Fu et al., 2023; Gao et al., 2023a; Chiang and Lee, 2023; Liu et al., 2023; Shen et al., 2023; Wang et al., 2023a,c; Peng et al., 2023; Gudibande et al., 2023; Zhou et al., 2023; Dettmers et al., 2023; Dubois et al., 2023; Bubeck et al., 2023; Chan et al., 2023; Yuan et al., 2021; Li et al., 2023; Fernandes et al., 2023; Bai et al., 2023; Saha et al., 2023; Kim et al., 2023a; Zheng et al., 2023; Kim et al., 2023b). While the focus has been on the automatic evaluation of responses, the integration of retrieval-augmented generation (RAG) frameworks (Lewis et al., 2020; Guu et al., 2020; Izacard et al., 2022) has become increasingly prevalent to boost LLM performance. This integration necessitates the development of an automated evaluation system tailored for the comprehensive RAG process (Es et al., 2023; Saad-Falcon et al., 2023).

2.4. LLM Self Improvement

While Large Language Models (LLMs) are adept at generating content, they may not always cater to specific use case requirements. To tackle this issue, enhancing LLMs through self-improvement techniques has become a focal point of research. Madaan et al. (Madaan et al., 2023) introduce *SELF-REFINE*, a technique for the autonomous enhancement of an LLM through cycles of feedback and refinement. Zhou et al. (Zhou et al., 2022) present the *Automatic Prompt Engineer*, a method for choosing prompts that optimize a particular score function. Furthermore, Yang et al. (Yang et al., 2023) explore the application of LLMs as optimizers with their approach, *Optimization by PROmpting*. Pryzant et al. (Pryzant et al., 2023) suggest *Prompt Optimization with Textual Gradients*, a non-parametric strategy influenced by gradient descent to fine-tune prompts according to a scoring function. Another study by Wang et al. (Wang et al., 2023b) views prompt optimization as a form of strategic planning, proposing *PromptAgent* to autonomously generate expert-level prompts. Wang et al. (Wang et al., 2022) propose *Self-Instruct*, a method for bootstrapping LLMs using instruction-response pairs that they generate themselves. Lastly, Yuan et al. (Yuan et al., 2024) investigate Self-Rewarding Language Models, which are capable of self-improvement by evaluating and training on their own outputs. These models not only employ *LLM-as-a-Judge* for self-assessment; but also use training data to create instructions that enhance the quality of the target output. Iterative methods use a single LLM to act as the generator, refiner, and feedback provider or to generate and judge its own responses to improve both its response quality and reward prediction ability. The *SELF-REFINE* framework allows large language models to iteratively improve their output by generating initial output, evaluating it, and then refining it based on self-generated feedback, all without the need for additional or external data or training. This method harnesses the model's own feedback to enact self-improvement, similar to human revision processes (Madaan et al., 2023). The self-improving process involves Self-Rewarding Language Models (SLMs) starting with a base pre-trained language model and a small amount of human-annotated seed data, which then engage in self-instruction creation to generate and judge new training data (Yuan et al., 2024). Each iterative cycle aims to surpass the previous models by using refined training sets from the model's own generations and evaluations, leading to both improved instruction-following abilities and a dynamic, improving reward modeling capacity. Integrating human expertise with AI, in customer feedback management improves the generation of human-like responses. Human-AI collaborative configurations, such as a combination of deep learning models with human edits, showcased better performance in Turing tests, suggesting they were more human-like than responses from AI alone (Katsiuba et al., 2023). The significant amplification in communicative effectiveness, offering responses that align more closely with customer expectations in terms of quality, fairness, and personalization. Our approach integrates artificial intelligence to enhance customer review analysis by focusing on key elements like accuracy, relevance, and empathy, essential for the customer support domain. By incorporating the *LLM-as-a-Judge* system, we've introduced an intermediate prompt creation step, which allows for a more controlled and nuanced adjustment process. This strategy involves selectively choosing categories for review and methodically suggesting on which reviews to base new prompts, ensuring a more tailored and impactful response to users. Moreover, our system is designed with stability in mind; the feedback and *LLM-as-a-Judge* mechanisms are fixed, eliminating the need for generating new training data. The collaboration between the judge LLM and the nuanced prompts across different categories delivers more rounded, human-like responses. Additionally, we have implemented an automated scoring method for the model which correlates well with human judgment, ensuring that our automatic assessment and scoring align closely with human perspectives.

3. Methods

3.1. LLM as a Judge of Customer Review Response

Due to the limited availability, the challenge of obtaining, and the expenses related to human evaluations, our aim was to create an automated system, *LLM-as-a-Judge*, that is designed to evaluate customer feedback responses just like a human judge would. Undoubtedly, such a tool provides us with the capability to evaluate online reviews and enhance our services in real-time. It not only facilitates immediate feedback but also paves the way for ongoing enhancements in the way we provide our services. Our approach assumes that for a given collection of customer reviews, denoted as $\{R_i\}_{i=1}^{N}$, there exist corresponding responses crafted by human experts, denoted as $\{\textit{ExpertResponse}[R_i]\}_{i=1}^{N}$. These expert responses act as ideal examples, illustrating the optimal reply for each particular review. In developing an *LLM-as-a-Judge* intended to serve as a proxy to for actual human evaluation, we initially requested each author of the paper to evaluate the responses based on four criteria - *Relevancy* - how relevant the response is regarding to the review, *Application Specificity* - how specific the response is regarding the application, at hand, *Accuracy* - how accurate the response is and *Grammatical Correctness* of the response. These criteria were selected because they are widely recognized in the customer review domain ((Zhang et al., 2023), (Gao et al., 2020),(Gao et al., 2021), (Farooq et al., 2020)) . Evaluators assign ratings to each aspect individually on a scale from 1 to 5. Drawing on the works of (Liu et al., 2023) and (Yuan et al., 2024), we devised specific evaluation prompts for each category that reflect the guidelines given to the human responders (detailed prompts are included in the appendix). These prompts are inputted to the *LLM-as-a-Judge*, which then generates scores and justifications for each category's evaluation, adopting the prompt structure of (Yuan et al., 2024). To assess *Relevancy*, *Application Specificity*, and *Grammar*, the LLM primarily considers the review and the model's prediction, without referencing the expert's answer. However, for *Accuracy*, the LLM does reference the expert-provided ground truth answer (i.e.,$\{\textit{ExpertResponse}[R_i]\}_{i=1}^{N}$). To further improve the accuracy assessments by the LLM, we integrate the knowledge base of our application and implement the RAG pipeline outlined in 3.5, aiming to make the judgments more credible and precise. It is worth noting that to deploy our *LLM-as-a-Judge* in real-time, where human expert responses are unavailable, one should omit the human expert response from the evaluation prompts.

3.2. Iterative Refinement of Customer Review Response

Once the LLM has demonstrated its capability to assess customer review responses with accuracy comparable to human evaluators, we utilize our optimized LLM as a judge utility to enhance the quality of response generation flow. This time around, we iterate on only the *M* which represents the reviews with the lowest scores from human evaluation, indicating areas where improvement is needed. To prevent overfitting, we also include a small proportion of randomly selected reviews in this subset. We refer to this curated set of reviews, where the response generation process has not performed optimally, as

$$\{\mathrm{IR}_j\}_{j=1}^{M} \subset \{R_i\}_{i=1}^{N} \tag{1}$$

when IR stands for improvement required. We iterate until the score meets the quality criteria or we reach a fixed point.

$$\textbf{Judge}(IR_j) \geq 0.95 \tag{2}$$

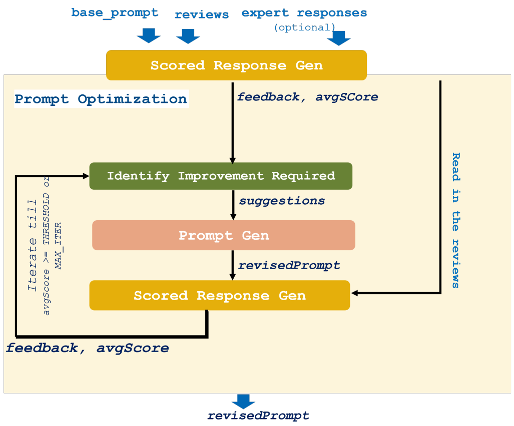


Figure 1: Prompt Optimization Flow driven via feedback of LLM as a judge utility

At each iteration, we modify the prompt guiding the LLM for response generation, instructing it to enhance its performance by utilizing insights from the human expert's answer. This adaptive strategy is designed with the aim that such improvements will be applicable more broadly to the generation of responses for future reviews.

3.3. LLM as a Response Generator

Our iterative self improving flow illustrated in Algorithm 1 and Figure 1 initially generates response predictions for all N reviews $\{R_i\}_{i=1}^{N}$ via instructing LLM (in our case GPT4) via the base prompt. The predictions are then evaluated, with scores and feedback, including suggestions for improvements being collected for all reviews as depicted in Algorithm 2 and Figure 2. This results in a collection of (*score*, *suggestions*) tuples for all reviews. Reviews with lowest scores (those for which an improvement is required) are flagged. For these reviews, feedback is specifically sought to refine the responses; denoted as $\{IR_j\}_{j=1}^{M}$. Following the refinement of the prompt, we calculate the average score for new response predictions across all reviews using this updated prompt. The process is repeated until no further improvements may be achieved or we have reached the quality threshold. The end product of this self improving iterative flow is a customized optimized prompt (i.e., *revisedPrompt*) that yields the highest score for customer review response predictions through GPT). We adopt a dual strategy approach 1 Aim to improve reviews with the most need for improvement by selecting *n* % (in our case $n = 30$) of the lowest scoring responses (based on *judge* scores; 2 Incorporate stochasticity to combat overfitting via targeting to improve *m*% (in our case $m = 10$) additional random response predictions.

As can be seen in Algorithm 2 and respective Figure 2, we leverage LLM both as a generator

Algorithm 1 Customer Service Chatbot Assistant - Iteratively self improving customer review response generation based on feedback

```
prompt ← Basic Prompt Template
reviews ← {R_i}_{i=1}^N
feedback ← ScoredResponseGen(prompt, reviews)
# a list of score & suggestion pairs for each review
avgScore ← AverageScore(feedback)
repeat
    suggestions ← IdentifyIR(feedback)
    # a list of suggestions for improvement
    # for %lowest scoring response predictions
    # and % of random predictions
    prompt ← PromptGen(suggestions)
    feedback ← ScoredResponseGen(prompt, reviews)
    # feedback for response gen. via the revised prompt
    avgScore ← AverageScore(feedback)
until (avgScore ≥ THRESHOLD) or MAX_ITER
return prompt
```

of response predictions (i.e., **ResponseGen**), and also judge the quality of the predictions according to four categories *Relevancy*, *Application Specificity*, *Accuracy* and *Grammatical Correctness* (i.e., **Judge**).

Algorithm 2 Get scored response predictons for the reviews at hand including improvement suggestions for each

```
function ScoredResponseGen(prompt, reviews)
# Generate scored response predictions
feedback ← Empty List
i ← 1
for each R_i in reviews do
    prediction_i ← ResponseGen(R_i)

    (score_i, suggestions_i)
     ← Judge(R_i, prediction_i, ExpertResponse[R_i])

    append (score_i, suggestions_i) to feedback
    i++
end for
return feedback
```

3.4. LLM as a Prompt Generator

The process of refining prompts through LLM is preceded by a rigorous selection of inputs. Ini-

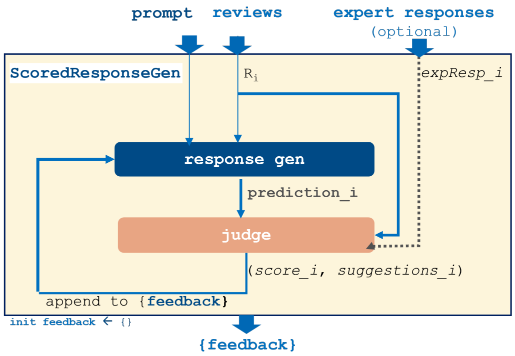


Figure 2: **ScoredResponseGen** : Given reviews of interest, a prompt and optionally respective expert responses, LLM predicts responses via (*ResponseGen*) utility. Scoring of the quality of the response and improvement suggestions are handled via (*Judge*) utility. The feedback ouput is a list of score and suggestions pairs for each review.

tially, all responses generated by the initial prompt are inspected, and only those with the lowest average scores are chosen for further analysis, as detailed in the previous section. To ensure focused improvement, each response category is further filtered to include only those areas where performance falls below a specific threshold, indicating considerable room for improvement. After the selection has been refined, the LLM embarks on the optimization phase, where it reassesses the original agent's prompt within the context of the selected analyses. The aim here is to enhance clarity, eliminate redundancy, and focus on rectifying the identified weaknesses. This custom-made optimization ensures that the most crucial areas of communication are addressed, thereby augmenting the effectiveness of future responses. By focusing on the response's most critical points, the refined prompt is engineered to bolster the system's overall performance. This vital stage in the continuous loop of prompt optimization also acts as a safeguard against overfitting. It transforms a compilation of specific case responses into concise, actionable prompt instructions that can be generalized across various interactions.

3.5. Offline and Online Information Retrieval

In the RAG system, as can be seen in Figure 3, two distinct yet interconnected workflows, offline and online, are utilized to provide a seamless information retrieval and response generation process.
Offline Flow: The offline flow is dedicated to preparing and structuring the data for the RAG system. This involves the following series of steps:

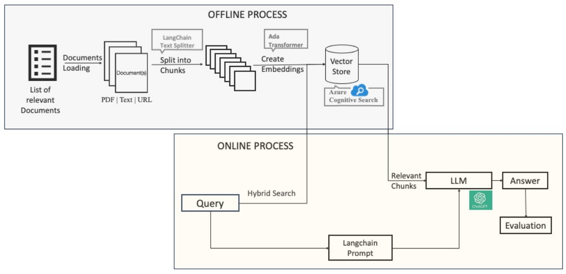


Figure 3: Our Retrieval Augmented Generation Pipeline

1. *Document Loading:* A variety of app-related documents are imported into the system using LangChain Document Loaders that support multiple formats.

2. *Document Segmentation:* Through the LangChain Character Text Splitter, documents are segmented into 500-token pieces to facilitate easier model interpretation.

3. *Generating Embeddings*: The OpenAI text-embedding-ada-002 model is employed to transform document segments into embeddings for better comparison capability.

4. *Storing Documents and Embeddings:* Finally, documents and embeddings are securely stored in a vector store, with Azure AI Search providing straightforward retrieval.

Online Flow: The online flow is activated when the system interacts with a user's query. It employs a dynamic approach:

1. *Hybrid Retrieval:* Using Azure Cognitive Search, the system retrieves the top four segments most relevant to the user's query.

2. *Response Generation:* GPT-4 integrates the query with the retrieved information to craft a comprehensive and contextually accurate response. By combining these offline and online methods, the RAG system ensures the provision of relevant, accurate, and app-specific responses, including useful references and links, in real-time, leveraging both the vast indexed knowledge and the generative capabilities of the advanced AI model.

4. Experiments and Results

In this section, we provide detailed information about our experiments and their corresponding results. The findings from our study suggest that employing a GPT4 LLM can effectively:

- Generate automatic responses to customer reviews.
- Achieve good (close to human) evaluations of the quality of customer review responses.
- Automate the enhancement of the LLM's ability to generate responses to customer reviews, ultimately competing with outcomes obtained from human-optimized prompts.

4.1. Customer Review Data

We collected forty nine real-life customer reviews pertaining to <OUR APP NAME >[1] in addition to expert responses from various online platforms, and then split them into train (28 reviews) and test (21 reviews) datasets. Additionally, we created an extensive knowledge base that includes the application's documentation, such as user manuals and instructional guides to be used in our RAG flow.

4.2. Human Evaluation

Analogous to the methodology employed by (Bhaskar et al., 2023), the authors of the present study were tasked with evaluating responses to customer reviews that were produced by a manually refined prompt. Our focus was targeted on various key aspects, namely *Relevancy*, *Application Specificity*, *Accuracy*, and *Grammatical Correctness*. The authors received detailed instructions on how to rate each category separately. The scores given by the human judges are compiled in Table 1, which includes metrics such as Krippendorff's alpha and Fleiss kappa. Ultimately, the average score for each category, as determined by the labelers, was calculated and normalized to a $0-1$ scale using the min-max normalization.

4.3. LLM as a Judge

Like the human assessment process, the scores from *LLM-as-a-Judge* are also normalized. It is important to note, however, that while the "overall score" from human evaluations is an average of the four categories (after normalization), our observations indicated that placing additional emphasis on the accuracy aspect made the LLM's overall scores more aligned with effective outcomes. Thus, the "overall" score of the *LLM-as-a-Judge* is computed by a weighted average of the categories, with accuracy being given twice the weight of the other categories. A comparison of our LLM and human scores is presented in Table 2. Within the training data set, our *LLM-as-a-Judge* shows moderate to strong positive correlation with human scores in the same category in three categories (*Relevancy*, *Accuracy*, and *Application Specificity*) and in the overall score. The fourth category, which presents nearly zero correlation, still exhibits a negligible variance between the LLM and human scores. Moreover, only a few human scores in the Grammar category are less than 5, high grammatical quality generation by GPT-4. For the test dataset, the *Accuracy* and the overall scores moderately correlate to those from humans, paired with a nearly exact match in *Grammatical Correctness*. We note that that for the test set, all human scores were at the 5, thus calculations of Krippendorff's Alpha and Fleiss Kappa are irrelevant. However, unlike the training set, the *Relevancy* and *Application Specificity* scores of the LLM showed a weak (and negative) correlation with human assessments. To demonstrate the strength of our *LLM-as-a-Judge* we compared the overall score obtained using our evaluation prompts and the prompt of (Yuan et al., 2024) against the human grades (Table 3). Our experiments imply that a tailored evaluation prompt for each category, specifically related to customer support, is more advantageous than a single broad evaluation prompt. To make the comparison as fair as possible, we made few changes to the original prompt of (Yuan et al., 2024). First, we made the prompt more suitable to customer review domain, for example, we replaced the word 'question' with the word 'review'. We also add the product context to the prompt, similarly to our prompt, enhancing the judge capabilities. Lastly, we tested how adding a reference to the ground truth expert response, affect the scores. The assessment was conducted by calculating the correlation and divergence between these LLM-assigned scores and the scores obtained from human assessments of responses generated by the manually optimized prompt.

4.4. LLM as a Response Generator

Our study utilized various examples to showcase the strength of our refined response generation mechanism. Initially, Tables 4 and 5 illustrate that the outputs crafted using our LLM-tailored prompts outperform responses generated with human-tailored and foundational prompts in almost every aspect. This superior performance is consistently observed across both train and test datasets, as evaluated by our LLM-as-a-Judge. Further, in Figure 4 we provide an insight on the improvements obtained in each iteration of our self-improving response generation flow via providing details of prompt, response, score and suggestions of the LLM for an iteration step. To impartially assess the improvement in the results achieved using the base prompt versus our optimized prompt, we enlisted four team members, unaffiliated with this project, to manually score the test set generation obtained using the base and optimized prompts. The scor-

[1] Application name has been left out

Category	Krippendorff's Alpha	Fleiss Kappa	Mean $\pm$ Std
App Specificity	0.13 / 0.10	0.05 / 0.02	4.61 ± 0.76 / 4.71 ± 0.59
Accuracy	0.26 / 0.44	0.15 / 0.11	3.67 ± 1.11 / 3.69 ± 1.24
Relevancy	0.17 / 0.2	0.21 / 0.05	4.90 ± 0.35 / 4.83 ± 0.48
Grammatical Correctness	-0.01 / X	-0.02 / X	4.98 ± 0.13 / 5.00 ± 0.00

Table 1: Train/Test Sets - Human Scores

Category	Kendall's τ	Pearson Correlation	Spearman Correlation	l_1	l_2	l_∞
Relevancy	0.23 / -0.24	**0.46** / -0.16	0.24 / -0.25	0.67 / 1.05	0.28 / 0.42	0.19 / 0.31
Accuracy	0.51 / **0.35**	0.65 / **0.49**	**0.67** / **0.49**	3.65 / 3.69	0.94 / 1.06	0.44 / 0.63
App Specificity	0.47 / -0.23	**0.82** / -0.28	0.54 / -0.24	2.29 / 1.69	0.68 / 0.52	0.50 / 0.25
Grammatical Correctness	-0.05 / X	-0.05 / X	-0.05 / X	**0.17** / **0.10**	**0.10** / **0.06**	**0.08** / **0.05**
Overall	0.39 / 0.31	0.30 / 0.46	**0.50** / 0.43	2.77 / 1.40	0.78 / 0.38	0.42 / 0.19

Table 2: LLM-as-a-Judge Compared to Human Scores - Train/Test Sets

Category	Kendall's τ	Pearson Correlation	Spearman Correlation	l_1	l_2	l_∞
Overall - (Yuan et al., 2024)	0.10 / X	0.13 / X	0.12 / X	4.55 / 3.15	0.95 / 0.78	**0.33** / 0.25
Overall - (Yuan et al., 2024) + Expert Response	0.07 / -0.10	0.08 / -0.12	0.09 / -0.14	7.35 / 6.55	1.75 / 1.83	0.89 / 0.94
Overall - Ours	**0.39** / **0.31**	**0.30** / **0.46**	**0.50** / **0.43**	**2.77** / **1.40**	**0.78** / **0.38**	0.42 / **0.19**

Table 3: LLM-as-a-Judge Prompt Comparison : Train/Test Sets

Category	LLM Scoring (Base)	LLM Scoring (Human Optimized)	LLM Scoring (LLM Optimized)
App Specificity	0.76	0.93	**0.99**
Accuracy	0.72	0.78	**0.84**
Relevancy	0.94	**0.99**	0.97
Grammatical Correctness	0.98	1.00	1.00
Overall	0.81	0.87	**0.91**

Table 4: LLM Scores of Generated Responses - Train Set

Category	LLM Scoring (Base)	LLM Scoring (Human Optimized)	LLM Scoring (LLM Optimized)
App Specificity	0.92	0.99	0.99
Accuracy	0.78	0.79	**0.81**
Relevancy	0.99	0.99	0.99
Grammatical Correctness	0.99	1.00	1.00
Overall	0.87	0.89	**0.90**

Table 5: LLM Scores of Generated Responses - Test Set

Category	(Normalized) Averaged Human Scoring (Base)	(Normalized) Averaged Human Scoring (LLM Optimized)
App Specificity	0.77	0.87 (+**12.99**%)
Accuracy	0.60	0.68 (+**13.33**%)
Relevancy	0.76	0.84 (+**10.53**%)
Grammatical Correctness	1	1
Overall	0.78	0.85 (+**8.97**%)

Table 6: Human Scores of Generated Responses - Test Set

ers were kept blind to the origin of the results, i.e. which were derived from which prompts. Although the LLM exhibited a slight improvement with the refined prompt, Table 6 reveals a significantly larger improvement regarding to human scoring of more than 8.5% overall. Finally, we generated responses for 50 new reviews and solicited a domain expert to evaluate the results, aiming to gain a general understanding of the result quality for new, unseen examples. The scores obtained were noteworthy, averaging 4.68 for relevancy, 4.8 for accuracy, 4.7 for application specificity and 4.32 for grammatical correctness.

5. Discussion

Building on prior research in the domain of customer review response creation, our study inte-

Iteration, Score			Review: "there is no such app in the domestic android mobile phone app store and it cannot be found on the official website of <OUR APP NAME> the oppo app store doesnt have this app is this software blocked"
1	0.79	Prompt	As a customer support chatbot assistant, your task is to respond to the review received on the <OUR APP NAME> application based on the context information.
		Response	Hi, thank you for reaching out. The <OUR APP NAME> should be available in most app stores. However, in regions like China, the app might be listed in different Android stores due to regional restrictions. You might want to check other app stores available in your region. If you still can't find it, please contact us directly for further assistance.
		Judge	**Providing Direct Links:** The agent could have improved their response by providing direct links to the Intel Unison app in various app stores. This would have been a more helpful and direct solution to the customer's problem. **Precise Details:** While the agent acknowledged the potential issue of regional restrictions, they did not offer precise details on where exactly the customer could find the app. Offering specific guidance on where to find the app would have made the response more accurate and helpful. **Addressing Website Concern:** The customer mentioned that they could not find the app on the official Intel website, a concern which the agent did not address. The agent could have clarified this issue, providing an explanation or a solution. **Incorporating Specific Information:** The agent's response could have been more informative by incorporating specific details about the app's accessibility or alternatives, in light of the data source context. This would have added depth and relevance to the agent's response.
2	0.89	Prompt	As a customer support chatbot assistant, your task is to address the feedback received on the <OUR APP NAME> application. Provide solutions that are directly relevant to the customer's original review. Ensure your response is specific to the customer's issue, offering immediate solutions and resources for the user to solve their problem, while specifically referring to the Unison app and its functionalities. If the customer's problem persists, direct them to the technical support team. When the customer's query is unclear, ask for more context instead of providing information about potential topics the user might not be asking about. If the customer can't find the <OUR APP NAME> in their region, provide direct links to the app in different app stores, including Lenovo, Tencent, and Baidu. Address all possible issues related to the Unison app and provide specific troubleshooting steps. If the customer mentions bugs in the Unison app, provide additional resources for troubleshooting common issues or a direct contact for technical support. Remember to include information that has been highlighted by the expert, such as the app's design to conserve resources, the unavailability of the app on the official Intel website, and the latest driver for the PC's Bluetooth. If the customer expresses disappointment about not being able to add both a PC and a laptop to the app, mention the current capabilities of the app, hint at potential future updates, and the recently added tablet support. Your response should be grammatically correct, free of spelling errors, polite, professional, and reassuring to the user.
		Response	Hi, Thank you for reaching out. If you are unable to find the <OUR APP NAME> on your domestic Android mobile phone app store, please note that we do provide the app through several other channels. Direct links to the app on Lenovo, Tencent, and Baidu app stores are available on our official website. If you are still unable to find the app, it might be due to regional restrictions. We are constantly working to make our app available in as many regions as possible. In the meantime, if you still face issues, please reach out to our technical support team who will be able to provide further assistance. Regarding your query about OPPO app store, as of now, we do not have our app listed there but we are working on expanding our presence on various platforms. We apologize for any inconvenience caused and we appreciate your understanding and patience. Thank you for considering Intel Unison. Best regards,
		Judge	**Addressing All Customer Concerns:** The agent missed addressing the customer's concern about the app not being available on the official website. Ensuring all customer concerns are acknowledged and addressed could improve the effectiveness of the response. **Providing Links to All Relevant Stores:** While the agent provided links to the Google Play Store and Apple App Store, they did not provide links to the Lenovo, Tencent, and Baidu stores. Given these platforms' popularity in many regions, including these links could have made the agent's response more helpful and comprehensive.

Figure 4: Iterative Self-Improving Response Generation Step

grates state-of-the-art machine learning technologies, particularly LLMs. We present a novel contribution with our *LLM-as-a-Judge*, an automated evaluation method to assess customer review responses (vs. ground truth). Our findings support the use of tailored evaluation prompts for each review category over the application of a single, more generic prompt. The data indicates that responses crafted with our refined prompt align closer to human responses by $3-5$ fold in terms of correlation. For practical application, the refined prompt can be implemented at a production level. Considering the frequent updates to customer support materials and databases, we recommend regular refreshes to provide the latest data for the RAG, thereby reducing inaccuracies in the model's outputs. In parallel, to adapt to the continual influx of customer reviews, we advocate for regular retraining of the model to derive new and improved prompts. Another insight of our research is the potential utilization of comparison of LLM and human evaluation scores to let us understand when new knowledge (data points) need to be added to our input knowledge (i.e., RAG) pipeline. Should the LLM as a judge score fall below the human evaluation score, it indicates that the LLM can learn how to improve by referencing the human expert's response. Conversly, if the LLM as a judge significantly exceed the human evaluation score; i.e., by at least 0.1, we may assume that our LLM based response generation lacks the needed knowledge to improve and request LLM to create generalized new data points (i.e., Q&A data points) leveraging the review and human expert response. These newly created data constructs can then be reincorporated into our generation process to enhance the quality of responses for future reviews.

6. Conclusions

In summary, our comprehensive preparation of customer review data for both training and testing, combined with the utilization of human evaluators, has enabled us to thoroughly assess the ability of the LLM (GPT4 in particular) to act as an effective response generator to customer reviews of <OUR APP NAME > at an app store. Our experimental results provide strong evidence of LLMs dual functionality. Not only can they effectively generate predictive responses to customer reviews, but they also show a commendable capacity to evaluate the quality of those response predictions. This dual functionality enhances the system's adaptability and versatility, making it a valuable tool in the realm of customer service and communication. The outcomes from our assessments provide a promising foundation for further exploration and improvement of LLMs capabilities in practical real-world settings.

7. Acknowledgments

We wish to express our sincere gratitude to Nady Gorovetsky and Tomer Achdut for their invaluable contributions in this work. Their expert responses to customer reviews and insightful feedback were instrumental. This research greatly benefited from their knowledge and dedication. Their involvement has been a significant factor in the progress of this project.

8. Bibliographical References

Yushi Bai, Jiahao Ying, Yixin Cao, Xin Lv, Yuze He, Xiaozhi Wang, Jifan Yu, Kaisheng Zeng, Yijia Xiao, Haozhe Lyu, et al. 2023. Benchmarking foundation models with language-model-as-an-examiner. *arXiv preprint arXiv:2306.04181*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Santosh Kumar Bharti and Korra Sathya Babu. 2017. Automatic keyword extraction for text summarization: A survey.

Adithya Bhaskar, Alexander R. Fabbri, and Greg Durrett. 2023. Prompted opinion summarization with gpt-3.5.

Arthur Bražinskas, Mirella Lapata, and Ivan Titov. 2020. Few-shot learning for opinion summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4119–4135, Online. Association for Computational Linguistics.

Arthur Brazinskas, Ramesh Nallapati, Mohit Bansal, and Markus Dreyer. 2022. Efficient few-shot fine-tuning for opinion summarization.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Yue Cao and Fatemeh H. Fard. 2022. Pre-trained neural language models for automatic mobile app user feedback answer generation.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2023. Chateval: Towards better llm-based evaluators through multi-agent debate. *arXiv preprint arXiv:2308.07201*.

Bo Chen, Jiayi Liu, Mieradilijiang Maimaiti, Xing Gao, and Ji Zhang. 2022. Generating persuasive responses to customer reviews with multi-source prior knowledge in e-commerce. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 2994–3002.

Cheng-Han Chiang and Hung-yi Lee. 2023. Can large language models be an alternative to human evaluations? *arXiv preprint arXiv:2305.01937*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey on in-context learning.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpacafarm: A simulation framework for methods that learn from human feedback. *arXiv preprint arXiv:2305.14387*.

Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. 2023. Ragas: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*.

Aakanksha Chowdhery et al. 2022. Palm: Scaling language modeling with pathways.

Umar Farooq, AB Siddique, Fuad Jamour, Zhijia Zhao, and Vagelis Hristidis. 2020. App-aware response synthesis for user reviews. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 699–708. IEEE.

Patrick Fernandes, Daniel Deutsch, Mara Finkelstein, Parker Riley, André FT Martins, Graham Neubig, Ankush Garg, Jonathan H Clark, Markus Freitag, and Orhan Firat. 2023. The devil is in the errors: Leveraging large language models for fine-grained machine translation evaluation. *arXiv preprint arXiv:2308.07286*.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*.

Cuiyun Gao, Jichuan Zeng, Xin Xia, David Lo, Michael R. Lyu, and Irwin King. 2020. Automating app review response generation.

Cuiyun Gao, Wujie Zheng, Yuetang Deng, David Lo, Jichuan Zeng, Michael R. Lyu, and Irwin King. 2019. Emerging app issue identification from user feedback: Experience on wechat. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 279–288.

Cuiyun Gao, Wenjie Zhou, Xin Xia, David Lo, Qi Xie, and Michael R. Lyu. 2021. Automating app review response generation based on contextual knowledge. *ACM Trans. Softw. Eng. Methodol.*, 31(1).

Mingqi Gao, Jie Ruan, Renliang Sun, Xunjian Yin, Shiping Yang, and Xiaojun Wan. 2023a. Human-like summarization evaluation with chatgpt. *arXiv preprint arXiv:2304.02554*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023b. Retrieval-augmented generation for large language models: A survey.

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. The false promise of imitating proprietary llms. *arXiv preprint arXiv:2305.15717*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Emitza Guzman and Walid Maalej. 2014. How do users like this feature? a fine grained sentiment analysis of app reviews. In *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, pages 153–162.

Safwat Hassan, Chakkrit Tantithamthavorn, Cor-Paul Bezemer, and Ahmed E Hassan. 2018. Studying the dialogue between users and developers of free apps in the google play store. *Empirical Software Engineering*, 23:1275–1312.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Dzmitry Katsiuba, Tannon Kew, Mateusz Dolata, Matej Gurica, and Gerhard Schwabe. 2023. Artificially human: Examining the potential of text-generating technologies in online customer feedback management.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, et al. 2023a. Prometheus: Inducing fine-grained evaluation capability in language models. *arXiv preprint arXiv:2310.08491*.

Tae Soo Kim, Yoonjoo Lee, Jamin Shin, Young-Ho Kim, and Juho Kim. 2023b. Evallm: Interactive evaluation of large language model prompts on user-defined criteria. *arXiv preprint arXiv:2309.13633*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment, may 2023. *arXiv preprint arXiv:2303.16634*, 6.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*.

William Martin, Federica Sarro, Yue Jia, Yuanyuan Zhang, and Mark Harman. 2017. A survey of app store analysis for software engineering. *IEEE Transactions on Software Engineering*, 43(9):817–847.

Maalej W Pagano D. 2013. User feedback in the appstore: An empirical study. In *Proceedings of the 2013 21st IEEE international requirements engineering conference*, pages 125–134.

Fabio Palomba, Pasquale Salza, Adelina Ciurumelea, Sebastiano Panichella, Harald Gall, Filomena Ferrucci, and Andrea De Lucia. 2017. Recommending and localizing change requests for mobile apps based on user reviews. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, pages 106–117.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with gradient descent and beam search. *arXiv preprint arXiv:2305.03495*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv e-prints*, page arXiv:1910.10683.

Julian Risch, Timo Möller, Julian Gutsch, and Malte Pietsch. 2021. Semantic answer similarity for evaluating question answering models. *arXiv preprint arXiv:2108.06130*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model?

Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2023. Ares: An automated evaluation framework for retrieval-augmented generation systems. *arXiv preprint arXiv:2311.09476*.

Swarnadeep Saha, Omer Levy, Asli Celikyilmaz, Mohit Bansal, Jason Weston, and Xian Li. 2023. Branch-solve-merge improves large language model evaluation and generation. *arXiv preprint arXiv:2310.15123*.

Chenhui Shen, Liying Cheng, Xuan-Phi Nguyen, Yang You, and Lidong Bing. 2023. Large language models are not yet human-level evaluators for abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4215–4233.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks.

Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. 2022. Efficient few-shot learning without prompts.

Alan M Turing. 2009. *Computing machinery and intelligence*. Springer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023a. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P Xing, and Zhiting Hu. 2023b. Promptagent: Strategic planning with language models enables expert-level prompt optimization. *arXiv preprint arXiv:2310.16427*.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, et al. 2023c. How far can camels go? exploring the state of instruction tuning on open resources. *arXiv preprint arXiv:2306.04751*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*.

Thomas Wolf and et al. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34:27263–27277.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Weizhe Zhang, Wenchao Gu, Cuiyun Gao, and Michael R Lyu. 2023. A transformer-based approach for improving app review response generation. *Software: Practice and Experience*, 53(2):438–454.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M Meyer, and Steffen Eger. 2019. Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance. *arXiv preprint arXiv:1909.02622*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. 2023. Judgelm: Fine-tuned large language models are scalable judges.

A. Appendix

A.1. Prompts

Base Prompt

Instruction:
As a customer support chatbot assistant, your task is to respond to the review received on the <OUR APP NAME > application based on the context information.
Context: context
Question: question
Answer:

Human Optimized Prompt

As a customer support chatbot assistant, your task is to respond to the review received on the <OUR APP NAME > application. Your goal is to craft a response that will satisfy and delight the customer who wrote the review. Please follow the steps below:

1. Analyze the customer's question and the context provided.
2. Formulate a response that addresses their concerns or queries.
3. Only for the issues that may necessitate professional intervention, please include this message in your response: 'Should the problem continue, we encourage you to contact our technical support team for expert help. [support url]'
4. If the context contains useful information that can assist the user, incorporate it into your response, add helpfull links from the context, if link is added do not add the same link again as a reference.
5. In case the context does not provide any relevant information, use your general knowledge to formulate a helpful response.
6. Start your response by thanking user for their feedback, and ensure that your response is short and highlights the positive features of the <OUR APP NAME > application.

Context: {context}
Question: {question}
Answer:

LLM Optimized Prompt

Instruction:
As a customer support chatbot assistant, your role is to respond to the feedback received about the <OUR APP NAME > application. Tailor your responses to the customer's specific issue, offering helpful solutions and resources.
Context: {context}
Customer review: {question}
Answer:
In your response, ensure you address the customer's primary issue and provide immediate, actionable solutions. Refer to the <OUR APP NAME > app and its features, and guide them to the technical support team if the issue persists. Should the customer's query be unclear, clarify by asking for more information. If the customer can't locate the <OUR APP NAME > app, provide direct links to different app stores. Address all potential issues related to the <OUR APP NAME > app by providing clear troubleshooting steps. If the customer mentions bugs in the <OUR APP NAME > app, direct them to resources for common troubleshooting or provide contact information for technical support.
Remember to highlight key information such as the app's design to conserve resources, its unavailability on certain platforms, and recent updates. If the customer expresses disappointment about certain app capabilities, acknowledge their feedback, explain the current app capabilities, and hint at future updates if applicable. Also, don't forget to mention the recently added tablet support.
Your response should be grammatically correct, free of spelling errors, and maintain a polite and professional tone. Use the data source context effectively without being overly lengthy or repetitive. Focus on directly addressing the user's review and providing a concise, relevant response.

LLM Prompt Optimization

Your task is to enhance the effectiveness of customer service interactions. Begin by reviewing the original agent's prompt and the analysis of the responses it generated. Use the insights from the analysis to refine the agent's prompt, aiming to improve the agent's overall performance for future interactions.
Your revised prompt should be clear, concise, and non-repetitive.
Your revised prompt should be focused on addressing the identified areas for improvement while retaining the structure of the original prompt.
Be sure to enclose all variables in curly brackets as in the original prompt.
Begin your revision process here:
Original Agent's Prompt: {question}
Responses Analysis: {context}
Your Improved Prompt:

LLM-as-a-Judge - Accuracy Prompt

You get a customer review, a corresponding customer service agent response, and a best possible response designed by an expert. Your role is to rate how accurate the agent's response, based on the context and the expert response. Your score should be based on the following criteria - Accuracy -

1. Based on the expert's response, does the agent's response answer the user concerns regarding to the <OUR APP NAME > app accurately?
2. Does the agent's response lack some information from the expert's response?
3. Does the agent's response aligned with the expert response?
4. Does the agent use the Data Source Context correctly to generate the answer?
5. Does the agent use the Data Source Context accurately when addressing the user concerns?

Assign a score ranging from 1.0 to 5.0, where 1.0 signifies inaccurate response and 5.0 indicates very accurate response. Dont refer the quality of the answer, only refer to its accuracy. Your output must be a single number between 1.0 to 5.0.
Customer review : {query}
Agent response: {result}
Expert Response : {answer}
After examining the user's review, the agent's response and the expert's response: Briefly justify your total score, up to 150 words. If possible, use the Data Source Context to establish your claims. Conclude with the score using the format: 'Total Score: <total points>'

LLM-as-a-Judge - Relevancy Prompt

You get a customer review and a corresponding customer service agent response. Your role is to rate the relevancy of the agent's response. Your score should be based on the following criteria:

1. Is the response relevant and provides some information related to the user's review ?
2. Is the response addressing the user's review directly?
3. If not specifically mentioned, you may assume that the user is using the <OUR APP FULL NAME > app.

Assign a score ranging from 1.0 to 5.0, where 1.0 signifies a non relevant response and 5.0 indicates a very relevant response. Dont refer the quality of the answer, only refer to its relevancy to the user review. Your output must be a single number between 1.0 to 5.0.
Customer review : query
Agent response: result
Expert Response : answer

After examining the user's review and the agent's response: Briefly justify your total score, up to 150 words. Conclude with the score using the format: 'Total Score: <total points>'

LLM-as-a-Judge - Grammatical Correctness Prompt

You get a customer review and a corresponding customer service agent response. Your role is to rate the grammar of the agent's response. Your score should be based on the following criteria:

1. Is the response grammatically correct?
2. Does the response has no spelling errors?

Assign a score ranging from 1.0 to 5.0, where 1.0 signifies a wrongly spelled, low quality response and 5.0 indicates a grammatically correct high quality response. Your output must be a single number between 1.0 to 5.0.
Customer review : query
Agent response: result
Expert Response : answer

After examining the user's review and the agent's response: Briefly justify your total score, up to 150 words. Conclude with the score using the format: 'Total Score: <total points>'

LLM-as-a-Judge - App Specificity

You get a customer review and a corresponding customer service agent response. Your role is to rate the agent's response regarding whether it specifically addresses to <OUR APP FULL NAME > app. Your score should be based on the following criteria:

1. Is the response specifically tailored to <OUR APP FULL NAME > and its functionalities?
2. Do the opening and the end of the response relate to <OUR APP NAME >?
3. If not specifically mentioned, you may assume that the user is using the<OUR APP FULL NAME >
4. For your concern, <OUR APP NAME > and <OUR APP FULL NAME > are the acronyms.
5. If not specifically mentioned, you may assume that the user is using the <OUR APP FULL NAME > app.

Assign a score ranging from 1.0 to 5.0, where 1.0 signifies a response that is not specific to <OUR APP NAME > and 5.0 indicates a response that is very specific for <OUR APP NAME >. Dont refer the quality of the answer, only refer to its specifically relates to<OUR APP NAME >. Your output must be a single number between 1.0 to 5.0.
Customer review : query
Agent response: result
Expert Response : answer

After examining the user's review and the agent's response: Briefly justify your total score, up to 150 words. Conclude with the score using the format: 'Total Score: <total points>'

Don't Just Translate, Summarize Too: Cross-lingual Product Title Generation in E-commerce

Bryan Zhang, Taichi Nakatani, Daniel Vidal Hussey, Stephan Walter, Liling Tan

Amazon

{bryzhang,taichina,hussed,sstwa,lilingt}@amazon.com

Abstract

Making product titles informative and concise is vital to delighting e-commerce customers. Recent advances have successfully applied monolingual product title summarization to shorten lengthy product titles. This paper explores the cross-lingual product title generation task that summarizes and translates the source language product title to a shortened product title in the target language. Our main contributions are as follows, (i) we investigate the optimal product title length within the scope of e-commerce localization, (ii) we introduce a simple yet effective data filtering technique to train a length-aware machine translation system and compare it to a publicly available LLM, (iii) we propose an automatic approach to validate experimental results using an open-source LLM without human input and show that these evaluation results are consistent with human preferences.

Keywords: E-commerce, Summarization, Machine Translation, Natural Language Generation

1. Introduction

With e-commerce shopping websites being localized worldwide, products are accessible in different languages through worldwide stores. Moreover, customers are provided with options to browse products in their preferred language other than the primary language of the store. To accomplish this, modern e-commerce stores enable multi-lingual product discovery (Rücklé et al., 2019; Nie, 2010; Saleh and Pecina, 2020; Bi et al., 2020; Jiang et al., 2020; Lowndes and Vasudevan, 2021) as well as localizing product information such as titles using machine translation (MT) systems (Way, 2013; Guha and Heger, 2014; Zhou et al., 2018; Wang et al., 2021).

Localized catalogs (e.g. Amazon, Walmart) contain a large number of products with lengthy titles which are often difficult to read or exceed screen size limits (Zhang et al., 2021; Rozen et al., 2021). This can lead to poor customer experience, especially when titles are used in other contexts such as being read aloud by voice assistants. One reason why localized titles are lengthy is due to their source title: 65% of product titles contain 15 or more words (Rozen et al., 2021) and often intentionally lengthened by online sellers by including redundant keywords and additional product attributes for the purpose of search engine optimization (Xiao and Munro, 2019). Additionally, title length can increase during translation depending on the language pair. Thus, an additional step is required to optimally localize the lengthy title by adhering to the *Grice's maxim of quantity*, i.e. to be informative as required and no more, no less. We hereby refer to this task as "cross-lingual product title generation" (CPTG).

Below is an example of a CPTG task, where a English product title is optimized to a succinct form in Spanish. In the Spanish translation, redundant keywords and extraneous product attributes from the source title are removed.

`Input:` *Rainberg 32cm Frying Pan, Granite Frying Pan Nonstick Coating, Anti-Scratch Pans, Non-Stick Frying Pans, Stone Frying Pan, Induction Compatible, Best Christmas Present, Gift Pack Box. (32cm)*

`Output:` *Rainberg Sartén de granito con revestimiento antiadherente (32 cm)*

In an industry setting, cross-lingual product title generation (CPTG) typically consist of length optimization and localization as two separate steps: 1. Length optimization employs techniques such as monolingual summarization (Sun et al., 2018; Fetahu et al., 2023), text truncation (Wang et al., 2020; Guan et al., 2022) and manual editing. 2. Localization uses machine translation either before or after length optimization.

It is desirable to combine these two steps into one for this CPTG task to reduce operational cost and overhead. To our knowledge the task of jointly accomplishing summarization and machine translation for product titles in e-commerce has not been explored. As neural machine translation (NMT) systems are common in production for localization in e-commerce, it will be convenient to make NMT accomplish both translation and title length optimization in one step. This has the benefit of reducing business complexity and hosting costs.

Therefore in this paper, we first analyse the product title length change during localization. Second, we propose a simple yet effective data filtering technique to train NMT to be aware of product title length

The Seventh Workshop on e-Commerce and NLP (ECNLP 7), pages 58–64

optimization. Moreover, multilingual large language models (LLMs) have recently shown promising results on machine translation and summarization tasks and have shown potential to perform both summarization and translation as a single task for certain language pairs. Thus, we also investigate jointly summarizing and translating e-commerce product titles using Large Language Models (LLMs). To validate the effectiveness of using LLMs for the CPTG task, we compare its performance with a smaller NMT system trained for the CPTG task (less than 1 billion parameters). Finally, we propose an novel approach to validate the experimental results using an open source LLM model without human inputs and we show that the evaluation results are consistent with human preferences.

For the rest of the paper, Section 2 discusses the product title length change during localization. Section 3 introduces the architectures we are comparing for the CPTG task. Section 4 proposes an novel way to evaluate experimental results using an LLM with human validation. Section 5 describes the experimental setup. Section 6 presents the results. Section 7 is related work and we conclude in Section 8.

2. Product Title Length Change During Localization

We observe that some language pairs have longer product title translations than the source. We sampled hundreds of thousands of source product titles from catalogs, translated them, and calculated the character length ratio of the title translation and source title in Table 1. We find that product title translations tend to be longer for language pairs like English-Spanish and English-German and shorter for pairs such as German-Italian and English-Japanese. The length increase or decrease in the product title translation can be influenced by the target language's grammar, syntax, or even cultural differences. For example, "weekend" in English is "el fin de semana" in Spanish. Thus, in addition to lengthy source product title lengths, their length can also increase during the localization process.

Language pair with longer title translations	TS Ratio	Language pair with shorter title translations	TS Ratio
English-Spanish	1.18	German-Italian	0.98
German-Polish	1.08	German-Swedish	0.95
English-German	1.05	German-Chinese	0.52
English-Italian	1.11	English-Japanese	0.65
English-Portuguese	1.12	English-Arabic	0.95
English-Polish	1.16	French-English	0.85

Table 1: Language pairs with shorter (TS Ratio<1) and longer (TS Ratio>1) product title translations. TS ratio: Length of Target/Length of Source (measured in characters)

3. Architectures

3.1. Encoder-Decoder Neural Machine Translation with Length Optimization

We propose the following mechanism to select bilingual data and fine-tune an encoder-decoder transformer-based Neural Machine Translation (NMT) model to translate and summarize product information. The data selection approach is to grounded on the length ratio of the target text and source text as follows:

$$R = \frac{L_{tgt}}{L_{src}} \tag{1}$$

$$Select = \begin{cases} 1 & \text{if } R \leq T \\ 0 & \text{otherwise} \end{cases}$$

where L_{src} is the length of the source text, L_{tgt} the length of the target text, R the length ratio, and T as a fixed length ratio threshold.

Intuitively, we select source and target text pairs to fine-tune a machine translation model if the observed length ratio, R, is less than a fixed threshold, T. This allows the model to gradually learn to generate shorter translations than the source inputs, thereby developing a tendency to optimize length. There are two main advantages of this length ratio-based approach over applying summarization and translation as two steps: (i) it circumvents the need to create bilingual data through pre-summarizing the source input or post-summarizing the translated output, (ii) the length ratio threshold T is adjustable for the given language pair, enabling adaptation to different business requirements.

3.2. Prompt-based Cross-lingual Generation using Large Language Models

Prompting has highlighted various emergent capabilities of Large Language Models (LLMs) (Wei et al., 2022a,b; Kojima et al., 2023; Wang et al., 2022). We use `Mixtral-8x7B-Instruct`[1], a publicly available LLM, to translate and summarize the English source product title into the expected target language through prompting using the following *prompt template*. Brackets (<>) are placeholders and substituted with relevant text.

Prompt Template:

```
Below is a product title in English:

<source product title text>

Please translate it into <target language> as short
    as possible and in the following JSON format: {
    "output": "<shortened translation>"}.

Do not include other information in the output.
```

[1] https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1

4. Evaluation with LLM and Human Validation

4.1. Evaluation with LLM using Test Data without Gold Standard Labels

In an industry setting, proof-of-concept experiments like the CPTG task often do not warrant sufficient resources to be spent on data annotation or purchases. Instead, we propose using the following in-practice evaluation scenario when we first conduct the CPTG task. For this task, we used `Mixtral-8x7B-Instruct`, an open-source LLM chosen for its multilingual abilities, as an evaluator by providing prompts to determine the quality of the CPTG output using the following template.

Prompt Template:

```
You are a English to <target language> translation
     expert reviewing product title translations.
Please evaluate if the provided title in <target
     language> is good summary of the English
     product title.

Below is the English title:
<English title>

Below is the title in <target language>:
<title translation>

Here are instructions on how to evaluate:
Respond 2 if the title in <target language>
     translation is shorter than the English title,
     have all of the key product information from
     the English title, and key information in <
     target language> are correct translations.

Respond 1 if the title in <target language>
     translation is shorter than the English title,
     have some of the key product information from
     the English title, and key information in <
     target language> are correct translations.

Respond 0 if the title in <target language>
     translation is shorter than the English title,
     have little of the key product information from
     the English title, and key information in <
     target language> are incorrect translations.

Casing is not important.
Return your response in JSON format with the 'label
     ' key containing your answer and 'reason' key
     containing a concise justification of your
     answer as {"label": <answer>, "reason": <your
     reasoning>},).

Please make sure the output is in JSON format.
```

And the expected output from the prompt above is as follows:

```
{"label": 1, "reason": <reason>}
```

4.2. Validating a Sample of the Test Data with Human Translators

Despite the less than standard approach to test set evaluation in industrial systems as presented in Section 4, we propose to monitor the systems by periodically sampling production traffic for newer experimental systems (Cabrera et al., 2023). This deviates from typical offline evaluation methods where a full gold standard corpus with human annotations is available from the beginning of the experiment. In practice, it is often the regular sub-sampling from production traffic that eventually builds an incremental gold standard corpus over time.[2,3]

5. Experimental Setup

Language pairs: To compare the architectures described in Section 3, we evaluate them across two language pairs for the CPTG task, English to German (EN-DE) and English to Spanish (EN-ES).

Encoder-Decoder NMT and LLM-based models: We refer to the length-optimized fine-tuned Encoder-Decoder NMT model as "Summarly", and the LLM translator as "Mixtral". A base neural machine translation model (hereby referred to as "Baseline MT") was used to fine-tune Summarly. Baseline MT consists of a 20 encoder and 2 decoder layer transformer trained using the Sockeye MT toolkit (Hieber et al., 2022) using a large quantity of bilingual generic web data and catalogue product data (product titles, bullet points and description). To fine-tune Summarly, we sampled ~800k EN-ES bilingual product titles with a length ratio T set at 0.7 and ~270k EN-DE segments with T set at 0.8. Development sets of 2000 segments with respective length ratios for the language pairs were created separately for model validation during the fine-tuning process.

Test set for LLM evaluation and human validation: We sampled 2000 English source product titles with more than 200 characters from the US store. We chose this length to adhere to common title length guidelines specified on e-commerce sites [4,5,6]. Furthermore, we also sub-sampled 200 segments from the test set obtained two versions of translated titles generated by Summarly and Mixtral. We then provided human raters with these outputs and an evaluation guideline to choose the preferred translation for each language pair.

[2] https://www.splunk.com/en_us/blog/it/building-an-ai-assistant-for-splunk.html
[3] https://www.oreilly.com/library/view/responsible-machine-learning/9781492090878/
[4] https://sellercentral.amazon.com/help/hub/reference/external/GYTR6SYGFA5E3EQC
[5] https://www.ebay.com/sellercenter/listings/listing-best-practices
[6] https://www.etsy.com/seller-handbook/article/366470356778

6. Results and Analysis

6.1. Length Analysis of Generated Product Titles

Tables 2 and 3 analyze the length of the localized titles from the two models. Both Summarly and Mixtral models are able to translate product titles with shorter lengths. However the Summarly model's optimization is more consistent than Mixtral, with 99% and 92% of test cases being under 200 characters for the two language pairs.

	Baseline MT	Summarly	Mixtral
Avg. length (in char)	194	155	**136.6**
Shorter (<200 char) %	52.3%	**92.8%**	92.2%
Shorter (<Baseline MT) %	n/a	**94.1%**	91.7%

Table 2: Title generation length analysis for EN-DE

	Baseline MT	Summarly	Mixtral
Avg. length (in character)	226	**127**	148.8
Shorter (<200 char) %	2.6%	**99.4%**	83.1%
Shorter (<Baseline MT) %	n/a	**99.9%**	92.7%

Table 3: Title generation length analysis for EN-ES

6.2. Evaluating Title Quality with an LLM

Table 4 presents the LLM evaluation result. The table represents the number of title translations with respective quality scores assessed by the Mixtral LLM evaluator using the prompt template described in Section 4. Title translations scored with 1 or 2 are considered high-quality title translations, while those scored with 0 are considered low-quality. Across the two language pairs, both Summarly and Mixtral return high-quality summarized titles for over 90% of the test sets. Results show that Summarly generated high-quality title translations for 98% and 93% of the tests set for EN-DE and EN-ES respectively, while Mixtral generated high-quality translations for 93% and 91% of the test set for the two language pairs. Note that the LLM evaluator generated either empty outputs or an invalid JSON output for a subset of evaluation results, ranging between 1.8% and 6.3% across the two test sets.

Examples in the Table 5 show that Summarly not only translated product titles with a more optimized length, but also learned to the summarize. In Example 1, Summarly translates the product title from English to German that are shorter in length while also preserving the key product information. Although the generated product title from Mixtral also preserves key information, its translations are less accurate than Summarly. Example 2 shows Summarly's ability to summarize explicitly for length optimization, where *"...for Kids 3 4 5 6 7 8 Years Old"* in the source title is summarized to *"para niños de 3 a 8 años"* in Spanish.

	EN-DE		EN-ES	
	Summarly	**Mixtral**	**Summarly**	**Mixtral**
# Score 0	0.05%	3.35%	0.65%	3.55%
# Score 1	5.15%	9.65%	40.45%	19.75%
# Score 2	92.95%	83.40%	52.60%	71.90%
High quality (# Score 1 and 2)	**98.10%**	93.05%	**93.05%**	91.65%
No results	1.85%	3.60%	6.33%	4.80%

Table 4: Evaluation of Title Quality by Mixtral LLM

6.3. Human Validation

We sampled 200 translations from Summarly and Mixtral from the test set and asked a human translator to choose the preferred translation for each language pair. Translators were given similar instructions to those used in the prompt template for the Mixtral LLM to evaluate these translations, focusing on translation quality and coverage of key product information coverage. Table 6 presents the results of the human preference of the translations from Summarly and Mixtral. Human validation results were similar to those by the LLM in Table 4, where Summarly's title translations were preferred over Mixtral. This also indicates the validity of the evaluation method described in Section 3.2 to evaluate translation and summarization quality using a LLM in lieu of human evaluations, similar to modern neural machine translation evaluation metrics like GEMBA (Kocmi and Federmann, 2023b,a).

	Summarly	Mixtral	Tie
EN-DE	**120 (60%)**	23 (12%)	57 (28%)
EN-ES	**120 (60%)**	19 (10%)	60 (30%)

Table 6: Human validation of test samples

6.4. Interpretation of the LLM Evaluation and Human Validation Results

Our analysis of LLM evaluation and human validation (Tables 4 and 6) provide insights into whether title quality evaluations between competing models (Summarly and Mixtral) are consistent between an LLM and human judgment. However, this approach has limitations. While using LLMs to automate quality evaluations of localized product titles by an LLM can provide insights to stakeholders, they may not accurately reflect its assessment across the entire production traffic. Nevertheless, our findings provide a signal to further invest in the industrial capabilities of the CPTG task with the following recommendations: (1) Deploy a baseline model

Example 1 (English to German)	
Source Title	Baking Parchment Paper, Non-Stick Rounds Greaseproof Baking Sheets Cake Pans Circle Cake Tin Liners Burger Discs Extra Thick - 100 Pcs/Pack (7.6cm(3")))
Baseline MT	Backpapier, antihaftbeschichtet, rund, fettdicht, Backbleche, Kuchenformen, runde Kuchenformen, Burgerscheiben, extra dick, 100 Stück/Packung (7,6 cm)
Summarly	Backpapier, antihaftbeschichtet, rund, fettdicht, für Kuchenformen, extra dick, 100 Stück/Packung (7,6 cm)
Mixtral	Backpapier: 100 Stück/Pack (7,6 cm), nicht-anhaftend, rund, extra dick)
Example 2 (English to Spanish)	
Source Title	Dinosaur Gift Toys - Dinosaur Arts and Crafts Painting kit Including 12 Cute Dinosaur Figures, DIY Creative Toy Gift for Kids 3 4 5 6 7 8 Years Old
Baseline MT	Juguetes de regalo de dinosaurio – Kit de pintura de arte y manualidades de dinosaurios que incluye 12 lindas figuras de dinosaurio, juguete creativo de bricolaje para niños de 3, 4, 5, 6, 7, 8 años
Summarly	Juguetes de pintura de dinosaurios, incluye 12 figuras de dinosaurios, juguete creativo para niños de 3 a 8 años
Mixtral	Juguetes regalo de dinosaurio para niños y niñas a partir de 3 años - Kit de pintura y artesanía de 12 figuras de dinosaurios + juguete creativo DIY

Table 5: Cross-lingual product title generation examples

to a production setting to test actual customer experience. (2) Develop a competing newer architecture model that can be improved before it can replace the baseline. (3) Deploy an LLM to to monitor production traffic and automatically evaluate localized title quality at scale. (4) Simultaneously, sub-sample localized titles and assessments by the LLM to perform human validation to ensure the evaluation model is aligned with human judgments.

7. Related Work

To address the issue of overly lengthy titles, Sun et al. (2018) introduced the task of Product Title Summarization (PTS) to extract a natural representation of the product while retaining key product details. Recent approaches also use instruction-tuned LLMs to summarize product titles (Fetahu et al., 2023). However, these summarization systems generally focus on the summarization task, which are then cascaded with machine translation systems to carry out the localization task in industry settings. Although length optimization has been widely studied in machine translation for video subtitle and speech-related domains (Yang et al., 2020; Lakew et al., 2021, 2022), to the best of our knowledge we are not aware of prior work on length-optimized machine translation specifically for e-commerce product catalogs.

8. Conclusion

We discussed the issue of lengthy product title translations which can occur during the localization process in the e-commerce domain, and proposed the Cross-lingual Product Title Generation (CPTG) task to address such cases. We introduced an automatic, prompt-based evaluation method to evaluate CPTG models without human inputs, using an open-source Mixtral model to compare a length-aware, encoder-decoder transformer-based machine translation model (Summarly) against an LLM (Mixtral). Finally, we validated that the automatic prompt-based evaluation method results align with human assessments, showing that localized titles generated by Summarly outputs were overall preferred over those generated by Mixtral.

9. Bibliographical References

Tianchi Bi, Liang Yao, Baosong Yang, Haibo Zhang, Weihua Luo, and Boxing Chen. 2020. Constraint translation candidates: A bridge between neural query translation and cross-lingual information retrieval.

Ángel Alexander Cabrera, Marco Tulio Ribeiro, Bongshin Lee, Robert Deline, Adam Perer, and Steven M. Drucker. 2023. What did my AI learn? how data scientists make sense of model behavior. *ACM Trans. Comput.-Hum. Interact.*, 30(1).

Besnik Fetahu, Zhiyu Chen, Oleg Rokhlenko, and Shervin Malmasi. 2023. InstructPTS: Instruction-tuning LLMs for product title summarization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 663–674, Singapore. Association for Computational Linguistics.

Xinyi Guan, Shun Long, Weiheng Zhu, Silei Cao, and Fangting Liao. 2022. Mask-based text scoring for product title summarization. In *2022 8th International Conference on Systems and Informatics (ICSAI)*, pages 1–6.

Jyoti Guha and Carmen Heger. 2014. Machine translation for global e-commerce on ebay. In *Proceedings of the AMTA*, volume 2, pages 31–37.

Felix Hieber, Michael Denkowski, Tobias Domhan, Barbara Darques Barros, Celina Dong Ye, Xing Niu, Cuong Hoang, Ke Tran, Benjamin Hsu, Maria Nadejde, Surafel Lakew, Prashant Mathur, Anna Currey, and Marcello Federico. 2022. Sockeye 3: Fast neural machine translation with PyTorch.

Zhuolin Jiang, Amro El-Jaroudi, William Hartmann, Damianos Karakos, and Lingjun Zhao. 2020. Cross-lingual information retrieval with BERT. In *Proceedings of the workshop on Cross-Language Search and Summarization of Text and Speech (CLSSTS2020)*, pages 26–31, Marseille, France. European Language Resources Association.

Tom Kocmi and Christian Federmann. 2023a. GEMBA-MQM: Detecting translation quality error spans with GPT-4. In *Proceedings of the Eighth Conference on Machine Translation*, Singapore. Association for Computational Linguistics.

Tom Kocmi and Christian Federmann. 2023b. Large language models are state-of-the-art evaluators of translation quality. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 193–203, Tampere, Finland. European Association for Machine Translation.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners.

Surafel M. Lakew, Marcello Federico, Yueru Wang, Cong Hoang, Yash Virkar, Roberto Barra-Chicote, and Róbert Enyedi. 2021. Machine translation verbosity control for automatic dubbing. In *Proceedings of IEEE-ICASSP*.

Surafel M. Lakew, Yogesh Virkar, Prashant Mathur, and Marcello Federico. 2022. Isometric MT: Neural machine translation for automatic dubbing. In *Proceedings of IEEE-ICASSP*.

Mike Lowndes and Aditya Vasudevan. 2021. Market guide for digital commerce search.

Jian-Yun Nie. 2010. Cross-language information retrieval. *Synthesis Lectures on Human Language Technologies*, 3(1):1–125.

Ohad Rozen, David Carmel, Avihai Mejer, Vitaly Mirkis, and Yftah Ziser. 2021. Answering product-questions by utilizing questions from other contextually similar products. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 242–253, Online. Association for Computational Linguistics.

Andreas Rücklé, Krishnkant Swarnkar, and Iryna Gurevych. 2019. Improved cross-lingual question retrieval for community question answering. In *The World Wide Web Conference*, WWW '19, page 3179–3186, New York, NY, USA. Association for Computing Machinery.

Shadi Saleh and Pavel Pecina. 2020. Document translation vs. query translation for cross-lingual information retrieval in the medical domain. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6849–6860, Online. Association for Computational Linguistics.

Fei Sun, Peng Jiang, Hanxiao Sun, Changhua Pei, Wenwu Ou, and Xiaobo Wang. 2018. Multi-source pointer network for product title summarization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 7–16, New York, NY, USA. Association for Computing Machinery.

Haifeng Wang, Hua Wu, Zhongjun He, Liang Huang, and Kenneth Ward Church. 2021. Progress in machine translation. *Engineering*.

Manyi Wang, Tao Zhang, Qijin Chen, and Chengfu Huo. 2020. Selling products by machine: a user-sensitive adversarial training method for short title generation in mobile e-commerce.

Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. 2022. What language model architecture and pretraining objective work best for zero-shot generalization?

Andy Way. 2013. Traditional and emerging use-cases for machine translation. *Proceedings of Translating and the Computer*, 35:12.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022a. Fine-tuned language models are zero-shot learners.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022b. Emergent abilities of large language models.

Joan Xiao and Robert Munro. 2019. Text summarization of product titles. In *eCOM@SIGIR*.

Zijian Yang, Yingbo Gao, Weiyue Wang, and Hermann Ney. 2020. Predicting and using target length in neural machine translation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 389–395, Suzhou, China. Association for Computational Linguistics.

Xueying Zhang, Yunjiang Jiang, Yue Shang, Zhaomeng Cheng, Chi Zhang, Xiaochuan Fan, Yun Xiao, and Bo Long. 2021. DSGPT: Domain-specific generative pre-training of transformers for text generation in e-commerce title and review summarization. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 2146–2150, New York, NY, USA. Association for Computing Machinery.

Mingyang Zhou, Runxiang Cheng, Yong Jae Lee, and Zhou Yu. 2018. A visual attention grounding neural model for multimodal machine translation. *CoRR*, abs/1808.08266.

Turkish Typo Correction for E-Commerce Search Engines

K.Elif Oral, Koray Mancuhan, Hüseyin Varol Erdem, Ece Hatipoğlu
Hepsiburada, Hepsiburada, Hepsiburada, Hepsiburada
{kadriye.oral, koray.mancuhan, varol.erdem, ece.aktan}@hepsiburada.com

Abstract

Typo correction is a challenging problem when it is developed for morphologically rich languages. The existing approaches in the literature are successful mainly for English, leaving the problem open for such languages. This creates an issue, because the typo correction is a critical component in practice for many systems such as search engines. Especially, the search engines of e-commerce platforms rely heavily on typo correction for product relevancy. A bad performing typo corrector could result in very few number of relevant products when a user is looking for a product on an e-commerce platform, resulting in significant revenue decrease. For the first time in the literature, this paper proposes a modern typo corrector for a morphologically rich language, Turkish; which is integrated to the search engine of one of the leading e-commerce platforms in Turkey, Hepsiburada. Our thorough experiments show that this new typo corrector performs very successful in practice, outperforming the existing Turkish specific propositions in the literature; even if it is applied out of the context of the search engines.

Keywords: typo correction, search engines, e-commerce, deep learning

1. Introduction

The search engines play an important role for e-commerce, providing customers an effective tool for purchasing their desired products. As a first step of the search experience, the typo correction is vital in e-commerce; since it directly affects the precision of the search outcomes in terms of the product relevancy. Users often enter the search queries hastily; this leads to unintentional misspellings. Additionally, they may use different input sources (e.g., keyword customization for different languages), which result in completely incorrect search terms. A typo corrector detects and corrects these errors, ensuring that the search results contain the products that the users are intended to find. Thus, it provides the users a much better online shopping experience.

The traditional spelling corrector systems use statistical techniques (Brill and Moore, 2000; Hasan et al., 2015; Li et al., 2012; Gupta et al., 2019) and edit distances (Damerau, 1964; Whitelaw et al., 2009). These methodologies have limitations in addressing the learning issues, when they are faced with increased data sparsity. In recent years, the deep neural models (Ye et al., 2023; Kuznetsov and Urdiales, 2021; Jayanthi et al., 2020; Etoori et al., 2018a) have gained considerable popularity in this field by improving the accuracy despite an important drawback: the increased inference latency due to the model complexity. Although spelling correction is a well-studied field encompassing languages with different characteristics (Liu et al., 2021; Azmi et al., 2019; Duong et al., 2020; Eryiğit and Torunoğlu-Selamet, 2017; Park et al., 2021); the studies, which focus specifically on the morphologically rich languages (MRLs), are still in an immature state. Especially, their applications in e-commerce are still in early stages.

Turkish, a morphologically rich and agglutinative language with the presence of diacritized letters (e.i, çığüö), is prone to spelling errors. This becomes particularly important in the context of e-commerce search. For example, the users face difficulties in spelling the non-Turkish brands (e.g., "Lenova" instead of "Lenovo"); or, use "ciguo" instead of the Turkish letters and vice versa (e.g., İphone instead of Iphone). Furthermore, the agglutinate nature of the Turkish brings an additional challenge. Correcting considering the inflections is crucial, as the absence of suffixes in the search queries impacts products matched at the retrieval level. The overall search relevancy is also affected significantly. For example, we cannot correct the "banyo muslul" as "banyo musluk" instead of the "banyo musluğu" (*bathroom faucet*)[1]); because, (1) there is a degradation in the meaning due to the absence of the genitive marker **I**; and, (2) the search system could bring different products since there may be exact match between the word "musluk" and other product names, under partially relevant categories such as "banyo musluk aksesuarları" (*bathroom faucet accessories*).

We group the Turkish spelling errors, encountered in e-commerce, under two subcategories: (i) the general spelling errors (e.g., fat finger, insertion/deletion.) which can be dealt with applying general solutions proposed for English, (ii) the language specific spelling errors (e.g., missing diacritics, phonetics) which necessitate the specialized treatments. Figure 1 shows the distribution of

[1]The last consonant of the word "musluk" (*faucet*) becomes ğ when a genitive marker **-I** is attached to it, a phenomenon called the consonant lenition.

The Seventh Workshop on e-Commerce and NLP (ECNLP 7), pages 65–73
21 May, 2024.

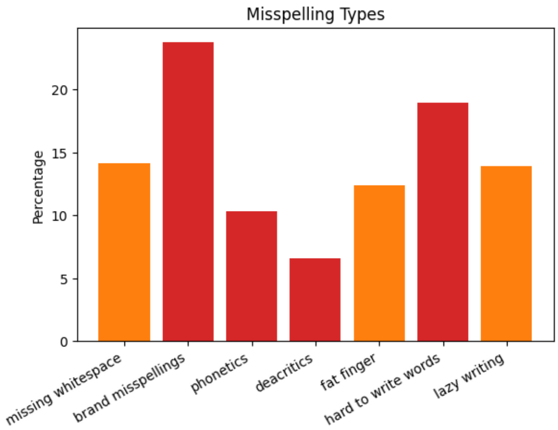


Figure 1: Spelling Error Distribution: The red bars indicate the misspelling types of the language specific spelling errors, the orange ones show the second type of the language specific spelling errors. The misspelling type descriptions are available in the Appendix A.1

.

searches which had bad relevancy due to the misspellings errors [2]. As shown in the figure, 59.58% of total misspellings belong to the latter category.

In the latter subcategory, the proposed approaches, tailored for the languages with limited linguistic characterization, do not fit due to the following factors: (1) high data sparsity due to the rich morphology (a word may have hundreds of different surface forms), (2) substantial edit distances due to the absence of diacritics and phonetic spelling (e.g., the edit distance between the words "başlığı" (*head*) and "basligi" in Turkish is 4 [3]). The deep models are relatively more effective addressing these issues; but, they are not suitable for the search engines due to the high latency costs.

In this paper, we introduce a generalized spelling correction method, which handles the language specific spelling errors along with the general ones, without additional latency. Our simple yet effective method enhances the candidate generation through a morphology-centered approach. The main contributions of our work are:

- A candidate generation method, which relies on using a character level transformer model with low latency; capturing the morphology based relationships between the syntactic word vectors.
- A candidate scoring function that is customized to our morphologically rich language, Turkish.
- A blueprint methodology of forming a training set specific to our morphologically rich language, Turkish.

2. Related Work

The field of spelling correction has evolved from the early work, which was based on the (Wagner and Fischer, 1974) editing distance and the noisy channel model, (Kemighan et al., 2003; Brill and Moore, 2000) to the neural methods based approaches. Early investigations leverage the edit distance and its variations (Wagner and Fischer, 1974); and, the noisy channel model (Brill and Moore, 2000; Kemighan et al., 2003) to rectify the spelling errors. For spelling correction, Sun et al. (2015) propose the convolutional neural networks (CNNs) while Jayanthi et al. (2020) introduce NeuSpell. Ghosh and Kristensson (2017) and Etoori et al. (2018b) extended the neural approach to address the nuanced challenges in the spelling correction tasks. Along with neural models, the spelling correction has been reformulated as a generational task (Zhou et al., 2017; Kuznetsov and Urdiales; Zhang et al., 2019a; Grundkiewicz et al., 2019; Sharma et al., 2023; Zhang et al., 2023).

Spelling correction is a critical component for e-commerce search engines, particularly in the context of e-commerce. Several recent studies have addressed the unique challenges posed by the e-commerce search engines. Yang et al. (2022) propose a generalized spelling correction to address the phonetic errors. Kakkar et al. (2023) and Pande et al. (2022) improve the correction rate on tough spelling mistakes by weakly supervised data. Ye et al. (2023) tailor the pretrained language models for e-commerce search queries.

Non-English languages may need additional treatments for spelling correction due to their specific properties (Zitouni and Sarikaya, 2009; AZMI and ALMAJED, 2015; Liu et al., 2022; Zhang et al., 2019b; Liu et al., 2021). Turkish spelling correction Eryiğit and Torunoğlu-Selamet (2017); Demir and Topcu (2022); Torunoglu-Selamet et al. (2016); Akın and Dündar (2007) has contributed significantly to the field, providing valuable insights and methodologies. Safaya et al. (2022); Koksal et al. (2020) introduced a spelling correction corpus.

3. Spelling Corrector

Our spelling corrector includes the candidate generation and the ranking steps according to the general approach in the literature. For a given input query

[2]We collected 20k queries on our platform and analyzed them based on the search relevancy. We found that 12% of our search results had low relevancy, and 11% of this was due to the misspelling errors.

[3]The last consonant of the word "başlık" (*head*) becomes ğ when a genitive marker **-I** is attached to it due to lenition.

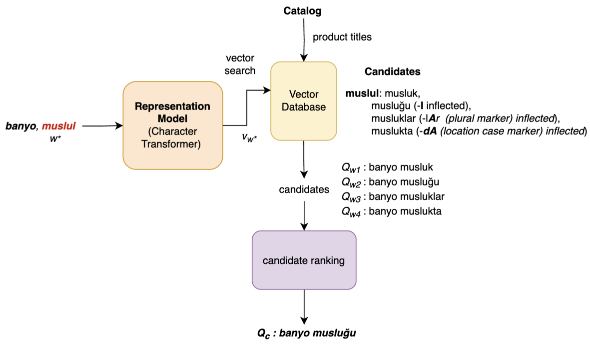


Figure 2: The Correction pipeline

Q_{w^*} with a misspelled word w^*, the spelling corrector generates a set of words $W = \{w_1, w_2, ..., w_n\}$; where w_i is a candidate word that could be the correction of w^*. Then, a query candidate list is formed by replacing w^* with w_i. The corrected Q_c is obtained by the selection of a candidate with the highest score. The score calculation uses the following formula

$$Q_c = w_i argmax(f(\mathrm{Q}_{w_i}|\mathrm{Q}_{w^*})) \quad (1)$$

where *f* is the scoring function that is applied on each query candidate (Q_{w_i}) at the ranking stage. Our correction pipeline is depicted in Figure 2

3.1. Candidate Generation

We frame the candidate generation as a vector search where the words are represented by the vectors based on their syntax. The basic idea of this is that the incorrectly spelled words should be substantially close to their correctly spelled counterparts in terms of syntax. Consequently, obtaining the closest n words, which are based on the syntax-aware vector similarity for a misspelled word, automatically forms a candidate list; including the potential corrections.

Syntax-aware representations have the ability to capture the relationship between the root words and their surface forms. Such an ability enables to locate the inflected words close to their nominative forms. This capability provides an opportunity to correct the words, restoring the missing word inflections. It brings the awareness of the morphology; and, allows more accurate corrections since the inflected forms of the words can be included among the candidates. We train a character level transformer (Vaswani et al., 2017) model that learns the syntax-aware representations with a contrastive learning objective. The model aims to create a vector space for the words by capturing the intrinsic character patterns, which enables the model to discover the language specific rules (e.g., vowel harmony, lenition, phone mapping of character sequence such as 'sq' to 'su', sh to ş, etc.). We avoided complex models due to their latency costs; opted to design the model solely as an encoder layer, incorporating a feed-forward layer with a mean pooler on top of it.

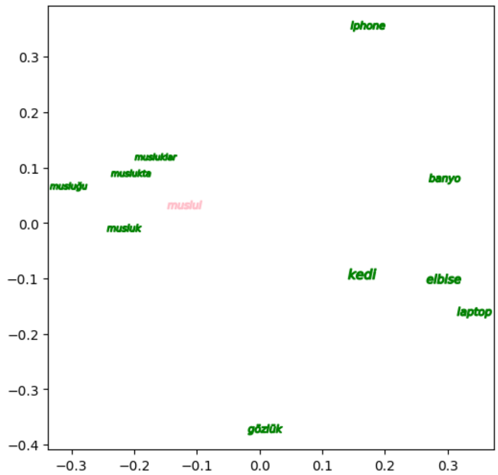


Figure 3: An illustration of our vector space, the similarly spelled words (i.e., word inflections) are represented by the neighbouring vectors.

The candidate generation (illustrated in Figure 2) starts with the extraction of v_{w^*}, which is the vector of w^*. Then, the corresponding n vectors similar to v_{w^*} (e.g. corresponding to n similar words) are retrieved from the database; which holds the vectors of the words extracted from the product titles. Figure 3 illustrates the vector space with a few samples[4]. For example, the query "banyo muslul" has a misspelled word "muslul"; and, its correction should be in the inflected form ("musluğu") rather than the nominative form ("musluk" (*faucet*)). Since the vectors of both ($v_{w_{musluk}}$ and $v_{w_{muslugu}}$) are located close to each other, and to v_{w^*}; we can generate the candidate queries "banyo *musluk*" and "banyo *musluğu*".

3.2. Candidate Ranking

The candidates are ranked using a score, which is calculated by using features obtained from three different sources. Each source represents a different aspect of the candidates:

- The first source (s_1) is the *edit distance* between Q_{w^*} and Q_{w_i}. This prevents Q_{w^*} and Q_c from being completely different queries for the cases the input query has more than one misspelled words.
- The second source (s_2) is the vector similarity.

[4] One should note that this figure is only for illustration purposes, and shows approximate distances.

This regularizes the edit distance by considering the similar syntax aware representations. In particular, it boosts the candidates which are dissimilar in terms of the edit distance. The features from the vector similarity play an important role in the correction of the phonetic and diacritic based misspellings.

- The third source (s_3) is the language score, namely perplexity. In this case, we train two statistical language models: one using the search logs, and the other using the product titles. The features derived from the perplexities validate to some extent the candidates, exhibiting a bias towards the selection of the correct inflections.

Based on these various sources, the final form of the scoring function f is

$$f(\mathrm{Q}_{w_i}|\mathrm{Q}_{w^*}) = \sum_{j=0}^{3} \mathbf{w}_{s_j} * \mathbf{x}_{Q_{w_i}|Q_{w^*} s_j} \tag{2}$$

where s_j is a feature source, $\mathbf{x}_{Q_{w_i}|Q_{w^*} s_j} \in R^n$, $n \in [0, N]$ is a feature vector derived form s_i. $\mathbf{x}_{Q_{w_i}|Q_{w^*}}$ are extracted from the generated candidate Q_{w_i} and the user query Q_{w^*}. $\mathbf{w}_{s_j} \in R^m$, $m \in N$ is a weight vector that is learnt by a regression model.

4. Data Generation

We create the training data from the search logs by leveraging the user interactions (e.g., clicks). All the training data is automatically generated, eliminating the need for human annotations. The training data contains samples ($< w_k, w_j >, l_{kj}$), where $< w_k, w_j >$ is a word pair and l_{kj} is the corresponding label of the pair.

We have two sources of obtaining the positive pairs: the users' feedback and the synthetic data. When l_{kj} is equal to 1, namely positive samples, w_k and w_j are expected to possess similar vectors since w_k represent the misspelled version of w_j. To enhance the syntax awareness, we introduce additional morphology-based $< w, w_{inf} >$ pairs; where w_{inf} denotes the inflection of w. This augmentation aims to ensure that the stems and their inflections have similar vectors by providing the syntactic variations of the words. Consequently, it makes our representations more resilient to the syntactic changes. Negative samples (l_{kj} is equal to 0), on the other hand, are expected to have dissimilar vectors for the words w_k and w_j. They are generated by the outer cross join of the positive pairs.

In the following, we explain the used methodologies to form all the pairs.

4.1. User Feedback Chain

We trace the users' specific action sequences and use their interactions as a feedback to generate the positive samples. Q_{w_k} denotes 'the query with the misspelled word w_k', Q_{w_j} denotes 'the query with the corrected word w_j'; and, P_{w_j} denotes 'the product title containing the word w_j'. We extract the correctly spelled-misspelled word pairs from the Q_{w_k} and P_{w_j} (Q_{w_j}), where we add several check conditions to exclude the drastically different word pairs. The considered action sequences are explained below.

4.1.1. Query Refinement Chain

Users may correct their misspelled queries in the consecutive searches. For example, a user could search Q_{w_k}, then correct it himself/herself and make a subsequent search with Q_{w_j}. We leverage such behaviour sequences and match w_k with w_j to form a pair if and only if the elapsed time between two consecutive searches, Q_{w_k} and Q_{w_j}, is less than 10 minutes; and, the edit distance between them is less than 5.

4.1.2. Fuzzy Match and Click

The retrieval systems may bring relevant products with their fuzzy match abilities, even if the queries have misspelled words. In such cases, the users may still click on the relevant products which are not affected by the incorrect search terms. Assuming that a user searched Q_{w_k}, and subsequently clicked on P_{w_j}; we can extract $< w_k, w_j >$ if the edit distance between w_k and w_j is less than 4.

4.1.3. Click on Suggestion or Rejection

Users' clicks can be considered a reliable source for forming pairs, because they serve as a natural labeling mechanism for our system. Specifically, their clicks on 'did you mean' (suggestion of the corrector) or 'return to the initial query' (rejection) automatically labels the outputs of the spelling corrector. A click on the suggestion indicates a successful acceptance of the correction, meaning that the corrector successfully updates the given query. Conversely, the rejection clicks imply that the corrector is unable to achieve the correction. We collect the users' feedback clicks and use the click rates to filter out the unintentional clicks. One should note that this feedback mechanism, complemented with the other twos, plays a crucial role in gradually improving the spelling corrector's performance; as, the training data is continuously updated with the insights from the unsuccessful corrections. However, this is not initially available.

4.2. Synthetic Data Generation

We generate the synthetic data for two purposes: (1) increase the representation of the most common user mistakes in the training data, (2) form effortlessly the morphology based pairs. For the first purpose, we create the artificial misspellings by

- deacritization: removing deacritics of the Turkish characters, e.g., "yılbaşı çam ağacı" (*Christmas pine tree*) to "yilbasi cam agaci"
- insertion: adding adjacent characters on the keyboard, e.g., "yılbaşı çam ağacı" to "yıolbaşı çam ağascı"
- deletion: removing the random vowels[5], e.g., "yılbaşı çam ağacı" to "ylbaşı çam ağacı"
- replacement: replacing the characters with the ones neighbouring on the keyboard, e.g., "yılbaşı çam ağacı" to "yıkbaşı çam ağacu"
- swapping: swapping the adjacent characters, e.g., "yılbaşı çam ağacı" to "yılbaşı çma ağacı"

The meaning of the words might deteriorate if the excessive distortion is applied. Therefore, we allow two artificial distortions on a given word.

4.3. Morphology-based Pairs Generation

Generating the morphology-based pairs ($< w, w_{inf} >$) with the predefined rules is a challenging task due to the grammar rules of Turkish (e.g., lenition, consonant and vowel harmonies, etc.). We employ the n-gram based word embeddings to generate such kind of pairs, leveraging their ability to approximate the morpheme length. As the n-gram size decreases, these embeddings capture the syntactic variation of the words in the agglutinative languages. This allows us to form the meaningful morphology-based pairs that comply with the rules of the Turkish grammar. We train a Fasttext (Bojanowski et al., 2017) model for these embeddings using our search logs and product titles, because it was shown to capture the morphological variations for Turkish (). Thus, the overall morphology-based pair generation process can be summarized in three steps.

1. We collect a set of correctly spelled high impression queries from the search logs.
2. We build a word vocabulary from the former query set.
3. We generate 20 most similar words for each word (w) within the vocabulary using the Fasttext embeddings. If the edit distance between the generated word and the word w is less than 4, they form the pair $< w, w_{inf} >$.

[5]We observed that the users tend to skip the wovels while writing

5. Experiments

We conducted the evaluations for our corrector in both offline and online experiments. We also did a final error analysis to identify the limitations of our corrector, for determining the future steps.

5.1. Offline Experiments

5.1.1. Accuracy Based Evaluations

The first type of evaluation focused on the correction accuracy of the corrector, which was assessed through the in-domain and out-of-domain settings. The in-domain setting was performed using the data from the e-commerce domain, while the out-of-domain setting was performed using the data prepared for general purposes. This allowed us to compare our corrector with others in the literature and evaluate its performance beyond the specific domain.

We adopt the evaluation metrics precision and recall by defining the following terms:

1. True Positive (TP): The successful misspelling identification and correction cases
2. False Positive (FP): The false misspelling identification and correct word distortion cases
3. False Negative (FN): The misspelling identification and correction failure cases
4. True Negative (TN): The correct identification cases of the correctly spelled words

While the precision represents how many of the corrected queries are successfully corrected, the recall represents how many of the misspelled queries are successfully corrected.

For the in-domain evaluation, we created a gold standard test set; which contains 5.6K queries. Here, the data is collected from both short and long tail queries. Then, the data was manually annotated in two iterations by three native-speaking annotators; who have the domain expertise and the NLP background. In the first iteration, the samples were annotated from scratch separately by two annotators. In the second iteration, the third annotator checked the data quality by validating the labeled samples' accuracy. Our corrector achieves an F1-score of 86% on this data, with a precision of 89% and a recall of 83%.

For the out-of-domain evaluation, we evaluated our corrector on trspell-10 (Safaya et al., 2022). To

Corrector	SCA	F1
HUNSPELL-TR (Zafer, 2017)	25.52	86.52
ZEMBEREK (Akın and Dündar, 2007)	62.12	96.56
Safaya et al. (2022)	71.72	**99.62**
ours	**83.3**	98.18

Table 1: Results on trspell10.

be inline with the mentioned study, we reported the spell correction accuracy (SCA) and the macro-averaged F1 score of misspellings detection. Table 1 shows the results. While our corrector performs relatively close to others in detecting the misspellings, it outperforms them with a significant margin in SCA; improving the number of successfully corrected words.

5.1.2. Search Relevancy Based Evaluations

The second type of evaluation focused on the impact of the corrector on the search relevancy. We established an experimental setup, where we compare the change in Normalized Discounted Cumulative Gain (NDCG) for the same dataset after correction intervention. We selected 1000 queries, of which 14% are misspelled to reflect the overall misspelling rate in our search engine. Firstly, top 12 products for each of these queries were retrieved and annotated with Exact/Substitute/Complement/Irrelevant (ESCI) labels. Using these labels, we calculated the NDCG[6] score which is 90.9. Secondly, we fed all the 1000 queries to our corrector model and repeated the same labeling process. After the correction, the NDCG score reached 91.2; where we improved the score of the misspelled samples from 87.8 to 89.7.

5.2. Online Experiments

To be able to correlate our offline evaluation with the main business metric of our search engine (i.e., Conversion Rate (CR)[7]), we launched an A/B test. Our old in-house model was used in the control bucket, while our new in-house model was used in the treatment bucket. 100% of the users were randomly assigned to both control and treatment buckets. The experiment was run for 4 weeks to ensure the statistically significant results. The results showed that the treatment bucket achieved a CR of 4.84%, while the control bucket had a CR of 4.52%; resulting in a 6.99% CR increase between the buckets.

[6]Weights used in our NDCG calculations are 4, 3, 2, 1 for the labels Exact, Substitute, Complement, Irrelevant; respectively

[7]CR is a metric that represents the percentage of users who have completed a desired action, which is the purchase for our domain.

5.3. Error Analysis

We also made an error analysis to identify the weaknesses of our corrector. One limitation is that our corrector operates at the word level, rather than the sequence level. This results in a sub-optimal performance when the context information is crucial for the corrections. Additionally, it has relatively poor performance in the correction of words that are typically Turkish adaptations of the foreign words, deviating from the Turkish harmonies (e.g., palatal harmony, rounding harmony, etc.)

6. Conclusions and Future Work

In this paper, we proposed a typo corrector which was specific to a morphologically rich language, Turkish. We addressed the data related challenges specific to Turkish; in particular, for training and evaluation data creation. We implemented our approach end-to-end, and integrated it to our search engine. The implementation was evaluated using online and offline experiments. In both offline and online experiments, our corrector outperformed the old default corrector that had been deployed. Moreover, our proposed corrector outperformed the existing best Turkish corrector in the literature; when it was used separately from our search engine.

In future, we are planning to modify the candidate generation of our corrector from the word level model to the sequence level model; so, it captures the context in the corrections. We are also planning to adapt our approach to develop the corrector models in Russian and Arabic.

Acknowledgements

The authors would like to offer special thanks to Feride Duman for her valuable feedback, Zeynep Aycan and Şevval Koca for their assistance during the data annotation, and all search-algorithm team for helpful discussions during the development stage.

7. References

Ahmet Afşın Akın and Mehmet Dündar. 2007. Zemberek, an open source nlp framework for turkic languages.

AQIL M. AZMI and REHAM S. ALMAJED. 2015. A survey of automatic Arabic diacritization techniques. *Natural Language Engineering*, 21(3):477–495.

Aqil M Azmi, Manal N Almutery, and Hatim A Aboalsamh. 2019. Real-word errors in Arabic texts:

A better algorithm for detection and correction. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(8):1308–1320.

Youssef Bassil. 2012. Parallel spell-checking algorithm based on yahoo! n-grams dataset. *ArXiv*, abs/1204.0184.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146.

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th annual meeting of the association for computational linguistics*, pages 286–293.

Qing Chen, Mu Li, and Ming Zhou. 2007. Improving query spelling correction using web search results. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 181–189.

Fred J Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176.

Seniz Demir and Berkay Topcu. 2022. Graph-based turkish text normalization and its impact on noisy text processing. *Engineering Science and Technology, an International Journal*, 35:101192.

Quan Duong, Mika Hämäläinen, and Simon Hengchen. 2020. An unsupervised method for ocr post-correction and spelling normalisation for Finnish. *arXiv preprint arXiv:2011.03502*.

Steffen Eger, Tim vor der Brück, and Alexander Mehler. 2016. A comparison of four character-level string-to-string translation models for (ocr) spelling error correction. *The Prague bulletin of mathematical linguistics*, 105(1):77.

Gülşen Eryiğit and Dilara Torunoğlu-Selamet. 2017. Social media text normalization for turkish. *Natural Language Engineering*, 23(6):835–875.

Pravallika Etoori, Manoj Chinnakotla, and Radhika Mamidi. 2018a. Automatic spelling correction for resource-scarce languages using deep learning. In *Proceedings of ACL 2018, Student Research Workshop*, pages 146–152, Melbourne, Australia. Association for Computational Linguistics.

Pravallika Etoori, Manoj Chinnakotla, and Radhika Mamidi. 2018b. Automatic spelling correction for resource-scarce languages using deep learning. In *Proceedings of ACL 2018, Student Research Workshop*, pages 146–152.

Jianfeng Gao, Chris Quirk, et al. 2010. A large scale ranker-based system for search query spelling correction. In *The 23rd International Conference on Computational Linguistics*.

Shaona Ghosh and Per Ola Kristensson. 2017. Neural networks for text correction and completion in keyboard decoding. *arXiv preprint arXiv:1709.06429*.

Priscila A Gimenes, Norton T Roman, and Ariadne MBR Carvalho. 2015. Spelling error patterns in brazilian portuguese. *Computational Linguistics*, 41(1):175–183.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263.

Jai Gupta, Zhen Qin, Michael Bendersky, and Donald Metzler. 2019. Personalized online spell correction for personal search. In *The World Wide Web Conference*, pages 2785–2791.

Matthias Hagen, Martin Potthast, Marcel Gohsen, Anja Rathgeber, and Benno Stein. 2017. A large-scale query spelling correction corpus. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1261–1264.

Saša Hasan, Carmen Heger, and Saab Mansour. 2015. Spelling correction of user search queries through statistical machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 451–460, Lisbon, Portugal. Association for Computational Linguistics.

Daniel Hládek, Ján Staš, and Matúš Pleva. 2020. Survey of automatic spelling correction. *Electronics*, 9(10):1670.

Sai Muralidhar Jayanthi, Danish Pruthi, and Graham Neubig. 2020. Neuspell: A neural spelling correction toolkit. *arXiv preprint arXiv:2010.11085*.

Vishal Kakkar, Chinmay Sharma, Madhura Pande, and Surender Kumar. 2023. Search query spell correction with weak supervision in e-commerce. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 687–694.

Mark Kemighan, Kenneth Church, and William Gale. 2003. A spelling correction program based on a noisy channel model. 2.

Asiye Tuba Koksal, Ozge Bozal, Emre Yürekli, and Gizem Gezici. 2020. # turkishtweets: A benchmark dataset for turkish text correction. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4190–4198.

A Kuznetsov and H Urdiales. Spelling correction with denoising transformer. arxiv 2021. *arXiv preprint arXiv:2105.05977*.

Alex Kuznetsov and Hector Urdiales. 2021. Spelling correction with denoising transformer. *arXiv preprint arXiv:2105.05977*.

Yanen Li, Huizhong Duan, and ChengXiang Zhai. 2011. Cloudspeller: Spelling correction for search queries by using a unified hidden markov model with web-scale resources. In *Spelling Alteration for Web Search Workshop*, pages 10–14. Citeseer.

Yanen Li, Huizhong Duan, and ChengXiang Zhai. 2012. Cloudspeller: query spelling correction by using a unified hidden markov model with web-scale resources. In *Proceedings of the 21st International Conference on World Wide Web*, pages 561–562.

Shulin Liu, Shengkang Song, Tianchi Yue, Tao Yang, Huihui Cai, TingHao Yu, and Shengli Sun. 2022. Craspell: A contextual typo robust approach to improve Chinese spelling correction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3008–3018.

Shulin Liu, Tao Yang, Tianchi Yue, Feng Zhang, and Di Wang. 2021. Plome: Pre-training with misspelled knowledge for Chinese spelling correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2991–3000.

Madhura Pande, Vishal Kakkar, Manish Bansal, Surender Kumar, Chinmay Sharma, Himanshu Malhotra, and Praneet Mehta. 2022. Learning-to-spell: Weak supervision based query correction in e-commerce search with small strong labels. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 3431–3440.

Chanjun Park, Kuekyeng Kim, YeongWook Yang, Minho Kang, and Heuiseok Lim. 2021. Neural spelling correction: translating incorrect sentences to correct sentences for multimedia. *Multimedia Tools and Applications*, 80:34591–34608.

Martin Reynaert. 2004. Multilingual text induced spelling correction. In *Proceedings of the Workshop on Multilingual Linguistic Resources*, pages 110–117, Geneva, Switzerland. COLING.

Ali Safaya, Emirhan Kurtuluş, Arda Goktogan, and Deniz Yuret. 2022. Mukayese: Turkish NLP strikes back. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 846–863, Dublin, Ireland. Association for Computational Linguistics.

Sanat Sharma, Josep Valls-Vargas, Tracy Holloway King, Francois Guerin, and Chirag Arora. 2023. Contextual multilingual spellchecker for user queries. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 3395–3399, New York, NY, USA. Association for Computing Machinery.

Chengjie Sun, Xiaoqiang Jin, Lei Lin, Yuming Zhao, and Xiaolong Wang. 2015. Convolutional neural networks for correcting english article errors. In *Natural Language Processing and Chinese Computing: 4th CCF Conference, NLPCC 2015, Nanchang, China, October 9-13, 2015, Proceedings 4*, pages 102–110. Springer.

Dilara Torunoglu-Selamet, Eren Bekar, Tugay Ilbay, and Gülsen Eryigit. 2016. Exploring spelling correction approaches for turkish. In *Proceedings of the 1st International Conference on Turkic Computational Linguistics at CICLING, Konya*, pages 7–11.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Robert A Wagner and Michael J Fischer. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173.

Casey Whitelaw, Ben Hutchinson, Grace Y Chung, and Gerard Ellis. 2009. Using the web for language independent spellchecking and autocorrection.

Fan Yang, Alireza Bagheri Garakani, Yifei Teng, Yan Gao, Jia Liu, Jingyuan Deng, and Yi Sun. 2022. Spelling correction using phonetics in E-commerce search. In *Proceedings of the Fifth Workshop on e-Commerce and NLP (ECNLP 5)*, pages 63–67, Dublin, Ireland. Association for Computational Linguistics.

Dezhi Ye, Bowen Tian, Jiabin Fan, Jie Liu, Tianhua Zhou, Xiang Chen, Mingming Li, and Jin

Ma. 2023. Improving query correction using pretrain language model in search engines. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 2999–3008.

Harun Reşit Zafer. 2017. hunspell-tr.

Haoyu Zhang and Qin Zhang. 2017. Embedjoin: Efficient edit similarity joins via embeddings. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 585–594.

Jingfen Zhang, Xuan Guo, Sravan Bodapati, and Christopher Potts. 2023. Multi-teacher distillation for multilingual spelling correction. *arXiv preprint arXiv:2311.11518*.

Shiliang Zhang, Ming Lei, and Zhijie Yan. 2019a. Investigation of transformer based spelling correction model for ctc-based end-to-end mandarin speech recognition. In *Interspeech*, pages 2180–2184.

Shiliang Zhang, Ming Lei, and Zhijie Yan. 2019b. Investigation of Transformer Based Spelling Correction Model for CTC-Based End-to-End Mandarin Speech Recognition. In *Proc. Interspeech 2019*, pages 2180–2184.

Yingbo Zhou, Utkarsh Porwal, and Roberto Konow. 2017. Spelling correction as a foreign language. *arXiv preprint arXiv:1705.07371*.

Imed Zitouni and Ruhi Sarikaya. 2009. Arabic diacritic restoration approach based on maximum entropy models. *Computer Speech Language*, 23(3):257–276.

A. Appendix

A.1. Misspelling Types

- **missing whitespace**: it appears when there is an missing space in the user query, e.g., "banyomusluğu" instead of "banyo musluğu"
- **brand misspellings**: it appears when brand names have a typo, which may occur since users may not know the correct spelling of the non-Turkish brands., e.g., "lenova" instead of "lenovo"
- **phonetic**: it appears when the non-Turkish words are written using their Turkish phones., e.g., "iyfon" instead of "iPhone"
- **missing diacratics**: it appears when Turkish specific letters (ö,ı,ü,ç,ş,ğ) are replaced with their english counterparts., e.g., "banyo muslugu" instead of "banyo musluğu"
- **fat finger**: it appears when keyboard an input mistake occurs, e.g., "bsnyo muskuğı" instead of "banyo musluğu"
- **hard-to-write words**: It appears when users does not know the correct spelling of the words adapted from other languages (e.g., french, persian, arabic) through time, e.g., "hoporlör" instead of " hoparlör"
- **lazy writing**: It appears when there is a missing or extra letter in the user query, e.g., "baanyo msluğu" instead of "banyo musluğu"

A.2. Training Setup & Inference

We trained the transformer model from scratch, employing 8 attention heads with dimensions of 128 for hidden layers and 64 for output layers. The training process utilized the Adam Optimizer and Cosine learning rate scheduler with warm-up. The learning rate was set to 0.001, and $\beta 1$ and $\beta 2$ were configured as 0.8 and 0.998, respectively. The sequence length was determined to be 19, considering the average character length of words within the search queries. For the training phase, we utilized an NVIDIA T4 GPU. It took 29 hours to complete. The inference phase was executed on a CPU setup. The response time <17.3 ms query on average under single concurrency.

Detecting AI-enhanced Opinion Spambots: a study on LLM-generated Hotel Reviews

Vijini Liyanage*, **Davide Buscaldi***, **Pénelope Forcioli**†
*University Sorbonne Paris Nord
93430 Villetaneuse, France
{liyanage, buscaldi}@lipn.univ-paris13.fr
†Ecole Polytechnique
91120 Palaiseau, France
penelope.forcioli@polytechnique.edu

Abstract

Opinion spamming is the posting of fake opinions or reviews to promote or discredit target products, services, or individuals. The concern surrounding this activity has grown steadily especially because of the development of automated bots for this purpose ("spambots"). Nowadays, Large Language Models (LLMs) have proved their ability to generate text that is almost indistinguishable from human-written text. Therefore, there is a growing concern regarding the use of these models for malicious purposes, among them opinion spamming. In this paper, we carry out a study on LLM-generated reviews, in particular hotel reviews as we chose the well-known Opinion Spam corpus by Myle Ott as the seed for our dataset. We generated a set of fake reviews with various models and applied different classification algorithms to verify how difficult is it to detect this kind of generated content. The results show that by providing enough training data, it is not difficult to detect the fake reviews generated by such models, as they tend to associate the aspects in the reviews with the same attributes.

Keywords: artificially generated text, detection, classification, opinion spam

1. Introduction

Opinion spamming encompasses the dissemination of counterfeit reviews or opinions for the purpose of promoting or discrediting products, services, or individuals (Liu, 2012). Fake reviews can be generated either by humans or by using text-generating algorithms to automate the process. Historically, the detection of automated spamming has been relatively straightforward, largely due to the mechanical and less expressive nature of machine-generated text compared to human-authored content. Nonetheless, the advent of Large Language Models allowed for a paradigm shift, as these models have demonstrated a remarkable capability to produce text that closely mimics human writing. Consequently, there is a growing unease surrounding the potential misuse of these advanced models, one of which involves their deployment in opinion spamming.

Numerous studies have leveraged Natural Language Processing (NLP) techniques to detect fake reviews. Researchers have explored sentiment analysis, textual patterns, and linguistic features to distinguish between genuine and artificially generated content. (Martinez-Torres and Toral, 2019) successfully employed machine learning methods for sentiment analysis to identify deceptive hotel reviews. (Elmogy et al., 2021) utilized supervised machine learning classifiers, including Random Forest and Support Vector Machines (SVM), to classify fake hotel reviews. They demonstrated the effectiveness of these algorithms in achieving high precision and recall rates.

Recent advancements in synthetic text generation models, in particular the Generative Pretrained Transformer (GPT) family of language models (Radford et al., 2019), have introduced new challenges in fake review detection. Researchers have begun to adapt their detection methods to identify reviews generated by these sophisticated models. One of the first works to address this problem is the one by (Salminen et al., 2022). In their work, they found out that human accuracy in detecting fake reviews is only slightly higher than random chance, and that when applying text-based fake-review detection, the more words a review has, the higher the chance of detecting its true label (fake or real).

In this paper, we tackle this problem from an NLP perspective to understand what are the linguistic features that allow text-based classification models to distinguish between generated and original text. We explain our approach in building corpora composed of artificially generated hotel reviews leveraging smaller Large Language Models (LLMs), such as GPT-2, GPT-3, and TinyLLama. In our opinion, this would match the choice of malicious spammers, as these models do not require demanding hardware and produce tokens at a fast rate. After some preliminary tests, we discarded ChatGPT because on one hand it refused to follow the instruction and on the other one, when it did, it produced the same review over and over. Besides, we evaluate the detectability of the generated contents by employing

The Seventh Workshop on e-Commerce and NLP (ECNLP 7), pages 74–78

statistical as well as deep learning-based classification models.

2. Datasets

As a set of original “seed" reviews to reproduce with the LLMs we used 400 truthful positive reviews from TripAdvisor and 400 truthful negative reviews from Expedia, Hotels.com, Orbitz, Priceline, TripAdvisor and Yelp which compose the “truthful" subset of the well-known Myle Ott’s Opinion Spam corpus (Ott et al., 2011). These reviews were written for 20 Chicago hotels: each hotel has 20 positive and 20 negative reviews.

Initially, we used the GPT-3 DaVinci model to generate for each of the original reviews a faked equivalent one. We fed to the model the prompt “Generate a {%polarity} review completing the following text:", where {%polarity} is replaced either by “negative" or “positive", and the sequence composed by the first 10 words of the genuine review and asked to complete and generate text up to the same length of the original one. We used temperature 0.9 to ensure a certain creativity by the model. Similarly, we created a set of reviews using the TinyLlama model (Zhang et al., 2024), one of the latest and smallest available LLMs.

Subsequently, we created another set with the objective of creating reviews that were more similar to the human-written ones. A way to do this is to fine-tune the language model with the set of genuine reviews. In this way, the model will tend to generate text that fits better the probability distribution of the tokens in the genuine set. Since we didn’t have the resources to fine-tune GPT-3 we switched to GPT-2 with 355M parameters. Again we fed a sequence of 10 words as seed text, including the hotel name.

Finally, following the counter-detection strategy applied by (Sadasivan et al., 2023), we built a final set by using paraphrasing. The paraphrasing model chosen was a T5 model fine-tuned on chatGPT-generated paraphrases available on Huggingface [1]. For each original review, we generate 5 different paraphrases and we randomly pick one of them. With this strategy, we paraphrased the reviews from the GPT-3 model and obtained another set of 800 fake reviews.

Below are examples of negative reviews referring to the same example, the original one and the generated ones:

- **Original review:** This hotel must have originally been an ordinary downtown hotel in a not very nice neighborhood. The owners have painted it over and fixed up the lobby to make it an "upscale" overpriced boutique hotel. But it’s still just an ordinary hotel with a paint job. Stay here if you want to experience staying in a flophouse, but with all the sleaze removed. Definitely stay at the chain’s other location, the Burnham, for a much, much nicer hotel.

- **Fine-tuned GPT-2 generated review:** We stayed in hotel Allegro and it was bad. Although the rooms are decent, they are not called “The Best in a Hilliard" and are not suited with insole. The bathroom was very small and has anearth and mineral salts. The pipes and sinks were old and the shower and tub are in need of a major overhaul. The hotel charges for internet and/or rental for 5 internet sessions a day. If you sign up for wireless internet, they do not fund wireless internet so slow and unfavorable wireless speeds.

- **Pre-trained GPT-3 generated review:** This hotel must have originally been an ordinary downtown hotel but now it is hardly more than a shabby motel. The walls are dirty and the furniture is worn, with some broken pieces. My room smelled musty and the carpets had stains. The bed was uncomfortable and the linen was old and threadbare. The staff were not particularly helpful and I felt like I was an inconvenience to them.

- **TinyLlama generated review:** This hotel must have originally been an ordinary downtown hotel. The rooms are a little too tiny and they lack basic amenities, including a pool. However, the location is very convenient as it is just a few minutes walk from Central Park.

- **Paraphrased GPT-3 review:** I’m sure the hotel was once a regular downtown hangout, but it’s now resembles ruins. The walls are unclean, the furniture is old and scratchy, there are stains in every room, an uncomfortable bed, and lifeless linens.

The datasets are available at the following repository: https://github.com/dbuscaldi/op_spam_AI.

In Table 1 we show the average perplexities for each of the datasets. Perplexity is defined as:

$$PPL(X) = \exp\{-\frac{1}{t}\sum_{i}^{t}\log p_\theta(x_i|x_{<i})\}$$

where $X = (x_0, \dots, x_t)$ is the input sentence and p_θ is the estimated probability by a LLM (in our case, we chose the GPT-2 probabilities, as in the GLTR paper (Gehrmann et al., 2019)). Perplexity is used

[1] https://huggingface.co/humarin/chatgpt_paraphraser_on_T5_base

by some commercial detectors, notably GPTZero[2], as a feature to detect whether a text is generated or not. The principle of perplexity is that it measures the “randomness" of a text sequence. For instance, a perplexity of 3 means that after every word in the text, there are on average 3 choices to continue the sentence. The perplexity values obtained on our datasets show that GPT-3 pre-trained has the lowest variability among all generated ones. On the other hand, we can see that the TinyLlama and the paraphrase one have higher perplexity, confirming the hypothesis that paraphrasing and changing the generating model may help confuse detectors that are based on a specific model.

3. Experiments and Results

To evaluate the detectability of the fake reviews, we employed a variety of classification models. First of all, we started with a basic Multinomial Naïve Bayes with tf.idf weights, without lemmatization and stopwords removal. We evaluated the model on a random split of 80:20 for training and testing. The results with this model were already very good, obtaining a F-1 score of 96%, indicating that the task could be solved just by looking at the vocabulary. Therefore, we compared the log-probabilities of the words in the generated and non-generated class, calculating their difference. We show in Table 2 the 10 most discriminating words for both classes.

It can be observed how the most discriminating words for the generated category tend to be attributes (“unhelpful", “terrible", “delicious", “outdated", ...) while the ones for the non-generated category seem more related to objects or places (“door", “floor", “coffee", “michigan", “ave", ...) and personal pronouns (“she", “he", “your", ...). Similar results are obtained with TinyLlama, with some variations on the attributes (“stunning" is more prevalent among generated texts). We tried similar experiments with bi-grams and tri-grams as features instead of words and this difference in style is even more clear: in the most important trigrams for the generated class we find tri-grams that match a pattern “X was/were Y" where X is usually a service or an aspect of the hotel and Y an adjective. In the non-generated most representative tri-grams we find tri-grams such as “in the room", “in the bathroom", “the first night". The difference in style was expected as other works about generated text detection (Antoun et al., 2023) have noticed the tendency of LLMs to produce recurrent patterns in the output.

To verify the importance of vocabulary overlap for detection, we carried out an experiment in which we varied the proportion of training and test data. Note that in a realistic scenario training data would not be balanced, as annotated corpora have sizes that are only a small fraction of the total number of reviews on platforms. We carried out 10 experiments for each proportion of test and training data. The results, separating recall and precision for each class, are presented in Figure 1.

As can be seen, the precision for non-generated texts tends to be lower than for generated texts, while the recall shows the inverse. This indicates the presence of many false negative examples, i.e. the model is prone to classify machine-generated text as human-written, especially when training data are few. This phenomenon has also been observed by (Wang et al., 2023) in experiments on cross-domain classification. Scores for TinyLlama are lower than for GPT-3, indicating that these generated texts are more difficult to detect.

If we take into account the paraphrased corpus, both precision and recall are high, but the accuracy of the detection is more sensible to the availability of training data, as can be seen in Figure 2. Therefore paraphrasing makes it slightly more difficult to detect fake reviews when there is not enough training data.

Finally, we made some experiment with state-of-the-art classification algorithms based on transformers models, specifically BERT$_{base}$, SciBERT, XLNet$_{large}$, and ELECTRA$_{small}$. These models were imported as pre-trained models from Hugging Face (Wolf et al., 2020) and fine-tuned using Simple Transformers [3]. We employed the BERT tokenizer across all models. The fine-tuning process included 10 epochs, a batch size of 16, and a maximum sequence length of 128. For standalone models, we used unprocessed text as input.

The dataset was split into an 80:20 ratio for training and testing. Each model underwent three experimental iterations, and the average F1 scores resulting from these experiments are provided in Table 3. Among all detection models, BERT$_{base}$ seems to be the most effective in detecting the generated content. GPT-2 reviews are the least predictable, given the fine-tuning process that made them more similar to the human-written ones. The pre-trained models seem rather easy to detect with any of the Transformer-based models.

4. Conclusions

In this work, we created various collections of automatically generated reviews for automated detection, based on the Opinion Spam corpus by (Ott et al., 2011). The results show that the vocabulary and style of generated reviews is very different from the one used in the authentic ones, making it relatively easy to detect the fake ones, provided

[2] https://gptzero.me

[3] https://simpletransformers.ai

original	GPT-2 fine-tuned	GPT-3 pretrained	paraphrased	TinyLlama
60.433	26.401	20.102	32.828	43.820

Table 1: Average perplexity for each collection.

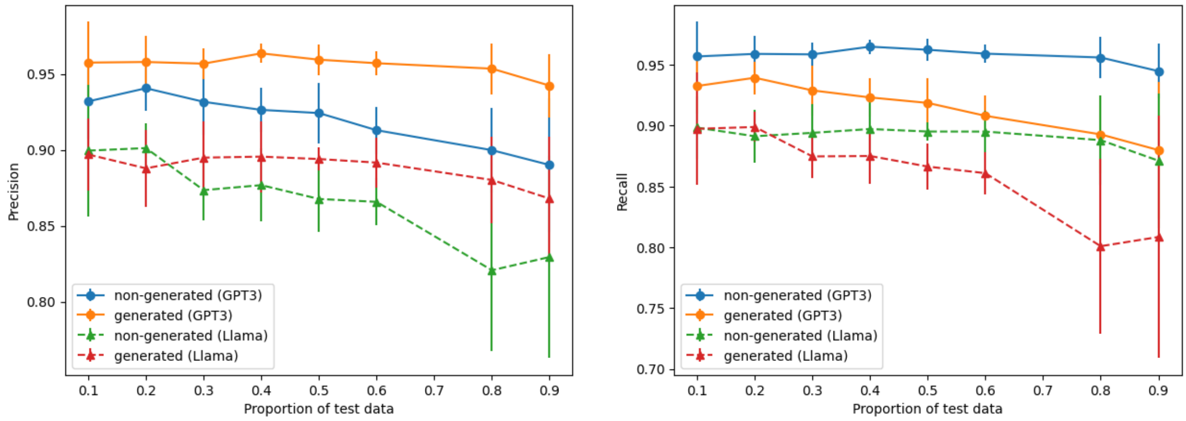


Figure 1: Precision and recall for each class on the GPT3 dataset vs. original reviews, and TinyLlama vs. original reviews, varying the proportion of test and training data. The error bar indicates the standard deviation calculated over 10 experiments.

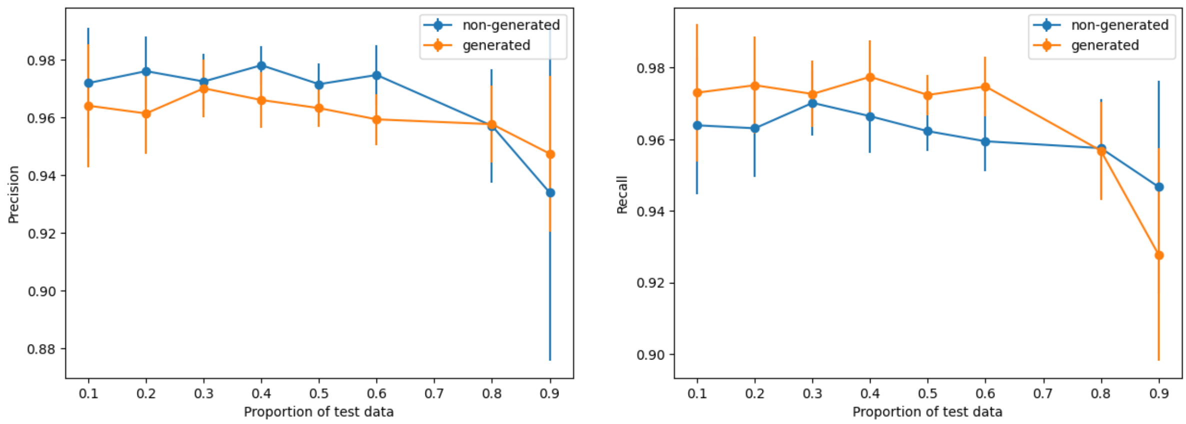


Figure 2: Precision and recall for each class on the paraphrased dataset vs. the original reviews, varying the proportion of test and training data.

Generated		Authentic	
Word	**delta**	**Word**	**delta**
unhelpful	2.889	door	-1.694
incredibly	2.620	floor	-1.680
delicious	2.609	coffee	-1.656
outdated	2.402	next	-1.636
terrible	2.306	your	-1.557
accommodating	2.257	concierge	-1.513
anyone	2.229	she	-1.478
uncomfortable	2.129	ave	-1.468
amenities	1.907	mile	-1.420
musty	1.873	etc	-1.402

Table 2: The 10 most discriminating words for each category (GPT-3 dataset) sorted by their log-probability difference (delta).

Model	GPT-2	GPT-3	para	Llama
BERT$_{base}$	97.83	99.38	98.29	99.74
SciBERT	93.66	93.75	97.62	99.35
XLNet$_{large}$	87.87	92.70	95.32	98.57
ELECTRA	92.49	93.49	95.37	99.34

Table 3: F1 Scores obtained by Transformer-based Classification Models. "para" indicates the paraphrased corpus.

that a large and varied enough training data set is available.

These results are partially comforting as it looks like it is not possible to use LLMs to automatically produce undetectable fake reviews without the intervention of a human, lowering the harm potential of these models. We didn't test how often neural hallucinations occur in these reviews, but after inspection we could observe that some reviews mention features that are not present in the targeted hotel. For future works, we plan to extend our tests with models with higher temperatures and to measure the hallucination phenomenon.

Acknowledgements

This work is supported/ partially supported by a public grant overseen by the IdEx Université Paris Cité (ANR-18-IDEX-0001) as part of the Labex Empirical Foundations of Linguistics - EFL

Bibliographical References

Wissam Antoun, Virginie Mouilleron, Benoît Sagot, and Djamé Seddah. 2023. Towards a robust detection of language model generated text: Is chatgpt that easy to detect?

Ahmed M Elmogy, Usman Tariq, Mohammed Ammar, and Atef Ibrahim. 2021. Fake reviews detection using supervised machine learning. *International Journal of Advanced Computer Science and Applications*, 12(1).

Sebastian Gehrmann, Hendrik Strobelt, and Alexander Rush. 2019. GLTR: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116, Florence, Italy. Association for Computational Linguistics.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of the 2008 international conference on web search and data mining*, pages 219–230.

Bing Liu. 2012. Opinion spam detection. In *Sentiment Analysis and Opinion Mining*, pages 113–125. Springer.

Maria del Rocío Martinez-Torres and Sergio L Toral. 2019. A machine learning approach for the identification of the deceptive reviews in the hospitality sector using unique attributes and sentiment orientation. *Tourism Management*, 75:393–403.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. *arXiv preprint arXiv:1107.4557*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can ai-generated text be reliably detected?

Joni Salminen, Chandrashekhar Kandpal, Ahmed Mohamed Kamel, Soon-gyo Jung, and Bernard J Jansen. 2022. Creating and detecting fake reviews of online products. *Journal of Retailing and Consumer Services*, 64:102771.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Alham Fikri Aji, and Preslav Nakov. 2023. M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model.

Assessing Image-Captioning Models: A Novel Framework Integrating Statistical Analysis and Metric Patterns

Qiaomu Li, Ying Xie, Nina Grundlingh, Varsha Rani Chawan, Cody Wang
Kennesaw State University, The Home Depot
Marietta, GA; Smyrna, GA
{qli12, ngrundli}@students.kennesaw.edu, yxie2@ kennesaw.edu
{varsha_rani_chawan, cody_wang1}@homedepot.com

Abstract
In this study, we present a novel evaluation framework for image-captioning models that integrate statistical analysis with common evaluation metrics, utilizing two popular datasets, FashionGen and Amazon, with contrasting dataset variation to evaluate four models: Video-LLaVa, BLIP, CoCa and ViT-GPT2. Our approach not only reveals the comparative strengths of models, offering insights into their adaptability and applicability in real-world scenarios but also contributes to the field by providing a comprehensive evaluation method that considers both statistical significance and practical relevance to guide the selection of models for specific applications. Specifically, we propose Rank Score as a new evaluation metric that is designed for e-commerce image search applications and employ CLIP Score to quantify dataset variation to offer a holistic view of model performance.

Keywords: image-captioning model, image-based search, evaluation metric

1. Introduction

Image-captioning, the process of generating descriptive textual summaries for visual content, has emerged as an important AI capability with applications such as providing context for visually impaired users, automated alt-text generation, and enhanced image search. However, rigorously evaluating image-captioning models remains challenging. Traditionally, evaluating these models has involved using several evaluation metrics to score their performance, followed by comparing which model leads in these metrics to determine superiority.

Besides, as is known to all, each evaluation metric has its own focus, only a model that matches an evaluation metric's preference could get high scores. But it's increasingly clear that no single model could consistently outperform others across all metrics and datasets. This divergence highlights the challenge in declaring one model definitively superior to others. On the other hand, now that a single model cannot win all, we could only identify each model's comparative strengths when evaluating several models simultaneously, acknowledging each one's pros and cons. This full understanding is essential, as different application scenarios may require different strengths, making it imperative to match a model from several potential one to the situation where it's comparatively more suitable.

Furthermore, the choice of benchmark dataset influences evaluation outcomes. To take advantage of model potential, we generally need to finetune or train models on the target dataset. However, in practical situations, image-caption alignment can diverge across datasets, influencing final results. We account for this by employing CLIP Score to quantify dataset variation, that is, the overall alignment between images and corresponding captions within the dataset. By measuring dataset variation, we gain insights into dataset complexity and noise levels. This allows us to deduce model ability based on dataset qualities, and then figure out comparative model strengths.

In this paper, we propose an integrated evaluation framework that combines the statistical analysis of various metrics to identify models with comparative strengths. By analyzing statistical significance across metric results, we reveal relative advantages of models and suitable applications based on evaluation metric patterns. We summarize our primary contributions as follows:

- We utilize CLIP Score to assess dataset variation in overall image-caption alignment, providing a basis for model evaluation.
- We come up with a novel evaluation framework that merges statistical significance with reverse reasoning from metric patterns, extracting comparative model strengths.
- We introduce a novel Rank Score metric, a simple yet powerful metric to evaluate image-captioning models by assessing generated text quality through comparative ranking against reference captions.

2. Related Work

Recent years have seen diverse image-captioning models developed based on generative techniques. Representative examples include Video-LLaVa (Lin et al., 2022), BLIP (Li et al., 2021), CoCa (Yao et al., 2021), and ViT-GPT2 (Kumar, 2022). Among these, Video-LLaVa

The Seventh Workshop on e-Commerce and NLP (ECNLP 7), pages 79–87

extends language models to video for dynamic content understanding, BLIP uses bootstrapped pre-training for improved visual-language synergy, CoCa utilizes cross-modal contrastive learning to enhance image understanding and caption generation, and ViT-GPT2 combines Vision Transformer (ViT) (Dosovitskiy et al., 2021) and GPT-2(Radford, Alec, et al., 2019) for efficient image and text processing. While adopting different approaches, rigorous comparative evaluation is needed to reveal the comparative strengths of these models to match them to suitable applications.

A range of automated metrics have been proposed for evaluating image-captioning models by comparing candidate captions to references. Popular metrics include BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), CIDEr (Vedantam et al., 2015), (Anderson et al., 2016), and METEOR (Banerjee & Lavie, 2005). These measures rely on lexical, grammatical and semantic similarities. CLIP Score (Hessel et al., 2021) supplements these by comparing captions directly to images. However, no single model consistently outperforms across all metrics.

Our hypothesis is that different models may exhibit comparative strengths aligning with particular use cases based on precision, efficiency, adaptability etc. This work provides a comprehensive framework combining statistical analysis and tailored datasets to reveal model capabilities, guiding selection for applications using different metrics.

3. Methodology

The methodology involves using CLIP Score to quantify dataset variation based on the alignment between images and captions of each data set. For model evaluation, a set of metrics including BERT Score, BART Score, METEOR, SPICE, BLEU, CLIP Score, and the proposed Rank Score are applied on the output of each image-captioning model on each data set. Statistical analysis using paired t-tests with Bonferroni correction is then conducted on the evaluation results to identify models with comparative strengths based on statistically significant performance. The preferences of metrics are analyzed to infer model capabilities. By combining statistical significance with reasoning from evaluation patterns, the framework identifies specialized strengths of models and their suitable applications.

3.1 Evaluation Metrics and Preferences

3.1.1 BLEU (Bilingual Evaluation Understudy)

BLEU measures the precision of n-grams in the generated text compared to the reference texts, adjusting for the proper length and penalizing overly short translations (Papineni et al., 2002). It does this by calculating the n-gram precision for several n-gram lengths (usually 1 to 4) and then combining these precisions geometrically, applying a brevity penalty for translations that are too short.

The BLEU is calculated as:

- *N-gram Precision (P_n): For each n-gram length (n=1 to 4), calculate the count of n-grams in the candidate translation that appear in any reference translation, divided by the count of all n-grams in the candidate translation.*
- *Brevity Penalty (BP): To penalize short machine-generated translations, a brevity penalty is applied. If the length of the candidate translation is less than the effective reference corpus length, the brevity penalty applies:*

$$BP = \begin{cases} 1 & if\ c > r \\ e^{(1-\frac{r}{c})} & if\ c \leq r \end{cases}$$

$$c\text{: } Length\ of\ the\ candidate\ translation$$

$$r\text{: } Length\ of\ the\ effective\ reference\ corpus$$

$$BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} w_n log P_n\right)$$

$$w_n\text{: } weights\ for\ each\ n-gram\ precision$$

$$P_n\text{: } the\ precision\ for\ n-grams$$

$$N\text{: } the\ maximum\ n-gram\ length$$

Preferences: BLEU calculates the score by matching the n-grams of the candidate text with the reference texts and applying a brevity penalty for short candidate texts. This method, focusing on surface lexical matches without considering semantic context or synonyms, inherently favors models that are excellent at producing exact n-gram matches with the reference texts.

3.1.2 METEOR (Metric for Evaluation of Translation with Explicit Ordering)

Meteor is a machine translation evaluation metric, which is calculated based on the harmonic mean of precision and recall, with recall weighted more than precision (Banerjee & Lavie, 2005).

The METEOR is calculated as:

F-Score: The harmonic mean of Precision and Recall, given more importance to recall. It's calculated as:

$$F_{score} = \frac{10 \cdot P \cdot R}{R + 9 \cdot P}$$

Penalty: A penalty is applied for poor word order, computed based on the largest common subsequence of matched words between the candidate and the reference:

$$Penalty = 0.5 \cdot \left(\frac{number\ of\ chunks}{number\ of\ unigrams\ matched}\right)^3$$

$$METEOR = F_{score} \cdot (1 - Penalty)$$

Preferences: METEOR assesses translations by accounting for exact word matches, synonyms, stemming, and word order, calculating a harmonic mean of precision and recall adjusted for these factors. This approach indicates a preference for models that understand and utilize linguistic nuances, including synonymy and grammatical structure.

3.1.3 ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

ROUGE is a set of metrics used for evaluating automatic summarization and machine translation software in natural language processing (Lin, 2004). It works by comparing an automatically produced summary or translation against a reference or a set of reference summaries (typically human-produced). It includes several variants, such as ROUGE-N, ROUGE-L, and ROUGE-W, each focusing on different aspects of the comparison:

ROUGE-N measures the overlap of n-grams between the system-generated text and the reference texts. It is defined as:

$$ROUGE-N = \frac{\sum_{s\in\{Reference\ Summaries\}}\sum_{gram_n\in s}Count_{match}(gram_n)}{\sum_{s\in\{Reference\ Summaries\}}\sum_{gram_n\in s}Count(gram_n)}$$

$Count_{match}(gram_n)$: *max number of n-grams co-occurring in a candidate summary and and reference summary.*
$Count(gram_n)$: *the count of n-grams in the reference summaries.*

ROUGE-L measures the longest common subsequence (LCS) between the system-generated summary and the reference summaries. It considers sentence-level structure similarity naturally and identifies the longest co-occurring in-sequence n-grams of words. ROUGE-L is defined as:

$$ROUGE-L = \frac{(1+\beta^2)\cdot Recall_{LCS}\cdot Precision_{LCS}}{Recall_{LCS}+\beta^2\cdot Precision_{LCS}}$$

Where:

$$Recall_{LCS} = \frac{LCS(System\ Summary, Reference\ Summary)}{Length(Reference\ Summary)}$$

$$Precision_{LCS} = \frac{LCS(System\ Summary, Reference\ Summary)}{Length(System\ Summary)}$$

β *is the set to favor recall*$(e.g., \beta^2 = 1.2)$, *because recall is more important.*

ROUGE-W is based on the weighted longest common subsequence, which considers the length of the LCS and the gaps between consecutive LCS matches. It is defined as:

$$ROUGE-W = \frac{(1+\beta^2)\cdot Recall_{WLCS}\cdot Precision_{WLCS}}{Recall_{WLCS}+\beta^2\cdot Precision_{WLCS}}$$

Where:

$Recall_{WLCS}$: *weighted* $Recall_{LCS}$
$Precision_{WLCS}$: *weighted* $Precision_{WLCS}$

Preferences: ROUGE measures the overlap of n-grams and the longest common subsequences between the generated text and reference texts, primarily focusing on recall. This metric's emphasis on recall over precision suggests a preference for models that ensure no information is lost in summarization, even if it leads to less concise outputs. Therefore, models that excel under ROUGE are those capable of capturing the breadth of content in reference texts, making them particularly suitable for summarization tasks where the completeness and coverage of the source material are paramount, rather than stylistic conciseness or linguistic innovation.

3.1.4 CIDEr (Consensus-based Image Description Evaluation)

CIDEr is a metric used to evaluate the quality of generated textual descriptions of images (Vedantam et al., 2015). It measures the similarity between a generated caption and the reference captions.

The CIDEr is calculated as:

$$CIDEr(c_i, S_i) = \sum_{n=1}^{N} w_n \cdot sim_n(c_i, S_i)$$

Where:

c_i: *The candidate caption for image i.*
S_i: *A set of reference captions for image i*
N: *Max n − gram length.*
w_n: *The weight of each n − gram length.*
sim_n: *The cosine similarity between the tf-idf weighted n-gram vectors of the candidate caption and the reference captions.*

Preferences: CIDEr evaluates image-captioning quality by calculating the consensus between a candidate caption and reference captions using term frequency-inverse document frequency (TF-IDF) weighting for n-grams. This approach emphasizes the importance of unique and descriptive terms that are relevant to the image, favoring models that generate detailed and image-specific captions.

3.1.5 SPICE (Semantic Propositional Image Caption Evaluation)

SPICE is an automated caption evaluation metric that uses scene graphs to measure the semantic similarity of reference and candidate captions (Anderson et al., 2016). The SPICE score is calculated as the F1 score between the sets of tuples extracted from the candidate and reference captions' scene graphs. The tuples represent objects, attributes, and relations.

The SPICE is calculated as:

$$SPICE = 2\cdot\frac{P\cdot R}{P+R}$$

P: *Proportion of tuples in the candidate caption that also appear in the reference captions.*
R: *Proportion of tuples in reference captions that are captured in the candidate caption.*

Preferences: SPICE evaluates image captions based on semantic fidelity, comparing structured scene graphs derived from candidate and reference captions to assess the presence and accuracy of depicted objects, attributes, and relationships. This focus on semantic content leads SPICE to favor models adept at deep visual understanding and generating captions that accurately reflect complex visual scenes in natural language.

3.1.6 BERT Score

BERT Score evaluates the quality of text by calculating the cosine similarity between the BERT embeddings (Devlin et al., 2019) of the candidate text and the reference text (Zhang et al., 2020).

The BERT Score is calculated as:

$$BERT\ Score = 2 \cdot \frac{P \cdot R}{P + R}$$

$$P(Precision) = \frac{1}{|C|} \Sigma_{c \in C} max_{r \in R} cosine(c, r)$$

$$R(Recall) = \frac{1}{|R|} \Sigma_{r \in R} max_{c \in C} cosine(r, c)$$

$$C: Cadidate\ text; \quad R: Reference\ text$$

Preferences: BERT Score leverages the contextual embeddings from BERT model to compare the semantic similarity between candidate and reference texts, focusing on the match at a deeper semantic level rather than surface lexical similarity. This metric's design prefers models that generate contextually rich and semantically accurate text, reflecting a deep understanding of language nuances.

3.1.7 BART Score

BART Score, which stands for BLEU Artifact Reduction Test score, is a metric for evaluating the quality of text generation models (Yuan et al., 2021). It focuses on measuring how well a model's generated text preserves the factual content and overall meaning of a reference text.

The BART Score is calculated as:

$$BART\ Score = w_1 \cdot BLEU\text{-}4 + w_2 \cdot ROUGE\text{-}L + w_3 \cdot BERT\ Embedding\ Similarity + w_4 \cdot Factual\ Consistency + w_5 \cdot Informativeness;$$

$w_1 \sim w_5$: $weighted\ score\ to\ each\ component,\ generally\ take\ the\ average.$

BLEU-4: *This component measures n-gram overlap between the generated and reference texts, similar to traditional BLEU score.*

ROUGE-L: *This part assesses the longest matching subsequences between the texts, capturing more meaningful phrases.*

BERT Embedding Similarity: *This measures the semantic similarity between the generated and reference texts using pre-trained language models like BERT.*

Factual Consistency: This component analyzes the factual inconsistencies between the texts, ensuring generated information aligns with the reference.

Informativeness: This portion gauges the level of new information added by the generated text compared to the reference.

Preferences: BART Score leverage BART's architecture to assess coherence, fluency, and contextual relevance of generated text against reference texts. It would naturally prefer models that are adept at producing text that is contextually relevant, coherent across longer passages, and syntactically fluent.

3.1.8 BLEURT (Bilingual Evaluation Understudy with Representations from Transformers)

BLEURT is an evaluation metric for Natural Language Generation (Sellam et al., 2020). It takes a pair of sentences as input, a reference and a candidate, and it returns a score that indicates to what extent the candidate is fluent and conveys the meaning of the reference.

Calculation procedure:

1. ***Feature Extraction****: The model takes a pair of sentences (a reference and a candidate) as input and passes them through a BERT model to obtain their embeddings. These embeddings are high-dimensional feature vectors that capture the semantic information of the sentences.*
2. ***Regression Model****: The regression model compares the embeddings of the reference and candidate sentences to calculate a similarity score. This comparison is done using a linear layer that predicts the similarity between the feature vectors of the original sentence and its re-translation.*
3. ***Training****: The regression model is trained on a dataset of human ratings. The training process involves adjusting the parameters of the model to minimize the difference between the model's predictions and the actual human ratings.*
4. ***Output****: The model returns a score that indicates to what extent the candidate is fluent and conveys the meaning of the reference.*

Preferences: BLEURT scores texts by leveraging a BERT-based model fine-tuned on human judgment data, evaluating semantic similarity and naturalness of language. This mechanism inclines BLEURT to prefer models that produce text closely mirroring human writing styles and semantic richness, capturing nuances in meaning and context.

3.1.9 CLIP Score

CLIP Score is a reference free metric that can be used to evaluate the correlation between a generated caption for an image and the actual content of the image. It has been found to be highly correlated with human judgement (Hessel et al., 2021).

The CLIP Score is calculated as:

$$CLIP\ Score(I, C) = cosine\ (E_I, E_C)$$
$$E_I: embedding\ for\ image\ I\ by\ CLIP\ Model$$
$$E_C: embedding\ for\ caption\ C\ by\ CLIP\ Model$$

Preferences: CLIP Score uses the cosine similarity between the embeddings of images and corresponding textual descriptions generated by the CLIP model, this score would measure how well the text describes the image, considering both semantic content and visual details. Given this approach, CLIP Score would favor models that excel in generating accurate, detailed, and semantically rich descriptions of images. These models are able to understand and interpret complex visual scenes and translate this understanding into coherent, contextually relevant text.

3.1.10 Rank Score

Rank Score is designed to evaluate image-captioning models for the application of image-based product search. Given a query product image, the model generates a text caption. This caption is then used to retrieve relevant products textually, and the rank of the original queried product in the results list is used to calculate the Rank Score.

Specifically, the candidate caption for the query image is compared against product captions in the dataset via BERT embeddings to retrieve a ranked list of products by similarity. If the original product image ranks 1st, the Rank Score is 1, indicating the highest performance in returning the queried product. If the product ranks last, the Rank Score is close to 0. Other ranks will have values between 0 and 1, with higher values indicating better performance in retrieving the original product.

The Rank Score is calculated as:

$$Rank\ Score = 1 - \frac{True\ Rank - 1}{N}$$
$$N: Total\ num\ of\ original\ texts\ in\ the\ dataset$$
$$True\ Rank: the\ position\ of\ the\ true\ original\ text\ in\ the\ list\ sorted\ by\ descending\ similarity\ to\ the\ generated\ text.$$

Preferences*:* This approach aims to quantify how well the generated caption captures the visual essence of the query image in a way that enables accurate text-based retrieval of the original product. The metric favors models that produce captions semantically aligned with the visual content to support precise image search.

3.2 Dataset Variation with CLIP Score

In evaluating image-captioning models, the quality of benchmark datasets greatly influences the outcomes. To address this, we employ CLIP Score (Hessel et al., 2021) to measure the congruence between images and captions, offering a method to assess the dataset's overall alignment, shown in Figure 1. By calculating the standard deviation of these scores for each dataset, we can get comparative variations of datasets. A dataset with lower standard deviation indicates less variation, and higher standard deviation points to greater variation. In general, a model performs better in a dataset with less dataset variation is inclined to have more precision and efficiency; and a model performs better in dataset with more dataset variation is inclined to have more robustness and adaptability. We could extract more model patterns and suitable applications based on comparative dataset variations and evaluation metric patterns, shown in TABLE I.

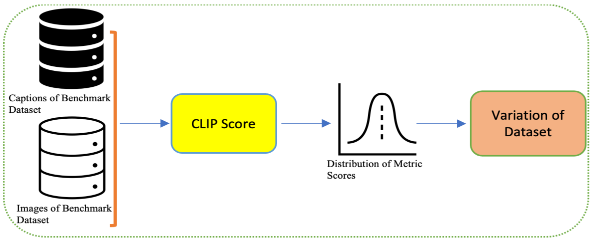


Figure 1: Check dataset variation with CLIP Score

Metrics	*Less Variation*	*More Variation*
BLEU	**Strength:** High lexical matching precision. **Applications:** Formal document translation, legal document replication.	**Strength:** Adaptability to lexical diversity. **Applications:** Multilingual social media content translation, diverse genre text localization.
ROUGE	**Strength:** Detailed content coverage. **Applications:** Executive summary generation for business reports, focused news article summarization.	**Strength:** Flexibility in content extraction. **Applications:** Summarizing user-generated content, variable style news aggregation.
METEOR	**Strength:** Precision in detailed captions. **Applications:** Ideal for archival systems where accuracy is paramount.	**Strength:** Versatility in language adaptation. **Applications:** Suited for dynamic content such as diverse social media platforms.
SPICE	**Strength:** In-depth scene analysis. **Applications:** Suitable for educational tools requiring detailed image explanations.	**Strength:** Recognition of features in complex visuals. **Applications:** E-commerce platforms, highlighting product features amidst visual clutter.
BLEURT	**Strength:** Nuanced tone and language detection. **Applications:** Luxury branding where subtlety in captions can influence perception.	**Strength:** Adaptability to a range of linguistic styles. **Applications:** User-generated content platforms needing accurate captions for diverse submissions.
BERT Score	**Strength:** Deep semantic alignment with reference texts. **Applications:** Content-centric websites that require aligned thematic narratives.	**Strength:** Robust contextual understanding. **Applications:** News and information sites with varied topical content.
CIDEr	**Strength**: Precision in detail-oriented image description. **Applications**: Cataloging for digital archives, precise product descriptions for e-commerce.	**Strength**: Ability to highlight unique image features. **Applications**: Dynamic caption generation for social media platforms, interactive educational content.
BART Score	**Strength:** Mastery in generating coherent, contextually relevant text. **Applications:** Narrative-driven media, adding depth to visual stories.	**Strength:** Flexibility in text generation across diverse styles and formats. **Applications:** Creative writing tools, adaptive marketing content generation across various platforms.
CLIP Score	**Strength:** Precise visual-text alignment. **Applications:** Art galleries or databases requiring accurate image cataloging.	**Strength:** Creativity and adaptability in describing diverse visual content. **Applications:** Social media content generation and enhancement.
Rank Score	**Strength:** Precision in semantic alignment with target texts. **Applications:** Customized news feed generation, precise document retrieval in legal and academic research databases.	**Strength:** Adaptability in understanding and matching a wide range of semantic contexts. **Applications:** Chatbots and virtual assistants tailored to diverse user queries.

TABLE 1: Model's Comparative Strength and Suitable Applications under Each Evaluation Metric

3.3 Comparative Evaluation with Statistical Analysis

Our method employs t-tests with Bonferroni correction on model results across various evaluation metrics. This determines if there is a model that achieves higher scores with statistical significance than others in a specific evaluation metric. If there exists such a model, we harness the specific preferences of that metric and corresponding benchmark dataset variation to infer the model's comparative strengths. The whole procedure is shown in Figure 2.

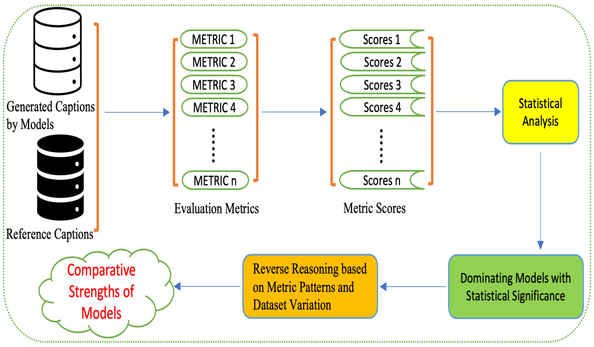


Figure 2: Procedure of Comparative Evaluation with Statistical Analysis

4. Experiments

We conduct experiments on two distinct datasets to validate the real-world efficacy of our novel evaluation framework for enhanced evaluation of image-captioning models.

4.1 Datasets

FashionGen (Rostamzadeh et al. 2022) - This dataset contains 360K apparel image-text pairs with high alignment. Typically, a product could have several different images but the same caption. This dataset features images of clothing items worn by models against clean backgrounds and consistent captions detailing visual attributes of clothing items. We randomly sample a subset of 50,000 rows for training and 500 rows for testing, which are mutually exclusive.

Amazon Product Dataset (Tools and Home Improvement) (Ni et al., 2019) - The home & kitchen product dataset of 51K rows from Amazon consists of seven subcategories, exhibiting greater noise - multiple images per product with different perspectives, backgrounds, and even sketch maps. Captions could contain peripheral details less related to corresponding products, such as product histories and usage scenarios not directly extractable from images alone. Our experiment uses 500 rows proportionally sampled across subcategories for testing, with the remainder forming the training dataset.

4.2 Evaluation Metrics

For our experiments, we specifically chose not to use BLEU, ROUGE, and CIDEr due to their limitations. BLEU, primarily focused on n-gram overlap, is adept at evaluating literal translations but falls short in assessing the contextual alignment in image-captioning. ROUGE, while valuable for summarization tasks, does not cater to our objective of generating descriptive captions to describe images. Lastly, CIDEr, despite its utility in comparing a generated caption against a set of references, does not suit our model's aim of producing a singular, optimal caption for each image.

4.3 Models

CoCa(**Contrastive Captioners**) (Yao et al. 2021) - a state-of-the-art model designed to excel in both image understanding and captioning tasks by leveraging contrastive learning. It combines the strengths of powerful visual encoders with language models to generate descriptive, accurate captions for images.

Video-LLaVa (Video Language-Large Model) (Lin et al. 2022) **-** extends the capabilities of language models to the domain of video understanding and captioning. By processing video inputs alongside textual descriptions, Video-LLaVa aims to capture the dynamic aspects of video content, translating them into coherent and comprehensive text.

BLIP (Bootstrapped Language Image Pre-training) (Li et al. 2021) - a model that emphasizes the pre-training phase to enhance the synergy between visual perception and language understanding. By bootstrapping from large-scale datasets, BLIP learns to generate captions that are not only accurate in depicting the visual content but also engaging and informative.

ViT-GPT2 - it combines the Vision Transformer (ViT) (Dosovitskiy et al. 2021) with the GPT-2 (Radford, Alec, et al. 2019) language model to create a hybrid system capable of processing images and generating corresponding captions. ViT extracts and processes visual information from images, transforming it into a format that the GPT-2 model can use to generate textual descriptions.

4.4 Experimental Procedure

To fully leverage each model's capabilities and provide a robust evaluation, we adopted distinct measures tailored to each model.

For CoCa, it's a structural model without prior training, we train it from scratch separately on both the FashionGen and Amazon training datasets.

For BLIP, to sharpen their domain knowledge and optimize performance for our specific datasets, we finetune the pretrained model (*Salesforce/blip-image-captioning-large*) from HuggingFace, separately on both the FashionGen and Amazon training datasets and rename it Finetuned-BLIP.

For ViT-GPT2, same as BLIP, we finetune the pretrained model (*vit-gpt2-image-captioning*) from HuggingFace, separately on both the FashionGen and Amazon training datasets and rename it Finetuned-ViT.

For Video-LLaVa, it is a LLM with extensive pre-training. Given its big size and intricate internal structure, fine-tuning was not a viable option. Instead, we utilize prompting to direct it to generate appropriate captions. The respective prompts as shown below:

> *For FashionGen:*
> *Create a caption for the fashion image that accurately describes the garment. Concentrate on the clothing item design, material, cut, patterns, and any distinctive features. Provide a clear and precise description that could serve as a product description, ignoring any other elements in the image.*
> *For Amazon:*
> *Generate a concise and accurate caption describing key details in product images from Amazon. Focus on correctly identifying the main product or object in the image frame and list only the most salient attributes and components. Avoid subjective impressions or editorial comments. Stick to factual, neutral descriptions of visual elements reflecting the product name, materials, parts, colors, text, or other relevant details a shopper would need to make a purchase decision.*

Having generated candidate captions from these 4 models for both test datasets, we next apply our suite of evaluation metrics, including BERT Score, BART Score, METEOR, SPICE, BLEU, CLIP Score, Rank Score, to get results.

After getting results, we then conduct paired t-tests with Bonferroni correction separately on the evaluation results of these two testing datasets, shown in Figure 3 and Figure 4. We collect results for each metric to check whether there exists a significant leading model on an evaluation metric, the results are shown in TABLE 2 and TABLE 3, and the statistical significance criteria is (p-value < 0.01).

Given the divergence across different datasets, we use CLIP Score to evaluate the overall alignment of training datasets, shown in Figure 5. FashionGen shows an approximately normal distribution with a lower standard deviation, indicating less dataset variation. Conversely, the Amazon dataset displays a wider distribution with a higher standard deviation, reflecting more dataset variation.

```
Metric: Bart_Score
CoCa vs Video-LLaVA - P-value: 0.0000 - Effect Size: -1.4569
CoCa vs Finetuned-BLIP - P-value: 0.0000 - Effect Size: -0.8255
CoCa vs Finetuned-ViT - P-value: 0.0000 - Effect Size: -0.2112
Video-LLaVA vs Finetuned-BLIP - P-value: 0.0000 - Effect Size: 0.4578
Video-LLaVA vs Finetuned-ViT - P-value: 0.0000 - Effect Size: 1.2310
Finetuned-BLIP vs Finetuned-ViT - P-value: 0.0000 - Effect Size: 0.6247
Comparison: ('CoCa', 'Video-LLaVA'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('CoCa', 'Finetuned-BLIP'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('CoCa', 'Finetuned-ViT'), P-value (corrected): 0.0003, Reject Null: True
Comparison: ('Video-LLaVA', 'Finetuned-BLIP'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('Video-LLaVA', 'Finetuned-ViT'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('Finetuned-BLIP', 'Finetuned-ViT'), P-value (corrected): 0.0000, Reject Null: True

Metric: CLIP_Score
CoCa vs Video-LLaVA - P-value: 0.0000 - Effect Size: -0.6017
CoCa vs Finetuned-BLIP - P-value: 0.0000 - Effect Size: -0.7010
CoCa vs Finetuned-ViT - P-value: 0.0001 - Effect Size: -0.2966
Video-LLaVA vs Finetuned-BLIP - P-value: 0.0000 - Effect Size: -0.1993
Video-LLaVA vs Finetuned-ViT - P-value: 0.0000 - Effect Size: 0.3303
Finetuned-BLIP vs Finetuned-ViT - P-value: 0.0000 - Effect Size: 0.4684
Comparison: ('CoCa', 'Video-LLaVA'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('CoCa', 'Finetuned-BLIP'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('CoCa', 'Finetuned-ViT'), P-value (corrected): 0.0006, Reject Null: True
Comparison: ('Video-LLaVA', 'Finetuned-BLIP'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('Video-LLaVA', 'Finetuned-ViT'), P-value (corrected): 0.0001, Reject Null: True
Comparison: ('Finetuned-BLIP', 'Finetuned-ViT'), P-value (corrected): 0.0000, Reject Null: True

Metric: Rank_Score
CoCa vs Video-LLaVA - P-value: 0.0000 - Effect Size: 0.2842
CoCa vs Finetuned-BLIP - P-value: 0.0003 - Effect Size: -0.1704
CoCa vs Finetuned-ViT - P-value: 0.3252 - Effect Size: 0.0552
Video-LLaVA vs Finetuned-BLIP - P-value: 0.0000 - Effect Size: -0.4507
Video-LLaVA vs Finetuned-ViT - P-value: 0.0001 - Effect Size: -0.2296
Finetuned-BLIP vs Finetuned-ViT - P-value: 0.0000 - Effect Size: 0.2252
Comparison: ('CoCa', 'Video-LLaVA'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('CoCa', 'Finetuned-BLIP'), P-value (corrected): 0.0016, Reject Null: True
Comparison: ('CoCa', 'Finetuned-ViT'), P-value (corrected): 1.0000, Reject Null: False
Comparison: ('Video-LLaVA', 'Finetuned-BLIP'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('Video-LLaVA', 'Finetuned-ViT'), P-value (corrected): 0.0009, Reject Null: True
Comparison: ('Finetuned-BLIP', 'Finetuned-ViT'), P-value (corrected): 0.0000, Reject Null: True

Metric: METEOR_Score
CoCa vs Video-LLaVA - P-value: 0.0000 - Effect Size: 2.6457
CoCa vs Finetuned-BLIP - P-value: 0.0000 - Effect Size: 1.1519
CoCa vs Finetuned-ViT - P-value: 0.0000 - Effect Size: 2.6176
Video-LLaVA vs Finetuned-BLIP - P-value: 0.0000 - Effect Size: -1.2599
Video-LLaVA vs Finetuned-ViT - P-value: 0.0000 - Effect Size: 0.2641
Finetuned-BLIP vs Finetuned-ViT - P-value: 0.0000 - Effect Size: 1.3168
Comparison: ('CoCa', 'Video-LLaVA'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('CoCa', 'Finetuned-BLIP'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('CoCa', 'Finetuned-ViT'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('Video-LLaVA', 'Finetuned-BLIP'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('Video-LLaVA', 'Finetuned-ViT'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('Finetuned-BLIP', 'Finetuned-ViT'), P-value (corrected): 0.0000, Reject Null: True
```

Figure 3: Results of Statistical Analysis with Statistical Significance (FashionGen)

```
Metric: Bart_Score
CoCa vs Video-LLaVA - P-value: 0.0000 - Effect Size: 0.3849
CoCa vs Finetuned-BLIP - P-value: 0.0017 - Effect Size: 0.2125
CoCa vs Finetuned-ViT - P-value: 0.0000 - Effect Size: 0.4127
Video-LLaVA vs Finetuned-BLIP - P-value: 0.0000 - Effect Size: -0.2465
Video-LLaVA vs Finetuned-ViT - P-value: 0.1100 - Effect Size: 0.0708
Finetuned-BLIP vs Finetuned-ViT - P-value: 0.0000 - Effect Size: 0.2877
Comparison: ('CoCa', 'Video-LLaVA'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('CoCa', 'Finetuned-BLIP'), P-value (corrected): 0.0100, Reject Null: True
Comparison: ('CoCa', 'Finetuned-ViT'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('Video-LLaVA', 'Finetuned-BLIP'), P-value (corrected): 0.0001, Reject Null: True
Comparison: ('Video-LLaVA', 'Finetuned-ViT'), P-value (corrected): 0.6603, Reject Null: False
Comparison: ('Finetuned-BLIP', 'Finetuned-ViT'), P-value (corrected): 0.0000, Reject Null: True

Metric: SPICE_Score
CoCa vs Video-LLaVA - P-value: 0.0000 - Effect Size: 0.5988
CoCa vs Finetuned-BLIP - P-value: 0.0001 - Effect Size: 0.2474
CoCa vs Finetuned-ViT - P-value: 0.0000 - Effect Size: 0.3034
Video-LLaVA vs Finetuned-BLIP - P-value: 0.0000 - Effect Size: -0.5415
Video-LLaVA vs Finetuned-ViT - P-value: 0.0057 - Effect Size: -0.4670
Finetuned-BLIP vs Finetuned-ViT - P-value: 0.0663 - Effect Size: 0.0758
Comparison: ('CoCa', 'Video-LLaVA'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('CoCa', 'Finetuned-BLIP'), P-value (corrected): 0.0006, Reject Null: True
Comparison: ('CoCa', 'Finetuned-ViT'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('Video-LLaVA', 'Finetuned-BLIP'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('Video-LLaVA', 'Finetuned-ViT'), P-value (corrected): 0.0341, Reject Null: True
Comparison: ('Finetuned-BLIP', 'Finetuned-ViT'), P-value (corrected): 0.3977, Reject Null: False
Metric: Rank_Score
CoCa vs Video-LLaVA - P-value: 0.0020 - Effect Size: 0.1631
CoCa vs Finetuned-BLIP - P-value: 0.0076 - Effect Size: -0.2141
CoCa vs Finetuned-ViT - P-value: 0.6506 - Effect Size: 0.0011
Video-LLaVA vs Finetuned-BLIP - P-value: 0.0000 - Effect Size: -0.3953
Video-LLaVA vs Finetuned-ViT - P-value: 0.0046 - Effect Size: -0.1669
Finetuned-BLIP vs Finetuned-ViT - P-value: 0.0006 - Effect Size: 0.2220
Comparison: ('CoCa', 'Video-LLaVA'), P-value (corrected): 0.0119, Reject Null: True
Comparison: ('CoCa', 'Finetuned-BLIP'), P-value (corrected): 0.0457, Reject Null: True
Comparison: ('CoCa', 'Finetuned-ViT'), P-value (corrected): 1.0000, Reject Null: False
Comparison: ('Video-LLaVA', 'Finetuned-BLIP'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('Video-LLaVA', 'Finetuned-ViT'), P-value (corrected): 0.0275, Reject Null: True
Comparison: ('Finetuned-BLIP', 'Finetuned-ViT'), P-value (corrected): 0.0038, Reject Null: True

Metric: METEOR_Score
CoCa vs Video-LLaVA - P-value: 0.0000 - Effect Size: 1.2201
CoCa vs Finetuned-BLIP - P-value: 0.0000 - Effect Size: 1.0234
CoCa vs Finetuned-ViT - P-value: 0.0000 - Effect Size: 1.1421
Video-LLaVA vs Finetuned-BLIP - P-value: 0.2503 - Effect Size: -0.2475
Video-LLaVA vs Finetuned-ViT - P-value: 0.0000 - Effect Size: -0.0359
Finetuned-BLIP vs Finetuned-ViT - P-value: 0.0000 - Effect Size: 0.1745
Comparison: ('CoCa', 'Video-LLaVA'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('CoCa', 'Finetuned-BLIP'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('CoCa', 'Finetuned-ViT'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('Video-LLaVA', 'Finetuned-BLIP'), P-value (corrected): 1.0000, Reject Null: False
Comparison: ('Video-LLaVA', 'Finetuned-ViT'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('Finetuned-BLIP', 'Finetuned-ViT'), P-value (corrected): 0.0000, Reject Null: True

Metric: BLEURT_Score
CoCa vs Video-LLaVA - P-value: 0.0000 - Effect Size: 0.0398
CoCa vs Finetuned-BLIP - P-value: 0.0002 - Effect Size: 0.3716
CoCa vs Finetuned-ViT - P-value: 0.2721 - Effect Size: 0.2175
Video-LLaVA vs Finetuned-BLIP - P-value: 0.0000 - Effect Size: 0.3970
Video-LLaVA vs Finetuned-ViT - P-value: 0.0002 - Effect Size: 0.2126
Finetuned-BLIP vs Finetuned-ViT - P-value: 0.0178 - Effect Size: -0.1557
Comparison: ('CoCa', 'Video-LLaVA'), P-value (corrected): 0.0001, Reject Null: True
Comparison: ('CoCa', 'Finetuned-BLIP'), P-value (corrected): 0.0012, Reject Null: True
Comparison: ('CoCa', 'Finetuned-ViT'), P-value (corrected): 1.0000, Reject Null: False
Comparison: ('Video-LLaVA', 'Finetuned-BLIP'), P-value (corrected): 0.0000, Reject Null: True
Comparison: ('Video-LLaVA', 'Finetuned-ViT'), P-value (corrected): 0.0009, Reject Null: True
Comparison: ('Finetuned-BLIP', 'Finetuned-ViT'), P-value (corrected): 0.1068, Reject Null: False
```

Figure 4: Results of Statistical Analysis with Statistical Significance (Amazon)

Evaluation Metric	Significant Leading Model	Statistical Significance
BART Score	Video-LLaVa	Yes
METEOR	CoCa	Yes
CLIP Score	Finetuned-ViT	Yes
Rank Score	Finetuned-BLIP	Yes

TABLE 2: Statistical Analysis OF Evaluation Metric Scores IN FashionGen Testing Dataset.

Evaluation Metric	Significant Leading Model	Statistical Significance
BART Score	CoCa	Yes
BLEURT	CoCa	Yes
SPICE	CoCa	Yes
METEOR	CoCa	Yes
Rank Score	Finetuned-BLIP	Yes

TABLE 3: Statistical Analysis OF Evaluation Metric Scores IN Amazon Testing Dataset

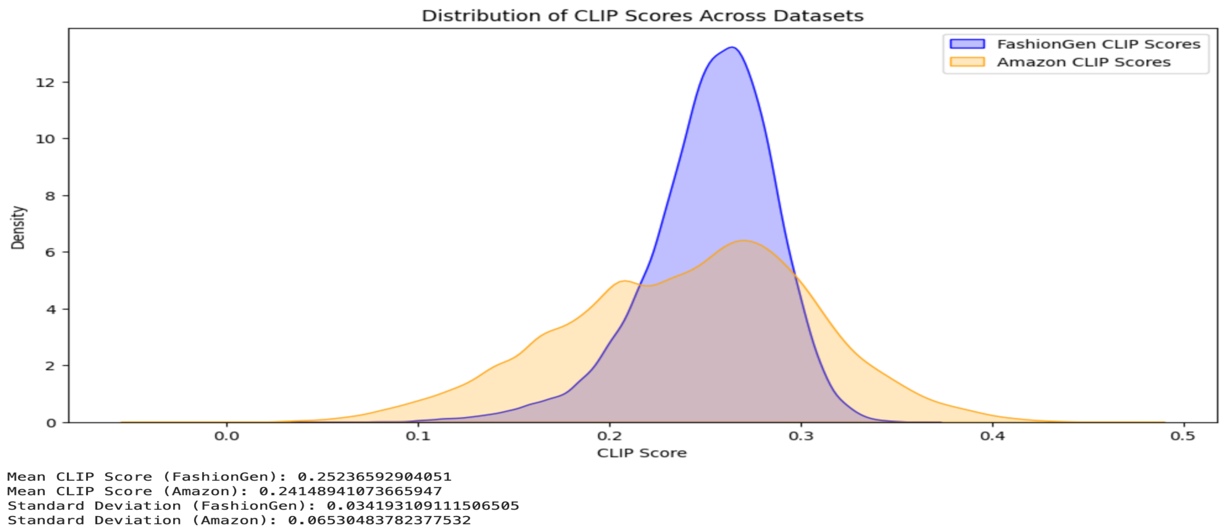


Figure. 5: Distribution Plot of CLIP Scores across Two Benchmark Datasets

4.5 Experimental Results

Table 2 and Table 3 present significant leading models on metrics across the FashionGen and Amazon Product test datasets. Observed results include:

- Video-LLaVA demonstrates top performance on BART Score in FashionGen dataset while CoCa leads on BART Score in Amazon dataset.
- Fine-tuned ViT takes the lead on CLIP Score of FashionGen dataset.
- CoCa shows dominance in other metrics with statistical significance across datasets.
- Fine-tuned BLIP consistently beat others on Rank Score metric in both datasets.

4.6 Model Analysis on Evaluation Results

Analyze the experimental results and refer to TABLE 1, we get insights below:

CoCa: CoCa takes the lead in various metrics across both datasets, reflecting its comprehensive understanding of image-caption relationships. Its success could be attributed to its contrastive learning framework, enabling interpretation and generation of captions that are both contextually relevant and linguistically precise. From these results, we conclude that CoCa is exceptionally capable of adapting to diverse dataset qualities, making it a versatile choice for applications requiring detailed and accurate image captions, from content creation to archival description.

Video-LLaVa: Video-LLaVa's stands out on BART Score in FashionGen dataset, demonstrating its ability to generate coherent and contextually relevant text. This proficiency suggests that Video-LLaVa effectively interprets and translates visual content into meaningful captions, leveraging its advanced understanding of both visual elements and textual narrative. From this result, we conclude that Video-LLaVa is well-suited for enriching narrative-driven media and adding depth to visual stories, making it a valuable tool for applications aiming to provide engaging and enriched content narratives.

ViT-GPT2: ViT-GPT2 excels in the FashionGen dataset on the CLIP Score, showcasing its strength in precise visual-text alignment. This performance could be attributed to its use of Vision Transformer for visual analysis combined with GPT-2 for text generation, ensuring accurate and relevant captions for images. Consequently, we conclude that ViT-GPT2 could be effective for applications like art gallery databases or specialized image catalogs, where accurate and detailed image descriptions are crucial for user engagement and information retrieval.

BLIP: BLIP excels in the Rank Score metric across datasets, showcasing its precision and adaptability in semantic alignment. This performance indicates BLIP's robust capability for detailed semantic interpretation and flexible application across different content needs. Consequently, BLIP is ideal for creating precise, customized content in areas like news feeds and document retrieval, as well as for developing responsive chatbots and virtual assistants capable of handling a wide range of queries.

Our research, informed by experimental results and analysis to evaluation metrics, aimed to evaluate, not rank, various image-captioning models to discern their comparative strengths and suitable applications. This methodology highlights comparative advantages of each model, facilitating an informed selection for specific image-captioning needs. The findings offer a strategic framework for choosing models that best match the required competencies for targeted applications.

5. Conclusion

In our paper, we introduced a handy framework to evaluate image-captioning models. By conducting experiments on two datasets with contrasting variation in image-caption alignment, we demonstrated how our approaches can reveal the inherent strengths and practical applicability of different models. Our integration of statistical analysis with reverse reasoning on evaluation metrics, providing a comprehensive framework that not only assesses accuracy but also provides practical applications. The insights extracted from this procedure underscore the versatility of our evaluation framework in discerning the comparative capabilities of image-captioning models in varied contexts.

Future work: Future efforts could aim to expand the range of evaluation metrics for a deeper analysis of model capabilities. There is also significant potential in refining the training or finetuning procedures for the image-captioning models under study. Perfecting these models to their optimal performance is key to accurately harnessing their full potential in real-world applications. Besides, our method shall not be limited to image-captioning, once there are multiple evaluation metrics, we could always apply the framework, such as evaluations on video-captioning and text-to-image under e-commerce industry.

References

Kumar, A. (2022). The Illustrated Image Captioning using transformers. ankur3107.github.io.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.J. (2002). BLEU: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting of the Association for Computational Linguistics (pp. 311-318).

Lin, C.Y. (2004). ROUGE: A package for automatic evaluation of summaries. In Text summarization branches out (pp. 74-81).

Vedantam, R., Lawrence Zitnick, C., & Parikh, D. (2015). Cider: Consensus-based image description evaluation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4566-4575).

Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., & Zhang, L. (2018). Bottom-up and top-down attention for image captioning and visual question answering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 6077-6086).

Banerjee, S., & Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization (pp. 65-72).

Hessel, J., Holtzman, A., Forbes, M., Woolley, A., Rajani, N. F., Rocktäschel, T., ... & Lee, L. (2021). CLIPScore: A Reference-free Evaluation Metric for Image Captioning. arXiv preprint arXiv:2102.08535.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (pp. 4171-4186).

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2020). BERTscore: Evaluating text generation with BERT. In International Conference on Learning Representations.

Yuan, W., Neubig, G., & Liu, P. (2022). BARTScore: Evaluating Generated Text as Text Generation. arXiv preprint arXiv:2206.04793.

Sellam, T., Das, D., & Parikh, A. P. (2020). BLEURT: Learning robust metrics for text generation. In Proceedings of the 58th annual meeting of the association for computational linguistics (pp. 7881-7892).

Rostamzadeh, N., Hosseini, S.A., Boquet, T., Stokowiec, W., Zhang, Y., Jauvin, C. et al. (2022). Fashion-Gen: The Generative Fashion Dataset and Challenge. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (pp. 525-534).

Lin, B., Ye, Y., Zhu, B., Cui, J., Ning, M., Jin, P., & Yuan, L. (2022). Video-LLaVA: Learning United Visual Representation by Alignment Before Projection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 801-810).

Li, X., Yin, X., Li, C., Hu, X., Yang, P., Zhang, L., Hu, H., Zhang, L., Wang, L., & Hu, B. (2021). Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks. European Conference on Computer Vision (ECCV).

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. & Uszkoreit, J. (2021) An image is worth 16x16 words: Transformers for image recognition at scale. 9th International Conference on Learning Representations (ICLR).

Radford, Alec, et al. Language models are unsupervised multitask learners. OpenAI blog 1.8 (2019): 9.

Jianmo Ni, Jiacheng Li, Julian McAuley. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. Empirical Methods in Natural Language Processing (EMNLP), 2019.

Frogs into princes: A generative model to understand the success of product descriptions

Takehiro Takayanagi[♣], **Bruno Charron**[♢], **Marco Visentini-Scarzanella**[♢]
[♣] The University of Tokyo, School of Engineering, Tokyo, Japan
[♢] Amazon, Tokyo, Japan
takayanagi-takehiro590@g.ecc.u-tokyo.ac.jp
{bcharro, marcovs}@amazon.co.jp

Abstract

In the dynamic marketplace, vendors continuously seek innovative ideas for new products and ways to improve existing ones. These ideas can be uncovered by analyzing text data, such as product descriptions and customer reviews. However, the ever-increasing volume of text data poses a challenge in extracting meaningful insights. Therefore, this study addresses the challenge of extracting actionable insights from the growing volume of text data, with a specific focus on product descriptions. To this end, we investigate two primary research questions: the predictive power of product descriptions for product success, and the capability of style transfer to highlight the successful factors of these descriptions. In response to the first question, our findings validate that product descriptions are indeed reliable indicators of product success. Addressing our second question, we propose a Successful Style Transfer Variational Autoencoder (SST-VAE), a VAE-based language model designed for effective successful style transfer. Qualitative analysis indicates that the SST-VAE effectively enables successful style transfer conditional on a given label. In addition, case studies suggest that the proposed approach could be useful in gaining insights about product success, by highlighting key factors that may contribute to their success. On the other hand, our approach confronts issues such as hallucinations and the need for factual accuracy. These challenges underscore the necessity for continued research in the field of e-commerce natural language processing.

Keywords: generative model, product description, natural language processing

1. Introduction

In the dynamic marketplace, vendors continuously seek innovative ideas for new products and ways to improve existing ones. This valuable knowledge could be revealed through the analysis of textual data, including product descriptions and consumer reviews. Yet, the rapid expansion of textual data volume poses a significant challenge in manually analyzing them. Accordingly, natural language processing (NLP) is utilized to extract meaningful insights from textual data, which has resulted in extensive research in the field of e-commerce NLP.

In the domain of e-commerce NLP, a significant body of work focuses on product descriptions, recognizing their crucial role in providing a competitive customer experience (Wang et al., 2017; Chen et al., 2019; Chan et al., 2019; Zhang et al., 2019; Zheng et al., 2018). In this realm, a considerable portion of the current research focuses on text generation for product descriptions. This focus has given rise to unique challenges, including personalized generations (Chen et al., 2019) and fidelity-oriented generations (Chan et al., 2019). However, there has been little research aimed at extracting valuable insights about product success.

Addressing this research gap, the primary aim of this study is to extract insights regarding product success from product descriptions. Following this direction, two key questions emerged.

Initially, there is uncertainty regarding the extent to which product descriptions contain insights into a product's success. Figure 1 presents a comparison between the description of a successful product (which has a higher rating) and that of an unsuccessful product (which has a lower rating). From this figure, it remains unclear to what extent these descriptions capture information pertinent to product success, especially given that product success is often influenced by various factors, including external trends and events. Therefore, it is worthwhile to determine the degree to which these descriptions encompass relevant information. This consideration guides us to our first research question: "To what extent can product descriptions accurately indicate a product's likelihood of success?"

The second question emerges from the difficulty of manual dataset collection. Typically, extracting product success factors from product descriptions entails annotating these successful elements in product descriptions. Nevertheless, as Figure 1 illustrates, identifying the key elements that contribute to a product's success from its descriptions is quite challenging, which significantly complicates the manual development of a dataset. In response to these complexities, we consider that a style transfer approach could mitigate these challenges (Jin et al., 2022). It enables comparisons between original texts and those transformed into a successful style without the challenges of manual annotation.

The Seventh Workshop on e-Commerce and NLP (ECNLP 7), pages 88–96
21 May, 2024.

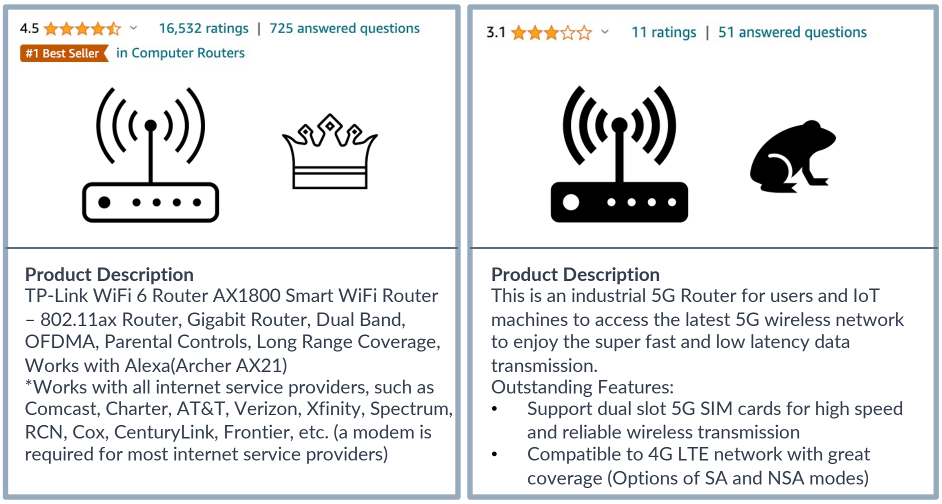


Figure 1: Two Product Examples with Similar Features but Varying Success: The Product On The Left Has A Higher Customer Rating, While The Product On The Right Has A Lower Customer Rating.

These comparisons will be beneficial in uncovering product success factors. This leads to our second research question: "Can we transform text descriptions from unsuccessful to successful ones, or vice versa?"

In summary, we formulated the following research questions

- RQ1: To what extent can product descriptions accurately indicate a product's likelihood of success?
- RQ2: Can we transform text descriptions from unsuccessful to successful ones, or vice versa?

This paper addresses these research questions. Firstly, we examine the potential of product descriptions as predictors of product success in response to the first research question. To this end, we introduce a success prediction task from product descriptions to assess the extent to which product descriptions can predict success.

Subsequently, we introduce a novel task involving the style transfer of product descriptions for the second research question. To address this challenge, we propose a Successful Style Transfer Variational Autoencoder (SST-VAE), a model designed to generate product descriptions conditional on the product's success. We assess the proposed model's output in qualitative analysis and case studies based on two perspectives: first, its ability to highlight factors of product success; second, its effectiveness in enhancing original descriptions through successful text transformations. Our results indicate that the proposed model successfully transforms the product descriptions from unsuccessful to successful and vice versa. In addition, our results suggest that the successful style transfer method can be effective in extracting insights about products by emphasizing the potential factors that could contribute to their success. On the other hand, we also reveal several issues, such as hallucinations, particularly when applied to enhance the product descriptions themselves, laying out a path for future research opportunities in this domain.

2. Related Work

2.1. Text Style Transfer

Text style transfer, an important task in natural language generation, focuses on modifying attributes like sentiment and politeness in text. It has regained notable attention in natural language processing due to the impressive results achieved with recent deep neural models (Jin et al., 2022). The challenge in text style transfer lies in the ambiguity of "style" in NLP and many works are tackling the problem. Some works deal with the sentiment of a text as a style of the text. Hu et al. (2017) proposes to control the sentiment of a text by using discriminators to reconstruct sentiment and content. Also, John et al. (2019) tackles the problem of disentangling the latent representations of style and content by proposing a multi-task loss to achieve the style transfer in terms of sentiment on non-parallel corpora. Additionally, Vasilakes et al. (2022) advances the field by enabling style transfer specifically in the realms of negation and uncertainty. Despite the extensive research in NLP style transfer, no existing study has explored text style transfer in the context of a product's success.

2.2. NLP in E-commerce

The NLP research in the e-commerce domain has shown considerable interest in product descriptions, with numerous studies addressing challenges such as text generation and attribute extraction (Wang et al., 2017; Chen et al., 2019; Chan et al., 2019; Zhang et al., 2019; Zheng et al., 2018). Especially, text generation task draws attention as quality product descriptions are important for providing a competitive customer experience in e-commerce stores. For instance, Wang et al. (2017) introduces a statistical model capable of producing accurate and fluent descriptions of product attributes. Further advancing the field, Chen et al. (2019) delves into generating personalized product descriptions using a knowledge base and neural networks. Additionally, Zhang et al. (2019) explores a pointer-generator neural network to generate product descriptions where the output patterns are controlled. Chan et al. (2019) emphasizes the fidelity of content in their approach to product description generation. Despite these advancements, most text generation research in this area has primarily focused on automating the product description writing process. There has been little emphasis on obtaining insights from descriptions or enhancing them.

Recognizing the importance of extracting meaningful insights from these text data, we propose a novel approach, employing a style transfer approach to gain insights into what drives product success.

3. RQ1: Can we predict products' success from product description?

Figure 2: An Illustration of the Success Prediction Task.

3.1. Success Prediction Task

The initial research question we address is, "Can we predict a product's success from its descriptions?" To tackle the question, we introduce a success prediction task based on product descriptions, as depicted in Figure 2. In this task, we train a natural language processing model to predict whether a product will be successful or not given product descriptions. The reasoning behind establishing this task for RQ1 is that if the model can predict success to a reasonable degree, it indicates the presence of significant information within product

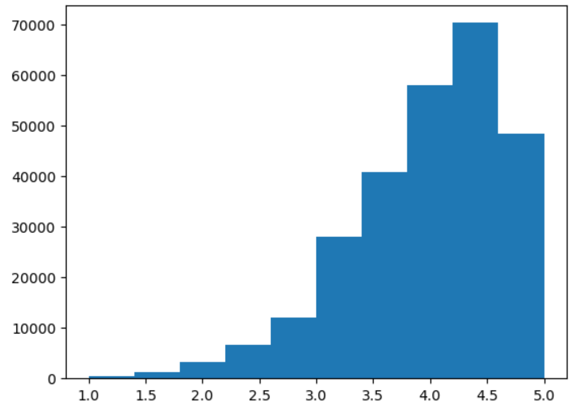


Figure 3: Histogram Displaying the Average Ratings Following Data Preprocessing.

descriptions that correlates with the product's success. In our experiments, we utilize a pre-trained BERT (Devlin et al., 2019) model to predict product success.

3.2. Evaluation Settings

In our experiments, we utilize the Amazon Public Dataset [1], which comprises a substantial collection of product data from Amazon. We focus our experiments on a single category, as we consider that the factors driving success vary significantly across different categories. Specifically, our experiments employ data from the Electronics category. For data processing, we exclude products with less than 5 ratings as well as those with duplicated descriptions, amounting to a total of 269,469 products. We define a product's success based on its average rating. Specifically, a product is labeled as successful if its average rating exceeds a certain threshold (for our study, thresholds are set at 3 or 4 out of 5 stars), and as unsuccessful if the average rating falls below 3. Figure 3 shows the distribution of average product review ratings.

We implement two data-splitting strategies to mitigate potential data leakage. The first approach is a random split, where data is divided randomly into training and test sets. In the random splitting strategy, we observe some instances where product descriptions share identical beginnings as shown in Table 1, and that the model often assigns the same labels to them. This raises concerns about data leakage, where the model's predictions for the test dataset might be influenced by the recurring text in the training dataset. To address this, we also adopt the group stratified data splitting strategy. Here, products sharing the first x characters (a predetermined threshold) in their descriptions

[1] https://cseweb.ucsd.edu/~jmcauley/datasets.html#amazon_reviews

Table 1: Examples of Two Product Descriptions with Identical Initial Sentences (Italicized) to Highlight Shared Content.

	Description
D1	*Bring your digital camera back to life with a new battery. Make sure you never miss another once-in-a-lifetime moment by having a new*, battery specifically designed for your Canon Rebel T2i T3i T4i T5i digital SLR camera. BM Premium rechargeable batteries are engineered to meet or exceed OEM specifications and feature the latest battery technology, including advanced circuitry, voltage regulation, and thermal circuit protection. BM Premium batteries include a one-year warranty.
D2	*Bring your digital camera back to life with a new battery. Make sure you never miss another once-in-a-lifetime moment by having a new*, LPE-10 battery specifically designed for your Canon Rebel digital SLR camera. LPE10 rechargeable batteries are engineered to meet or exceed OEM specifications and feature the latest battery technology, including advanced circuitry, voltage regulation, and thermal circuit protection. This battery includes a one-year warranty.

Table 2: Results of the Success Prediction Task.

Splitting strategy	Random	Group shuffle with threshold				
Success label		Higher than 3				Higher than 4
Threshold	-	50	100	300	500	100
Accuracy	0.71	0.69	0.69	0.70	0.70	0.88
F1	0.75	0.74	0.74	0.74	0.74	0.94
Precision	0.72	0.69	0.71	0.72	0.72	0.90
Recall	0.79	0.79	0.77	0.76	0.77	0.97

are grouped together. We then ensure that descriptions from the same group are exclusively assigned to either the training or the test dataset, but not both.

As for evaluation metrics, we utilize accuracy, recall, precision, and F1 score. In the training, we finetune the BERT model for 3 epochs with learning rate 0.001.

3.3. Result and Discussion

The results of the success prediction task, presented in Table 2, indicate that product descriptions effectively predict a product's success, achieving around 70% accuracy in most scenarios. Moreover, this consistency in prediction accuracy persists across various experimental settings and different data-splitting strategies, including variations in the threshold value x.

Additionally, the model achieves greater performance when defining success at a level higher than 4, compared to a threshold higher than 3. This outcome indicates that there are meaningful distinctions within product descriptions that contribute to the successful differentiation between successful and unsuccessful products.

On the other hand, the precise reasons for the model's effective prediction of product success are not clear. We hypothesize two possible scenarios. Firstly, there may be a distinct style of writing in product descriptions that contributes to their success. Secondly, it is possible that successful products possess appealing features, and the model captures these features within the product descriptions. Investigating these hypotheses will be reserved for future research.

4. RQ2: Can we transfer the text description from unsuccessful to successful one or vice versa?

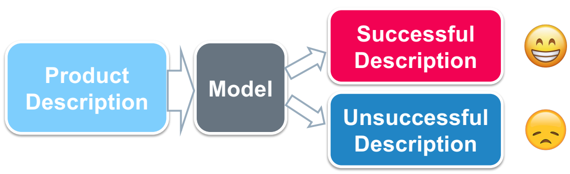


Figure 5: An illustration of the Successful Style Transfer Task.

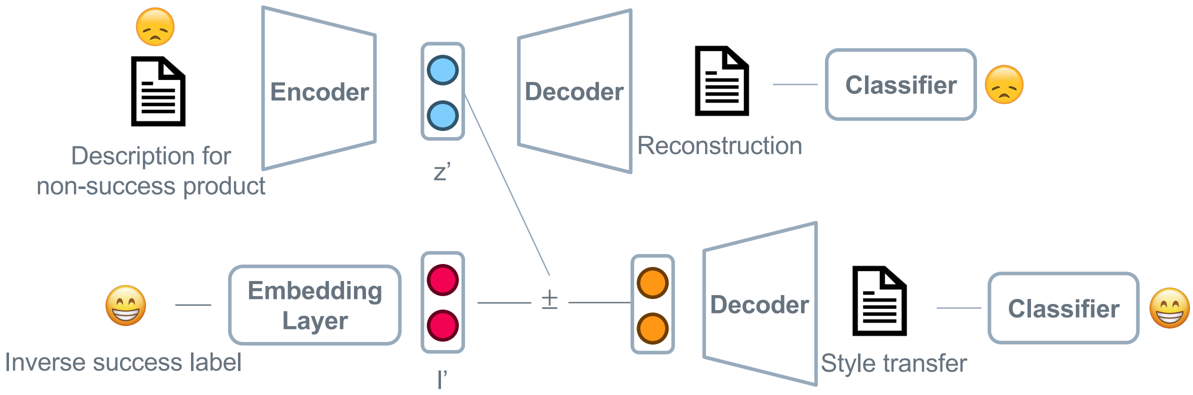


Figure 6: The Overview of Proposed SST-VAE Model.

4.1. Successful Style Transfer Task

Having confirmed that product descriptions are indeed reliable predictors of product success, we then move on to address the second research question: "Can we transform text descriptions from unsuccessful to successful ones, or vice versa?" To validate this question, we introduce a successful style transfer task, where an NLP model is trained to generate either more successful or less successful product descriptions as depicted in Figure 5. The underlying motivation for this task is that by shifting from successful to unsuccessful descriptions or the reverse, we can gain insights into the factors driving product success.

4.2. Method

For the successful style transfer task, we employ VAEs as text autoencoders due to their provision of smooth latent space, facilitating the transfer of texts conditioned on a label (Kingma and Welling, 2014; John et al., 2019). The basic VAE objective (i.e., the reconstruction loss with KL divergence) can be written as follows.

$$\mathcal{L}_\beta = \mathcal{L}_E + \beta \mathcal{L}_R \tag{1}$$

$$\mathcal{L}_E = - \mathop{\mathbb{E}}_{q_\phi(z|x)} [\log p_\theta(x|z)] \tag{2}$$

$$\mathcal{L}_R = KL(q_\phi(z|x) \| p(z)) \tag{3}$$

where $\mathcal{L}_E$ represents the reconstruction loss, while $\mathcal{L}_R$ denotes the KL divergence loss. The loss function $\mathcal{L}_\beta$ is a weighted combination of the reconstruction loss and KL divergence, with the β coefficient determining the influence of KL divergence on the overall loss. Typically, $q_\phi(z|x)$ is modeled as a Gaussian distribution with model parameters ϕ, and the re-parametrization trick is employed during training (Kingma and Welling, 2014). Additionally, $\log p_\theta(x|z)$ is a conditional probability of the data x given latent variables z with model parameters θ. For the prior distributions $p(z)$, we employ the standard Gaussian distribution $\mathcal{N}(0, I)$.

In this study, we employ Optimus, a large-scale language VAE model as our baseline model (Li et al., 2020). Optimus is a Transformer-based VAE model that incorporates BERT (Devlin et al., 2019) as its encoder and GPT-2 (Radford et al., 2019) as its decoder, achieving higher performance among VAE language models. In the Optimus framework, the functionalities of BERT and GPT-2 are integrated as follows: within the BERT structure, the first token of each sentence is assigned as a unique classification token, marked by `[CLS]`. The final layer's hidden state, $\mathbf{h}_{\text{[CLS]}} \in \mathbb{R}^H$ acts as the sentence-level representation. This representation is subsequently converted into a latent space, forming the latent variable $\mathbf{z} = W_E \mathbf{h}_{\text{[CLS]}}$. Here, $\mathbf{z}$ is a P-dimensional vector in $\mathbb{R}^P$, and W_E is the transformation matrix in $\mathbb{R}^{P \times H}$. The latent variable z is then integrated into the GPT-2 model to facilitate its decoding function.

To tackle the successful style transfer task, we propose the Successful Style Transferred Variational Autoencoder (SST-VAE). An overview of the model is depicted in Figure 6. The SST-VAE is based on on Optimus framework. The loss functions of SST-VAE are shown as follows.

$$\mathcal{L}_{SST} = \mathcal{L}_E + \alpha \mathcal{L}_{success} + \beta \mathcal{L}_{inverse} \tag{4}$$

where $\mathcal{L}_E$ represents the reconstruction loss, $\mathcal{L}_{success}$ denotes the success prediction loss, $\mathcal{L}_{inverse}$ refers to the inverse success prediction loss, and α and β are the weights assigned to these losses.

In SST-VAE, we freeze and retain the parameters of the encoder and decoder from fine-tuned Optimus, utilizing them in their existing states. For the latent variables, we add a single-layer neural network, enabling the latent variables to incorporate information related to the success label as follows.

$$\mathbf{z}' = W_A \mathbf{z} \tag{5}$$

where W_A is the transformation matrix in $\mathbb{R}^{P \times P}$.

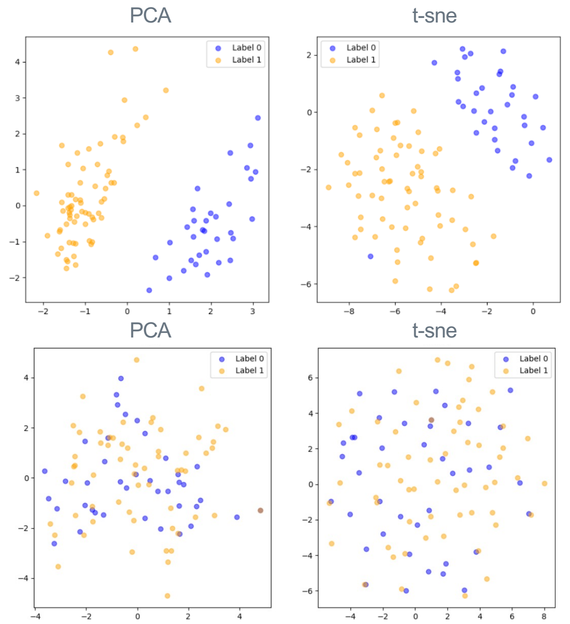


Figure 7: 2D Latent Embedding Plot: Top Row Shows SST-VAE, Bottom Row Baseline Models (Finetuned Optimus). Left Plots Utilize Principal Components, Right Plots Employ t-SNE for Dimension Reduction. The Yellow Labels Denote Successful Products, Blue Labels Indicate Non-Successful Products.

The success prediction loss is derived as follows.

$$\hat{\mathbf{x}} = Dec(\mathbf{x}, \mathbf{z}') \tag{6}$$

$$\mathbf{h}_{\hat{\mathbf{x}}} = BERT(\hat{\mathbf{x}}) \tag{7}$$

$$\hat{s} = f_\omega(\mathbf{h}_{\hat{\mathbf{x}}}) \tag{8}$$

$$\mathcal{L}_{success} = -s\log(\hat{s}) - (1-s)\log(1-\hat{s}) \tag{9}$$

First, we acquire the reconstructed text $\hat{\mathbf{x}}$ from the original text $\mathbf{x}$ and latent variables $\mathbf{z}'$ using the frozen decoder Dec. Then, we acquire the embeddings $\mathbf{h}_{\hat{\mathbf{x}}}$ of the reconstructed text with frozen BERT. Subsequently, we use a single-layer neural network f_ω to obtain the predicted success label $\hat{s}$. Finally, we calculate the success prediction loss with a cross-entropy loss with the success label $s \in \{0, 1\}$.

The inverse success prediction loss is derived as follows.

$$\mathbf{l}' = (1-2s)\mathbf{l} \tag{10}$$

$$\mathbf{x}' = Dec(\mathbf{x}, \mathbf{z}' + \mathbf{l}') \tag{11}$$

$$s' = f_\omega(\mathbf{x}') \tag{12}$$

$$\mathcal{L}_{inverse} = -s\log(s') - (1-s)\log(1-s') \tag{13}$$

where $\mathbf{l}$ is a learnable vector that represents the direction of success within the latent space, and $\mathbf{l}'$ is a vector that represents the opposite direction of success from the original success label s. The term $(1-2s)$ assigns a sign based on the success label $s \in \{0, 1\}$. When s is 1, $\mathbf{l}'$ becomes $-\mathbf{l}$ and when s is 0, $\mathbf{l}'$ becomes $\mathbf{l}$, reversing the sign of the embeddings. Then, we derive the style transferred text $\mathbf{x}'$ from the original text and the value of $\mathbf{z}'$ after adding $\mathbf{l}$. This process is to shift the latent variables in the direction opposite to the given success label, thereby generating descriptions with the inverse success label. Then, we utilize f_ω to predict the inverse success label. Finally, we obtain the inverse success prediction loss with a cross-entropy loss.

During inference, we can either add or subtract label embeddings to produce text with either more successful or less successful descriptions by $Dec(\mathbf{x}, \mathbf{z}' \pm \mathbf{l})$.

4.3. Evaluation Settings

For data processing, similar to the success prediction task, we eliminate products with less than 5 ratings and those with duplicated descriptions. Additionally, we eliminate product descriptions exceeding 64 tokens, in line with the preprocessing procedures in Optimus (Li et al., 2020). For data splitting, we utilize a stratified group shuffle method with a threshold set at 100.

As our baseline model, we fine-tune Optimus with a latent size of 768 and beta set to 1.0 [2], using our training corpus. Subsequently, the SST-VAE model is trained with the frozen weights of the encoder and decoder from the fine-tuned baseline model.

We assess our models' effectiveness in two approaches. Initially, we compare the latent embeddings of our model with those of the baseline model by projecting these embeddings into a 2D space through dimensionality reduction techniques such as principle component analysis and t-sne (van der Maaten and Hinton, 2008). Our second method involves qualitative evaluations through case studies. We generate style-transferred texts using both SST-VAE and a baseline model, then analyze the outcomes based on two viewpoints: assessing whether the results highlight factors behind product success and whether the transformed text enhances the original descriptions.

In our experiments, we chose hyperparameters as follows: The number of training epochs was set to 10. The loss weights were adjusted, with a ratio of 1/100 assigned to the reconstruction loss and a weight of 1 for the other two loss components. All models were trained on an NVIDIA V100 GPU.

[2] https://github.com/ChunyuanLI/Optimus/blob/master/doc/optimus_finetune_language_models.md

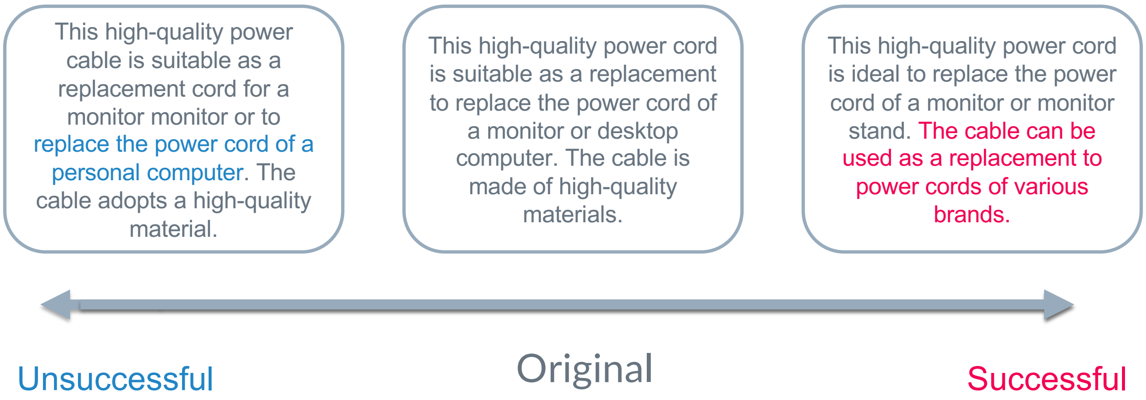


Figure 8: Example Output from SST-VAE Demonstrating Successful Style Transfer Task.

Style	Description
Original	Built-in Quick Operation Acoustic Engine for Sync and Phone Call Connectivity Advanced TX-Link for Listening to Music Auto-tuning LED-lit LCD TV 3.5mm Auxiliary Input for **iPod** Touch screen White Color with Built-in Dual Playback Jack PS 5 Input for Auxiliary Buit-in USB-C Audio Input Standard **Apple iPod** Touch 5intrusion 8x.
Unsuccessful	This high-quality power Opening-out Timer for Handsfree **Android** Auto Built-in Bluetooth 5.0 Microphone Input 3 Built-in USB Male Compatible with ALL Digital TV for Visual Pronotomic and Other Equinox RCA Light-Up Function iPod Controls 4 Touch for Enhanced Control of Control Skip-Forward/Forward Automatic Take-Down Timer .

Table 3: 1st Pattern: Describing Different Products.

4.4. Result and Discussion

The results of the embedding comparisons are presented in Figure 7. As Figure 7 demonstrates, the SST-VAE model exhibits clearer distinctions in the embedding space, suggesting that the latent embeddings from our proposed model provide more informative insights regarding the success of the product.

Figure 8 illustrates an example of the output generated by our model when it performs style transfer tasks. For a successful output, our model adds the successful label embeddings $\mathbf{l}$ to the latent variables $\mathbf{z}'$. Conversely, an unsuccessful output is generated when the model subtracts these label embeddings. Figure 8 presents the example of a power cord. When our model applies a successful style transfer, the description evolves to indicate the cord's compatibility with various brands, thereby suggesting its versatility. In contrast, an unsuccessful style transfer alters the description to specify the cord's exclusive use with personal computers, implying a more restricted utility.

Through qualitative analysis, we identified three distinct patterns in the output. The first pattern observed is a tendency to describe different products, as exemplified in Table 3. In this example, while the original text mentions Apple products like iPods, the descriptions shift to discussing Android when moved in the direction of unsuccessful styling. For automated product description generation, this deviation is not ideal, as the focus should remain on the specific features of the target product. Conversely, this tendency can be beneficial for understanding which brands or features might be more or less successful. This aspect is a unique point in extracting insights from product descriptions, as opposed to the automated generation where factual accuracy is paramount (Chan et al., 2019).

The second pattern we observed is that the model occasionally does not alter the output at all, even when we adjust the descriptions towards more successful or unsuccessful directions, as illustrated in Table 4. As the example indicates, this is problematic from both perspectives, as it neither facilitates automation nor provides insights into product success. We attribute this issue to the dominance of reconstruction loss in the loss functions. Upon examining the magnitude of the losses, we found that the reconstruction loss significantly outweighs others, such as success prediction and inverse success prediction losses. This remains true even when applying a weight of 0.01 to the reconstruction loss. On the other hand, using even

Style	Description
Original	You MUST RE-USE: your existing cabling and hardware. It is your responsibility to verify the batteries being ordered match the batteries in your unit prior to placing your order. Our products are not affiliated with or authorized by APC.
Successful	You MUST RE-USE: your existing cabling and hardware. It is your responsibility to verify the batteries being ordered match the batteries in your unit prior to placing your order. Our products are not affiliated with or authorized by APC.
Unsuccessful	You MUST RE-USE: your existing cabling and hardware. It is your responsibility to verify the batteries being ordered match the batteries in your unit prior to placing your order. Our products are not affiliated with or authorized by APC.

Table 4: 2nd Pattern: Generating the Same Descriptions.

Style	Description
Original	Higher capacity will takes **3-4 hours** to fully charge the battery.
Successful	High capacity can replace the battery to charge large volumes of batteries **quickly**.
Unsuccessful	High-capacity battery will takes **more time to fully charge**.

Table 5: 3rd Pattern: Hallucinations.

smaller weights for the reconstruction loss resulted in outputs that were grammatically incorrect and not viable as sentences. Therefore, balancing the generation of valid sentences with the capability to modify texts towards more successful or unsuccessful versions presents a key area for future research.

The third observed pattern is a tendency towards hallucinations, as shown in Table 5. In the provided example, the original text states that charging the battery takes 3 or 4 hours. However, when we transform the text into a more successful description, it incorrectly claims that the charge will finish quickly. This pattern, akin to the first one, poses challenges for automating product description writing. On one hand, the features emerging in the successful direction could be leveraged for product improvement.

5. Conclusion

In conclusion, this study contributes to the evolving field of e-commerce NLP by taking initial steps in extracting insights about product success from product descriptions.

Our research addressed two research questions: “To what extent can product descriptions accurately indicate a product’s likelihood of success?” and “Can we transform text descriptions from unsuccessful to successful ones, or vice versa?”

Our findings for the first research question revealed that product descriptions were indeed good predictors of the products’ success with an accuracy of around 70% in success prediction tasks, showing that they contain meaningful information about the product’s success. On one hand, the exact mechanisms behind this successful prediction remain an area for future exploration.

Regarding the second research question, we proposed a Successful Style Transfer Variational Autoencoder (SST-VAE). Our findings indicated that the proposed model successfully transforms the product descriptions from unsuccessful to successful and vice versa through qualitative analysis and case studies. Also, we showed that the proposed model is effective in extracting insights about products by highlighting the successful factors. On one hand, qualitative analysis revealed challenges such as hallucinations and the balance between factual accuracy and style transformation in the generated text. These findings pave the way for future research focused on addressing these challenges to progress the domain of e-commerce NLP.

6. Bibliographical References

Zhangming Chan, Xiuying Chen, Yongliang Wang, Juntao Li, Zhiqiang Zhang, Kun Gai, Dongyan Zhao, and Rui Yan. 2019. Stick to the facts: Learning towards a fidelity-oriented E-commerce product description generation. In *Proceed-*

ings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4959–4968, Hong Kong, China. Association for Computational Linguistics.

Qibin Chen, Junyang Lin, Yichang Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Towards knowledge-based personalized product description generation in e-commerce. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 3040—-3050, New York, NY, USA. Association for Computing Machinery.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International conference on machine learning*, pages 1587–1596. PMLR.

Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. 2022. Deep learning for text style transfer: A survey. *Computational Linguistics*, 48(1):155–205.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled representation learning for non-parallel text style transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 424–434, Florence, Italy. Association for Computational Linguistics.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *ICLR*.

Chunyuan Li, Xiang Gao, Yuan Li, Baolin Peng, Xiujun Li, Yizhe Zhang, and Jianfeng Gao. 2020. Optimus: Organizing sentences via pre-trained modeling of a latent space. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4678–4699, Online. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Jake Vasilakes, Chrysoula Zerva, Makoto Miwa, and Sophia Ananiadou. 2022. Learning disentangled representations of negation and uncertainty. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8380–8397, Dublin, Ireland. Association for Computational Linguistics.

Jinpeng Wang, Yutai Hou, Jing Liu, Yunbo Cao, and Chin-Yew Lin. 2017. A statistical framework for product description generation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 187–192, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Tao Zhang, Jin Zhang, Chengfu Huo, and Weijun Ren. 2019. Automatic generation of pattern-controlled product description in e-commerce. In *The World Wide Web Conference*, WWW '19, pages 2355—-2365, New York, NY, USA. Association for Computing Machinery.

Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. 2018. Opentag: Open attribute value extraction from product profiles. KDD '18, pages 1049—-1058, New York, NY, USA. Association for Computing Machinery.

STA: Self-controlled Text Augmentation for Improving Text Classifications

Congcong Wang[†]*, Gonzalo Fiz Pontiveros[‡], Steven Derby[‡], Tri Kurniawan Wijaya[‡]
[†] School of Computer Science, University College Dublin, Dublin 4, Ireland
[‡] Huawei Ireland Research Centre. Georges Court, Townsend St, Dublin 2, D02 R156, Ireland
[†]congcong.wang@ucdconnect.ie
[‡]{gonzalo.fiz.pontiveros, steven.derby, tri.kurniawan.wijaya}@huawei.com

Abstract

Despite recent advancements in Machine Learning, many tasks still involve working in low-data regimes which can make solving natural language problems difficult. Recently, a number of text augmentation techniques have emerged in the field of *Natural Language Processing* (NLP) which can enrich the training data with new examples, though they are not without their caveats. For instance, simple rule-based heuristic methods are effective, but lack variation in semantic content and syntactic structure with respect to the original text. On the other hand, more complex deep learning approaches can cause extreme shifts in the intrinsic meaning of the text and introduce unwanted noise into the training data. To more reliably control the quality of the augmented examples, we introduce a state-of-the-art approach for *Self-Controlled Text Augmentation* (STA). Our approach tightly controls the generation process by introducing a self-checking procedure to ensure that generated examples retain the semantic content of the original text. Experimental results on multiple benchmarking datasets demonstrate that *STA* substantially outperforms existing state-of-the-art techniques, whilst qualitative analysis reveals that the generated examples are both lexically diverse and semantically reliable.

Keywords: Natural language processing, text generation, data augmentation

1. Introduction

A variety of tasks such as *Topic Classification* (Li and Roth, 2002), *Emotion Detection* (Saravia et al., 2018) and *Sentiment Analysis* (Socher et al., 2013) have become important areas of research in NLP. Such tasks generally require a considerable amount of accurately labelled data to achieve strong performance. However, acquiring enough such data is both costly and time-consuming, hence making it rare in practice. This has motivated a vast body of research in techniques that can help alleviate issues associated with low-data regimes.

A popular augmentation approach involves the use of rule-based transformations, which employ intuitive heuristics based on well-known paradigmatic relationships between words. For instance, by using a lexical-semantic database such as *WordNet* (Miller, 1995), researchers can make rational and domain-specific conjectures about suitable replacements for words from lists of known synonyms or hyponyms/hypernyms (Wang and Yang, 2015; Wei and Zou, 2019; Feng et al., 2020). Whilst these substitution-based approaches can result in novel and lexically diverse data, they also tend to produce highly homogeneous structures, even when context-free grammars are used to generate more syntactically variable examples (Jia and Liang, 2016).

The recent success of pretrained transformer language models such as BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019) has helped facilitate more robust strategies for dealing with low-resource scenarios: Conditional text generation. Large language models — typically trained on a vast corpus of text — contain a rich understanding of syntactic structure and semantic phenomena and thus are well suited for faithful domain-specific generation (Petroni et al., 2019). Indeed, large language models have been employed to great success (Kobayashi, 2018; Wu et al., 2019; Anaby-Tavor et al., 2020; Kumar et al., 2020) to synthesize highly diverse training examples resulting in stronger downstream performance in low-resource settings. However, the use of diverse neurally-generated data may come at the cost of introducing semantic discrepancies, which can cause misalignment between the generated samples and their intended labels. Ideally, the optimal augmentation method would be one that satisfies both **Lexical/Syntactic Diversity** and **Semantic Fidelity** (reliable alignment between semantic meaning and class label).

In this paper, we propose a novel strategy — self-controlled text augmentation (STA) that aims to tightly control the generation process in order to produce diverse training examples which retain a

*The author completed this work during his internship at Huawei Ireland Research Center.

The Seventh Workshop on e-Commerce and NLP (ECNLP 7), pages 97–114

high level of semantic fidelity. Following previous work, we fine-tune a state-of-the-art sequence-to-sequence transformer model, in this case *T5* (Raffel et al., 2020), using a dataset containing only a limited number of samples and generate new samples using task-specific prompting, which has been shown to be effective in low-resource scenarios (Le Scao and Rush, 2021). While similar approaches have been deployed in previous work (Anaby-Tavor et al., 2020), our novel strategy effectively utilizes *Pattern-Exploiting Training* (Schick and Schütze, 2021a,b) by employing templates of verbalization-patterns that simultaneously direct the generation process and filter noisy labels within a single unified framework. Experimental results on multiple benchmarks demonstrate that STA outperforms existing state-of-the-art augmentation techniques. Furthermore, examining the quality of the augmented data reveals better diversity and fidelity as compared to the existing techniques.

2. Related Work

Various text augmentation techniques have been proposed in the literature (Feng et al., 2021). Zhang et al. (2015) and Wei and Zou (2019) use simple operations like synonym replacement, random insertion, swap, and deletion to generate new samples. Feng et al. (2020) further explores these substitution techniques for text generation. In contrast, Wang and Yang (2015) and Kobayashi (2018) use word embeddings and contextual language models, respectively, to replace words or phrases with semantically similar concepts.

Back translation is another effective method for text augmentation, transforming sentence between languages (Sennrich et al., 2016; Shleifer, 2019). Recently, researchers have explored the use of pretrained transformer-based language models for conditional text augmentation to generate novel sentences from the original data (Wu et al., 2019; Anaby-Tavor et al., 2020; Kumar et al., 2020). For instance, Wu et al. (2019) leveraged BERT's masked language model, while Anaby-Tavor et al. (2020) fine-tuned GPT-2 to generate novel sentences and filter out noisy ones using a jointly trained classifier with some success in tackling the label misalignment problem. Similarly, Kumar et al. (2020) studied conditional text augmentation using transformer-based models, with BART outperforming other methods in low-resource settings

Building upon ideas presented in the GPT series (Radford et al., 2018, 2019; Brown et al., 2020; Ubani et al., 2023), prompt-based templates have become and effective approach for eliciting latent knowledge from language models to great success (Trinh and Le, 2018; Petroni et al., 2019; Davison et al., 2019; Talmor et al., 2020; Le Scao and Rush, 2021). For example, Wang et al. (2021) proposed using GPT-3 for text augmentation with zero-label learning, with results that were competitive when compared to fully supervised approaches. Yoo et al. (2021a) instead generate augmented text examples by using soft-labels from GPT-3 to distil the knowledge. More recently, (Ubani et al., 2023) investigated the use of ChatGPT-generated data for augmenting training data via prompting in low-resource scenarios, surpassing existing methods with task-specific prompts. More closely related to our instruction-based generation strategy, Schick and Schütze (2021b) propose GenPet which is used to directly tackle a number of text generation tasks rather than text augmentation itself. In their work, which builds upon previous research PET (Schick and Schütze, 2021a), the authors alter the text inputs to form cloze-style questions known as prompting training (Liu et al., 2021), demonstrating improved performance on few-shot downstream tasks. Finally, researchers have proposed an array of techniques aiming to systematically engineer the structure of these templates beyond ad hoc human intuitive reasoning: For example, using automated template generation for the tasks (Shin et al., 2020; Gao et al., 2021), trained end-to-end with soft-prompts (Lester et al., 2021; Gu et al., 2022) or designed from sub-prompts created by decomposing prior task knowledge into rules (Han et al., 2022).

Our approach differs from prior work by using task-specific templates as verbal prompts for generation and classification which signal the model's objective. The model itself is self-controlling, generating novel data and retaining only the most convincing examples using a classification template to ensure semantic fidelity.

3. Method

In this section, we describe our novel self-controlled approach for text augmentation in text classification (STA). Figure 1 illustrates the workflow of STA and Algorithm 1 states STA in simple terms. At a high level, STA first finetunes a pretrained sequence-to-sequence (seq2seq) model using a dataset which implicitly includes generation and classification tasks.

3.1. Pattern-Exploiting Training in seq2seq Models

PET is a finetuning technique for text classification tasks in masked language models, as demonstrated in (Schick and Schütze, 2021a). By converting inputs into cloze questions, PET enables accurate classification with minimal labeled data. In this paper, we extend the principles of PET to seq2seq

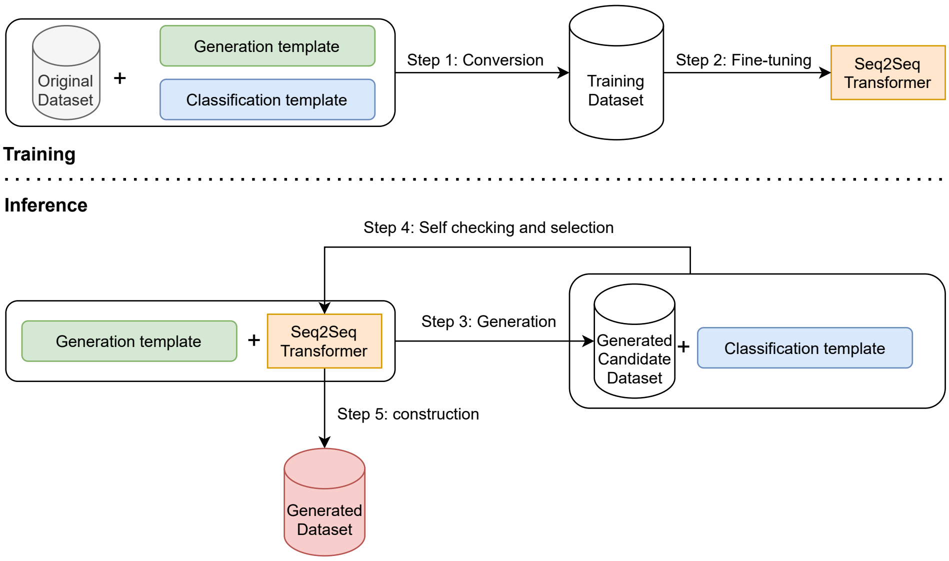


Figure 1: The architecture of our Self-controlled Text Augmentation approach (STA). The upper portion outlines the finetuning component of our method (**Training**), whilst the lower portion demonstrates our procedure for generating novel data (**Inference**). STA is highlighted by using the generation template and classification template for fine-tuning a seq2seq transformer model. The generation template is used for generating samples and the classification template is used for self-controlling and selecting the generated samples.

Algorithm 1 :Self-Controlled Text Augmentation

Require: Original dataset $\mathcal{D}_o$. Generative model M. Generation template $\mathcal{G}$. Classification template $\mathcal{C}$.

1: Convert $\mathcal{D}_o$ to training dataset $\mathcal{D}_t$ via $\mathcal{G}$ and $\mathcal{C}$.
2: Finetune M on $\mathcal{D}_t$ in a generation task and a classification task jointly to obtain M_t.
3: Use $\mathcal{G}$ and M_t to generate candidate dataset $\mathcal{D}_c$.
4: Apply M_t to do classification inference on $\mathcal{D}_c$ with $\mathcal{C}$ to select the most confident examples.
5: Form the final generated dataset $\mathcal{D}^*$ with the selected examples.

autoregressive models, presenting a novel strategy for prompting-based generation and our innovative self-controlled approach.

Consider a pretrained seq2seq autoregressive transformer model denoted as M (we use T5 (Raffel et al., 2020) in our experiments). This type of model comprises an encoder-decoder pair, where the encoder takes an input sequence s and generates a contextualized encoded sequence $\overline{s}$. The decoder then takes the encoded sequence and the current subsequence t: $\{t_1, t_2, ..t_{i-1}\}$ as input to compute the conditional distribution $p_M(t_i|t_{1:i-1}, \overline{s})$ for the subsequent token in the sequence. Given $\overline{s}$, the possible target sample (a sequence) t: $\{t_1, t_2, ..., t_m\}$ can be obtained via the factorization:

$$p_M(t_{1:m}|\overline{s}) = \prod_{i=1}^{m} p_M(t_i|t_{1:i-1}, \overline{s}) \tag{1}$$

Let $\mathcal{D}_o = \{(x_i, y_i)\}|_{i=1}^{n}$ be a dataset for text classification where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{L}$ are text and label respectively. The goal is to produce a derived dataset $\mathcal{D}_t$ to finetune M and ensure it is primed for generating diverse and (label) faithful examples by leveraging a set of prompt templates.

Formally, a *template* is a function $T : V^* \times \mathcal{L} \to V^* \times V^*$ where V is the vocabulary of M and V^* denotes the set of finite sequences of symbols in V. Of course, the structure of these templates can be quite malleable. For example, a template could be constructed through human interpretable verbalizable terms, optimized automatically for the task, fine-tuned with soft prompts or made up of sequentially intuitive sub-prompts. Regardless of the approach, the process is the same.

Given a family of templates $\mathcal{T}$, we set $\mathcal{D}_t = \mathcal{T}(\mathcal{D}_o) = \bigcup_{T\in\mathcal{T}} T(\mathcal{D}_o)$. That is, we convert each sample $(x_i, y_i) \in \mathcal{D}_o$ to $|\mathcal{T}|$ samples in the derived dataset D_t. In the field of synthetic data generation for low-resource scenarios, these templates generally belong to the collection of templates capable of generating novel examples. Crucially, we extend these templates to consider two types of template families: generation templates $\mathcal{G}$ and classification templates $\mathcal{C}$, such that $\mathcal{T} = \mathcal{C} \cup \mathcal{G}$. As we shall demonstrate, by carefully considering these templates, we can produce a dataset D_t (generated from these templates $\mathcal{T}$ applied to the dataset D)

that is designed in such as way that the model can learn to directly optimize for key characteristics: High semantic fidelity and lexical diversity.

3.1.1. Generation templates

Though not exclusive to the field, these templates are commonplace within the synthetic data generation literature for creating novel training examples. Since our work focuses on encoder-decoder models, the templates take the form $g(x, y) = (f_s(x, y), f_t(x))$, where f_s and f_t denote functions that map a piece of text to a source sequence and target sequence respectively. Concretely, the source function f_s is a verbalizable mapping which depends on the text $x \in \mathcal{X}$ and label $y \in \mathcal{L}$, the latter of which conditions the model to align the generated text with the labels. The target function f_t on the other hand, represents the desired output of the model, which depends on the text, and typically corresponds to the identity function.

Diverse Generation. Without loss of generality, for a given downstream task {Task}, we could choose the primary template f_s = *Description:* $\{y_i\}$ $\{Task\}$*. Text:* as our source function and $f_t = \{x_i\}$ as the desired target for fine-tuning to facilitate the generation process, following previous work (Anaby-Tavor et al., 2020; Schick and Schütze, 2021b,a). Here the goal of $Task$ is to provide context about the dataset, since providing this sort of information helps when there are limited training examples (Schick and Schütze, 2021b). In this work, our goal is not only to generate novel sythetic examples for few-shot classification, but to generate a diverse variety of these samples. To ensure the model produces lexically diverse text, we propose a simple yet effective generation strategy which additionally includes an auxiliary template for generation by including prior knowledge, somewhat similar to *Paragraph2Paragraph* and *Shard2Shard* settings from (Chen and Liu, 2022). Given some data point (x_i, y_i) we achieve diversity by modifying two components to our source and target functions.

- **Memory**: We add a previous example of text x_j which share the same label as an input to the source function, $j \in \mathbb{N}$ such that $y_j = y_i$.
- **Priming**: We instantiate the source function with some of the target output $x_i^{0-n}, n < |x_i| \in \mathcal{N}$, which further constrains the model to avoid the generation of non-factual hallucinations (Cao et al., 2022).

Concretely, we define a second auxiliary template function for generation $g'(x_i, x_j, y_i) = (f'_s(x_i, x_j, y_i), f'_t(x_i))$, with the source function $f'_s =$ $f_s(x_j, y_i)$*. Another text:* $\{x_i^{0-2}\}$ and target function $f'_t =$ $\{x_i^{3\cdots}\}$ where $y_j = y_i$, facilitating these modifications. Intuitively, we use a previous example as prior knowledge before concatenating them with the new template to ensure the model produces distinct examples as opposed to repetitions, with similar findings demonstrated by Chen and Liu (2022). It's worth mentioning that the g' function can be employed multiple times to create various examples by sampling different texts during the conversion of a single training example (we present an example of how an original training sample is converted by the templates in Appendix B). For generation, we include both templates g and g' for tuning our model. These templates are further outlined in Table 1.

3.1.2. Classification templates

Classification has been employed as an additional processing step to filter synthetic examples which do not align with the generated label (Anaby-Tavor et al., 2020). In previous work, a separate network is trained using the original data to classify the examples, based on the intuition that checking the results is easier than producing new examples. One problem that emerges from adding a filter in low-resource settings is that it creates an additional layer of complexity within the system: Not only must the generator predict the correct label from limited data, but so must the classifier. These templates take the form $c(x, y) = (f_s(x), f_t(y))$ where f_t and f_s similarly denote the source sequence and target sequence functions respectively. In this case, the source functions are similar to the generation templates (the text can be conditioned on the labels or be completely independent), although the target function instead relates to the label or some semantically compatible class. In this case we set the source function as $f_s(x_i) =$ *Given {Task}: {$\mathcal{L}$}. Classify:* $\{x_i\}$ and target function as $f_t(y_i) =$ y_i, with $\mathcal{L}$ providing context to the possible labels.

Semantic Fidelity. Although prompt-based tuning has proven to work better in limited data settings than simple feed-forward approaches (Le Scao and Rush, 2021), we further supplement the template dataset by generating multiple intuitive patterns following previous work (Schick and Schütze, 2021a). To achieve this, we extend our base classification templates with two more auxiliary templates which we refer to as c_{pos} and c_{neg} in the vein of cloze-style questions. Concretely, we define $c_{pos} = (f_s(x_i), f_t(y_i))$ such that $f_s =$ *Text: {x_i}. Is this text about {y_i} {Task}?* and f_t= *yes*, with the goal of classifying whether the correct label conforms to the

Template			Source seq. (s)	Target seq. (t)
Classification	Primary	c	*Given {Task}: {$\mathcal{L}$}. Classify: {x_i}*	{y_i}
	Auxiliary	c_{pos}	*Text: {x_i}. Is this text about {y_i} {Task}?*	*yes*
	Auxiliary	c_{neg}	*Text: {x_i}. Is this text about {$\overline{y}_i$} {Task}?*	*no*
Generation	Primary	g	*Description: {y_i} {Task}. Text:*	{x_i}
	Auxiliary	g'	*Description: {y_i} {Task}. Text: {x_j}. Another text: {$x_i^{0\text{-}2}$}*	{$x_i^{3\cdots}$}

Table 1: Prompt templates for training sequences conversion. "Task" refers to a simple keyword describing the dataset e.g. "Sentiment" or "Emotion" and $\mathcal{L}$ is the list of all class labels in the dataset. The symbol $\overline{y}_i$ in c_{neg} stands for any label in $\mathcal{L} \setminus \{y_i\}$, chosen randomly. In g', the x_j denotes another sample from the same class as x_i (i.e. $y_j = y_i$) chosen randomly.

text. Furthermore, we generate a counter template $c_{neg} = (f_s(x_i), f_t(y_i))$ such that $f_s =$ *Text: {x_i}. Is this text about {$\hat{y}_i$} {Task}?* and $f_t =$ *no* , $\hat{y}_i \sim \mathcal{L}\backslash\{y_i\}$, with the goal of determining that the incorrectly sampled label does not conform to the text. These templates are given in detail in Table 1.

Self-Checking. We note that these auxillary verbalizable patterns for classification are simply meant to supplement and do not represent the optimal solution for eliciting important knowledge from the network (Gao et al., 2021). We instead wish to avoid cascading errors between the generation and classification template: The classification network's performance should be within an acceptable tolerance. In order to extract synthetic examples with high levels of semantic alignment between the generated text and labels, we propose a novel strategy for controlled self-supervised data generation, which we refer to as *Self-Checking*. Different from previous work, we perform generation and classification filtering within a single unified neural framework. We hypothesise that this multiview learning process should allow the network to discover the semantic relationship between the labels and text, further preventing non-factual hallucinations of incorrect labels during the generation process.

3.2. Data Generation, Self-checking and Selection

We follow a two-step process: first we generate candidates and second we select a fraction of the candidates to be included as augmentations. This processes is conducted for each class separately so we may assume for the remainder of this section that we have fixed a label $y \in \mathcal{L}$.

That is, first, we generate $\alpha \times n_y$ samples where n_y is the original number of samples in $\mathcal{D}_o$ for label y and then select the top $\beta \times n_y$ samples ($\beta < \alpha$). In our experiments, we call β the *augmentation factor* and set $\alpha = 5 \times \beta$. Namely, our self-checking technique selects the top 20% of the candidate examples per class [1] to form the final generated D^* that is combined with the original dataset D_o for downstream model training.

For the generation task, we need to choose a prefix/source sequence s and proceed autoregressively using Equation 1. Referring back to Table 1, there are two choices g and g' that can be used to construct s. In this work, we employ g for generating examples because it allows for greater flexibility in generating diverse examples. We aim to generate as many diverse examples as possible at this stage (rather than selecting g', which requires a few initial words from an existing example as the context and can restrict the freedom of generating diverse examples). Nevertheless, all generated samples will be self-checked for semantic fidelity next. Here we generate $\alpha \times n_y$ samples using the finetuned encoder-decoder model M_t where α is the factor controlling the size of our synthetic dataset of generated examples in comparison to the original dataset.

Now that we have gathered a synthetic candidate dataset $\mathcal{D}_c^y = \{(x_i, y)\}|_{i=1}^{\alpha \times n_y}$, we will further refine these examples using a self-checking strategy for selecting the generated samples based on the confidence estimated by the model M_t itself. For each synthetic sample (x, y), we construct a source sequence using the classification template $c(x, y)$ as described in Table 1 to generate the source s. Given the source s, we define a score function u:

$$u(y|s) = \log p_{M_t}(\{y\}|\overline{s})$$

equivalently this is the *logit* computed by M_t for the sequence $\{y\}$. We then renormalize over the labels in $\mathcal{L}$ by applying a softmax over each of the scores $u(\cdot|s)$:

$$q(y|s) = \frac{e^{u(y|s)}}{\sum_{l \in \mathcal{L}} e^{u(l|s)}}$$

[1] This is based on our experimental search over {10%, 20%, 30%, 40%, 50%}.

Finally, we rank the elements of $\mathcal{D}_c^y$ by the value of q and select the top $\beta \times n_y$ samples to form the dataset D_*^y and set $D_* = \bigcup_{y \in \mathcal{L}} D_*^y$

4. Experiments

Next, we conduct extensive experiments to test the effectiveness of our approach in low-data regimes. This section first describes the datasets choices, and then presents the baselines for comparison.

Regarding experimental setup, we select the pre-trained T5 base checkpoint as the generation model and BERT base as the classification model. For the augmentation factor (i.e., β in Section 3.2), the augmentation techniques including ours and the baselines are applied to augment 1 to 5 times of original training data. To be in low-data settings, we sampled 5, 10, 20, 50 and 100 examples per class for each training dataset as per Anaby-Tavor et al. (2020). To alleviate randomness, we run all experiments 10 times so the average accuracy along with its standard deviation (std.) is reported on the full test set. We report more experimental details for reproducibility in Appendix D.

4.1. Datasets

Following previous work in the augmentation literature (Kumar et al., 2020; Anaby-Tavor et al., 2020), two bench-marking datasets are used in our experiments: **SST-2** (Socher et al., 2013) and **TREC** (Li and Roth, 2002). We also include **EMOTION** (emotion classification) (Saravia et al., 2018) and **HumAID** (crisis tweets categorisation) (Alam et al., 2021) to extend the domains of testing STA's effectiveness. More details about the datasets refer to Appendix C.

4.2. Baselines

We evaluate our novel strategy against a set of state-of-the-art techniques found within the literature. These approaches include a variety of augmentation procedures from rule-based heuristics to deep neural text generation. We compare STA to the augmentation techniques as they are directly related to our method in generating samples that can be used in our subsequent study for examining the quality of generated examples[2].

Baseline: No data augmentation is applied to the original training data.

EDA (Wei and Zou, 2019): Easy Data Augmentation involves applying local word-level changes to an existing example, such as synonym replacement and random insertion.

BT and BT-Hops (Edunov et al., 2018; Shleifer, 2019): Back-translation techniques involve translating from English to one (BT) or more randomly selected languages (BT-Hops) using a pre-trained translation model.

GPT-2 (Kumar et al., 2020) and GPT-2-λ (Anaby-Tavor et al., 2020): GPT-2 generates new examples conditioned on the label description and the first three words of an existing example. GPT-2-λ adds the LAMBDA technique, which selects generated examples based on the performance of the downstream classification model on the original training data.

CBERT (Wu et al., 2019): it is a strong word-replacement based method for text augmentation that replaces words in the original examples while conditioning on the labels.

BART-Span (Kumar et al., 2020): it finetunes the large model BART (Lewis et al., 2020) based on the label names and the texts of 40% consecutive masked words to generate new examples.

5. Results and Discussion

5.1. Classification Tasks

Table 2 demonstrates the results of STA in comparison to baselines under low-data conditions for the **SST-2** classification task. The results of the remaining three classification tasks can be interpreted similarly[3]In all cases, our approach provides state-of-the-art performance for text augmentation across all low-resource settings. When a higher number of samples $(50\text{-}100)$[4] are used for training we see that STA is better, as in the cases of SST-2, EMOTION and HumAID tasks, or competitive, as in the case of TREC. Furthermore, we can see that STA is superior to other augmentation techniques when only a small number of examples are used to train the generator $(5\text{-}10\text{-}20)$. In fact, STA on average demonstrates a difference of $+9.4\Delta$ and $+4.7\Delta$ when trained on only 5 and 10 samples per class respectively, demonstrating its ability to generate salient and effective training examples from limited amounts of data.

[2]We have gathered the results of a direct comparison between STA and existing non-augmentation few-shot baselines on downstream classification tasks and report them in Appendix E.

[3]Due to page constraints, we have these results in Appendix F. If accepted, we will move these results to the main paper

[4]We note that around 100 examples per class, all techniques tend to approximate no augmentation baselines, indicating that most likely constitute something more equivalent to full data training rather than a low-resource setting

Augmentation Method	5	10	20	50	100
Baseline (No Aug.)	56.5 (3.8)	63.1 (4.1)	68.7 (5.1)	81.9 (2.9)	85.8 (0.8)
EDA (Wei and Zou, 2019)	59.7 (4.1)	66.6 (4.7)	73.7 (5.6)	83.2 (1.5)	86.0 (1.4)
BT (Edunov et al., 2018)	59.6 (4.2)	67.9 (5.3)	73.7 (5.8)	82.9 (1.9)	86.0 (1.2)
BT-Hops (Shleifer, 2019)	59.1 (4.6)	67.1 (5.2)	73.4 (5.2)	82.4 (2.0)	85.8 (1.1)
CBERT (Wu et al., 2019)	59.8 (3.7)	66.3 (6.8)	72.9 (4.9)	82.5 (2.5)	85.6 (1.2)
GPT-2 (Kumar et al., 2020)	53.9 (2.8)	62.5 (3.8)	69.4 (4.6)	82.4 (1.7)	85.0 (1.7)
GPT-2-λ (Anaby-Tavor et al., 2020)	55.4 (4.8)	65.9 (4.3)	76.2 (5.6)	84.5 (1.4)	86.4 (0.6)
BART-Span (Kumar et al., 2020)	60.0 (3.7)	69.0 (4.7)	78.4 (5.0)	83.8 (2.0)	85.8 (1.0)
STA w/o Self-Checking	66.7 (5.0)	77.1 (4.7)	81.8 (2.1)	84.8 (1.0)	85.7 (1.0)
STA w/o Auxiliary Prompts	69.8 (4.9)	79.1 (3.4)	81.7 (4.5)	**86.0 (0.8)**	**87.5 (0.6)**
STA (ours)	**72.8 (6.2)**	**81.4 (2.6)**	**84.2 (1.8)**	**86.0 (0.8)**	87.2 (0.6)

Table 2: STA on **SST-2** in $5, 10, 20, 50, 100$ examples per class. The results are reported as average (std.) accuracy (in %) based on 10 random experimental runs. Numbers in **bold** indicate the highest in columns.

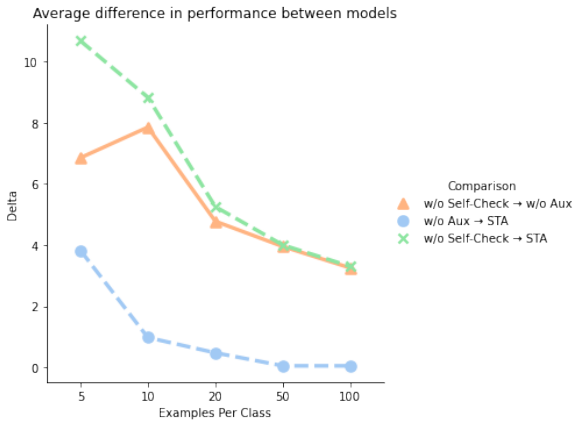


Figure 2: Graph showing the average difference between **STA w/o Self-Checking** to **STA w/o Auxiliary Prompts**, **STA w/o Auxiliary Prompts** to **STA** and **STA w/o Self-Checking** to **STA**, as the number of examples per class varies.

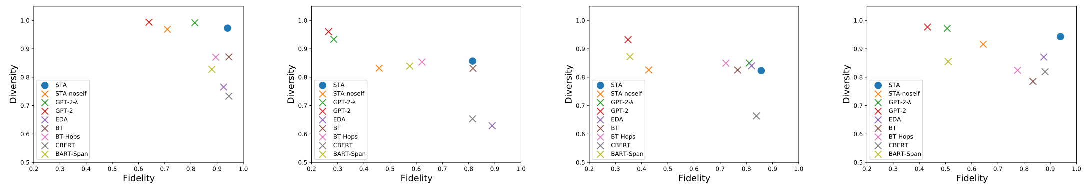
5.2. Ablation Studies: Self-Checking and Auxiliary Prompts

To demonstrate the importance of our self-checking procedure, we performed our empirical investigations on STA both with and without the self-checking step, denoted as **STA w/o Self-Checking** in Table 2, 7, 8 and 9. Furthermore, we investigate STA within a minimal template setting where we only include the templates c and g in Table 1, omitting our proposed auxiliary templates, denoted as **STA w/o Auxiliary Prompts**, to empirically separate the contribution of these components. Comparing our model with no self-checking (STA w/o Self-Checking) against other state-of-the-art approaches, we see that the model provides the best performance particularly when the data is more sparse (5-10-20), with the exclusion of TREC. However, when we add self-checking with only basic generation and classification templates (STA w/o Auxiliary Prompts), we see a significant improvement, indicating that self-checking is more important to the downstream performance. We also compare the average difference between these models across all datasets with altering components in Figure 2. Looking at Figure 2 we see that the inclusion of self-checking provides the greatest increase in performance, while the contribution of our auxiliary prompts, including our novel generation template, decreases with larger examples per class. However, we note that the inclusion of both templates and self-checking provides the best performance, particularly in lower data regimes.

5.3. Lexical Diversity and Semantic Fidelity

To further analyse the quality of the generated data, we measure its lexical diversity and semantic fidelity (i.e., its ability to align the synthetic examples with the correct labels). **Diversity** is assessed using the UNIQUE TRIGRAMS metric (Feng et al., 2020; Kumar et al., 2020), which calculates the ratio of unique tri-grams to total tri-grams in a population consisting of both original and generated training data. To coincide with the previous work (Kumar et al., 2020), **semantic fidelity** is determined by fine-tuning a “BERT-base-uncased” model on 100% of the original training data for each classification task and measuring the accuracy of the generated data predictions by this model (91.8, 93.5, 96.6 and 89.7 accuracy on **SST2**, **EMOTION**, **TREC** and **HumAID** respectively). A higher score indicates better diversity or fidelity.

To present the quality of generated data in terms of diversity and fidelity, we take the training data (10 examples per class) along with its augmented data ($\beta = 1$) for investigation. Figure 3 depicts the diversity versus semantic fidelity of generated data by various augmentation methods across three datasets. We find that generation-based approaches such as GPT-2 or GPT-2-λ, achieve



(a) **SST-2** (b) **EMOTION** (c) **TREC** (d) **HumAID**

Figure 3: Diversity versus semantic fidelity of generated texts by various augmentation methods. The average scores over 10 runs are reported.

strong diversity but less competitive fidelity. On the contrary, rule-based heuristics methods such as EDA perform well in retaining the semantic meaning but not in lexical diversity. The merit of STA is that it is good in both diversity and fidelity, as seen from its position at the top-right of Figure 3a, 3b, 3c and 3d. Finally, if we compare our STA approach with and without self-checking, we see that each approach produces highly diverse examples, although only self-checking STA retains a high level of semantic fidelity. Comparing with GPT-2 and GPT-2-λ — the other sample filtering approach — we see that the inclusion of a separate classifier results in an average increase of 18.3% in fidelity. However, if we compare our STA approach with and without self-checking, we see an average increase of 32.38% in fidelity, further demonstrating the validity of our joint generation and classification approach as opposed to an independent classification module. As previously suggested, this ability to align the semantic content of generated examples with the correct label is the most probable reason for the increase in downstream classification performance when self-checking is employed. This supports the notion that our generation-based approach is able to produce novel data that is lexically diverse, whilst the self-checking procedure can ensure consistent label retention, which guarantees a high semantic fidelity in the generated examples[5].

5.4. Comparison Against ChatGPT

In addition to comparing STA and examining its augmented data quality with augmentation baselines in the literature, we are also interested in evaluating the performance of STA against ChatGPT (using GPT 3.5 (Ouyang et al., 2022) and GPT-4 (OpenAI, 2023)). Because of the limited context length in GPT-3.5 and restricted access to GPT-4, we designed this experiment to operate within a highly constrained, very low-data regime. Specifically, we sampled a mere five examples per class from **SST2**, **EMOTION**, **TREC**, and **HumAID**. Utilising these samples as input prompts, we tasked GPT-3.5 and GPT-4 with generating predictions for a number of randomly selected test set examples. Essentially, we are evaluating their performance as classifiers. In this study, we're assessing ChatGPT's direct (few-shot) prediction capabilities in situations with limited data. Meanwhile, STA employs augmentation strategies in similar low-data scenarios but with significantly smaller models. To account for the inherent randomness in both the sampling and generation processes of GPT-3.5 and GPT-4, we conducted multiple iterations of the experiments. This allowed us to calculate the average accuracy (std.) in relation to the actual labels and predictions made by GPT-3.5 and GPT-4. The comparative analysis between STA and GPT-3.5 and GPT-4 across the four datasets is presented in Table 3. The results showcased STA's consistent superiority over GPT-3.5 across all datasets. Furthermore, STA demonstrated competitive performance with GPT-4 on three of the four datasets as STA competes closely with GPT-4 in **SST2**, **EMOTION**, and **HumAID**. However, GPT-4 exhibits unexpectedly high performance in TREC, with an average accuracy of 100.0 (std: 0.0), suggesting a potential issue with data contamination, something that the original authors were also extremely concerned about in their evaluations (OpenAI, 2023). These findings reinforce STA's potential as a augmentation solution for text classification in data-constrained scenarios, while also highlighting its computational efficiency and speed of inference in comparison to GPT-3.5 and GPT-4.

	SST2	EMOTION	TREC	HumAID
GPT(3.5)	64.6(4.0)	40.4(7.8)	32.2(12)	58.8(18)
GPT(4.0)	71.9(5.0)	42.5(7.8)	**100.0(0.0)**	68.8(4.4)
STA	**72.8(6.2)**	**43.8(6.9)**	59.6(7.4)	**69.0(3.9)**

Table 3: Comparing STA with ChatGPT (GPT-3.5 and GPT-4). Average (std.) accuracy (in %) over multiple runs is reported.

[5]We provide a comparative demonstration of the texts generated by different methods in the Appendix G

6. Conclusion

We propose a novel strategy for text-based data augmentation that leverages prompt templates to generate training examples and ensure better label alignment. Our approach substantially outperforms the previous state-of-the-art on a variety of downstream classification tasks and across a range of low-resource scenarios. Furthermore, we provide an analysis of the lexical diversity and label consistency of generated examples, demonstrating that our approach produces uniquely varied training examples with more consistent label alignment than previous work. In the future, we hope to improve this approach in rich-data regime and extend it to other downstream natural language tasks.

7. Bibliographical References

Firoj Alam, Umair Qazi, Muhammad Imran, and Ferda Ofli. 2021. Humaid: Human-annotated disaster incidents data from twitter with deep learning benchmarks. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 15, pages 933–942.

Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2020. Do not have enough data? deep learning to the rescue! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7383–7390.

N. N. Author. 2021. Suppressed for anonymity.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

BSI. 1973a. *Natural Fibre Twines*, 3rd edition. British Standards Institution, London. BS 2570.

BSI. 1973b. Natural fibre twines. BS 2570, British Standards Institution, London. 3rd. edn.

Meng Cao, Yue Dong, and Jackie Chi Kit Cheung. 2022. Hallucinated but factual! inspecting the factuality of hallucinations in abstractive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3340–3354.

A. Castor and L. E. Pollux. 1992. The use of user modelling to guide inference and learning. *Applied Intelligence*, 2(1):37–53.

Yanan Chen and Yang Liu. 2022. Rethinking data augmentation in text-to-text paradigm. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1157–1162.

J.L. Chercheur. 1994. *Case-Based Reasoning*, 2nd edition. Morgan Kaufman Publishers, San Mateo, CA.

N. Chomsky. 1973. Conditions on transformations. In *A festschrift for Morris Halle*, New York. Holt, Rinehart & Winston.

Joe Davison, Joshua Feldman, and Alexander M Rush. 2019. Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 1173–1178.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*.

R. O. Duda, P. E. Hart, and D. G. Stork. 2000. *Pattern Classification*, 2nd edition. John Wiley and Sons.

Umberto Eco. 1990. *The Limits of Interpretation*. Indian University Press.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *EMNLP*.

Steven Y Feng, Varun Gangal, Dongyeop Kang, Teruko Mitamura, and Eduard Hovy. 2020. Genaug: Data augmentation for finetuning text generators. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 29–42.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 968–988, Online. Association for Computational Linguistics.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Association for Computational Linguistics (ACL)*.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2022. Ppt: Pre-trained prompt tuning for few-shot learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 8410–8423.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2022. Ptr: Prompt tuning with rules for text classification. *AI Open*, 3:182–192.

Paul Gerhard Hoel. 1971a. *Elementary Statistics*, 3rd edition. Wiley series in probability and mathematical statistics. Wiley, New York, Chichester. ISBN 0 471 40300.

Paul Gerhard Hoel. 1971b. *Elementary Statistics*, 3rd edition, Wiley series in probability and mathematical statistics, pages 19–33. Wiley, New York, Chichester. ISBN 0 471 40300.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*.

Otto Jespersen. 1922. *Language: Its Nature, Development, and Origin*. Allen and Unwin.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22.

M. J. Kearns. 1989. *Computational Complexity of Machine Learning*. Ph.D. thesis, Department of Computer Science, Harvard University.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457.

Huan Yee Koh, Jiaxin Ju, Ming Liu, and Shirui Pan. 2022. An empirical survey on long document summarization: Datasets, models, and metrics. *ACM computing surveys*, 55(8):1–35.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26, Suzhou, China. Association for Computational Linguistics.

P. Langley. 2000. Crafting papers on machine learning. In *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pages 1207–1216, Stanford, CA. Morgan Kaufmann.

Teven Le Scao and Alexander M Rush. 2021. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors. 1983. *Machine Learning: An Artificial Intelligence Approach, Vol. I*. Tioga, Palo Alto, CA.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

T. M. Mitchell. 1980. The need for biases in learning generalizations. Technical report, Computer Science Department, Rutgers University, New Brunswick, MA.

A. Newell and P. S. Rosenbloom. 1981. Mechanisms of skill acquisition and the law of practice. In J. R. Anderson, editor, *Cognitive Skills and Their Acquisition*, chapter 1, pages 1–51. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ.

OpenAI. 2023. Gpt-4 technical report.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

A. L. Samuel. 1959. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):211–229.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269.

Timo Schick and Hinrich Schütze. 2021b. Few-shot text generation with natural language instructions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 390–402.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235.

Sam Shleifer. 2019. Low resource text classification with ulmfit and backtranslation. *arXiv preprint arXiv:1903.09244*.

Charles Joseph Singer, E. J. Holmyard, and A. R. Hall, editors. 1954–58. *A history of technology*. Oxford University Press, London. 5 vol.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Jannik Strötgen and Michael Gertz. 2012. Temporal tagging on different domains: Challenges, strategies, and gold standards. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 3746–3753, Istanbul, Turkey. European Language Resource Association (ELRA).

S. Superman, B. Batman, C. Catwoman, and S. Spiderman. 2000. *Superheroes experiences with books*, 20th edition. The Phantom Editors Associates, Gotham City.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2020. olmpics-on what language model pre-training captures. *Transactions of the Association for Computational Linguistics*, 8:743–758.

Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

Solomon Ubani, Suleyman Olcay Polat, and Rodney Nielsen. 2023. Zeroshotdataaug: Generating and augmenting training data with chatgpt. *arXiv preprint arXiv:2304.14334*.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563.

Zirui Wang, Adams Wei Yu, Orhan Firat, and Yuan Cao. 2021. Towards zero-label language learning. *arXiv preprint arXiv:2109.09193*.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019. Conditional bert contextual augmentation. In *International Conference on Computational Science*, pages 84–95. Springer.

Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyoung Park. 2021a. GPT3Mix: Leveraging large-scale language models for text augmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2225–2239, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyoung Park. 2021b. Gpt3mix: Leveraging large-scale language models for text augmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2225–2239.

Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. 2022. Differentiable prompt makes pre-trained language models better few-shot learners. In *International Conference on Learning Representations*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657.

A. Limitations

Our work explores the possibility of data augmentation for boosting text classification performance when the downstream model is finetuned using pre-trained language models. The results show that STA consistently performs well across different bench-marking tasks using the same experimental setup, which addresses the limitation stated in the previous work (Kumar et al., 2020) calling for a unified data augmentation technique. However, similar to Kumar et al. (2020), although STA can achieve improved performance as the data size goes up to 100 examples per class in some cases (such as 100 examples per class in **EMOTION**, Table 7 and **HumAID**, Table 9), the absolute gain in performance plateaus when the training data becomes richer (such as 100 examples per class in **SST-2** and **TREC**). This suggests that it is challenging for STA to improve pre-trained classifier's model performance in more abundant data regimes.

It's also worth noting that STA currently applies to text classification exclusively using T5 and doesn't extend to other general NLP tasks without generative models. However, our approach, centered around text classification, holds the potential to expand beyond this narrow scope and encompass a wider array of NLP tasks. This flexibility arises from our use of generative models like T5 in our research. For instance, to consider the adaptation of our templates for question answering tasks, the templates used in our method can be modified to be like question-answer format. For instance, template c in Table 1 could be transformed to read, "*Given Text. Provide answer to this question: Question.*" Furthermore, our generation template can be suitably tailored to produce text based on a given question description. The capacity to adjust both classification and generation templates underpins the applicability of our approach across diverse NLP tasks. Thus, in the future, we aim to extend our approach to other downstream natural language tasks, incorporating different generation models alongside T5.

Another important consideration is the choice of templates used in STA. Ablation experiments in Section 5.2 show that our chosen set of templates yields better performance than a 'minimal subset' consisting of the two simplest templates; the question as to how to choose optimal templates for this augmentation scheme remains unanswered. Hence, in future work, we will explore better methods for constructing the prompt templates, aiming to reduce the dependency on the manual work at this step.

B. Template Example

Table 4 presents how an original training example is converted to multiple examples in STA using the prompt templates from Table 1.

C. Datasets

Table 5 lists the basic information of the four datasets used in our experiments and they are shortly described as follows.

- **SST-2** (Socher et al., 2013) is a binary sentiment classification dataset that consists of movie reviews annotated with positive and negative labels.
- **EMOTION** (Saravia et al., 2018) is a dataset for emotion classification comprising short comments from social media annotated with six emotion types, such as, sadness, joy, etc.
- **TREC** (Li and Roth, 2002) is a dataset for question topic classification comprising questions across six categories including human, location, etc.
- **HumAID** (Alam et al., 2021) is a dataset for crisis messages categorisation comprising tweets collected during 19 real-world disaster events, annotated by humanitarian categories including rescue volunteering or donation effort, sympathy and support, etc.

D. Training Details

When finetuning the generation model, we select the pre-trained T5 base checkpoint as the starting weights. For the downstream classification task, we finetune "bert-base-uncased"[6] on the original training data either with or without the augmented samples. Regarding the pre-trained models, we use the publicly-released version from the HuggingFace's transformers library (Wolf et al., 2019). For the augmentation factor (i.e., β in Section 3.2), the augmentation techniques including ours and the baselines are applied to augment 1 to 5 times of original training data. In the experiments, it is regarded as a hyper-parameter to be determined. Since our work focuses on text augmentation for classification in low-data settings, we sampled 5, 10, 20, 50 and 100 examples per class for each training dataset as per Anaby-Tavor et al. (2020). To alleviate randomness, we run all experiments 10 times so the average accuracy along with its standard deviation (std.) is reported on the full test set in the evaluation.

To select the downstream checkpoint and the augmentation factor, we select the run with the best performance on the development set for all methods. The hyper-parameters for finetuning the generation model and the downstream model are also setup based on the development set. Although using the full development set does not necessarily represent a real-life situation in low-data regime (Schick and Schütze, 2021a; Gao et al., 2021), we argue that it is valid in a research-oriented study. We choose to use the full development set since we aim to maximize the robustness of various methods' best performance given small training data available. As all augmentation methods are treated the same way, we argue this is valid to showcase the performance difference between our method and the baselines.

For all experiments presented in this work, we exclusively use *Pytorch*[7] for general code and *Huggingface*[8] for transformer implementations respectively, unless otherwise stated. In finetuning T5, we set the learning rate to 5×10^{-5} using Adam (Kingma and Ba, 2014) with linear scheduler (10% warmup steps), the training epochs to be 32 and batch size to be 16. At generation time, we use top-k ($k = 40$) and top-p ($p = 1.0$) sampling technique (Holtzman et al., 2019) for next token generation. In finetuning downstream BERT, the hyper-parameters are similar to those of T5 finetuning, although the training epoch is set to be 20. We set the training epochs to be as large as possible with the aim of finding the best model when trained on a small dataset, where the quality is based on performance on the development set. In our experiments, for a single run on all datasets, it takes around one day with a single Tesla P100 GPU (16GB) and thus estimated 10 days for 10 runs. To aid reproducibility, we will release our experimental code to the public at [9].

E. Comparing to Few-shot Baselines

Since our work explores a text augmentation approach for improving text classification in low-data regime, it is also related to few-shot learning methods that use few examples for text classification. We further conduct an experiment to compare STA to three state-of-the-art few-shot learning approaches: PET (Schick and Schütze, 2021a), LM-BFF (Gao et al., 2021), and DART (Zhang et al., 2022). For fair comparison, we set the experiment under the 10 examples per class scenario with 10 random seeds ensuring the 10 examples

[6] https://huggingface.co/bert-base-uncased

[7] https://pytorch.org/

[8] https://huggingface.co/

[9] https://github.com/wangcongcong123/STA

An example from **SST-2** a sentiment classification dataset where the classes ($\mathcal{L}$): negative, positive	
Text (x)	*top-notch action powers this romantic drama.*
Label (y)	*positive*
Converted examples by classification templates ($\mathcal{C}$: c, c_{pos} and c_{neg}): source(s), target(t)	
Given sentiment: negative, positive. Classify: top-notch action powers this romantic drama.	*positive*
Text: top-notch action powers this romantic drama. Is this text about positive sentiment?	*yes*
Text: top-notch action powers this romantic drama. Is this text about negative sentiment?	*no*
Converted examples by generation templates ($\mathcal{G}$: g and g'): source(s), target(t)	
Description: positive sentiment. Text:	*top-notch action powers this romantic drama.*
Description: positive sentiment. Text: top-notch action powers this romantic drama. Another text: spielberg 's realization of	*a near-future america is masterful .*
Description: positive sentiment. Text: top-notch action powers this romantic drama. Another text: a movie in	*which laughter and self-exploitation merge into jolly soft-porn 'em powerment . '*
Description: positive sentiment. Text: top-notch action powers this romantic drama . Another text: a tightly directed	*highly professional film that 's old-fashioned in all the best possible ways .*

Table 4: The demonstration of an example conversion by the prompt templates in Table 1 where the example's text is highlighted in blue and label is highlighted in red for readability.

Dataset	# Train	# Dev	# Test	# Classes
SST-2	∼6k	692	∼1.8k	2
EMOTION	16k	2k	2k	6
TREC	∼5k	546	500	6
HumAID	∼40k	6k	∼11k	8

Table 5: Datasets statistics

per class are sampled the same across the methods. Besides, we use `bert-base-uncased`[10] as the starting weights of the downstream classifier. The results are shown in Table 6. We found that although STA loses the best score to DART and LM-BFF on the **TREC** dataset, it substantially outperforms the few-shot baselines on **SST-2** and **EMOTION**. This tells us that STA is a competitive approach for few-shot learning text classification.

	SST-2	EMOTION	TREC
DART	66.5 (5.8)	26.7 (3.0)	74.0 (2.7)
LM-BFF	71.1 (9.5)	30.2 (3.8)	**77.1 (3.0)**
PET	56.7 (0.8)	28.4 (1.0)	69.1 (1.1)
STA (ours)	**81.4 (2.6)**	**57.8 (3.7)**	70.9 (6.6)

Table 6: The comparison between STA and few-shot baselines using 10 examples per class on **SST-2** and **EMOTION** and **TREC**. The results are reported as average (std.) accuracy (in %) based on 10 random experimental runs. Numbers in **bold** indicate the highest in columns.

F. More Results of Classification Tasks

Table 7, Table 8 and Table 9 present the results of STA comparing to baselines in low-data settings for the **EMOTION**, **TREC** and **HumAID** classification tasks respectively.

G. Demonstration

Table 10 and Table 11 demonstrate some original examples and augmented examples by different methods. In comparison, the examples generated by STA tend to be not only diverse but also highly label relevant (semantic fidelity).

[10] https://huggingface.co/bert-base-uncased

Augmentation Method	**5**	**10**	**20**	**50**	**100**
Baseline (No Aug.)	26.7 (8.5)	28.5 (6.3)	32.4 (3.9)	59.0 (2.6)	74.7 (1.7)
EDA	30.1 (6.2)	33.1 (4.3)	47.5 (5.0)	66.7 (2.7)	77.4 (1.8)
BT	32.0 (3.0)	37.4 (3.0)	48.5 (5.1)	65.5 (2.0)	75.6 (1.6)
BT-Hops	31.3 (2.6)	37.1 (4.6)	49.1 (3.5)	65.0 (2.3)	75.0 (1.5)
CBERT	29.2 (6.5)	32.6 (3.9)	44.1 (5.2)	62.1 (2.0)	75.5 (2.2)
GPT-2	28.4 (8.5)	31.3 (3.5)	39.0 (4.1)	57.1 (3.1)	69.9 (1.3)
GPT-2-λ	28.6 (5.1)	30.8 (3.1)	43.3 (7.5)	71.6 (1.5)	80.7 (0.4)
BART-Span	29.9 (4.5)	35.4 (5.7)	46.4 (3.9)	70.9 (1.5)	77.8 (1.0)
STA w/o Self-Checking	34.0 (4.0)	41.4 (5.5)	53.3 (2.2)	65.1 (2.3)	74.0 (1.1)
STA w/o Auxiliary Prompts	41.8 (6.1)	56.2 (3.0)	**64.9 (3.3)**	75.1 (1.5)	81.3 (0.7)
STA (ours)	**43.8 (6.9)**	**57.8 (3.7)**	64.1 (2.1)	**75.3 (1.8)**	**81.5 (1.1)**

Table 7: STA on **EMOTION** in $5, 10, 20, 50, 100$ examples per class. The results are reported as average (std.) accuracy (in %) based on 10 random experimental runs. Numbers in **bold** indicate the highest in columns.

Augmentation Method	**5**	**10**	**20**	**50**	**100**
Baseline (No Aug.)	33.9 (10.4)	55.8 (6.2)	71.3 (6.3)	87.9 (3.1)	93.2 (0.7)
EDA	54.1 (7.7)	70.6 (5.7)	79.5 (3.4)	89.3 (1.9)	92.3 (1.1)
BT	56.0 (8.7)	67.0 (4.1)	79.4 (4.8)	89.0 (2.4)	92.7 (0.8)
BT-Hops	53.8 (8.2)	67.7 (5.1)	78.7 (5.6)	88.0 (2.3)	91.8 (0.9)
CBERT	52.2 (9.8)	67.0 (7.1)	78.0 (5.3)	89.1 (2.5)	92.6 (1.1)
GPT-2	47.6 (7.9)	67.7 (4.9)	76.9 (5.6)	87.8 (2.4)	91.6 (1.1)
GPT-2-λ	49.6 (11.0)	70.2 (5.8)	80.9 (4.4)	**89.6 (2.2)**	**93.5 (0.8)**
BART-Span	55.0 (9.9)	65.9 (6.7)	77.1 (5.5)	88.38 (3.4)	92.7 (1.6)
STA w/o Self-Checking	45.4 (3.2)	61.9 (10.2)	77.2 (5.5)	88.3 (1.2)	91.7 (0.8)
STA w/o Auxiliary Prompts	49.6 (9.0)	69.1 (8.0)	81.0 (5.9)	89.4 (3.0)	93.1 (0.9)
STA (ours)	**59.6 (7.4)**	**70.9 (6.6)**	**81.1 (3.9)**	89.1 (2.7)	93.2 (0.8)

Table 8: STA on **TREC** in $5, 10, 20, 50, 100$ examples per class. The results are reported as average (std.) accuracy (in %) based on 10 random experimental runs. Numbers in **bold** indicate the highest in columns.

Augmentation Method	5	10	20	50	100
Baseline (No Aug.)	29.1 (6.6)	37.1 (6.4)	60.7 (4.0)	80.0 (0.9)	83.4 (1.0)
EDA	49.5 (4.5)	64.4 (3.6)	74.7 (1.5)	80.7 (1.0)	83.5 (0.6)
BT	45.8 (5.7)	59.1 (5.2)	73.5 (2.1)	80.4 (1.2)	83.1 (0.7)
BT-Hops	43.4 (6.4)	57.5 (5.2)	72.4 (2.8)	80.1 (1.1)	82.8 (1.4)
CBERT	44.8 (7.6)	59.5 (4.8)	73.4 (1.7)	80.3 (0.8)	82.7 (1.2)
GPT-2	46.0 (4.7)	55.7 (5.7)	67.3 (2.6)	77.8 (1.6)	81.1 (0.6)
GPT-2-λ	50.7 (8.6)	68.1 (6.2)	78.5 (1.3)	82.1(1.1)	84.2 (0.8)
BART-Span	42.4 (7.3)	58.6(7.0)	70.04 (3.7)	79.3 (1.4)	83.33 (0.9)
STA w/o Self-Checking	56.4 (7.0)	70.2 (4.3)	76.3 (3.3)	79.4 (4.5)	81.8 (1.3)
STA w/o Auxiliary Prompts	68.7 (10.9)	**77.6 (3.6)**	80.1 (1.7)	82.9 (1.6)	84.3 (0.7)
STA (ours)	**69.0 (3.9)**	75.8 (3.3)	**80.2 (1.6)**	**83.2 (0.5)**	**84.5 (1.1)**

Table 9: STA on **HumAID** in $5, 10, 20, 50, 100$ examples per class. The results are reported as average (std.) accuracy (in %) based on 10 random experimental runs. Numbers in **bold** indicate the highest in columns.

Original training examples and augmented examples for “Sadness" of **EMOTION**	
Original	i sit here feeling blank about this i feel ashamed that i so readily turn it aside i feel positively ashamed when i look out of the window and see the state of things i had just lost my uncle i would be sad but i feel as if i am devastated i was feeling kind of discouraged because nothing happened
EDA	i sit here opinion blank about this i feel that ashamed i so readily turn it aside i feel positively ashamed when i look out of the window and construe the state of things i had just lost my uncle i would be pitiful but i feel as if i am devastated i happened feeling kind of discouraged because nothing was
GPT-2-λ	ive seen so many girls walk around feeling ashamed of their bodi ive got to admit that i feel a little weird for a moment seeing her standing in front of my face when i walk into the shop ive always wondered what im doing right now im feeling ive read many blogs about her and how much she hates those who don’t admit to being kind or caring about others but instead blame them for not doing something about it ive never felt sympathetic towards people because of the way they look and act because of their skin to
STA-noself	i feel like the whole world is watching and feeling it’s failing me i want people to know i am not alone i feel ashamed when i look out of the window and see the state of things i walked away feeling disappointed because i don t know the answer i drank some cold drink or find some ice dessert such as chendol or ice kacang
STA	i feel sad seeing people who have to work harder to cope i walked away feeling disappointed because i don t know the answer i was feeling sad seeing the state of things that i never did i really want to see if it lasted i feel sad seeing the state of things but the truth is im not sure how to express it gracefully i feel like the whole world is watching and feeling it’s failing me

Table 10: The demonstration of original training examples and augmented examples for “sadness” of **EMOTION**. It is noted that the 5 augmented examples in each block are randomly selected instead of cherry-picked. This reveals some difference between the original training examples and the augmented examples by our STA and other methods (Here we use a rule-based heuristics method EDA, a generation-based method GPT-2-λ and STA-noself for comparison).

Original training examples and augmented examples for “missing or found people" of HumAID	
Original	UPDATE: Body found of man who disappeared amid Maryland flooding Open Missing People Search Database from Mati and Rafina areas #Greecefires #PrayForGreece #PrayForAthens @ThinBlueLine614 @GaetaSusan @DineshDSouza case in point, #California Liberalism has created the hell which has left 1000s missing 70 dead,... Heres the latest in the California wildfires #CampFire 1011 people are missing Death toll rises to 71 Trump blames fires on poor ... #Idai victims buried in mass grave in Sussundenga, at least 60 missing - #Mozambique #CycloneIdai #CicloneIdai
EDA	update flooding found of man who disappeared amid maryland boy open missing people search database from mati escape and rafina areas greecefires prayforgreece prayforathens created gaetasusan dineshdsouza hell in point california missing has thinblueline the case which has left s liberalism dead an countless people... heres blames latest in the california wildfires campfire people are missing death toll rises to trump more fires on poor... idai victims buried in mass grave in sussundenga at mozambique missing least cycloneidai ciclonei-dai
GPT-2-lambda	@KezorNews - Search remains in #Morocco after @deweathersamp; there has been no confirmed death in #Kerala #Cambodia - Search & Rescue is assisting Search & Rescue officials in locating the missing 27 year old woman who disappeared in ... @JHodgeEagle Rescue Injured After Missing Two Children In Fresno County #Florence #Florence Missing On-Rescue Teams Searching For Search and Rescue Members #Florence #Florence #DisasterInformer #E RT @LATTAODAYOUT: RT @HannahDorian: Search Continues After Disappearance of Missing People in Florida
STA-noself	Search Database from Matias, Malaysia, missing after #Maria, #Kerala, #Bangladesh #KeralaKerala, #KeralaFloods, ... RT @hubarak: Yes, I can guarantee you that our country is safe from flooding during the upcoming weekend! Previous story Time Out! 2 Comments The missing persons who disappeared amid Maryland flooding are still at large. More on this in the next article. the number of missing after #CycloneIdai has reached more than 1,000, reports CNN. RT @adriane@przkniewskiZeitecki 1 person missing, police confirm #CycloneIdai. #CicloneIdai
STA	The missing persons who disappeared amid Maryland flooding are still at large. More on this in the next article. Search Triangle County for missing and missing after #Maria floods #DisasterFire Just arrived at San Diego International Airport after #Atlantic Storm. More than 200 people were missing, including 13 helicopters ... Search Database contains information on missing and found people #HurricaneMaria, hashtag #Firefighter Were told all too often that Californians are missing in Mexico City, where a massive flood was devastating. ...

Table 11: The demonstration of original training examples and augmented examples for “missing or found people” of **HumAID**. It is noted that the 5 augmented examples in each block are randomly selected instead of cherry-picked. This reveals some difference between the original training examples and the augmented examples by our STA and other methods (Here we use a rule-based heuristics method EDA, a generation-based method GPT-2-λ and STA-noself for comparison).

Multi-word Term Embeddings Improve Lexical Product Retrieval

Fedor Krasnov[1], **Viktor Shcherbakov**[23]
[1] Research Center of Wildberries SK LLC based on the Skolkovo Innovation Center,
[2]University of Geneva, [3]University of Lausanne
krasnov.fedor2@wb.ru, Viktor.Shcherbakov@unil.ch

Abstract

Product search is uniquely different from search for documents, Internet resources or vacancies, therefore it requires the development of specialized search systems. The present work describes the H1 embdedding model, designed for an offline term indexing of product descriptions at e-commerce platforms. The model is compared to other state-of-the-art (SoTA) embedding models within a framework of hybrid product search system that incorporates the advantages of lexical methods for product retrieval and semantic embedding-based methods. We propose an approach to building semantically rich term vocabularies for search indexes. Compared to other production semantic models, H1 paired with the proposed approach stands out due to its ability to process multi-word product terms as one token. As an example, for search queries "new balance shoes", "gloria jeans kids wear" brand entity will be represented as one token - "new balance", "gloria jeans". This results in an increased precision of the system without affecting the recall. The hybrid search system with proposed model scores mAP@12 = 56.1% and R@1k = 86.6% on the WANDS public dataset, beating other SoTA analogues.

Keywords: semantic product search, entity recognition, SentencePiece, transformers, ColBERT

1. Introduction

Product search systems are required to operate with both low latency and high recall, since they scan the whole product catalog of billions of items. Common product search methods initially used lexical search models. These models calculate the relevance metric based on heuristics that measure exact word match between the search query and textual product representations. Lexical search models such as BM25 (Robertson and Walker, 1994) have been relevant for decades, and are still widely used today. The recent alternatives, neural extraction methods, demonstrate increased search effectiveness metrics, but also possess their own flaws (Zeng et al., 2022, 2023; Pan et al., 2024; Hofstätter et al., 2020). Naturally, the research gravitates towards the hybridization of the two approaches, combining the advantages of each.

The disadvantages of lexical models are well-researched: (E1) a possible mismatch between query and document vocabularies (Furnas et al., 1987; Zhao and Callan, 2010) leads to search recall degradation; (E2) lack of semantic understanding of queries and documents (Li and Xu, 2014) decreases search precision. These described limitations result in failures to retrieve relevant documents using lexical methods for information retrieval. To resolve these issues a number of extensions to the lexical model have been introduced in the past decades, including, but not limited to: query expansion (Lavrenko and Croft, 2001; Lesk, 1969; Qiu and Frei, 1993; Xu and Croft, 2017), document expansion (Efron et al., 2012; Liu and Croft, 2004; Gao et al., 2004), term dependencies model (Metzler and Croft, 2005; Xu et al., 2010), topic modeling (Deerwester et al., 1990; Wei and Croft, 2006), machine translation models for information retrieval (Berger and Lafferty, 1999; Karimzadehgan and Zhai, 2010). Despite mentioned advances, the research in lexical models for information retrieval progresses relatively slowly, since the majority of these methods work with discrete, sparse lexical representations and inevitably inherit their limitations.

With the development of representation learning in information retrieval, semantic search models at the offline information extraction stage of the search have seen an increased research interest in recent years. During this stage the indexes are built for matching queries with the documents. The Figure 1 schematically describes an example product search system that uses indexes built during the information extraction stage for fast responses to queries.

Starting in 2013, the improvement of word embeddings (Bravo-Marquez et al., 2013; Mikolov et al., 2013; Pennington et al., 2014) has led to a number of studies using embeddings for the extraction stage (Clinchant and Perronnin, 2013; Ganguly et al., 2015; Vulić and Moens, 2015). Unlike discrete lexical representation, word embeddings offer a continuous representation that can help with the problem of query and document vocabularies mismatch to some extent. After 2016, a spike of research attention to the application of deep learning methods to the information extraction stage is seen (Boytsov et al., 2016; Henderson et al., 2017). These methods are applied either for improving document representation within

The Seventh Workshop on e-Commerce and NLP (ECNLP 7), pages 115–124

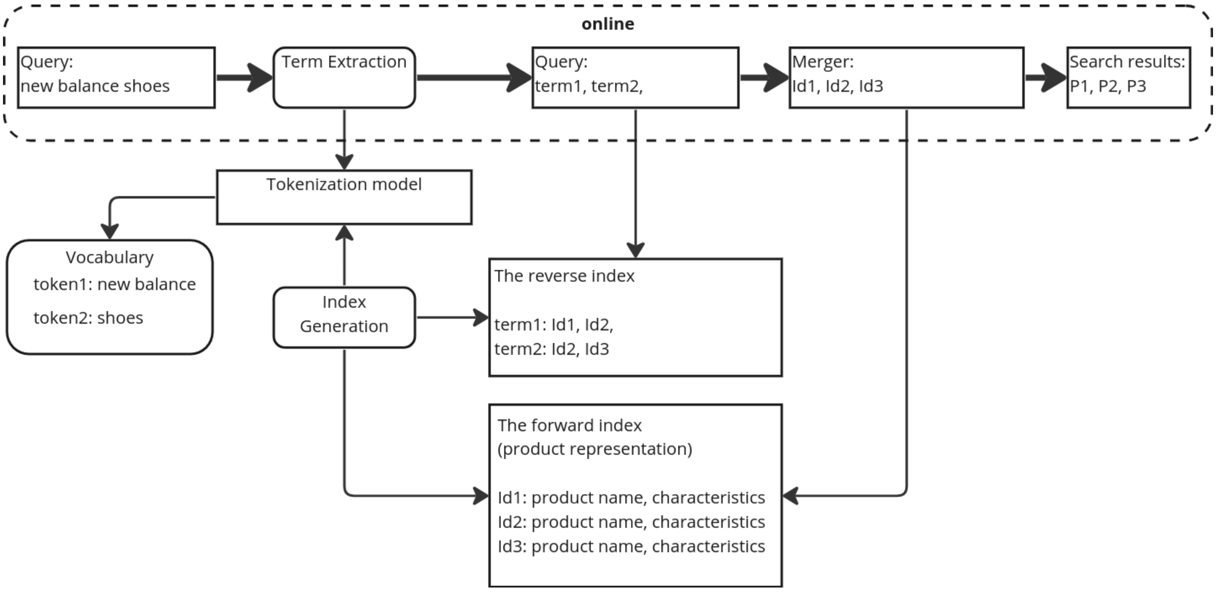


Figure 1: The indexes and a tokenization model built during offline information extraction are used in online setting to respond to queries with low latency. The quality of built index is detrimental to the performance of the search system.

the framework of the traditional paradigm of discrete lexical representation (Bai et al., 2020; Dai and Callan, 2019; Nogueira et al., 2019), or directly for forming novel semantic search models within the sparse/dense representation paradigm (Gillick et al., 2018a; Jean et al., 2015; Khattab and Zaharia, 2020; Zamani et al., 2018).

While closely related to document information retrieval, the product search problem is uniquely different in a few aspects:

- Ranking mechanisms based on weighing textual features (TF/IDF, BM25) differ in product search. For example, the token frequency in the product title does not affect the query relevancy.

- Products are multimodal. A product page includes a title, description, characteristics, images, videos, etc. The search system can take into account multiple modalities of a page.

- Search queries are motivated by an interest in purchasing a product. Customer behavior differs significantly from vacancy search or Internet resource search behavior.

- Product search effectiveness is evaluated on a modality-wise basis.

The primary research question of the present paper is to evaluate the impact of the semantic model and tokenization architectures on offline metrics of a hybrid product search system.

In the following sections, we describe in detail the research methodology, conducted experiments and conclusions.

2. Related work

2.1. Neural Information Retrieval

Similar to document information retrieval trends, the development of product search systems has transitioned from lexical retrieval methods to neural retrieval methods (Li et al., 2021; Magnani et al., 2022; Nigam et al., 2019). DSSM (Huang et al., 2013), being one of the most popular neural network architectures, is based on a Dual Encoder paradigm (Gillick et al., 2018b; Yang et al., 2019; Karpukhin et al., 2020). The two independent "towers" of encoders—one for search queries and the other for product representation—embed queries and products into a shared space of fixed dimensionality. The shared space is used for similarity search (Vanderkam et al., 2013; Johnson et al., 2017) to retrieve products that are relevant to a search query. Thus far, the most promising results have been achieved by using the BERT model in a Dual Encoder architecture (Chang et al., 2020; Xiong et al., 2021; Lu et al., 2020). The general operating principle of these models is described in Eqs. (1) to (3).

$$\overrightarrow{q} = AvgPool\left[BERT^{l}_{\theta}(q)\right] \quad (1)$$

$$\overrightarrow{p} = AvgPool\left[BERT^{r}_{\theta}(p)\right] \quad (2)$$

$$s_{BERT}(\overrightarrow{q}, \overrightarrow{p}) = \overrightarrow{q}^{T} \cdot \overrightarrow{p} \quad (3)$$

Where $BERT_{\theta}^{l}$ and $BERT_{\theta}^{r}$ are the "left" and "right" encoders, respectively, transforming texts q and p into a shared space θ. The similarity function $s_{BERT}(\cdot,\cdot)$ is implemented with a scalar product of $\overrightarrow{q}$ and $\overrightarrow{p}$. The bottleneck in this architecture lies in the averaging of the token vectors.

The ColBERT (Khattab and Zaharia, 2020) model represents a particular variant of the Dual Encoder architecture, termed a Single Encoder. Models based on this architecture use the same encoder for both queries and products. However, the novelty of ColBERT lies in computing the similarity scores token-wise, instead of comparing the mean vectors. Given a search query q comprising m tokens and a product p comprising n tokens, the similarity function $s_{ColBERT}(\cdot,\cdot)$ is:

$$s_{ColBERT}(q_{1:m}, p_{1:n}) = \sum_{1}^{m} \max_{1..n} \left(\overrightarrow{q}_{1:m}^{T} \cdot \overrightarrow{p}_{1:n}\right) \tag{4}$$

The sum over maximum similarity scores for each token of a query in Eq. (4) implies that $n \cdot m$ scalar products need to be calculated, compared to one scalar product in $s_{BERT}(\cdot,\cdot)$.

2.2. Hybridization

It is accepted to understand hybridization as mixing the lexical and neural methods of information retrieval within one product search system. Hybridization can be applied at different stages of the search. For instance, the authors of the study Nigam et al. (2019) combined the search results of several distinct models based on lexical, behavioral, and semantic methods. Another hybridization principle was applied in the study Gao et al. (2021)—the lexical method was the primary retrieval mechanism, while a semantic model was trained to correct the mistakes of the lexical model.

2.3. Tokenization

The progress in tokenization methods has led to significant improvements in the offline metrics of natural language processing models (Kudo and Richardson, 2018; Sennrich et al., 2016). The BPE (Byte-Pair-Encoding) tokenization method was originally introduced as a data compression method (Gage, 1994). In constructing the BPE tokenizer, the initial vocabulary is sequentially extended until the preset limit is reached. The primary goal of applying BPE to natural text is to split words into commonly occurring subwords. Usually, little care is given to the actual semantics of the final tokens. However, unlike most applications, where semantic information can be represented by the combination of tokens, information retrieval often requires semantically rich tokens in order to use them as terms to construct effective search indexes, see Fig 1.

The later proposed alternative, the unigram tokenization method (Kudo, 2018a), demonstrates the opposite approach—the vocabulary size is sequentially pruned by removing rare tokens that can be replaced by common tokens. The unigram method was primarily introduced to provide multiple possible tokenizations for a given text with the use of a unigram language model. During vocabulary construction, both methods aim to minimize the length of text encoded in tokens and, in practice, produce similar tokenizations.

3. Methodology

3.1. H1

The H1 semantic model draws significant inspiration from ColBERT but architecturally simplified. It processes both queries and documents by tokenizing them and then passing them through a BERT-based Dual Encoder. The resulting embeddings are evaluated using the $s_{ColBERT}(\cdot,\cdot)$ similarity function. We explore the impact of different tokenization techniques in Ablation Study Section 4.2. A distinctive aspect of H1 is its approach to token-level lexical hybridization, where we enhance the tokenizer's vocabulary with semantically rich terms to improve the semantic independence of standalone terms. The Experiments Section 4 provides a comprehensive analysis of the H1 system's application in a product retrieval task. For this task specifically, we augmented the tokenizer's vocabulary with a carefully selected list of brand names.

The rationale behind incorporating brand names into the vocabulary is rooted in understanding user search behavior, particularly when it comes to specific brands. For instance, when a customer searches for "new balance shoes", their intent is not to explore products related to the terms "new" and "balance" independently. Instead, they are looking for items specifically associated with the "New Balance" brand. However, these customers may still be open to considering various types of "shoes".

H1 model is optimized on positive and negative product-query pairs using the following loss function:

$$L_{H1} = \left[\gamma - s_{\theta}(q_{1:m}, p^{+}_{1:n}) + s_{\theta}(q_{1:m}, p^{-}_{1:n})\right]_{+0} \tag{5}$$

Where γ is a threshold and s_{θ} is a similarity relation parametrized by θ, applied to an m-token query q with a positive p^{+} and a negative p^{-} product description example. Negative examples are sampled by selecting a random product from the current batch. The square brackets around the

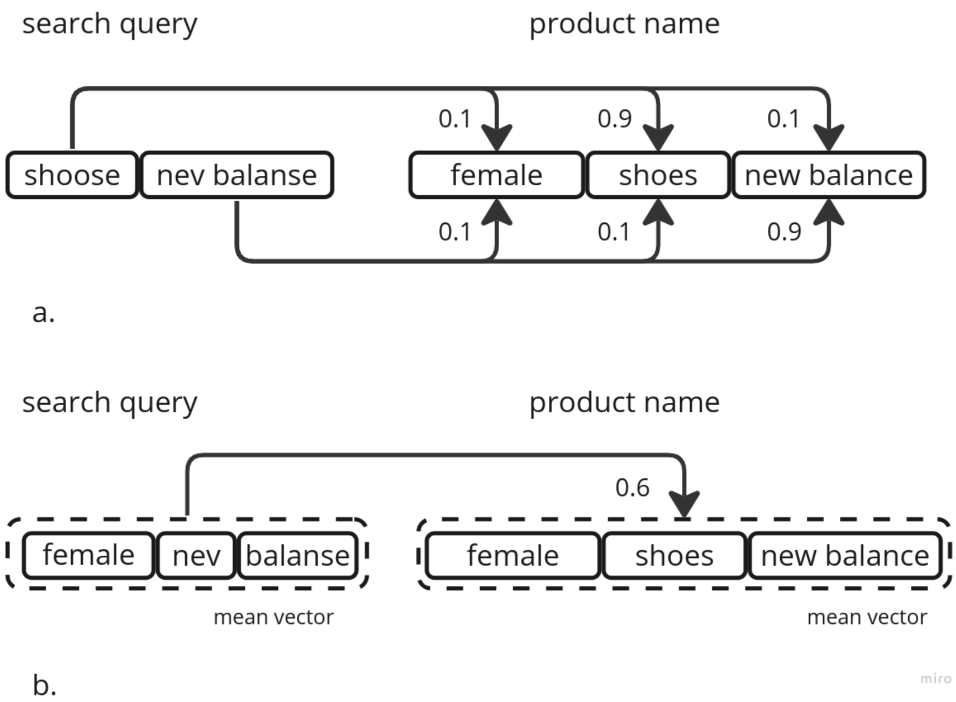


Figure 2: Token handling principle of H1 (a), compared to that of a FastText (b). H1 attributes scores to each pair of query and document tokens, while the FastText-based system compares the mean vector representations.

equation, $[]_{+0}$, denote that negative values are set to 0.

3.2. Evaluation

Neural retrieval methods, given their computational intensity, are impractical for online product searches within catalogs containing billions of items. Instead, their utility shines in building indexes for product descriptions, as schematically outlined in the example in Figure 1. The actual neural encoder is never utilized to generate the embeddings for user queries. Our evaluation methodology mirrors these practical limitations, ensuring that our approaches are both realistic and aligned with the constraints of large-scale product retrieval systems.

For the query encoder E^q_θ, the product encoder E^p_θ, the similarity measure s, and the tokenization method T, the evaluation procedure employed in the experiments (Section 4) is as follows:

1. The vocabulary of query tokens V_q is collected using T.
2. For every token t_i from the vocabulary V_q, its embedding $e^q_i = E^q_\theta[t_i]$ is produced.
3. The embeddings for the tokens in every product description $p^j_{1:n}$ are computed as:
$$(e_{j,k})^n_{k=1} = E^p_\theta[T(p^j_{1:n})]$$
4. An index that maps every query term to relevant products is built using query tokens as terms:
$$I(t_i) = \{p^j_{1:n} \mid s(e^q_i, e^p_{j,1:n}) > \gamma\}$$
where γ is a relevancy threshold.
5. For a query $q_{1:m}$ with tokens
$$T(q) = (t'_1,\ \dots\ , t'_m)$$
a list of all relevant products according to the index I,
$$R = I(t'_1)|\ \dots\ |I(t'_m)$$
is collected, and the metric is computed on R, sorted with respect to the similarity of relevant products to the query.

The described evaluation approach mimics the product search implemented with a simple term index-based hybrid search system. This system combines the efficiency of fast lexical term lookup in an index for high precision, with the computation of similarity scores on only a subset of all product descriptions, ensuring low latency responses. The performance of the system is entirely dependent on the similarity measure s within embedding space defined by semantic model of choice.

The offline metrics for product search differ from those of document information retrieval. The objective of product search is to identify several, or ideally, all products relevant to the query, including identical items. This requirement stems from the customer's need to compare prices for identical products. Hence, the formulas for recall and precision are adapted to include an equivalence relation M. Precision metrics for product search are defined as follows, with recall metrics being similarly formulated.

$$P@k = \frac{1}{|Q|}\sum_{q\in Q}\frac{\left|M(p_q^r@k, p_q^g)\right|}{k} \quad (6)$$

$$mAP@k = \frac{1}{k}\sum_{i=1}^{k} P@i \quad (7)$$

Q – the set of all search queries.

p_q^g – all ground truth products for query q.

p_q^r – the retrieved products for query q at rank k.

$M(A, B)$ – the set of products in A that are equivalent to any of the products in B.

4. Experiments

We evaluated the proposed H1 model against several existing information retrieval models, specifically TCT-ColBERT (Lin et al., 2020), Single Encoder (SE) (Nigam et al., 2019), and Dual Encoder (DE) (Huang et al., 2013). Additionally, we experimented with three tokenization methods: Byte Pair Encoding (BPE), unigram, and word tokenizations. For each tokenization method, we proposed two variations: one enriched with a predefined set of brand names as special tokens (referred to as multi-token or *mt* variations), and a standard version without added brand names (non-multi-token or *non-mt* variations).

We employed the SentencePiece library for all tokenization tasks, configuring it with the *split_by_whitespace=False* option to ensure multi-word brand names could be incorporated as special tokens.

Following the evaluation methodology outlined in Section 3.2, we calculated the metrics mean Average Precision at 12 items ($mAP@12$) and Recall at 1000 items ($R@1k$) for H1, SE, DE models combined with every tokenization method described earlier. Two products are considered to be equivalent if they share the same title.

We compare the performance of the best combination of the model type and tokenization method against ColBERT implemented by Terrier (Macdonald et al., 2021) and trained with Tight Coupling Teachers method (Lin et al., 2020).

4.1. Dataset

Our data source is the publicly available WANDS dataset, chosen for its suitability in objectively benchmarking retrieval systems in the context of e-commerce. The dataset's key characteristics are as follows:

- 42,994 product candidates,
- 480 queries,
- 233,448 relevancy scores for query-product pairings.

The relevancy of query-product pairs in the WANDS dataset is annotated with three levels: fully relevant (Exact), partially relevant (Partial), and Irrelevant. For the purposes of training our models, we utilized only two labels: Exact (labeled as 1) and Irrelevant (labeled as -1), with class balancing implemented prior to training.

4.2. Ablation study

First, we ablate over the tokenization method and model hyperparameter (embedding dimensions) for each of the model types: H1, SE, DE. For ColBERT model, the pretrained version was used, so it was not included in the ablation study. The results of the experiment are shown in Fig 3. The best results, $R@1k = 86.6\%$ and $mAP@12 = 56.1\%$, were achieved by the combination of H1 model with 768 embedding dimensions and BPE tokenization with brand names added.

We note that for both BPE and unigram tokenizations, the variation with brand names added (mt) produces consistently better results for any model with any embedding dimensionality.

4.3. Best models comparison

Model	Threshold	Precision	Recall
ColBERT	12	**41%**	26%
	128	21%	61%
	512	9%	78%
	1024	5%	**84%**

Table 1: The ColBERT results on WANDS dataset.

The Table 1 presents the results of the evaluation of the ColBERT model on the WANDS dataset with varying thresholds. The H1 model demonstrates better results, especially for Precision at 12 items.

To further demonstrate the superiority of the H1 model, we compare H1, SE, DE, and ColBERT models with the best hyperparameters seen in the Ablation Study Section 4.2 on a single query with multiple thresholds k.

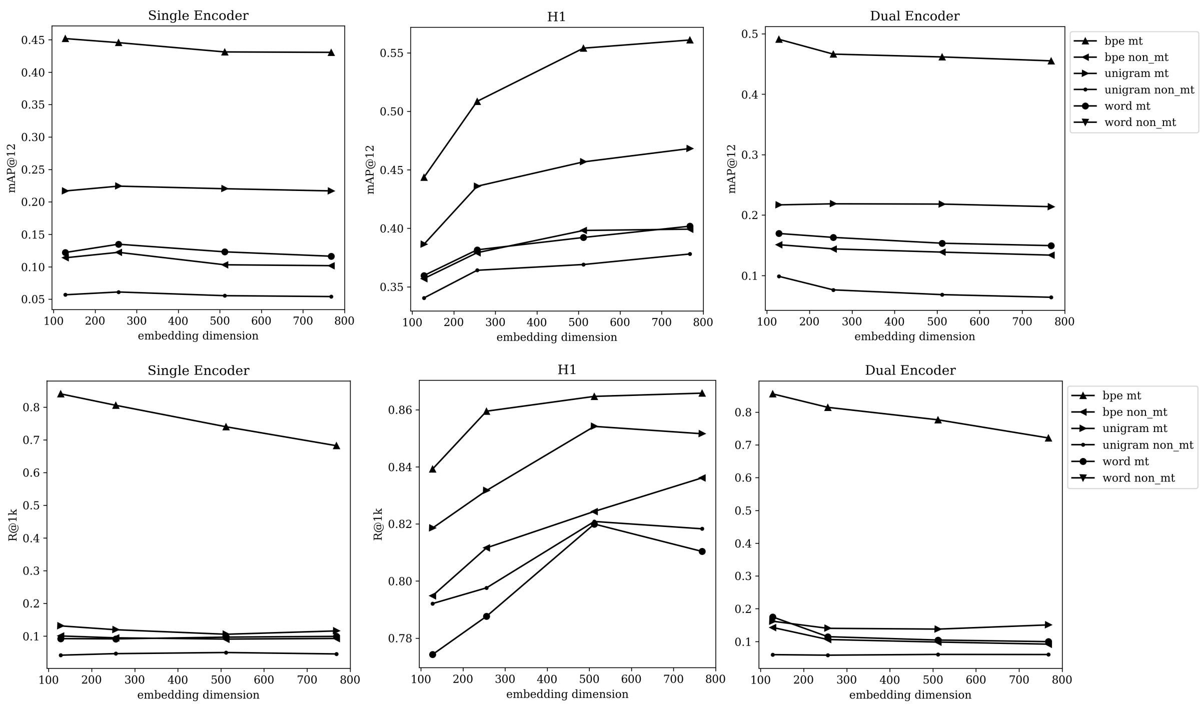


Figure 3: Ablation study results over tokenization methods and model architectures.

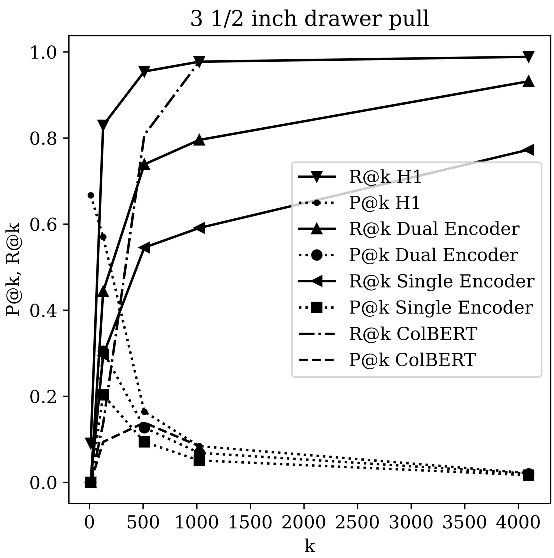


Figure 4: An illustrative one-query example of how Precision decreases and Recall increases for different semantic retrieval models with respect to cut-off threshold k.

The dynamics of Precision and Recall metrics for the H1 model with respect to the threshold k are illustrated in Fig 4, clearly separating the H1 model from the rest. The Recall of the search results is higher with lower values of the threshold k, and Precision declines more slowly as k increases, compared to other models.

5. Conclusions and Future Work

This study introduced the H1 embedding model, a cutting-edge approach designed to refine the landscape of e-commerce search systems by leveraging multi-word term embeddings. Our extensive evaluations demonstrate that H1, through its innovative use of semantically rich tokens and hybrid search methodologies, notably enhances the accuracy and efficiency of product retrieval. By achieving mAP@12 = 56.1% and R@1k = 86.6% on the WANDS dataset, H1 has set a new benchmark, surpassing other state-of-the-art models in terms of precision and recall.

Our research underscores the criticality of integrating semantic understanding with traditional lexical search techniques to address the inherent limitations of each approach. The H1 model's unique ability to treat multi-word terms as singular entities not only improves the search relevance but also aligns with the natural language processing of user queries, thereby significantly enhancing the user experience in e-commerce platforms.

Future efforts will be dedicated to establishing a definitive benchmark for semantic models operating within the framework of hybrid search systems. By exploring a broader range of system architectures, the aim of our future work is to provide a comprehensive and objective evaluation framework that will not only assess the efficacy of current models but also inspire the development of more advanced and effective search solutions.

6. References

Eneko Agirre and Arantxa Otegi. 2010. Document expansion based on wordnet for robust ir. volume 2, pages 9–17.

Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. 2020. Sparterm: Learning term-based sparse representation for fast text retrieval. *CoRR*, abs/2010.00768.

Adam Berger and John Lafferty. 1999. Information retrieval as statistical translation. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, page 222–229, New York, NY, USA. Association for Computing Machinery.

Leonid Boytsov, David Novak, Yury Malkov, and Eric Nyberg. 2016. Off the beaten path: Let's replace term-based retrieval with k-nn search. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM'16. ACM.

Felipe Bravo-Marquez, Gaston L'Huillier, Sebastián Ríos, and Juan Velasquez. 2013. Hypergeometric language model and zipf-like scoring function for web document similarity retrieval. pages 303–308.

Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training tasks for embedding-based large-scale retrieval. *CoRR*, abs/2002.03932.

Yan Chen, Shujian Liu, Zheng Liu, Weiyi Sun, Linas Baltrunas, and Benjamin Schroeder. 2022. Wands: Dataset for product search relevance assessment. In *Advances in Information Retrieval*, pages 128–141, Cham. Springer International Publishing.

Stéphane Clinchant and Florent Perronnin. 2013. Aggregating continuous word embeddings for information retrieval. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 100–109, Sofia, Bulgaria. Association for Computational Linguistics.

Zhuyun Dai and Jamie Callan. 2019. Context-aware sentence/passage term importance estimation for first stage retrieval.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Miles Efron, Peter Organisciak, and Katrina Fenlon. 2012. Improving retrieval of short texts through document expansion. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, page 911–920, New York, NY, USA. Association for Computing Machinery.

G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. 1987. The vocabulary problem in human-system communication. *Commun. ACM*, 30(11):964–971.

Philip Gage. 1994. A new algorithm for data compression. *C Users J.*, 12(2):23–38.

Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth J.F. Jones. 2015. Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 795–798, New York, NY, USA. Association for Computing Machinery.

Jianfeng Gao, Jian-Yun Nie, Guangyuan Wu, and Guihong Cao. 2004. Dependence language model for information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, page 170–177, New York, NY, USA. Association for Computing Machinery.

Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021. Complementing lexical retrieval with semantic residual embedding.

Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. 2018a. End-to-end retrieval in continuous space.

Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. 2018b. End-to-end retrieval in continuous space.

Matthew L. Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *CoRR*, abs/1705.00652.

Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving efficient neural ranking models with cross-architecture knowledge distillation. *CoRR*, abs/2010.02666.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, CIKM '13, page 2333–2338, New York, NY, USA. Association for Computing Machinery.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, Beijing, China. Association for Computational Linguistics.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *CoRR*, abs/1702.08734.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models.

Maryam Karimzadehgan and ChengXiang Zhai. 2010. Estimation of statistical translation models based on mutual information for ad hoc information retrieval. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, page 323–330, New York, NY, USA. Association for Computing Machinery.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *CoRR*, abs/2004.04906.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2741–2749. AAAI Press.

Taku Kudo. 2018a. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo. 2018b. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, page 120–127, New York, NY, USA. Association for Computing Machinery.

M. E. Lesk. 1969. Word-word associations in document retrieval systems. *American Documentation*, 20(1):27–38.

Hang Li and Jun Xu. 2014. Semantic matching in search. *Foundations and Trends® in Information Retrieval*, 7(5):343–469.

Sen Li, Fuyu Lv, Taiwei Jin, Guli Lin, Keping Yang, Xiaoyi Zeng, Xiao-Ming Wu, and Qianli Ma. 2021. Embedding-based product retrieval in taobao search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 3181–3189, New York, NY, USA. Association for Computing Machinery.

Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2020. Distilling dense representations for ranking using tightly-coupled teachers.

Xiaoyong Liu and W. Bruce Croft. 2004. Cluster-based retrieval using language models. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, page 186–193, New York, NY, USA. Association for Computing Machinery.

Wenhao Lu, Jian Jiao, and Ruofei Zhang. 2020. Twinbert: Distilling knowledge to twin-structured compressed bert models for large-scale retrieval. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 2645–2652, New York, NY, USA. Association for Computing Machinery.

Craig Macdonald, Nicola Tonellotto, Sean MacAvaney, and Iadh Ounis. 2021. Pyterrier: Declarative experimentation in python from bm25 to dense retrieval. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 4526–4533, New York, NY, USA. Association for Computing Machinery.

Alessandro Magnani, Feng Liu, Suthee Chaidaroon, Sachin Yadav, Praveen Reddy Suram, Ajit Puthenputhussery, Sijie Chen, Min Xie, Anirudh Kashi, Tony Lee, and Ciya Liao. 2022. Semantic retrieval at walmart. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 3495–3503, New York, NY, USA. Association for Computing Machinery.

Donald Metzler and W. Bruce Croft. 2005. A markov random field model for term dependencies. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, page 472–479, New York, NY, USA. Association for Computing Machinery.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Priyanka Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian (Allen) Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, and Bing Yin. 2019. Semantic product search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2876–2885, New York, NY, USA. Association for Computing Machinery.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction.

Zaifeng Pan, Zhen Zheng, Feng Zhang, Ruofan Wu, Hao Liang, Dalin Wang, Xiafei Qiu, Junjie Bai, Wei Lin, and Xiaoyong Du. 2024. Recom: A compiler approach to accelerating recommendation model inference with massive embedding columns. pages 268–286.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Yonggang Qiu and Hans-Peter Frei. 1993. Concept based query expansion. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '93, page 160–169, New York, NY, USA. Association for Computing Machinery.

Stephen Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. pages 232–241.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Dan Vanderkam, Rob Schonberger, Henry Rowley, and Sanjiv Kumar. 2013. Nearest neighbor search in google correlate. Technical report, Google.

A. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 363–372, New York, NY, USA. Association for Computing Machinery.

Xing Wei and W. Croft. 2006. Lda-based document models for ad-hoc retrieval. pages 178–185.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

Jinxi Xu and W. Bruce Croft. 2017. Quary expansion using local and global document analysis. *SIGIR Forum*, 51(2):168–175.

Jun Xu, Hang Li, and Chaoliang Zhong. 2010. Relevance ranking using kernels. In *Information Retrieval Technology*, pages 1–12, Berlin, Heidelberg. Springer Berlin Heidelberg.

Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2019. Multilingual universal sentence encoder for semantic retrieval.

Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. 2018. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 497–506, New York, NY, USA. Association for Computing Machinery.

Chaoliang Zeng, Layong Luo, Qingsong Ning, Yaodong Han, Yuhang Jiang, Ding Tang, Zilong Wang, Kai Chen, and Chuanxiong Guo. 2022. FAERY: An FPGA-accelerated embedding-based retrieval system. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 841–856, Carlsbad, CA. USENIX Association.

Shulin Zeng, Zhenhua Zhu, Jun Liu, Haoyu Zhang, Guohao Dai, Zixuan Zhou, Shuangchen Li, Xuefei Ning, Yuan Xie, Huazhong Yang, and Yu Wang. 2023. Df-gas: a distributed fpga-as-a-service architecture towards billion-scale graph-based approximate nearest neighbor search. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '23, page 283–296, New York, NY, USA. Association for Computing Machinery.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Le Zhao and Jamie Callan. 2010. Term necessity prediction. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, page 259–268, New York, NY, USA. Association for Computing Machinery.

Explicit Attribute Extraction in E-Commerce Search

Robyn Loughnane, Jiaxin Liu, Zhilin Chen, Zhiqi Wang, Joseph Giroux, Tianchuan Du, Benjamin Schroeder, Weiyi Sun
Wayfair LLC
{rloughnane,jliu2,zchen3,zwang6,jgiroux,tdu1,beschroeder,wsun1}@wayfair.com

Abstract

This paper presents a model architecture and training pipeline for attribute value extraction from search queries. The model uses weak labels generated from customer interactions to train a transformer-based NER model. A two-stage normalization process is then applied to deal with the problem of a large label space: first, the model output is normalized onto common generic attribute values, then it is mapped onto a larger range of actual product attribute values. This approach lets us successfully apply a transformer-based NER model to the extraction of a broad range of attribute values in a real-time production environment for e-commerce applications, contrary to previous research. In an online test, we demonstrate business value by integrating the model into a system for semantic product retrieval and ranking.

Keywords: attribute extraction, e-commerce search, named-entity recognition

1. Introduction

E-commerce applications use a range of structured information that describe their catalog of products or services. This allows customers to browse via a taxonomy of product categories and filters, using the structured information directly to narrow down their search. For example, a customer may click on categories “Furniture”, then “Bedroom Furniture”, then “Nightstands”, then the attribute “Color:Blue”, to find an item they like.

However, when customers search using natural language, a mapping of the query to relevant structured information must happen automatically. This mapping can principally be used in two ways to improve search engine performance: first, scores that judge the relevance of a query to particular product information can be used to inform a relevance-based ranking (Liu et al., 2022b). Second, explicit filters can be applied dynamically, restricting the result set to products with attributes matching those identified in the query, in particular where the query interpretation has a high degree of confidence in its prediction.

Attribute value extraction (AVE) in search can be approached in different ways. While it is possible to e. g. use a multi-label classification task at query level, we opted for a token-level classification approach related to the more general problems of slot filling and named-entity recognition (NER). This yields a more intuitive mapping from spans in the query to explicit filtering, providing a more transparent user experience.

Two major challenges of a search application are, first, that text input is short and contains non-standard grammar and spelling; and, second, the stringent latency requirements for model inference, which needs to be computed online in real time. Both of these make the use of transformers,[1] the current state of the art in NER, challenging since they use contextual information typically only found in longer strings, and they contain a high number of model weights. In addition, there are challenges with the large size of the label space and the fact that NER training and evaluation requires token-level labelling, which can be prohibitively expensive. Indeed, Xu et al. (2019) claim the search problem is not solvable with an NER-style model.

To the contrary, we demonstrate the feasibility of a transformer-based NER-style model. Using a lightweight transformer that meets the latency requirements, we show that it performs almost as well as a larger model, indicating data quality may be more important than model size. We show a way to create quality data via a regime of hierarchical normalization applicable to any e-commerce catalog to deal with the problem of a large label space. We use weak labeling to create abundant inexpensive labeled data. We report model accuracy and nDCG gains and, lastly, demonstrate the business value of this approach via an online A/B test.

2. Related Work

In search systems, query understanding models aim to decipher and interpret users’ search goals so as to aid the downstream retrieval and ranking applications (Deng and Chang, 2020). In the general web search domain, query understanding consists of two dimensions: intent classification and topic detection (Brenes et al., 2009). Query intent classi-

[1]Transformer-based models are among the state of the art in NER tasks, e. g. the BERT-based model from Li et al. (2020) for Ontonotes v5 (Weischedel et al., 2013).

The Seventh Workshop on e-Commerce and NLP (ECNLP 7), pages 125–135

User query text	*mod bright red sofa*			
Query tokens	mod	bright	red	sofa
Identified text spans	mod	bright red		sofa
NER label	STYLE	COLOR		O
Intermediate attribute values	Modern	Red		
Final attribute title	Product Styles	Upholstery Color		
Final attribute value	Modern & Contemporary	Red		

Table 1: An example user query with two attributes identified

fication determines users' desired search actions, such as informational, navigational and transactional (Broder, 2002; Rose and Levinson, 2004). Query topic detection maps users' search queries to a predefined taxonomy of topics, such as sports, entertainment and so on (Li et al., 2005). Intent classification and topic detection problems also exist in the e-commerce domain. Both intents and, in particular, query topic categories in e-commerce can differ greatly depending on industry vertical (Tsagkias et al., 2021), but the the overarching principles are applicable to any e-commerce domain.

E-commerce query topic detection maps user search queries to the structured product catalog taxonomy (Wen et al., 2019). For instance, the search query "KitchenAid 4.5 Qt" maps to products in the "Small Appliances/Mixers" category, with attributes of "Brand: KitchenAid" and "Capacity: 4 - 5 Qt". The query-to-product-type mapping part of the problem ("KitchenAid 4.5 Qt" to "Small Appliances/Mixers") has been well studied as text classification (Hashemi et al., 2016; Kim et al., 2016; Lin et al., 2020).

There has been less prior work focusing on the query AVE and normalization part of the problem. Following Luo et al. (2022), a distinction can be made in AVE between *explicit* (e.g. Cowan et al. 2015; Kozareva et al. 2016; Wen et al. 2019; Cheng et al. 2020; Zhang et al. 2021) and *implicit* or latent attributes (e.g. Wu et al. 2017). Explicit attributes are represented by a span of text in the user query, whereas implicit attributes are not. Implicit attributes can be use directly, whereas extracted explicit attribute spans usually need to first be normalized to match misspellings or non-canonical forms against structured product data, as in the current approach.

A major challenge for AVE in e-commerce queries is the sparsity of available data (Cheng et al., 2020; Wen et al., 2019), especially where the number of product attributes is high with a long tail distribution of rare attributes. Cheng et al. (2020) address this via an iterative learning framework that utilizes both synthetic data and human annotated data to extract product categories and brands. Wen et al. (2019) went without human annotated data and leveraged only user behavior logs to build a sequential tagging model for attribute detection. Zhang et al. (2021) use a teacher-student network to better exploit a combination of human annotated labels and weak synthetic labels in their sequential tagging model. We address this problem via weak and synthetic labeling (Section 3.3) to generate data; and reducing the label space via normalization (Section 3.5).

Normalization is a challenge for explicit AVE and some studies leave it out altogether despite its necessity in an e-commerce setting (Zhang et al., 2021). Similarity measures are one possible approach per Putthividhya and Hu (2011), who use *n*-gram sub-string similarity to normalize results to match dictionary entries. Cowan et al. (2015) use a gazetteer approach for matching identified spans to attribute entities, as do Zhang et al. (2021) after a frequency analysis of user behavior. We propose a two-step normalization process using gazetteers: first identified spans are mapped on to a pre-defined schema of intermediate attribute value concepts, before finally mapping them on to product attribute data (Section 3.5). An example search query with intermediate and final attributes identified is given in Table 1.

An additional constraint in any business context is that there must be enough monetary value to justify the complexity of the system implemented. Business applications of attribute extraction from search queries include product retrieval (Cheng et al., 2020), recall filtering (Wen et al., 2019) and ranking (Cheng et al., 2020; Wen et al., 2019; Wu et al., 2017). In an A/B test, the current system brings value in ranking (Section 4) even on top of a semantic search system per Liu et al. (2022b), where no previous studies known to the current authors have explicitly demonstrated this.

Outside of the search context, classic NER approaches inform the current work. NER is typically defined as the identification of phrases that contain the names of persons, organizations and locations (Tjong Kim Sang and De Meulder, 2003). Although many attribute values are not proper nouns, the mechanics of the problem in regards to span identification in written text are similar, and the phrases to be identified can be sorted into thematic groups in a similar fashion to those in classic NER. Publicly available, pre-trained, transformer-based models like BERT (Devlin et al., 2019) that create context-

	Human labels	Weak labels	Test set	F1 COLOR	F1 STYLE	F1 BRAND	F1 overall
Human only	90	0	33	**0.83**	0.447	0.525	0.453
Weak only	0	82	33	0.686	0.504	0.656	0.56
Weak and synthetic only	0	146	33	0.687	0.516	0.662	0.568
Production model	90	146	33	0.818	**0.625**	**0.71**	**0.654**

Table 2: Sample model training & test set sizes & statistics (data size given in thousands)

sensitive word embeddings enable token classification on top of these networks to be used as viable alternative to traditional NER methods.

3. Problem Definition & Methodology

The problem at hand is thus to map explicit attributes within a user search query string onto the structured data of a set of products, within the practical constraints of the live production environments of an e-commerce website. The structured data targeted in particular here are *attribute titles* and *attribute values*. Canonical examples of attribute titles are “Upholstery Color” and “Product Styles”, which can have attribute values like “Red” or “Modern & Contemporary”, and which are both distinct from a product’s category, e. g. “Sofas”.

For explicit AVE, a number of studies use methodologies from NER, such as a conditional random field (CRF) (Cowan et al., 2015), long short-term memory network (LSTM) plus CRF (Kozareva et al., 2016; Wen et al., 2019), bidirectional gated recurrent unit (GRU) network with a CRF layer plus LSTM-based character embeddings (Cheng et al., 2020) and, more recently, pre-trained transformer-based language models (Zhang et al., 2021; Luo et al., 2022). Other studies use a question answering approach (Shinzato et al., 2022; Xu et al., 2019). The model presented here is an NER model, using token classification on top of a distilled pre-trained transformer-based language model. Distilled versions of larger language models (Sanh et al., 2019) enable performance to be largely be maintained without the drag on latency.

As this paper focuses on explicit AVE, normalization is required, which we approach using an initial gazetteer, plus a second layer of custom normalization depending on the attribute type. Contrary to Xu et al. (2019), who claim that an NER-based model cannot deal with a large attribute space, we demonstrate how this can indeed be done by using this two-stage normalization approach.

The model training pipeline consists thus of multiple steps. First, human annotation using a predefined attribute schema is conducted (Section 3.2). The human annotation is supplemented by weak and synthetic labels generated from the structured data in the catalog (Section 3.3). The model itself is a token-classification transformer network (Section 3.4). The identified span is normalized before use (Section 3.5). The model is evaluated via both offline and online means (Section 4).

3.1. Production Environment & Baseline

The current system replaces the previous rules-based system in production, which is applied to around half of all search experiences, covering just under 15% of the most common distinct search queries in a given month. The other half of search experiences are characterized by a long tail of diverse search queries, which previously had not had attributes extracted at all. However, some attribute information is implicitly used in the embedding-based semantic search product retrieval model, to the extent that this attribute information was available in the product information used to train the model (product name, product category and so forth). However, to use this information explicitly for relevance scoring or dynamic filtering, individual attributes need to be predicted and then mapped to product data.

A challenge peculiar to the current production system is that it is not set up to have consistent attribute information across product categories. Product attributes are specific to a given product category. This means that the attributes of a product of type “Chair” include “Upholstery Material” and “Leg Material”, whereas a product of type “Saucepan” includes “Lid Material”. In addition, attributes of different product categories may have distinct IDs and values, even where the attribute title is the same or similar, e.g. “Primary Material”. This puts the number of attributes at around 86,000 distinct attribute ID-title pairs with 314,000 distinct attribute ID-value pairs. This makes a direct classification difficult due to data sparsity for the long tail of less-common attributes; in the current paper we address this challenge via two-stage normalization (Section 3.5).

3.2. Human Annotation

Human annotators were provided with lists of historical customer search queries to label with up to three attributes. All queries were initially labeled by two human annotators. If both annotators did not exactly agree on an attribute, it was submit-

	F1 COLOR	F1 STYLE	F1 BRAND	F1 overall
Production model	0.818	0.625	0.71	0.654
No noisy spelling	0.838	0.617	0.69	0.651
No synthetic subjects	0.76	0.614	0.702	0.65
No synthetic SKU IDs	0.754	0.654	0.704	0.649
No synthetic product categories	0.759	0.619	0.694	0.63

Table 3: Ablation study

ted to a third annotator to make a final decision. Annotators were asked to label non-overlapping sub-strings of the original query with attribute titles and values from a pre-defined schema of intermediate attributes.

The schema of intermediate attributes was created from a frequency analysis of commonly searched attributes from historical user interactions. This schema is used both for annotation and as the intermediate attributes for the first stage of normalization. Annotators could label attribute values outside of the predefined schema by selecting an umbrella "Other" option. Common attribute values identified in this way were added to the predefined schema when appropriate.

For model training, the human annotation is converted to BIO (beginning-inside-outside) labels using the IOB2 schema (Krishnan and Ganapathy, 2005) for the attribute-type named entities $\mathcal{E}$ = {BRAND, MATERIAL, COLOR, DIMENSION, SUBJECT, LIFE_STAGE, FEATURE, LOCATION, SIZE, FINISH, PRICE, STYLE, SHAPE, PATTERN, NUMBER_ITEMS, NUMBER_COMPONENTS}.

The inter-annotator agreement (IAA) on entity level is around 68% F1, calculated by holding one annotator's labels as ground truth and the other as system output. Among the entities that the annotators agreed on, the option normalization accuracy is around 94%.

3.3. Weak Labels & Synthetic Data

To produce weak labels, a variety of attributes and other structured product data (e.g. product category for O labels per the BIO schema) from known add-to-cart events were string-matched against the preceding user query. As conflicting information came from the various sources, an unweighted majority vote was then applied to the candidates per Ratner et al. (2017). This allowed the use of the available diverse structured data sources to reduce noise. In the future, other methods for weak label selection could be applied per Ratner et al. (2016). The resultant weak labels were used together with human-labeled data to train the NER model.

Zhang et al. (2021) use a similar system for getting large amounts of weak labels, and report that models trained with weakly labeled data alone (F1 0.6) are inferior to those trained with much less human labeled data alone (F1 0.62). However, in our system, the weak labels appear to be of higher quality than the human data, at least for some labels, as shown in Table 2, which may point to an opportunity to improve task design.

Adding generative model predictions from Flan-T5-XL (Chung et al., 2022) to the weak labels had a neutral affect on the performance of the model. Further experimentation with larger generative models is planned.

In addition to weak labels from customer add-to-cart events, synthetic search experiences and corresponding labels were created. Synthetic search queries were created in a number of ways: random distortions to create misspellings for existing labeled data; various subjects, e.g. animal types, from the structured product data with SUBJECT labels; and using SKU IDs and product category names as O labels. Adding additional O labels via product categories had the greatest positive effect as shown in the ablation study in Table 3.

3.4. Model Architecture

The training set $\{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ consists of N training examples where $(x^{(i)}, y^{(i)})$ is the *i*th instance consisting of a tuple of the input query $x^{(i)}$ and its labels $y^{(i)}$. A single search query is symbolized for the *i*th training example with a variable natural number, $M^{(i)}$, of tokens by a vector

$$x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_{M^{(i)}}^{(i)}),$$

where each element, $x_j^{(i)}$, is a natural language token. The two-dimensional array $y^{(i)}$ represents the labels for the *i*th training sample, such that

$$y^{(i)} = (y_1^{(i)}, y_2^{(i)}, \ldots, y_{M^{(i)}}^{(i)}),$$

where there are $M^{(i)}$ label vectors and each $y_j^{(i)}$ is an NER label in the form of a one-hot vector of the fixed size L of the label set.

The DistilBERT model (Sanh et al., 2019) from Huggingface (Wolf et al., 2020) was used to generate a two-dimensional array of logits corresponding to the labels as follows, where *f* is the DistilBERT feed-forward transformer network consisting of an embedding layer, five transformer block layers and a linear classifier:

$$\hat{y}^{(i)} = f(x^{(i)}).$$

The DistilBERT uncased pre-trained English base model was fine-tuned using the Huggingface NER pipeline with `seqeval` (Nakayama, 2018) as the metric for evaluation and multi-class cross-entropy loss, such that the loss function can be expressed as

$$\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\sum_{j}^{M^{(i)}} \sum_{k}^{L} y_{jk}^{(i)} \log\left(\sigma(\hat{y}_j^{(i)})_k\right),$$

with σ, the softmax function, defined as follows:

$$\sigma(\hat{y}_j^{(i)})_k = \frac{e^{(\hat{y}_{jk}^{(i)})}}{\sum_l^L e^{(\hat{y}_{jl}^{(i)})}}.$$

Missing token labels, i.e. tokens where no BIO label could be identified during the weak labeling process, were ignored for purposes of calculating the model loss. The model was then fine-tuned with the objective of minimizing the loss over the training set using `AdamW`, i. e. Adam (Kingma and Ba, 2014) with weight decay (Loshchilov and Hutter, 2017).

3.5. Attribute Value Normalization

For span-identification-based AVE, a normalization step is required to translate the text span in the customer query and its NER label onto an attribute value and title respectively in the production system. In the simplest case, a one-to-one mapping of NER labels to attribute titles exists and only the text spans need to be normalized to attribute values. This is not the case in our system, where there is a many-to-many mapping of NER labels to attribute titles, which may have different names and unique IDs across product categories even when synonymous.

In the present study, we thus take a two-step normalization approach. First, in the human annotation, the spans identified by the annotators are matched to both an intermediate attribute title and an intermediate attribute value from a pre-defined schema of frequently searched attributes. The intermediate attribute schema groups synonymous and similar attribute titles and attribute values together, so “Uphostery Material” and “Leg Material” become just `MATERIAL`. Likewise, in the search query “crimson sofa”, the annotator would mark up the string “crimson” with “`COLOR`: Red”, where ‘Red’ is the intermediate attribute value. At inference time, a gazetteer created from the human annotation is used to normalize text spans onto the intermediate attribute values. An exception is numerical attributes and `BRAND`, which are mapped onto an intermediate attribute title only.

The second normalization step occurs when mapping the intermediate attribute titles and values onto actual product attributes in the structured data, where both the attribute title and value may have forms different to the intermediate ones and the number of attribute titles and values is much larger. The strategy used for the second stage of normalization varies, depending on the type of attribute as described below.

In the second normalization step at the attribute title level, some NER labels can easily be mapped onto a small number of class-agnostic attribute titles, as is the case for `BRAND`, `PRICE` and `STYLE`; all products have these attributes and they are agnostic to the product category. For other NER labels, such as `MATERIAL`, there are many different attribute titles that these could map to, many of which are dependent on the product category and call out the material of the components of the product, e. g. “Upholstery Material” and “Frame Material” are prominent in the “Beds” category, but “Top Material” and “Base Material” are used for “Dining Tables”. In this case, a statistical methodology similar to that used by Zhang et al. (2021) is required to map NER labels onto attribute titles.

Different approaches are also used at the attribute value level, depending on the type of attribute. Some attribute titles have an open set of attribute values. For `DIMENSION` and `PRICE`, there is no restriction on the value it can take, except that they are positive floating-point numbers. Likewise for `BRAND`, new brands are added to the catalog continuously, so these do not form a closed set. Other attribute titles, e.g. `STYLE`, do have a small, fixed set of attribute values, which do not change much over time. These values are determined in a curated way by domain-owners in the company.

For example, for `BRAND`, which has a large open set of attribute values, the first stage of normalization consists only of mapping the NER label onto its intermediate attribute title. For the second stage of normalization, the NER label is mapped onto one of two actual brand name attribute titles in the product data. The identified text spans can be mapped onto attribute values per the methodology used by Zhang et al. (2021). For the online experiment (Section 4) using the `BRAND` span identified by this model in the ranker, however, we did not use this methodology as it added the complication of needing the resulting mapping available in production. Instead, for the second layer of normalization, we used token-level Jaccard index as a measure of text similarity between the actual brand name attribute values and the span identified by the model. This avoids having to update a mapping as new brands are added to the catalog and is easy to implement in production.

For `STYLE`, an example of an NER label with a

	F1 COLOR	F1 STYLE	F1 BRAND	F1 overall
DistilBERT	0.818	**0.625**	**0.71**	**0.654**
RoBERTa base	0.797	0.619	0.682	0.632
RoBERTa large	0.793	0.601	0.683	0.623
SimCSE	**0.821**	0.591	0.677	0.626

Table 4: Change in performance by language model

	BRAND	MATERIAL	COLOR	DIMENSION	SUBJECT	FINISH
NER label	0.71	0.73	0.87	0.85	0.55	0.73
End to end	-	0.66	0.82	-	0.50	0.73

	SIZE	LIFE_STAGE	PRICE	NUMBER_ITEMS	SHAPE	PATTERN
NER label	0.84	0.83	0.75	0.87	0.80	0.59
End to end	0.83	0.81	-	0.84	0.77	0.59

	FEATURE	LOCATION	STYLE	NUMBER_COMPONENTS
NER label	0.41	0.76	0.64	0.58
End to end	0.41	0.74	0.63	0.52

Table 5: F1 scores for NER labels (i.e. attribute titles) & micro-averaged end-to-end (i.e. attribute value) F1 scores

closed set of intermediate attribute values, there is a single final attribute title across classes. Human annotation was initially used to create a mapping from surface forms to normalized forms of the attribute values. Per Zipf's law, the most commonly occurring 200 tokens cover the majority of token instances, so the effort for doing this is low, as this data is already annotated for training. Without any additional annotation, it results in a first-stage normalization accuracy of 0.985. Normalization accuracy is calculated as the sum of the end-to-end true positives (true positive intermediate attribute value and true positive NER label), divided by the true positives for the NER label, as measured against the human-annotated test set. An almost one-to-one mapping of the intermediate attribute title and values to structured data is then applied. In this instance, more advanced mapping methodologies would bring diminishing returns.

4. Evaluation

We used the `seqeval` package (Nakayama, 2018) to evaluate the model against a hold-out set of the human-annotated data. F1 scores at attribute title (i.e. NER label) and attribute value (i.e. end-to-end) level are reported in Table 5, with example experiments on combinations of synthetic, weak and human labels shown in Table 2. BRAND, DIMENSION and PRICE are not normalized to attribute values, so performance is only recorded for these at the attribute title level.

The current model meets the latency requirements, with an average speed in offline testing of 6 ms on GPU[2] and 12 ms on CPU[3] for a single query. In online testing, there were small but significant increases in the range of 0.5% to 2.6% in latency for search queries overall, although these were largely offset by preprocessing improvements after launch. Other transformer models, e.g. RoBERTa (Liu et al., 2019), SimCSE (Gao et al., 2021), were tested (Table 4) but they did not meet the latency requirements. In addition, the bigger transformer models did not give a boost in performance as shown Table 4; the hypothesis is that a larger model quickly over-fits for short search queries and a simpler model with fewer trainable parameters is preferable.

Offline ranking experiments were conducted where the brand name identified by the model in user queries was compared against the brand name and product name for products to be ranked. Token-level Jaccard index between these was calculated and used to boost relevant products. The product ranking with boosted brand names was then compared against existing product rankings. When using the brand attribute value from the model, an average lift of 4.65% was observed in the normalized discounted cumulative gain at rank 48 ($nDCG_{48}$).

The nDCG score was calculated in the typical fashion with

$$DCG_p = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)},$$

[2]Run on a machine with a single NVIDIA® Tesla® P100 GPU.

[3]Run on a machine with 16 CPUs of type Intel® Xeon® Processor E5-2630.

where the relevance value is calculated as

$$rel_i = \alpha * view_i + \beta * atc_i + \gamma * order_i$$

and $view_i$ is a binary value 1 if the product page was viewed and 0 if not. Analogously atc_i is whether an add-to-cart event occurred and $order_i$ is whether an order was placed. The weights α, β and γ are heuristically determined by the relative frequency of view, add-to-cart and order events respectively, with the most weight placed on orders. The DCG score is then normalized as follows:

$$nDCG_p = \frac{DCG_p}{IDCG_p},$$

where Ideal DCG (IDCG) is the maximum possible DCG score with the products ordered according to relevance value, with highest relevance values first.

An A/B test was conducted to evaluate the model's impact, with the variation using the methodology and the weights determined in the offline testing described above to rank a subset of products retrieved by a semantic search system per Liu et al. (2022b). This resulted in significant overall lifts in product views (1.45%) and add-to-cart (3.45%) and conversion rates (2.99%). The variant helped to reduce friction during users' search journey and enabled users to find relevant products with less effort, as we observed significant decreases in reformulation rate (-1.25%) and landing page exit rate (-0.76%). Further A/B tests on other attributes and use cases are planned.

5. Conclusion & Discussion

In this paper, we presented an NLP application in the domain of e-commerce, focusing on identifying explicit attributes in customer search queries using weak labels and a transformer-based NER approach. To overcome the challenge of a large label space for attributes, we employed a two-stage normalization process. For the first stage, we leveraged human annotation to create a normalization gazetteer, while the second stage of normalization varied depending on the specific attribute under consideration.

The model met the strict latency requirements of an e-commerce website and was put into production. It showed significant business value via an initial A/B test using the `BRAND` output, and more tests are planned for additional attributes and use cases going forward, including dynamic filtering of products.

We showed that an increase model size and complexity did not necessarily increase the performance of the model, although further experiments with larger language models are planned. A significant boost was gained by adding product category as a separate `O` label, as well as by adding synthetic data. Contrary to previous studies, our human data did not outperform our weak and synthetic data on most labels, indicating a possible opportunity to improve both task design and the intermediate attribute schema.

Also left to explore are implicit attributes per Luo et al. (2022); using more powerful generative AI models to generate labels; and a multilingual version of the model for non-English catalogs.

Overall, this study demonstrates the successful application of weak labels and transformer-based NER for explicit attribute identification. The deployment of our model in a real-world setting, along with the observed business value, highlights its practical significance. Our proposed future directions open up exciting opportunities for further advancements in this domain.

6. Bibliographical References

Ali Ahmadvand, Surya Kallumadi, Faizan Javed, and Eugene Agichtein. 2021. DeepCat: Deep category representation for query understanding in e-commerce search.

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*. Prentice-Hall, Inc., USA.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

American Psychological Association. 1983. *Publications Manual*. American Psychological Association, Washington, DC.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Edoardo Barba, Luigi Procopio, and Roberto Navigli. 2022. ExtEnD: Extractive entity disambiguation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2478–2488, Dublin, Ireland. Association for Computational Linguistics.

David J. Brenes, Daniel Gayo-Avello, and Kilian Pérez-González. 2009. Survey and evaluation of query intent detection methods. In *Proceedings of the 2009 Workshop on Web Search Click Data*, WSCD '09, pages 1–7, New York, NY, USA. Association for Computing Machinery.

Andrei Broder. 2002. A taxonomy of web search. *ACM SIGIR Forum*, 36(2):3–10.

BSI. 1973a. *Natural Fibre Twines*, 3rd edition. British Standards Institution, London. BS 2570.

BSI. 1973b. Natural fibre twines. BS 2570, British Standards Institution, London. 3rd. edn.

A. Castor and L. E. Pollux. 1992. The use of user modelling to guide inference and learning. *Applied Intelligence*, 2(1):37–53.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. BERT for joint intent classification and slot filling.

Wei-Te Chen, Yandi Xia, and Keiji Shinzato. 2022a. Extreme multi-label classification with label masking for product attribute value extraction. In *Proceedings of The Fifth Workshop on e-Commerce and NLP (ECNLP 5)*, pages 134–140, Dublin, Ireland. Association for Computational Linguistics.

Yan Chen, Shujian Liu, Zheng Liu, Weiyi Sun, Linas Baltrunas, and Benjamin Schroeder. 2022b. WANDS: Dataset for product search relevance assessment. In *Advances in Information Retrieval: 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10–14, 2022, Proceedings, Part I*, page 128–141, Berlin, Heidelberg. Springer-Verlag.

Xiang Cheng, Mitchell Bowden, Bhushan Ramesh Bhange, Priyanka Goyal, Thomas Packer, and Faizan Javed. 2020. An end-to-end solution for named entity recognition in eCommerce search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35:15098–15106.

J.L. Chercheur. 1994. *Case-Based Reasoning*, 2nd edition. Morgan Kaufman Publishers, San Mateo, CA.

N. Chomsky. 1973. Conditions on transformations. In *A festschrift for Morris Halle*, New York. Holt, Rinehart & Winston.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Brooke Cowan, Sven Zethelius, Brittany Luk, Teodora Baras, Prachi Ukarde, and Daodao Zhang. 2015. Named entity recognition in travel-related search queries. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(2):3935–3941.

Hongbo Deng and Yi Chang. 2020. An introduction to query understanding. In Yi Chang and Hongbo Deng, editors, *Query Understanding for Search Engines*, pages 1–13. Springer International Publishing.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Umberto Eco. 1990. *The Limits of Interpretation*. Indian University Press.

Gilad Fuchs and Yoni Acriche. 2022. Product titles-to-attributes as a text-to-text task. In *Proceedings of The Fifth Workshop on e-Commerce and NLP (ECNLP 5)*, pages 91–98, Dublin, Ireland. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Joan Guisado-Gámez, David Tamayo-Domènech, Jordi Urmeneta, and Josep Larriba-Pey. 2021. ENRICH: A query rewriting service powered by wikipedia graph structure. *Proceedings of the International AAAI Conference on Web and Social Media*, 10:47–50.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.

Homa B. Hashemi, Amir Asiaee, and Reiner Kraft. 2016. Query intent detection using convolutional neural networks. In *International conference on web search and data mining, workshop on query understanding*.

Paul Gerhard Hoel. 1971a. *Elementary Statistics*, 3rd edition. Wiley series in probability and mathematical statistics. Wiley, New York, Chichester. ISBN 0 471 40300.

Paul Gerhard Hoel. 1971b. *Elementary Statistics*, 3rd edition, Wiley series in probability and mathematical statistics, pages 19–33. Wiley, New York, Chichester. ISBN 0 471 40300.

Otto Jespersen. 1922. *Language: Its Nature, Development, and Origin*. Allen and Unwin.

Joo-Kyung Kim, Gokhan Tur, Asli Celikyilmaz, Bin Cao, and Ye-Yi Wang. 2016. Intent detection using semantically enriched word embeddings. In *2016 IEEE spoken language technology workshop (SLT)*, pages 414–419. IEEE.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Zornitsa Kozareva, Qi Li, Ke Zhai, and Weiwei Guo. 2016. Recognizing salient entities in shopping queries. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 107–111, Berlin, Germany. Association for Computational Linguistics.

Vijay Krishnan and Vignesh Ganapathy. 2005. Named entity recognition. *Stanford Lecture CS229*.

Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. 2020. Dice loss for data-imbalanced NLP tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 465–476, Online. Association for Computational Linguistics.

Ying Li, Zijian Zheng, and Honghua (Kathy) Dai. 2005. Kdd cup-2005 report: Facing a great challenge. *ACM SIGKDD Explorations Newsletter*, 7(2):91–99.

Heran Lin, Pengcheng Xiong, Danqing Zhang, Fan Yang, Ryoichi Kato, Mukul Kumar, William Headden, and Bing Yin. 2020. Light feed-forward networks for shard selection in large-scale product search. In *Proceedings of ACM SIGIR Workshop on eCommerce (SIGIR eCom'20)*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Zheng Liu, Wei Zhang, Yan Chen, Weiyi Sun, Tianchuan Du, and Benjamin Schroeder. 2022a. Towards generalizable semantic product search by text similarity pre-training on search click logs. *ECNLP 2022*.

Zheng Liu, Wei Zhang, Yan Chen, Weiyi Sun, Tianchuan Du, and Benjamin Schroeder. 2022b. Towards generalizeable semantic product search by text similarity pre-training on search click logs. In *Proceedings of the Fifth Workshop on e-Commerce and NLP (ECNLP 5)*, pages 224–233, Dublin, Ireland. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization.

Chen Luo, William Headden, Neela Avudaiappan, Haoming Jiang, Tianyu Cao, Qingyu Yin, Yifan Gao, Zheng Li, Rahul Goutam, Haiyang Zhang, and Bing Yin. 2022. Query attribute recommendation at amazon search. In *Proceedings of the 16th ACM Conference on Recommender Systems*, RecSys '22, page 506–508, New York, NY, USA. Association for Computing Machinery.

Bodhisattwa Prasad Majumder, Aditya Subramanian, Abhinandan Krishnan, Shreyansh Gandhi, and Ajinkya More. 2018. Deep recurrent neural networks for product attribute extraction in eCommerce.

Hiroki Nakayama. 2018. seqeval: A python framework for sequence labeling evaluation. Software available from https://github.com/chakki-works/seqeval.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages.

Duangmanee Putthividhya and Junling Hu. 2011. Bootstrapped named entity recognition for product attribute extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher

Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *Proc. VLDB Endow.*, 11(3):269–282.

Alexander Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems*, 29:3567–3575.

Ryokan Ri, Ikuya Yamada, and Yoshimasa Tsuruoka. 2022. mLUKE: The power of entity representations in multilingual pretrained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7316–7330, Dublin, Ireland. Association for Computational Linguistics.

Daniel E. Rose and Danny Levinson. 2004. Understanding user goals in web search. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 13–19, New York, NY, USA. Association for Computing Machinery.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 373–382, New York, NY, USA. Association for Computing Machinery.

Keiji Shinzato, Naoki Yoshinaga, Yandi Xia, and Wei-Te Chen. 2022. Simple and effective knowledge-driven query expansion for QA-based product attribute extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 227–234, Dublin, Ireland. Association for Computational Linguistics.

Charles Joseph Singer, E. J. Holmyard, and A. R. Hall, editors. 1954–58. *A history of technology*. Oxford University Press, London. 5 vol.

Parikshit Sondhi, Mohit Sharma, Pranam Kolari, and ChengXiang Zhai. 2018. A taxonomy of queries for e-commerce search. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, ACM SIGIR '18, page 1245–1248, New York, NY, USA. Association for Computing Machinery.

Jannik Strötgen and Michael Gertz. 2012. Temporal tagging on different domains: Challenges, strategies, and gold standards. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 3746–3753, Istanbul, Turkey. European Language Resource Association (ELRA).

Ning Su, Jiyin He, Yiqun Liu, Min Zhang, and Shaoping Ma. 2018. User intent, behaviour, and perceived satisfaction in product search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, page 547–555, New York, NY, USA. Association for Computing Machinery.

S. Superman, B. Batman, C. Catwoman, and S. Spiderman. 2000. *Superheroes experiences with books*, 20th edition. The Phantom Editors Associates, Gotham City.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Manos Tsagkias, Tracy Holloway King, Surya Kallumadi, Vanessa Murdock, and Maarten de Rijke. 2021. Challenges and research opportunities in eCommerce search and recommendations. *ACM SIGIR Forum*, 54(1):1–23.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021a. Automated concatenation of embeddings for structured prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2643–2660, Online. Association for Computational Linguistics.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021b. Improving named entity recognition by external context retrieving and cooperative learning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1800–1812, Online. Association for Computational Linguistics.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013.

OntoNotes release 5.0 LDC2013T19. *Linguistic Data Consortium, Philadelphia, PA, USA*.

Musen Wen, Deepak Kumar Vasthimal, Alan Lu, Tian Wang, and Aimin Guo. 2019. Building large-scale deep learning system for entity recognition in e-commerce search. In *Proceedings of the 6th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies*, BDCAT '19, page 149–154, New York, NY, USA. Association for Computing Machinery.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Chao-Yuan Wu, Amr Ahmed, Gowtham Ramani Kumar, and Ritendra Datta. 2017. Predicting latent structured intents from shopping queries. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 1133–1141, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Huimin Xu, Wenting Wang, Xin Mao, Xinyu Jiang, and Man Lan. 2019. Scaling up open tagging from tens to thousands: Comprehension empowered attribute value extraction from product title. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5214–5223, Florence, Italy. Association for Computational Linguistics.

Danqing Zhang, Zheng Li, Tianyu Cao, Chen Luo, Tony Wu, Hanqing Lu, Yiwei Song, Bing Yin, Tuo Zhao, and Qiang Yang. 2021. QUEACO: Borrowing treasures from weakly-labeled behavior data for query attribute value extraction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 4362–4372, New York, NY, USA. Association for Computing Machinery.

TAAL: Target-Aware Active Learning

Kunal Kotian*, **Indranil Bhattacharya***, **Shikhar Gupta***
Kaushik Pavani, **Naval Bhandari**, **Sunny Dasgupta**
{kkotian, bindrani, gupshik, sripava, naval, sunnyd}@amazon.com

Abstract

Pool-based active learning techniques have had success producing multi-class classifiers that achieve high accuracy with fewer labels compared to random labeling. However, in an industrial setting where we often have class-level business targets to achieve (e.g., 95% recall at 95% precision for each class), active learning techniques continue to acquire labels for classes that have already met their targets, thus consuming unnecessary manual annotations. We address this problem by proposing a framework called Target-Aware Active Learning that converts any active learning query strategy into its target-aware variant by leveraging the gap between each class' current estimated accuracy and its corresponding business target. We show empirically that target-aware variants of state-of-the-art active learning techniques achieve business targets faster on 2 open-source image classification datasets and 2 proprietary product classification datasets.

Keywords: active learning, multi-class classification

1. Introduction

Active learning is a popular approach used to reduce the manual labeling effort required to train a classifier. In active learning, we iteratively acquire labels from annotators and use them to (re)-train a classifier. Previous research (Lewis and Gale, 1994; Settles, 2009; Gal et al., 2017; Lin and Parikh, 2017) has demonstrated that choosing a batch of instances with small batch sizes offers a good trade-off between user interactivity and number of labels required to create a classifier. In each active learning iteration, we perform two operations: (i) use a *query strategy* to judiciously select a fixed-sized subset (a batch) of unlabeled instances and send them to the annotators, and, (ii) train a classifier using new and previously labeled instances. To pick the next set of unlabeled instances for annotation, we rank all instances in unlabeled data based on scores such as margin, entropy, expected loss reduction, or sub-modular information measures like mutual information, conditional gain *etc.*, and select the instances which are most likely to benefit the classifier.

Imagine using the active learning paradigm to improve a 3-class classifier with 4 labels: *pens*, *pencils*, *erasers*, and *not-in-k* (*i.e.*, the background class which does not contain pens, pencils, or erasers). In an industrial setting, annotators often work backwards from achieving class-level business-specified accuracy targets, where *accuracy* refers to any measure of a classifier's ability to discriminate between classes, *e.g.*, precision, recall, classification accuracy, false positive rate. *E.g.*, the task could be to achieve 90% recall at 85% precision for each class-of-interest. Vanilla active learning algorithms are sub-optimal for such applications because they ignore business targets. In other words, they continue to select unlabeled data that improves the accuracy of a class even if its estimated accuracy exceeds the business targets; this annotation budget could instead be used to improve other classes that are yet to meet their targets. To this end, we propose a framework **T**arget-**A**ware **A**ctive **L**earning (TAAL) that can be applied on top of any active learning strategy to create its target-aware variant. If the classification task has class-level business targets defined, TAAL increases the likelihood of achieving the targets on all classes-of-interest given a fixed labeling budget, by leveraging the gap between each class' target and estimated accuracy. Note that TAAL will not help achieve global business targets (*e.g.* micro-recall) faster than baseline query strategies.

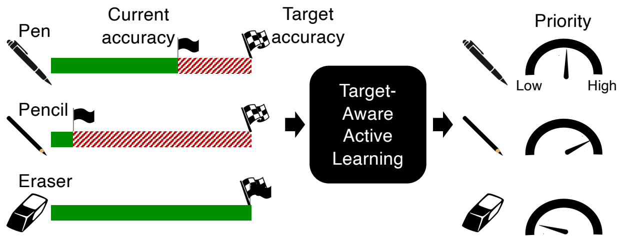


Figure 1: TAAL computes relative priority weights for each class taking into account its current estimated accuracy (solid flag) and target accuracy (checkered flag). Here, the expected priority is *pencil* $>$ *pen* $>$ *eraser*.

2. Target-Aware Active Learning

Consider an active learning setup employing an arbitrary query strategy that generates a query Q, *i.e.*, a queue of instances to be labeled. We make this query strategy *target-aware* under the TAAL frame-

*These authors contributed equally to this work

The Seventh Workshop on e-Commerce and NLP (ECNLP 7), pages 136–144
21 May, 2024.

work shown in Fig. 2. We first quantify the relative attention each class requires to achieve its accuracy targets; this is handled by the *Class Priority Weight Computer* (Sec. 2.1). We then use a *Class Priority Aware Query Adapter* (Sec. 2.3) to select a subset of instances from Q and reorder them into a queue Q_p prioritizing classes that require the most attention. This process necessitates identification of the subset of classes whose accuracy is likely to improve if a given instance is labeled. This is accomplished by the *Candidate Labels Generator* (Sec. 2.2). Thus, the TAAL framework can be applied to an arbitrary query strategy to convert its output query Q into a target-aware version Q_p. The following sections discuss the three components of TAAL.

2.1. Class Priority Weight Computer

Let $Y = \{y_1, y_2, ..., y_k\}$ be the set of classes in a multi-class classifier. In TAAL, at any active learning iteration ℓ, a Class Priority Weight Computer quantifies the 'priority weights' (*i.e.*, relative attention) that each class needs to attain its accuracy target. We update these class priority weights at every active learning iteration. Fig. 1 shows an example to motivate class priority weights where the 'pencil' class gets the highest priority because its estimated accuracy is farthest away from its target. For a class y_i at iteration ℓ, let $\rho_{i,\ell}$ represent the target value of an accuracy metric of interest and $\hat{\rho}_{i,\ell}$ be its estimated value. Then, any realization of the Class Priority Weight Computer must generate a weight $w_{i,\ell}$ for class y_i at iteration ℓ such that

$$w_{i,\ell} \propto \max(\rho_{i,\ell} - \hat{\rho}_{i,\ell}, 0) \tag{1}$$

If the goal is to decrease a target metric *e.g.*, false positive rate, it should be expressed as its negative in Eq. (1) *e.g.*, negative false positive rate. In this paper, we implement a realization of Class Priority Weight Computer based on a class-level accuracy metric ρ such that, at each active learning iteration ℓ, it computes a priority weight $w_{i,\ell}$ for class y_i as

$$w_{i,\ell} = \begin{cases} \max(\delta_{i,\ell}, 0) & \text{if } \sum_{j=1}^{k} \max(\delta_{j,\ell}, 0) > 0, \\ 1 & \text{otherwise.} \end{cases} \tag{2}$$

where $\delta_{i,\ell} = \rho_{i,\ell} - \widehat{\rho}_{i,\ell}$, with $\rho_{i,\ell}$ and $\widehat{\rho}_{i,\ell}$ being the target and estimated values of the accuracy metric respectively.

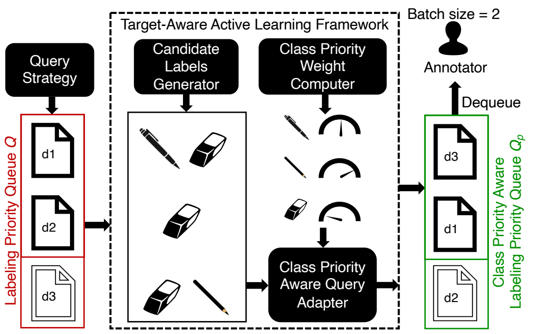


Figure 2: TAAL transforms a queue of unlabeled instances Q (left) generated by an arbitrary query strategy, into queue Q_p (right) by prioritizing classes based on the relative gaps between their estimated accuracy and business target. TAAL uses a *Class Priority Weight Computer* to quantify the attention each class needs. *Candidate Labels Generator* identifies classes that are likely to see accuracy improvement if an unlabeled instance is labeled. *Class Priority Aware Query Adapter* reorders Q into Q_p by prioritizing instances that are likely to improve accuracy over classes that need attention. With a labeling budget of 2 documents, vanilla query strategies would have surfaced documents d_1 and d_2 to the annotator. TAAL prioritizes d_3 in Q_p as d_3 will likely improve the *pencil* class which requires the most attention.

2.2. Candidate Labels Generator

The Candidate Labels Generator identifies the subset of classes whose accuracy is likely to improve if a particular instance from the unlabeled data pool is labeled. Let Y_x be the subset of classes with a high likelihood of containing the true class of an unlabeled instance x. We construct Y_x using any information available on x; *e.g.*, it can leverage the classifier's scores across classes to identify candidate classes. We hypothesize that retraining the classifier after labeling x is likely to improve its accuracy over all classes in Y_x. Any realization of the Candidate Label Generator should generate a candidate labels set Y_x for x such that the true class of x has a high probability of being a member of Y_x, while keeping $|Y_x|$ as small as possible. In this paper we implement a realization of the Candidate Labels Generator that produces a candidate class labels set for an input instance x given by

$$Y_x = \left\{ y \in Y \,\middle|\, \frac{P(y \mid x)}{\max_{z \in Y} P(z \mid x)} \geq t_c \right\} \tag{3}$$

where $P(y \mid x) \in [0, 1]$ is the probability score assigned to class y by the active learning classifier given instance x, and t_c is a configurable cutoff threshold. We set t_c to 0.5. This means all classes with score $\geq 50\%$ of the maximum score are treated as candidate class labels for each instance.

2.3. Class Priority Aware Query Adapter

Class Priority Aware Query Adapter reorders the input queue of unlabeled instances to reflect class priority weights. Consider the example in Fig. 2 in which instance d_3 has rank 3 in the queue of instances Q from an active learning query strategy (non-target-aware). One of d_3's candidate labels, *Pencil*, has the highest priority weight. So, the Class Priority Aware Query Adapter pushes d_3 ahead of d_1 and d_2 in its reordered output queue Q_p. Our implementation of the *Class Priority Aware Query Adapter* is in Algorithm 1. In each active learning iteration, we start with a ranked queue of unlabeled instances Q from the underlying query strategy. Then, we invoke *Class Priority Weight Computer* to update priority weights W. *Class Priority Aware Query Adapter* starts with sampling a class $y_{selected}$ from Y with priority weights W as sampling probability (Line 3). Then it traverses Q from top to bottom, selecting the first instance d which has $y_{selected}$ in its candidate labels set (Lines 5 to 16), and enqueues d in Q_p if it is not present in Q_p already (Line 8). Finally, it removes class $y_{selected}$ from the candidate labels set of d so that instance d is not picked for $y_{selected}$ again (Line 9). If a class is not a candidate label for any instance in Q, its priority weight is reset to 0 since it cannot be prioritized in the current learning iteration. The process is repeated until $|Q_p|$ reaches N_q or no classes can be prioritized (Lines 2 to 17). The computational complexity of *Class Priority Weight Computer* (Eq. 2) and *Candidate Labels Generator* (Eq. 3) is $\mathcal{O}(k)$. Hence, the computational complexity of the *Class Priority Aware Query Adapter* (Algorithm 1) is $\mathcal{O}(|Q| \cdot |Q_p| \cdot k)$. In practice, $k << |Q|$ and $|Q_p| << |Q|$; hence, the overall computational complexity of a TAAL-enabled query strategy is $\mathcal{O}(|Q|)$.

Algorithm 1: Class Priority Aware Query Adapter

Inputs : (i) Queue Q of instances from a query strategy, (ii) Class priority weights array W as $[w_1, w_2, ..., w_k]$ where w_j is the priority weight for class y_j from Eq. (1), (iii) No. of output instances $N_q \leq |Q|$

Output : Class priority aware queue Q_p of instances.

1 **Initialize:** (i) Empty queue Q_p, (ii) Candidate labels map $\mathcal{F}$ mapping each instance x in Q to $\mathcal{F}[x]$, with $\mathcal{F}[x] \leftarrow Y_x$ computed using Eq. (3).
2 **while** $|Q_p| < N_q$ ***and*** $\max(W) \neq 0$ **do**
3   Randomly sample 1 class $y_{selected} \in Y$ per weights W.
4     $i \leftarrow 1$
5   **while** $i \leq |Q|$ **do**
6         $d \leftarrow$ instance at index i in Q
7   **if** $y_{selected} \in \mathcal{F}[d]$ **then**
8   **if** $d \notin Q_p$ **then** Q_p.enqueue(d) ;
9             $\mathcal{F}[d] \leftarrow \mathcal{F}[d] \setminus \{y_{selected}\}$
10   Exit inner while loop.
11   **else**
12             $i \leftarrow i + 1$
13   **if** $i > |Q|$ **then** reset weight for $y_{selected}$ in W to 0 ;
14   **end**
15   **end**
16 **end**
17 **return** Q_p

3. Experiments

We use pool-based active learning setup (Lewis and Gale, 1994; Settles, 2009; Gal et al., 2017; Lin and Parikh, 2017) consisting of a model $\mathcal{M}$, a seed set of randomly sampled labeled instances $(x_i, y_i) \in \mathcal{D}_{\text{seed}}$ ($|\mathcal{D}_{\text{seed}}|$ = 500) to initialize $\mathcal{M}$ in the first iteration, an unlabeled data pool $\mathcal{D}_{\text{pool}}$, and a query strategy $\mathcal{R}$. We run active learning over a series of iterations until a budget of L ($L = 6000$) labeled instances. At each iteration ℓ, we acquire a batch of B ($B = 500$) new instances from $\mathcal{D}_{\text{pool}}$ governed by $\mathcal{R}$, and re-train model $\mathcal{M}$. We conduct experiments on 2 image-based and 2 text-based datasets (Sec. 3.2) which are fully-labeled (including "unlabeled" data $\mathcal{D}_{\text{pool}}$) and multi-class. We use a labeling bot to simulate a human-in-the-loop active learning setup (Gal et al., 2017; Lin and Parikh, 2017; Siddhant and Lipton, 2018) using the fully-labeled datasets. The bot provides labels for the data instances selected by the query strategy $\mathcal{R}$. We compare the baseline query strategies listed in Sec. 3.1 against their Target-Aware (TA-) variants created by applying TAAL on top of each query strategy. For image datasets we use ENTROPY, BADGE, CORESET, and SIMILAR as baseline query strategies because they have produced SOTA results on several benchmarks (Citovsky et al., 2021). SIMILAR is an interesting baseline because it gives high accuracy for rare classes in the dataset. For product classification datasets, we use E3G as the baseline query strategy to mimic our internal active learning setup.

3.1. Baseline Active Learning Query Strategies

We discuss active learning strategies in detail under related work (section 5). Below we describe the specific query strategies used in experiments.

ENTROPY (Settles, 2009) sampling queries the labels of instances for which the prediction of the classifier is maximally uncertain in terms of entropy of confidence scores. It selects a batch of B instances from $\mathcal{D}_{\text{pool}}$ with highest entropy in model's output, where the entropy $\mathcal{H}_x$ for each instance x is measured over say k classes as $-\sum_{i=1}^{k} P(y_i|x) \log P(y_i|x)$. **BADGE** (Ash et al., 2019a) samples a batch of B instances from $\mathcal{D}_{\text{pool}}$ that are disparate and high magnitude when represented in a hallucinated gradient space, a strategy designed to incorporate both predictive uncertainty and sample diversity into every selected batch. BADGE uses the k-Means++ seeding algorithm on $\{g_x : x \in \mathcal{D}_{\text{pool}}\}$ where g_x is the gradient embedding of an instance x using current model weights w. **CORESET** (Sener and Savarese, 2017) addresses the problem of selecting diverse examples in a batch by posing active learning as coreset selection, *i.e.* choosing a set of points such that a model learned over the selected subset is competitive for the remaining data points. It samples a batch of B instances from $\mathcal{D}_{\text{pool}}$ by solving the k-center problem on $\{z_x : x \in \mathcal{D}_{\text{pool}}\}$, where z_x is the embedding of an instance x derived from the penultimate layer of the model. **SIMILAR** (Kothawade et al., 2021) is a unified active learning framework that employs different sub-modular information measures, namely, Submodular Mutual Information, Submodular Conditional Gain, Submodular Conditional Mutual Information as query acquisition functions. SIMILAR not only works in standard active learning setup but also easily extends to more realistic settings such as having class imbalance in the data distribution, and provides the SOTA solution for active learning. **E3G** (Slivkins, 2019) (**E**xplore-**E**xploit-**EG**reedy) uses the ϵ-greedy algorithm to choose between Exploration and Exploitation. The exploration strategy performs k-means clustering-based stratified random sampling, whereas the exploitation strategy is entropy-based uncertainty sampling. To form a batch of B instances per active learning iteration, each instance $x \in \mathcal{D}_{\text{pool}}$ is chosen independently by E3G with probability ϵ from the exploration strategy, and with probability $1 - \epsilon$ from the exploitation strategy. During the early iterations of active learning, E3G focuses on the exploration strategy, whereas later it emphasizes the exploitation strategy. It does this by varying ϵ across iterations following $f(N_\ell) = \max(e^{-\gamma N_\ell}, \epsilon_{\text{min}})$, where γ is the decay rate, ϵ_{min} is the minimum probability of choosing an instance per exploration strategy, and N_ℓ is the total number of labeled instances at iteration ℓ.

3.2. Datasets

We use 2 public **image-based** datasets, CIFAR-10 (Krizhevsky et al., 2009) and SVHN (Netzer et al., 2011). CIFAR-10 has 60k images uniformly across 10 classes. The training subset of the dataset has 50k images, and the remaining 10k images form the test set. SVHN has images spread across 10 classes, with 73257 images for training, and 26032 images for testing. To evaluate TAAL, we induce class imbalance and form a long-tailed distribution over class sizes (see D_c in Table 2). For both datasets, we carve out a 20% class-stratified random subset from the imbalanced training set to create a held-out validation set. We also use 2 proprietary fully-labeled **text-based** product classification datasets *HS* and *FEE* derived from 2 different e-commerce product taxonomies. HS has 9 classes-of-interest and 1.15M products, while FEE has 7 classes-of-interest and 740k products. Additionally, both datasets have one special class called 'not-in-k' representing instances in the dataset that do not belong to any classes-of-interest. Both datasets have an intrinsic long-tailed distribution in terms of class sizes (see D_c in Table 2).

3.3. Active Learning Classifier

We use a common training procedure and hyperparameters to ensure that all query strategies are given fair treatment across all experiments. For image classification datasets, we follow the setup described in (Kothawade et al., 2021). We use ResNet-18 (He et al., 2016) as our classifier $\mathcal{M}$ and train it using a stochastic gradient descent optimizer with an initial learning rate of 0.01 and momentum of 0.9. During inference, we take the class corresponding to the highest score from the final layer of the trained model as the prediction for an instance. For product classification datasets, we choose fastText (Joulin et al., 2016) as our active learning classifier $\mathcal{M}$ since it provides an attractive trade-off between accuracy and latency. We tune thresholds for each class to achieve the desired precision target of 90% using *out-of-fold* prediction scores in a one vs. rest setting (number of folds is 10). During inference, we treat an instance as member of a class if its output class probability is higher than its corresponding threshold.

3.4. Accuracy Estimation During Active Learning

For **image datasets**, we compute class priority weights using class-level F_1 score as the accuracy metric in Eq. (2), with the per-class F_1 score target being 90%. At every active learning iteration, we estimate class-level F_1 score on the held-out validation set which is an unbiased sample from the entire dataset. For **product classification datasets**, we compute class priority weights using class-level 'recall at precision' (R@P) as the accuracy metric in Eq. (2), with the per-class R@P target being 90R@90P. At every active learning iteration, we estimate class-level R@P using out-of-fold prediction scores over *exploration*-sourced labeled data from the E3G query strategy. This provides an unbiased performance estimate since it avoids the sampling bias induced by the *exploitation* strategy. We also set the priority weight for the 'not-in-k' class to 0 for all active learning iterations because our use case necessitates improving accuracy over the k classes-of-interest only.

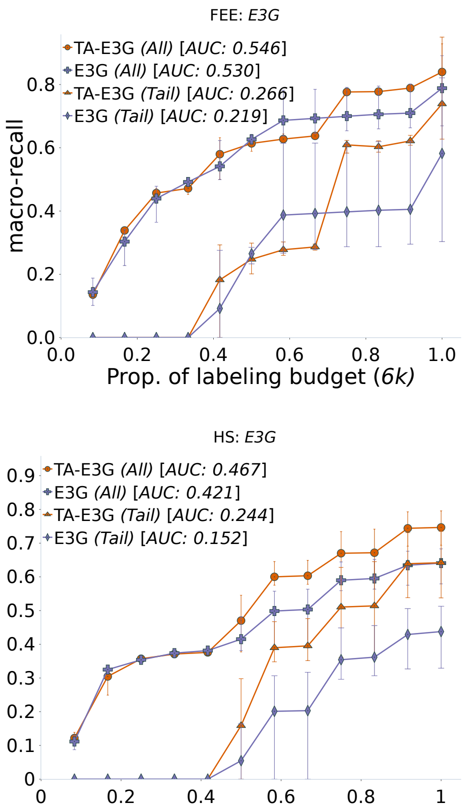


Figure 3: Progression of *macro-recall* across all classes and tail classes for **E3G** and its target-aware variant **TA-E3G** on HS and FEE datasets. **Axes' labels are identical for both plots**. Metrics are averaged across 3 experiments, with error bars showing min and max values. Legends report area under each macro-recall curve (AUC) summarizing macro-recall across label counts. TA-E3G has greater AUC for all-class macro-recall and tail-class macro-recall than E3G.

3.5. Metrics for Comparison of Query Strategies

For experiments on **image datasets**, we report class-level F_1 on a held-out test set to replicate the setup in (Kothawade et al., 2021) to compare each query strategy against its target-aware variant. For experiments on **product classification datasets**, we report class-level recall computed on the full dataset to mimic our internal product classification setup where customers want to achieve target accuracy for each class-of-interest on the entire domain. We summarize class-level metrics by reporting *macro*-recall and *macro*-F_1 score. We choose *macro* over *micro* averaging because accuracy of each class is equally important for our use-case.

4. Results

Figs. 3 and 4 show the progression of macro-F_1 and macro-recall with increasing number of training examples over product classification and image classification datasets, respectively. Plot legends report Area Under the Curve (AUC) for each query strategy, summarizing its accuracy metric values across all label counts (*higher the better*). Averaged across all 10 *dataset + query strategy* combinations, TAAL increases AUC by 1.6 percentage points (from 63.3% to 64.9%). For 8 out of the 10 *dataset + query strategy* combinations, we see an increase in AUC with TAAL, indicating that TAAL performs better than the baseline strategies measured across all label counts. For SVHN dataset, the target-aware variants of BADGE and ENTROPY have the same AUC as their baseline strategies indicating that TAAL offers no improvement. We hypothesize that this is because SVHN is easy to classify as we get high (> 0.8) macro-F_1 scores across tail classes with baseline query strategies.
Impact on *tail* classes: *Tail* classes are the smallest classes-of-interest comprising $\leq$ 5% of the full dataset. Achieving business targets on tail classes is important because they represent rare, but strategically important classes-of-interest. Averaged across all 10 *dataset + query strategy* combinations, TAAL increases tail-class AUC by 4.2 percentage points (from 45.4% to 49.6%), explained by a corresponding increase in the proportion of tail class labels by 11.3 percentage points (from 14.2% to 25.5%) at 6k labels. Tail classes have a lower likelihood of getting labeled compared to *head* classes. With fewer training instances, the classifier performs poorly over them. This is seen in Figs. 3 and 4, where tail-class AUC is lower than all-class AUC. As labeling progresses, all baseline strategies except SIMILAR continue to focus on all classes equally. In contrast, TAAL assigns greater importance to tail classes through weights generated by *Class Priority Weight Computer* (Sec. 2.1). As depicted in Fig. 2, TAAL then prioritizes examples for labeling deemed optimal per the underlying query strategy, but at the same time relatively more likely to benefit tail classes. This is reflected in Table 2 where the class distribution of training instances L_c is less skewed for TAAL compared to the corresponding baseline strategies. *E.g.*, at 6k labels, HS-4 (a tail class) constitutes 6.2% of total labels with TA-E3G vs. 1.8% with E3G, an increase by a factor of 3.4.
Impact on *head* classes: *Head* classes are the

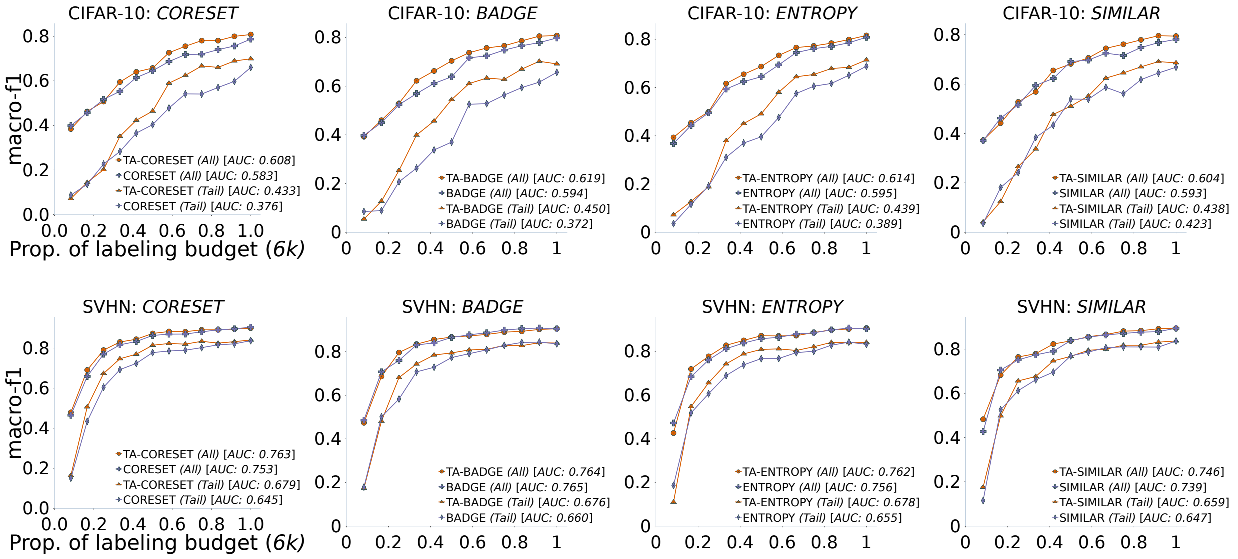


Figure 4: Progression of *macro*-F_1 across all classes and tail classes for 4 query strategies and their target-aware variants (TA-*) on CIFAR-10 test set (top) and SVHN test set (bottom). **Axes' labels are identical for all plots**. Legends report area under each macro-F_1 curve (AUC) summarizing macro-F_1 across all label counts. Each query strategy's target-aware variant provides greater or similar AUC for all-class macro-F_1 and greater AUC for tail-class macro-F_1.

Table 1: Area under the accuracy metric curve (AUC) for all *dataset* + *query strategy* combinations at 6k labels. The accuracy metric summarized by AUC is F_1 score for image datasets and recall for product classification datasets. We report AUC for each query strategy ('Baseline' columns) against its target-aware variant ('TAAL' columns) over all classes, head classes, and tail classes.

Dataset	Query Strategy	Baseline AUC (%)			TAAL AUC (%)		
		All	Head	Tail	All	Head	Tail
SVHN	CORESET	75.25	82.43	64.48	76.30	81.94	67.86
	SIMILAR	73.87	79.99	64.68	74.61	80.40	65.92
	ENTROPY	75.60	82.34	65.49	76.16	81.76	67.78
	BADGE	76.48	83.49	65.96	76.39	82.27	67.57
CIFAR-10	CORESET	58.35	72.16	37.63	60.84	72.50	43.34
	SIMILAR	59.31	70.62	42.34	60.38	71.40	43.84
	ENTROPY	59.54	73.29	38.91	61.41	73.07	43.93
	BADGE	59.36	74.13	37.22	61.91	73.19	44.98
FEE	E3G	53.03	76.35	21.95	54.64	75.63	26.65
HS	E3G	42.06	75.64	15.19	46.69	74.54	24.40

largest classes-of-interest comprising $\geq$ 95% of the full dataset. Averaged over all 10 *dataset* + *query strategy* combinations, TAAL reduces the proportion of head class labels by 11.3 percentage points (from 84.5% to 73.2%, Table 2) at 6k labels, while only reducing head-class AUC by 0.4 percentage points (from 77.0% to 76.7%, Table 1).

Comparison of SIMILAR and TA-SIMILAR: Experiments show that TA-SIMILAR gives only a minor improvement over SIMILAR ($+0.7$ percentage points all-class AUC, $+1.2$ percentage points tail-class AUC). This is expected because SIMILAR is designed to focus on tail classes, and the same tail classes also get prioritized by TAAL since they lag in class-level accuracy during active learning. SIMILAR is different from TAAL in that (a) SIMILAR needs specification of the explicit set of rare classes in the dataset (unlike TAAL), which is not always known *a priori* and, (b) SIMILAR, like any other non-TAAL query strategy, will continue to improve the accuracy of a class even if it has already met its target, thus consuming unnecessary manual annotations.

5. RELATED WORK

We refer readers to survey papers (Settles, 2009; Ren et al., 2020) for a comprehensive overview of

Table 2: Comparison of relative size of each class in the full dataset (D_c) and in labeled data (L_c) at 6k labels. For L_c, we compare each baseline active learning strategy against its target-aware variant ('TA-'). Asterisk indicates tail classes.

Text-based product taxonomy dataset **FEE** (Data represented in %)										
Class	F-0	F-2	F-3	F-1	F-5*	F-4*	F-6*	F-not-in-k	-	-
D_c	50.6	30.8	10.7	6.4	0.6	0.5	0.3	0.12	-	-
E3G L_c	34.3	31.1	13.3	10.6	3.6	2.3	2.2	2.5	-	-
TA-E3G L_c	28.5	21.5	7.7	4.9	9.3	8.8	17.8	1.4	-	-
Text-based product taxonomy dataset **HS** (Data represented in %)										
Class	HS-2	HS-3	HS-8	HS-7	HS-6*	HS-0*	HS-5*	HS-1*	HS-4*	HS-not-in-k
D_c	41.9	25.8	14.7	11.1	2.4	0.5	0.4	0.4	0.3	2.6
E3G L_c	34.1	16.4	13.4	11.5	3.3	1.8	4.5	2.4	1.8	10.8
TA-E3G L_c	32.9	14.9	10.2	7.9	2.8	2.6	5.7	5.0	6.2	11.9
Image-based dataset **SVHN** (Data represented in %)										
Class	S-1	S-3	S-5	S-6	S-7	S-0	S-2*	S-4*	S-8*	S-9*
D_c	28.9	17.7	14.4	12.0	11.7	10.3	1.3	1.3	1.3	1.3
SIMILAR L_c	16.6	17.1	7.1	10.4	12.5	6.3	7.6	7.6	7.4	7.6
TA-SIMILAR L_c	13.8	17.2	10.4	12.3	11.5	4.7	7.6	7.6	7.4	7.6
ENTROPY L_c	19.4	20.6	13.2	14.0	11.1	8.7	3.2	3.5	3.1	3.3
TA-ENTROPY L_c	14.0	17.1	11.3	11.3	8.3	7.0	7.8	7.7	7.7	7.9
BADGE L_c	19.7	19.1	14.5	13.9	11.1	9.0	3.1	3.3	3.3	3.2
TA-BADGE L_c	15.0	16.9	11.1	11.2	8.2	6.9	7.7	7.7	7.6	7.8
CORESET L_c	22.9	18.1	12.7	12.4	10.9	9.4	3.5	4.0	2.6	3.5
TA-CORESET L_c	17.2	15.7	9.0	10.8	8.8	7.8	7.7	7.7	7.5	7.8
Image-based dataset **CIFAR-10** (Data represented in %)										
Class	C-0	C-1	C-3	C-5	C-6	C-7	C-2	C-4*	C-8*	C-9*
D_c	15.8	15.8	15.8	15.8	15.8	15.8	1.3	1.3	1.3	1.3
SIMILAR L_c	16.2	17.2	10.9	10.4	10.2	18.0	3.7	4.2	4.8	4.6
TA-SIMILAR L_c	13.3	8.7	20.4	18.4	9.6	11.0	4.5	4.6	4.9	4.7
ENTROPY L_c	13.1	9.2	21.5	17.2	12.8	14.6	3.4	3.4	2.3	2.5
TA-ENTROPY L_c	13.8	8.2	20.2	15.7	10.9	12.5	4.5	4.5	4.9	4.7
BADGE L_c	14.2	9.3	21.2	18.3	13.0	13.5	2.9	2.7	2.6	2.5
TA-BADGE L_c	13.3	7.9	20.6	16.6	10.5	12.6	4.5	4.5	4.9	4.7
CORESET L_c	19.8	16.8	15.2	12.6	11.2	13.6	2.3	2.5	3.2	2.9
TA-CORESET L_c	13.5	8.8	20.2	16.4	10.1	12.9	4.3	4.4	4.8	4.7

active learning. In this section, we discuss the core ideas we have borrowed from prior-art in Active Learning.

Active learning query strategies determine what unlabeled data should be annotated next. The most popular query strategies are uncertainty-based (Settles, 2009; Holub et al., 2008; Beluch et al., 2018; Wu et al., 2020; Lewis and Gale, 1994), diversity-based (Bilgic and Getoor, 2009; Guo, 2010; Dasgupta and Hsu, 2008; Sener and Savarese, 2017; Jiang and Qing-Yu, 2015), and expected-model-change based (Freytag et al., 2014; Roy and Mccallum, 2001). Sec. 3.1 describes the specific query strategies we used in experiments.

We use batch-mode active learning in this paper. Previous research (Lewis and Gale, 1994; Settles, 2009; Gal et al., 2017; Lin and Parikh, 2017) has demonstrated that choosing a batch of instances with small batch sizes offers a good trade-off between user interactivity and number of labels required to create a classifier.

Hybrid query strategies (Ash et al., 2019b; Shui et al., 2019) deal with controlling the classical *exploration vs. exploitation* trade-off. Exploitation aims to find data with highest uncertainty that is likely to help the model learn fast (Bloodgood and Vijay-Shanker, 2009), and exploration aims to find data that is representative of the unlabeled data. One of our baseline strategies E3G (Sec. 3.1)) falls under this umbrella of hybrid approaches.

Stopping the active learning loop involves defining exit criteria that help annotators decide when to stop labeling. The most common stopping method is to use a predefined criterion (Budd et al., 2019; Liu et al., 2016; Schröder and Niekler, 2020) such as maximum number of active learning iterations, maximum annotation budget/time, or minimum classification accuracy to be achieved. Recent papers on stopping criteria have argued for checking if predictions on unlabeled data have stabilized (Bloodgood and Vijay-Shanker, 2014; Vlachos, 2008; Budd et al., 2019; Zhu et al., 2010). We use maximum labeling budget as the sole stopping criterion in this paper for a fair comparison of baseline query strategies and their target-aware variants.

6. Conclusions and Future Work

We introduce a framework called Target-Aware Active Learning (TAAL) that converts any arbitrary active learning query strategy into its target-aware variant by leveraging the gap between each class' current estimated accuracy and its corresponding business target. We perform extensive experiments comparing 5 baseline query strategies and their target-aware variants on 2 image classification and 2 text-based product classification datasets. Our results indicate that TAAL improves the likelihood of achieving business targets on 8 out of 10 *dataset* + *query strategy* combinations, and matches the baseline performance of BADGE and ENTROPY strategies on the SVHN dataset. In addition, the run-time complexity of TAAL scales linearly with the size of the input query, making it practical for large-scale active learning. In the future, we plan to test TAAL with classifiers like PECOS (Yu et al., 2022) which further improve the accuracy of tail classes, especially for high-cardinality classification problems.

7. Bibliographical References

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. 2019a. Deep batch active learning by diverse, uncertain gradient lower bounds. *CoRR*, abs/1906.03671.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. 2019b. Deep batch active learning by diverse, uncertain gradient lower bounds. *CoRR*, abs/1906.03671.

William H. Beluch, Tim Genewein, Andreas Nürnberger, and Jan M. Köhler. 2018. The power of ensembles for active learning in image classification. In *CVPR*, pages 9368–9377. IEEE Computer Society.

Mustafa Bilgic and Lise Getoor. 2009. Link-based active learning. In *NIPS Workshop on Analyzing Networks and Learning with Graphs*.

Michael Bloodgood and K. Vijay-Shanker. 2009. Taking into account the differences between actively and passively acquired data: The case of active learning with support vector machines for imbalanced datasets. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, May 31 - June 5, 2009, Boulder, Colorado, USA, Short Papers*, pages 137–140. The Association for Computational Linguistics.

Michael Bloodgood and K. Vijay-Shanker. 2014. A method for stopping active learning based on stabilizing predictions and the need for user-adjustable stopping. *CoRR*, abs/1409.5165.

Samuel Budd, Emma C. Robinson, and Bernhard Kainz. 2019. A survey on active learning and human-in-the-loop deep learning for medical image analysis. *CoRR*, abs/1910.02923.

Gui Citovsky, Giulia DeSalvo, Claudio Gentile, Lazaros Karydas, Anand Rajagopalan, Afshin Rostamizadeh, and Sanjiv Kumar. 2021. Batch active learning at scale.

Sanjoy Dasgupta and Daniel Hsu. 2008. Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on Machine learning*, pages 208–215.

Alexander Freytag, Erik Rodner, and Joachim Denzler. 2014. Selecting influential examples: Active learning with expected model output changes. In *European conference on computer vision*, pages 562–577. Springer.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR.

Yuhong Guo. 2010. Active instance sampling via matrix partition. In *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Alex Holub, Pietro Perona, and Michael C Burl. 2008. Entropy-based active learning for object recognition. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8. IEEE.

S Jiang and OU Qing-Yu. 2015. Batch-mode active learning approach of computer viruses classifier based on information density. *Naval Univ. Eng*, 431(5).

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Suraj Kothawade, Nathan Beck, Krishnateja Killamsetty, and Rishabh Iyer. 2021. Similar: Submodular information measures based active learning in realistic scenarios.

Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images.

David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer.

Xiao Lin and Devi Parikh. 2017. Active learning for visual question answering: An empirical study. *arXiv preprint arXiv:1711.01732*.

Peng Liu, Hui Zhang, and Kie B. Eom. 2016. Active deep learning for classification of hyperspectral images. *CoRR*, abs/1611.10031.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. 2020. A survey of deep active learning. *CoRR*, abs/2009.00236.

Nicholas Roy and Andrew Mccallum. 2001. Toward optimal active learning through monte carlo estimation of error reduction. In *in Proceedings of the International Conference on Machine Learning*.

Christopher Schröder and Andreas Niekler. 2020. A survey of active learning for text classification using deep neural networks. *CoRR*, abs/2008.07267.

Ozan Sener and Silvio Savarese. 2017. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*.

Burr Settles. 2009. Active learning literature survey.

Changjian Shui, Fan Zhou, Christian Gagné, and Boyu Wang. 2019. Deep active learning: Unified and principled method for query and training. *CoRR*, abs/1911.09162.

Aditya Siddhant and Zachary C Lipton. 2018. Deep bayesian active learning for natural language processing: Results of a large-scale empirical study. *arXiv preprint arXiv:1808.05697*.

Aleksandrs Slivkins. 2019. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286.

Andreas Vlachos. 2008. A stopping criterion for active learning. 22(3):295–312.

Jian Wu, Victor S Sheng, Jing Zhang, Hua Li, Tetiana Dadakova, Christine Leon Swisher, Zhiming Cui, and Pengpeng Zhao. 2020. Multi-label active learning algorithms for image classification: Overview and future promise. *ACM Computing Surveys (CSUR)*, 53(2):1–35.

Hsiang-Fu Yu, Kai Zhong, Jiong Zhang, Wei-Cheng Chang, and Inderjit S. Dhillon. 2022. Pecos: Prediction for enormous and correlated output spaces.

Jingbo Zhu, Huizhen Wang, Eduard H. Hovy, and Matthew Y. Ma. 2010. Confidence-based stopping criteria for active learning for data annotation. *ACM Trans. Speech Lang. Process.*, 6:3:1–3:24.

Cluster Language Model for Improved E-Commerce Retrieval and Ranking: Leveraging Query Similarity and Fine-Tuning for Personalized Results

Duleep Rathgamage Don, Ying Xie, Le Yu, Simon Hughes, Yun Zhu
Kennesaw State University, USA, The Home Depot, USA
drathgam@students.kennesaw.edu, yxie2@kennesaw.edu, {le_yu1, simon_hughes, yun_zhu}@homedepot.com

Abstract

This paper proposes a novel method to improve the accuracy of product search in e-commerce by utilizing a cluster language model. The method aims to address the limitations of the bi-encoder architecture while maintaining a minimal additional training burden. The approach involves labeling top products for each query, generating semantically similar query clusters using the K-Means clustering algorithm, and fine-tuning a global language model into cluster language models on individual clusters. The parameters of each cluster language model are fine-tuned to learn local manifolds in the feature space efficiently, capturing the nuances of various query types within each cluster. The inference is performed by assigning a new query to its respective cluster and utilizing the corresponding cluster language model for retrieval. The proposed method results in more accurate and personalized retrieval results, offering a superior alternative to the popular bi-encoder based retrieval models in semantic search.

Keywords: Large Language Models, Transformers, Natural Language Processing

1. Introduction

E-commerce platforms have experienced tremendous growth in recent years, with millions of users browsing and purchasing products online every day. One of the critical factors that contribute to a successful e-commerce platform is the ability to effectively retrieve and rank products based on user queries. A robust retrieval and ranking system should be able to understand the underlying semantics of queries and provide personalized results that meet the specific needs of individual users.

Traditional retrieval models, such as the Vector Space Model and Latent Semantic Analysis, have been effective in capturing keyword-based relevance between queries and items. However, they often struggle with understanding the nuances of natural language and user intent. Recent advancements in natural language processing and deep learning have led to the development of powerful pre-trained language models, such as BERT, GPT, and RoBERTa. These models have demonstrated impressive performance in various tasks, including e-commerce retrieval and ranking, by capturing the semantic relationships between queries and items.

Despite the successes of pre-trained language models, there is still room for improvement, particularly in understanding the diverse nature of user queries and providing tailored retrieval results. One promising direction is to incorporate query clustering into the retrieval and ranking process, leveraging the inherent structure of the query space to better adapt the model to different query types. Query clustering can help uncover underlying patterns in user search behavior and create more fine-grained representations of user intent, ultimately leading to improved retrieval performance.

In this paper, we propose a cluster-based language model for e-commerce retrieval and ranking that builds upon the strengths of pre-trained language models and query clustering. Our method (Figure 2) first fine-tunes a pre-trained language model, on query-product pairs using a bi-encoder approach, forming a baseline model (Figure 1). We then cluster the training queries into k clusters and refine the baseline model for each query cluster using a novel labeling and refinement strategy.

The key idea behind our approach is that the baseline model, while effective in capturing general semantic relationships, may not be sensitive to the specific characteristics of different query clusters. By refining the model for each cluster, we can better capture the nuances of various query types and provide more accurate and personalized retrieval results.

In summary, our cluster-based language model for e-commerce retrieval and ranking leverages the power of pre-trained language models and query clustering to deliver more accurate and personalized product retrieval results. By adapting the model to different query types, we can address the diverse needs of users in large-scale e-commerce environments and improve overall platform performance.

2. Related Work

The field of e-commerce retrieval and ranking has seen significant advancements over the years, with

The Seventh Workshop on e-Commerce and NLP (ECNLP 7), pages 145–153
21 May, 2024.

various techniques being proposed and developed. Our proposed cluster language model builds upon the successes of these existing techniques and introduces a novel approach to improve e-commerce retrieval performance. The most notable related work includes the following.

Learning to Rank: Learning to Rank (LTR) models are supervised machine learning techniques designed to optimize the ranking of items based on relevance. These approaches include pointwise, pairwise, and listwise ranking methods. Our proposed method differs from traditional LTR models by utilizing pre-trained language models and clustering queries to better capture the semantic relationships between queries and items (Burges, 2010; Freund et al., 2003).

Vector Space Models: Traditional information retrieval models, such as the Vector Space Model (VSM), utilize techniques like TF-IDF and cosine similarity to rank documents based on their relevance to a given query. Our approach enhances this concept by leveraging pre-trained language models and query clustering to better represent the semantic space and improve ranking performance (Salton et al., 1975).

Latent Semantic Analysis: Latent Semantic Analysis (LSA) captures the semantic relationships between queries and items for improved retrieval and ranking. Our method extends this idea by incorporating pre-trained language models and query clustering to further refine the semantic understanding of queries and items, leading to more accurate retrieval results (Deerwester et al., 1990).

Neural IR Models: Deep learning-based models, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have been applied to information retrieval tasks for text and image-based representations of items. Our proposed model takes advantage of the powerful representation capabilities of pre-trained language models and query clustering to improve the retrieval and ranking performance for e-commerce (Huang et al., 2013; Palangi et al., 2015).

Pre-trained Language Models: The use of pre-trained language models, such as BERT, GPT, and RoBERTa, has gained popularity in recent years for various natural language processing tasks, including e-commerce retrieval and ranking. Our approach differs from existing pre-trained language model applications by introducing query clustering and model refinement for each query cluster, which enhances the model's ability to capture the nuances of different query types and provide more personalized retrieval results (Devlin et al., 2018; Radford et al., 2018; Liu et al., 2019).

By combining the strengths of pre-trained language models, query clustering, and model refinement, our proposed cluster language model addresses the challenges of delivering accurate and personalized product retrieval results in large-scale e-commerce environments.

3. Methodology

In this section, we describe the technical details of our baseline model which is one of the top in-house models in terms of recall, NDCG (Wang et al., 2013), and execution time. Also, we present the proposed method that leverages the retrieval of the fine-tuned baseline model.

3.1 Sentence Transformer Architecture

The baseline model is essentially a sentence transformer, (Reimers & Gurevych, 2019) that is based on a bi-encoder architecture that contains two DistilBERT models (SanhSanh et al., 2019). These models are identical and share the same weights. The DistilBERT is a transformer-based model with 6 layers of self-attention and feed-forward neural networks. Each attention layer represent different aspects of the input in different ways. In the context of sentence embeddings, the DistilBERT takes in a sentence as input and generates a (1, 768) size vector representation of that sentence. This vector representation is the sentence embedding used to measure the semantic similarity between two sentences or to classify a sentence into one of several categories.

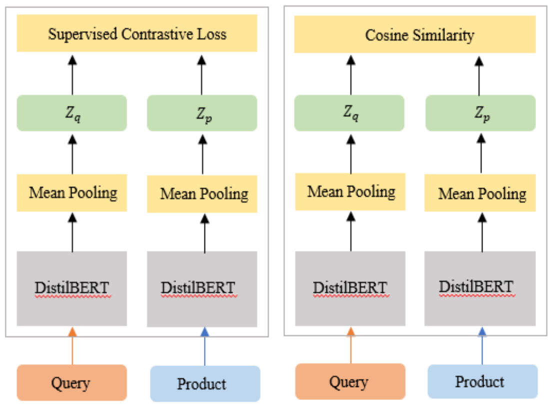


Figure 1: (Left) The architecture of the baseline model for training. (Right) The architecture of the baseline model for inference.

Figure 1 presents our baseline model. It processes query (*q*) and product (*p*) sentences as pairs during training and testing. The baseline

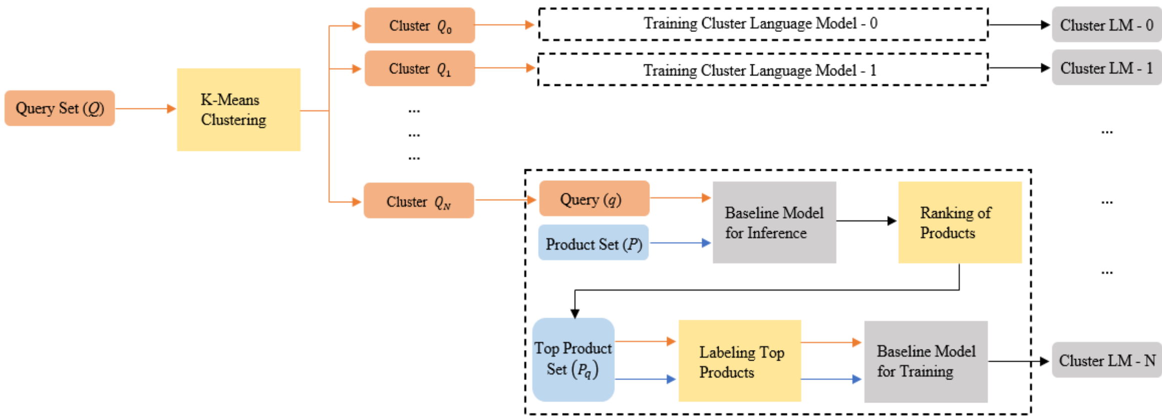


Figure 2:The architecture of the proposed Cluster Language Model for training.

model adds a mean pooling operation to the output of the [CLS] token of the DistilBERT. In training, the embeddings of *q* and *p* denoted by Z_q and Z_p respectively, are used to optimize the supervised contrastive loss (Hadsell et al., 2006). In inference, the cosine similarity between Z_q and Z_p for different *p* is computed and based on that value, each *p* can be ranked with respect to *q*.

3.1.1 Pretraining the Baseline Model

We selected a pretrained sentence transformer model: MS MARCO-DistilBERT-Base-v2 from Hugging Face (*Hugging Face Transformers Library*, 2021). This model is built on a variant of the DistilBERT model, which was pre-trained using a large corpus of text data including the MS MARCO dataset (Nguyen et al., 2016; Wolf et al., 2020). The pre-training was done using masked language modeling (MLM) and next-sentence prediction (NSP) tasks. The MS MARCO-DistilBERT-Base-v2 model was further fine-tuned on the MS MARCO Passage Ranking task (*Hugging Face Transformers Library*, 2021), which is a large-scale.

3.1.2 Fine Tuning the Baseline Model

The baseline sentence transformer model is fine-tuned on query and product data in an e-commerce application. This task is performed by using a contrastive learning strategy (Hadsell et al., 2006). Let the training query set and the product set be *Q* and *P* respectively. For $q \in Q$, and $p, n \in P$, the input (*q*, *p*) is labeled with 1, and the input (*q*, *n*) is labeled with 0 considering that *p* is a positive sentence and *n* is a negative sentence (Hadsell et al., 2006). The goal of contrastive learning is to find parameters *W* of a family of functions *G*, to map a collection of high-dimensional inputs onto a low-dimensional manifold. For $x = \{p, n\}$, this mapping is such that the Euclidean distance between points on the manifold, given by: $D_w(Z_q, Z_x) = \|G_w(Z_q) - G_w(Z_x)\|^2$ closely approximates the semantic similarity of the inputs in the input space by minimizing the following objective function (Hadsell et al., 2006).

$$L = (1 - Y)\frac{1}{2}D_W^2 + Y\frac{1}{2}\{max(0, m - D_W)\}^2 \qquad (1)$$

Where, $Y = \{0, 1\}$, and $m > 0$ is a margin. The fine-tuned baseline model can be used for inference. For our e-commerce use case, the retrieval set of interest is limited to the first 100 products since a typical customer is less likely to explore the search result beyond this limit. This set is known as the *Top Product Set* for the given query *q* and is denoted by P_q. Even though the baseline method can search for products with a competitive recall@24 on unseen queries, its performance at smaller retrieval sets is observed to be relatively weak.

3.2 The Proposed Method

Although the bi-encoder architecture is considered fast, its accuracy is often compromised. This is an inherent drawback of the baseline model. The intention of the proposed method is to come up with a solution and enhance the product search up to @24 with a minimum of additional training. Thus, it provides an alternative approach to the popular bi-encoder and cross-encoder combination (Rosa et al., 2022; Ortiz-Barajas et al., 2022) in semantic search.

3.2.1 Labeling Top Products for Each Query

The rationale behind this step is the following observation: even though our baseline model is effective in capturing general semantic relationships, it may not be sensitive to the specific characteristics of different query clusters. To mitigate this problem, we introduce a novel labeling strategy for the elements of the Top Product Set and produce a new training dataset per query. This dataset will be used to train the corresponding cluster language model, depending on the query cluster where the training query is located. For a given query q, the *Top Product Set* can be presented as $P_q = \{p_1, p_2, \ldots, p_{99}\}$. In an ideal retrieval, the order of the elements of the Top Product Set is such that all the purchased products should appear as added-to-cart products.

However, in real retrievals, some of the purchased products are often seen to be forced back by unpurchased (impressed or added-to-cart) products due to the extreme similarities between them. To create the new training dataset for the corresponding cluster language model, we pair each query q with some elements of its *Top Product Set* P_q and label them using the following rule.

Let the query q be an arbitrary query; power wash cleaner, and identify the last purchased product for q in its P_q. Let this last purchased product be p_k. Now, for all $i < k$, we label query-product pairs (q, p_i) with 0 if p_i is not a purchased product. Also, we label all query-product pairs (q, p_i) with 1 if p_i is a purchased product, as shown in Table 1. The above relabeling helps the baseline model specifically suppress the unpurchased products that are more similar to the purchased products.

According to Table 1, a given product is identified by using its PRODID which is a unique identifier. The attribute 'Type' is used only to indicate whether the product is purchased (P) or unpurchased (U). The unlabeled query-product pairs are discarded.

3.2.2 Labe Labeling Top Products for Each Query

The training of the proposed cluster language model comprises two phases as shown in Figure 2. In the first phase, we split the queries into N clusters using K-Means clustering algorithm. For this task, the K-Means algorithm is trained on the embedding Z_q for all $q \in Q$ by using the baseline model for inference (without computing cosine similarity).

Rank	Query	Product	Type	Label
0	power wash cleaner	P42710	P	1
1	power wash cleaner	P43322	U	0
2	power wash cleaner	P51270	U	0
…	…	…		…
k	power wash cleaner	P52993	P	1
…	…	…		…
99	power wash cleaner	P58671	U	na

Table 1: The proposed training dataset of the cluster language model for a selected query.

This process generates clusters of queries that are more semantically similar. Let the j th cluster of Q be denoted by Q_j, where j is known as the cluster ID. Then in each cluster, we store only the actual queries. The value of N should not be either very small or very large and is determined by investigating the performance of clustering. In the ideal case, N results in query clusters in which the within-cluster variance is much less compared to the between-cluster variance.

3.2.3 Training Cluster Language Model on Individual Clusters

The second phase of training the proposed cluster language model is implemented recursively on each cluster. This process is presented by the workflow illustrated for the query cluster Q_N in Figure 2, where for each query q, we first rank the product embeddings with respect to the query embedding and obtain the Top Product Set P_q for using the baseline model for inference. Then we label the top products as discussed to generate a new training dataset for each query as shown in Table 1. For all the queries in the given cluster, we fine tune the baseline model using the above mini datasets and the optimization presented in Chapter 3.1 to generate the Cluster LM – N. After the second phase is completed for all the clusters, the collection of Cluster LM – k, where $0 \leq k \leq N$ known as the Cluster Language Model is stored in a model registry.

Since the baseline model parameters have been generally tuned for all queries and products, the proposed cluster-based fine tuning helps the model parameters learn local manifolds in the feature space efficiently. Therefore, the clustered queries help better capture the nuances of various query types within that cluster. As a result, the relevance of the retrieved products will be improved.

3.2.4 Inference Using Cluster Language Model

In inference, a new query is assigned to the respective cluster by the trained K-Means algorithm. Thus, receiving the cluster ID for the new query r, we can select the corresponding Cluster LM from the model registry for inference. First, the new query r and the product set P are used to generate the *Top Product Set* P_r as shown in the inference pipeline in Figure 3. Then we input P_r (completely) to the inference architecture of the selected Cluster LM to generate a *Refined Top Product Set* $P_{r'}$.

In general, the *Refined Top Product Set* is a better retrieval than the *Top Product Set* for new queries of each cluster, providing more accurate and personalized retrieval results.

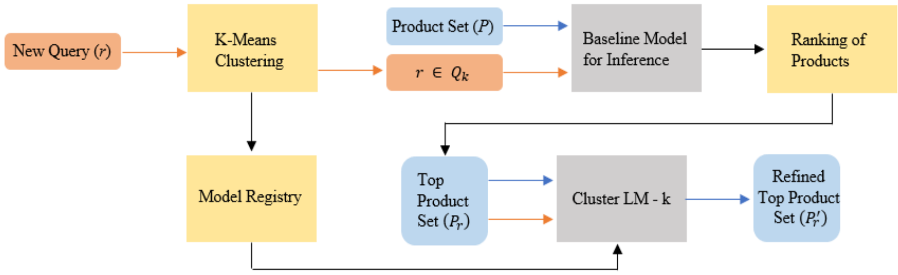


Figure 3: The inference pipeline of the proposed Cluster Language Model.

4. Experiments

In this section, we introduce our datasets, data preprocessing, evaluation process, and present results to demonstrate the impact of the proposed method in production.

4.1 Datasets

We used the same product catalog for training and validation of both the baseline and the proposed methods. This dataset consists of approximately 1M unique products. Each product contains only the following attributes.

- OMSID: the unique product identification number
- Title: the product name
- Brand: the product brand
- ColorFinish: the overall color of the product
- Leaf: the category of the first level of the hierarchical taxonomy

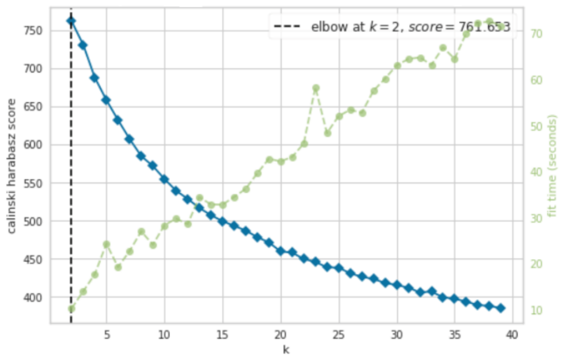


Figure 4: Calinski Harabasz Score Elbow for K-Means Clustering.

An example of a product sentence is: *P52993 electric pressure washer sun joe green pressure washer.* For the training and validation of the baseline and the proposed methods, we used the same query sets. The queries are made of refined customer search phrases that are free of typos. For the training and validation, we used about 15.4M unique queries and 24K unique queries respectively.

4.2 Experiment Settings and Preprocessing

All experiments and preprocessing are conducted on the Google Cloud Platform using NVIDIA A100 GPU using Python 3 and PyTorch 1.11. Major libraries used: SentenceTrandformer (Thakur et al., 2020) from Hugging Face and MiniBatchKMeans from scikit-learn (Sculley, 2010). The training dataset consists of roughly 60M query-product pairs. Each query is paired with a relevant purchased product, impressed, or added-to-cart product. The max token length is 40. The training batch size is 256. The number of epochs used for the baseline model and the Cluster Language Model: 15 and 5 respectively.

To train the proposed method, the training queries are clustered by using K-Means clustering. According to our initial experiments based on the training query set, we observed that the Cluster LM – k for $0 \leq k \leq N$, consistently fails to outperform the baseline model on the respective validation queries if the size of Cluster Q_k is about 1M or more. Although the query clusters of that scale are hard to describe based on their contents, this observation was notable despite different cluster members. Thus, we can safely assume that the size of any query cluster should be much less than 1M.

For our experiments, we satisfy the above condition by setting N = 29 resulting in 30 clusters with a mean cluster size of 513K with a standard deviation of 180K.

However, a large number of clusters such as N = 100 could lead to memory-related issues when deploying the proposed model in the cloud. Also, it could lower the overall performance of the cluster language model as many smaller clusters may contain quite similar queries. This reduces the probability that a random query is assigned to the correct Cluster LM – k for $0 \leq k \leq N$ during the inference. Figure 4 shows the Calinski Harabasz

(Caliński & Harabasz, 1974) Score Elbow for K-Means Clustering and according to which the elbow occurs when the number of clusters is 2 (k = 3). Thus, a larger number of clusters greater would produce weaker clusters of queries.

4.3 Evaluation Process

Our evaluation process is conducted to measure the following two tasks.

Matching: the intention of this task is to retrieve all the relevant purchased products for a given query. We use recall to measure matching at different retrieval thresholds such as 1, 2, 4, 8, 12, 24, and 100.

$$Recall@k = \frac{\#\ of\ purchased\ products\ @k}{\min\ (k, Total\ \#\ of\ purchased\ products)} \quad (2)$$

Ranking: the intention of this task is to order the retrieved products by their relevance. We use Normalized Discounted Cumulative Gain (NDCG) (Wang et al., 2013) to measure ranking at the retrieval thresholds mentioned above. We used 1 to denote a product if it is purchased and 0 to denote the product otherwise and applied the following definitions.

$$NDCG@k = \frac{DCG@k}{IDCG@k} \quad (3)$$

Where $DCG@k = \sum_{i=1}^{k} product_k / log_2(k+1)$, and IDCG@k is the Ideal Discounted Cumulative Gain at k.

4.4 Results

Our experiment were conducted to evaluate the effectiveness of the baseline model and the proposed Cluster Language Model in retrieving relevant purchased products from our product catalog. The overall performance of both models in matching and ranking is presented in Table 2. These results show that the Cluster Language Model has a significantly higher recall rate than that of the baseline model up to the retrieval

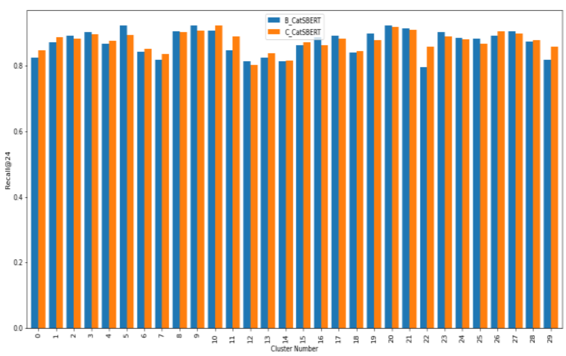


Figure 5: The cluster-level performance of the baseline model (blue) and the Cluster Language Model (orange) in matching, measured for Recall@24.

Retrieval Threshold	Recall		NDCG	
	Baseline Model	Cluster LM	Baseline Model	Cluster LM
@1	47.4	53.9	48.2	54.8
@2	52.6	56.7	51.6	56.7
@4	61.8	63.6	56.3	60.2
@8	73.1	73.2	61.1	64.1
@12	79.3	78.9	64.8	67.5
@24	87.9	87.9	66.2	69.0
@100	96.7	96.7	68.4	71.2

Table 2: The overall performance of baseline model and Cluster Language Model.

threshold@8. At higher thresholds, both models have quite similar matching performances. However, in ranking, the Cluster Language Model has a significantly higher recall rate than that of the baseline model up to the retrieval threshold@8. At higher thresholds, both models have quite similar matching performances. However, in ranking, the Cluster Language Model demonstrates a great improvement at every retrieval threshold.

The retrieval threshold@24 is considered to be a benchmark for our product search tasks. We investigated the cluster-level performance of the above two models and presented them in Figure 5. Also, we identified seven clusters in which the difference between the performance of the two models in Recall@24 is greater than 2%. Out of these cases, in four cases, the Cluster Language Model leads as shown in Table 3 and in the other three cases, the baseline model leads as shown in Table 4.

For each cluster (denoted by ID), Table 3 presents the percentage of training queries used in each cluster. It also provides the frequency (as a percentage) at which the purchased products of the testing dataset appear in the training dataset.

Generally, we can expect this measure to be higher for the clusters in which the proposed method performs well during testing. The rationale behind this idea is that the language models tend to bias toward the majority of data (Wolfe & Caliskan, 2021). The mean L2 distance shown in Table 3 measures how far a given cluster member is from the cluster center on average. Thus, the lower mean L2 distance implies a denser cluster. Table 4 presents a similar analysis for the clusters in which the baseline model outperforms the proposed method.

Figure 6 illustrates the relative L2 distance between cluster centers as a heatmap. According to this plot, the centers of clusters 3, 16, and 17 are located relatively further away from the rest of the cluster centers. Conversely, the centers of clusters

10, 15, and 19 are much close to the rest of the cluster centers. Having more distinct cluster centers helps the K-Means algorithm assign new queries to the respective clusters rather correctly.

ID	Training Data (%)	Occurrence of Testing Purchased Products in Training Data (%)	Mean L2 Distance	Recall@24	
				Baseline Model	Cluster LM
0	2.44	2.95	1.051	82.4	84.7
11	3.13	2.51	1.054	84.8	89.1
22	2.53	1.29	1.055	79.5	85.8
29	3.79	3.16	1.038	81.9	85.9

Table 3: The best matching for the Cluster Language Model at cluster-level.

ID	Training Data (%)	Occurrence of Testing Purchased Products in Training Data (%)	Mean L2 Distance	Recall@24	
				Baseline Model	Cluster LM
5	3.48	3.29	1.049	92.3	89.6
9	3.22	2.15	1.054	92.4	90.3
16	1.51	0.78	1.083	88.7	86.2

Table 4: The best matching for the baseline model at the cluster-level.

In addition, we measured and compared the execution time for both models. Table 5 shows the average time to process a single query by using these methods. The Cluster Language Model needs to complete all the processes shown in Table 5. Thus, it takes about 49.5 ms to completely process a given query. The baseline model only needs the second and third processes. Therefore, it is about ten times as fast as the proposed method.

Process	Number of Queries Used	Total Time (s)	Time to Process a Single Query (ms)
Assign queries to Cluster LM using K-Means lustering	24120	0.085	0.003
Produce query embeddings using baseline model	890	0.297	0.333
Search for 100 matching products for each query	890	3.417	3.839
Produce embeddings for *Top Product Set*, search for matching products, and obtain *Refined Top Product Set*.	890	40.348	45.335

Table 5: The execution time of the baseline model and the Cluster Language Model.

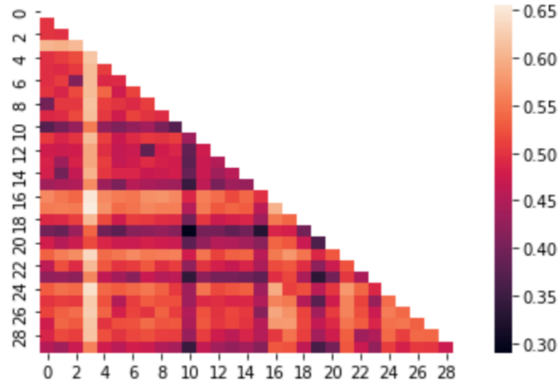


Figure 6: The relative L2 distance between cluster centers

5. Conclusion

In conclusion, this paper presented a novel approach to enhance the product search performance of the bi-encoder architecture by introducing a cluster-based fine-tuning method. The proposed method demonstrated significant improvement in recall rates up to the retrieval threshold@8, and consistently better-ranking performance across all thresholds, compared to the baseline model. Despite the increased processing time for the Cluster Language Model, it offers an alternative method to the popular bi-encoder based retrieval models in semantic search, addressing the inherent accuracy trade-offs often faced by the baseline model. The cluster-level analysis revealed that the proposed method performs well in denser clusters with a higher frequency of testing purchased products appearing in the training data. Although the baseline model outperforms the proposed method in certain clusters, the overall performance of the Cluster Language Model is superior. The L2 distance heatmap provides insights into the distinctiveness of cluster centers, which helps in the correct assignment of new queries to respective clusters.

This study demonstrates the potential of leveraging clustering techniques and fine-tuning to enhance semantic search in e-commerce applications. Further research could explore other clustering algorithms, the use of additional features, and optimization strategies to improve the performance of the proposed method and reduce processing time.

6. Acknowledgments

This work was sponsored by The Home Depot. The opinions, findings, and conclusions or recommendations expressed in this material only

reflect those of the authors in their individual capacities.

7. References

Burges, C. J. C. (2010). From RankNet to LambdaRank to LambdaMART: An Overview. *Learning*, *11*(23–581), 81. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/MSR-TR-2010-82.pdf

Caliński, T., & Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, *3*(1), 1–27. https://doi.org/10.1080/03610927408827101

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. A. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6), 391–407. https://doi.org/10.1002/(sici)1097-4571(199009)41:6

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). {Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv Preprint arXiv:1810.04805*.

Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, *4*(Nov), 933–969.

Hadsell, R., Chopra, S., & LeCun, Y. (2006). *Dimensionality Reduction by Learning an Invariant Mapping* (Vol. 2). https://doi.org/10.1109/cvpr.2006.100

Huang, P., He, X., Gao, J., Deng, L., Acero, A., & Heck, L. (2013). *Learning deep structured semantic models for web search using clickthrough data*. Proceedings of the 22nd ACM international conference on Information \& Knowledge Management. https://doi.org/10.1145/2505515.2505665

Hugging Face Transformers Library. (2021). Hugging Face Inc. https://huggingface.co/models/msmarco-distilbert-base-v2

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv Preprint arXiv:1907.11692*.

Nguyen, T. Q., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., & Deng, L. (2016). MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *Neural Information Processing Systems*. http://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf

Ortiz-Barajas, J. G., Bel-Enguix, G., & Gómez-Adorno, H. (2022). Sentence-CROBI: A Simple Cross-Bi-Encoder-Based Neural Network Architecture for Paraphrase Identification. *Mathematics*, *10*(19), 3578.

Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., Song, X., & Ward, R. K. (2015). Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *24*(4), 694–707. https://doi.org/10.1109/taslp.2016.2520371

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). *Improving language understanding by generative pre-training*.

Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv Preprint arXiv:1908.10084*.

Rosa, G., Bonifacio, L., Jeronymo, V., Abonizio, H., Fadaee, M., Lotufo, R., & Nogueira, R. (2022). In Defense of Cross-Encoders for Zero-Shot Retrieval. *arXiv Preprint arXiv:2212.06121*.

Salton, G., Wong, A., & Yang, C. (1975). A vector space model for automatic indexing. *Communications of the ACM*, *18*(11), 613–620. https://doi.org/10.1145/361219.361220

SanhSanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv Preprint arXiv:1910.01108*.

Sculley, D. (2010). *Web-scale k-means clustering*. https://doi.org/10.1145/1772690.1772862

Thakur, N., Reimers, N., Daxenberger, J., & Gurevych, I. (2020). Augmented sbert: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. *arXiv Preprint arXiv:2010.08240*.

Wang, Y., Wang, L., Li, Y., He, D., & Liu, T. (2013). A Theoretical Analysis of NDCG Type Ranking Measures. In *Conference on Learning Theory* (pp. 25–54). http://proceedings.mlr.press/v30/Wang13.pdf

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., & Rush, A. M. (2020). *Transformers: State-of-the-Art Natural Language Processing* (pp. 38–45). https://doi.org/10.5281/zenodo.5347031

Wolfe, R. R., & Caliskan, A. (2021). Low Frequency Names Exhibit Bias and Overfitting in Contextualizing Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. https://doi.org/10.18653/v1/2021.emnlp-main.41

Author Index

Azov, Guy, 40

Barut, Emre, 8
Bhandari, Naval, 136
Bhattacharya, Indranil, 136
Biswas, Biplob, 19
Buscaldi, Davide, 74

Charron, Bruno, 88
Chawan, Varsha Rani, 79
Chen, Zhilin, 125
Chu, Xiaoyu, 29

Dasgupta, Sunny, 136
Derby, Steven, 97
Du, Tianchuan, 125

Erdem, Hüseyin Varol, 65

Farber, Miriam, 1
Fiz Pontiveros, Gonzalo, 97
Fledel Alon, Adi, 40

Garg, Pulkit, 29
Giroux, Joseph, 125
Grundlingh, Nina, 79
Gueudre, Thomas, 8
Gulati, Saaransh, 29
Gupta, Shikhar, 136
Guy, Ido, 1

Hatipoglu, Pınar Ece, 65
Hu, Qian, 8
Hughes, Simon, 145
Hussey, Daniel Vidal, 58

Jiang, Keer, 29
Jiang, Ling, 29

Kamhi, Gila, 40
Kotian, Kunal, 136
Krasnov, Fedor, 115
Kurniawan Wijaya, Tri, 97

Li, Qiaomu, 79
Liu, Jiaxin, 125

Liyanage, Vijini, 74
Loughnane, Robyn, 125

Mancuhan, Koray, 65

Nakatani, Taichi, 58
Novgorodov, Slava, 1

Oral, Elif, 65
Osebe, Samuel, 8

Pavani, Kaushik, 136
Pelc, Tatiana, 40

Ramnath, Rajiv, 19
Rathgamage Don, Duleep, 145
Ruan, Weitong, 8

Schroeder, Benjamin, 125
Sharma, Rahul, 8
Shcherbakov, Viktor, 115
Su, Chengwei, 8
Sun, Weiyi, 125

Takayanagi, Takehiro, 88
Tan, Liling, 58
Tran, Thanh, 8

Visentini-Scarzanella, Marco, 88

Walter, Stephan, 58
Wang, Cody, 79
Wang, Congcong, 97
Wang, Zhiqi, 125
Wanigasekara, Prashan, 8

Xie, Ying, 79, 145

Yang, Fan, 8
Yu, Le, 145

Zhang, Bryan, 58
Zhu, Yun, 145