Search Query Spell Correction with Weak Supervision in E-commerce

Vishal Kakkar[1], **Chinmay Sharma**[2], **Madhura Pande**[2], **Surender Kumar**[2]
[1]Microsoft India
[2]Flipkart Internet Private Limited
vishalkakkar90@gmail.com
{chinmay.sharma,madhura.pande,surender.k}@flipkart.com

Abstract

Misspelled search queries in e-commerce can lead to empty or irrelevant products. Besides inadvertent typing mistakes, most spell mistakes occur because the user does not know the correct spelling, hence typing it as it is pronounced colloquially. This colloquial typing creates countless misspelling patterns for a single correct query. In this paper, we first systematically analyze and group different spell errors into *error classes* and then leverage the state-of-the-art Transformer model for contextual spell correction. We overcome the constraint of limited human labelled data by proposing novel synthetic data generation techniques for voluminous generation of training pairs needed by data hungry Transformers, without any human intervention. We further utilize weakly supervised data coupled with curriculum learning strategies to improve on tough spell mistakes without regressing on the easier ones. We show significant improvements from our model on human labeled data and online A/B experiments against multiple state-of-art models.

1 Introduction

An incorrectly spelt search query can return irrelevant products in e-commerce which hurts both the business and experience for a user who is unable to find the intended product. As per the latest English Proficiency Index report of written test data from 100 countries, [1] only 29% countries are proficient in English. Our platform operates in Asian countries like India which are ranked low in this survey. As per the latest Indian census only ∼10% of the Indian population is versed in the English language thus causing high spell errors in the user queries. Spelling errors are generally classified as — typographic and cognitive (Toutanova and Moore, 2002). Typographic errors emanate from the typing mistakes on the keyboard which worsens on mobiles due to smaller keypads. Cognitive errors happen when a user does not know how to correctly spell a word. This leads to phonetic errors like "cenityjer" for "sanitizer".

Edit-distance[2] based spell corrections at run-time are computationally expensive for higher edit distances at web-scale. Also, this approach typically works at an individual word level (Norvig, 2009; Garbe, 2021; Whitelaw et al., 2009) is not able to identify contextual mistakes like "greeting cart". Human labeling being expensive and time-consuming, these methods use a large amount of user query reformulations as training data for learning. Most of the web search work relies on this for generating correct-incorrect word pairs(Whitelaw et al., 2009; Gao et al., 2010). Query reformulations alone fail to cover all kinds of errors like phonetic (cognitive) ones - "metras" vs "mattress" or edit/phonetic+word-compounding like "ball pen" vs "bolpan" when the user herself does not know the correct spelling. Fig. 1 shows the distribution of different types of spell errors on our platform over a sample of $\sim 23k$ queries as classified by human judges. Inspired by the low-resource machine translation research, the latest spell systems create synthetic data to learn Neural (Zhou et al., 2017; Jayanthi et al., 2020) and statistical (Brill and Moore, 2000; Cucerzan and Brill, 2004) models for spell correction. To solve at internet scale, 3 billion search queries a month from over 450 million users, we too create a large amount of synthetic and user feedback data. Given significant colloquial phonetic mistakes (1), we develop a novel way of generating phonetic mistakes besides edit-distance and compounding ones. Along with the user query reformulations, the user clicks on the spell-corrected queries also provides another source of noisy labeled data. Curriculum learning (Bengio et al., 2009; Elman, 1993) has shown to generate robust models which show the improvements in their learning ability

[1]https://www.ef.com/wwen/epi

[2]https://en.wikipedia.org/wiki/Levenshtein_distance

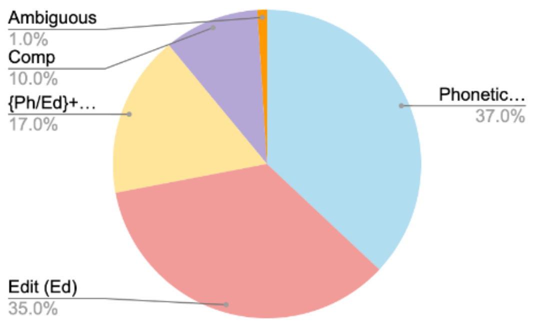

Figure 1: Spell Error Distribution

when they first learn on simpler tasks followed by tougher tasks. We devise a few simple curriculums to improve accuracy on tougher mistakes. Using Transformers, the main contributions of our work are:
(i) Generating large synthetic and weakly supervised labelled data for different spell mistakes including novel deep learning models for phonetic mistakes.
(ii) Curriculum learning to improve on tough error classes like edit/phonetic+word-compounding without degrading performance on the easier ones.

We report the effectiveness of our approach ReparoS (Reparo-Spell) through offline evaluation as well as online A/B experiment on user queries with significant improvements. In the following sections, we discuss related work followed by approach, the experiments, and finally the results.

2 Related Work

Context-free word-based spell checkers in web search have been commonly implemented in two ways: edit distance (Damerau, 1964) and statistical noisy channel. Damerau-Levenshtein edit distance finds the correct words by editing 'k' characters of the input word. On the other hand, statistical noisy channel methods (Kernighan et al., 1990) assume that the user inadvertently introduces some noise through keyboard errors. Brill and Moore improves the standard noisy channel model. Rule-based systems like Soundex(Knuth, 1973) and Metaphone(Atkinson, 2009) generate a phoneme sequence for a given word to match against the phoneme sequence of the correct word. Context-aware spell checkers (Whitelaw et al., 2009) incorporate the surrounding words to improve the correction. However, this approach still corrects each word individually and is not able to address word-compounding spell mistakes. Machine Translation ((Hasan et al., 2015)) based approaches treat a correct query as a translation of the misspelled query. Zhou et al. use an RNN with encoder-decoder architecture that improves upon the statistical approaches. To avoid expensive human-labeled data, it's become common to generate copious amounts of synthetic labeled data (Etoori et al., 2018; Jayanthi et al., 2020) for model training. NeuSpell (Jayanthi et al., 2020) formulates spell correction as a sequential labeling problem where a correct word is labelled as itself and a misspelled token is labeled as its correction. These approaches suffer due to synthetic data being mainly edit-distance based errors. Simple phonetic based corrections have been explored as well (Yang, 2022), (Brill and Moore, 2000). NeuSpell also ignores the compounding errors which form more than 25% of the spell errors. Our approach overcomes these by using state-of-art context-aware Transformer models with curriculum learning, user feedback-based data and synthetic data sets for different types of spell mistakes. In the following sections we describe data generation methods for training, model details followed by experiments and discussion.

3 Data generation

3.1 Spell Error Classes

In this section, we describe various error classes based on the patterns we observe in our e-commerce search logs. Broadly, any spell error can be classified as below:
Edit: These are induced by performing the following character operations on a correct word: (i) deletion (“nike” → “nke”), (ii) adjacent swap(“nike” → “nkie”), (iii) replacement like from neighbouring characters on the keyboard (“nike” → “bike”), (iv) insertion (“nike” → “nioke”).
Compounding: These errors are induced by the wrong usage of space ("back pack" vs "backpack").
Phonetic: When users write a query based on their pronunciation. This challenging class of errors (37%) is full of variations due to accents influenced by the regional, colloquial languages of the users ("shart", "sart" → "shirt").
{Edit/Phonetic}+Compounding: This is when edit or phonetic errors exist simultaneously with the compounding errors. For example, "air cooler" → "yercular", "dry fruits" → "drayfrut"

Figure 1 shows the distribution of various error classes based on monthly search queries received

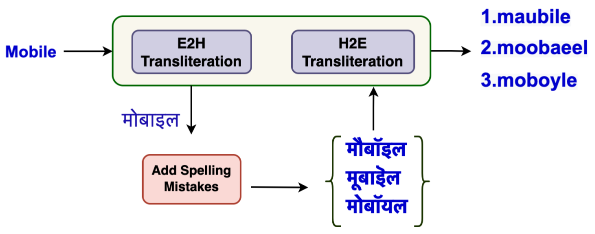

Figure 2: Pipeline for inducing phonetic errors

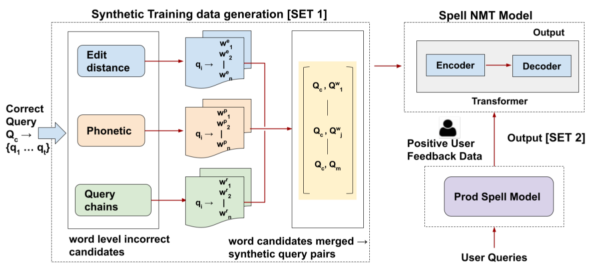
Synthetic Training data generation [SET 1]
Correct Query $Q_c \to \{q_1 \dots q_t\}$
Edit distance
Phonetic
Query chains
$q_i \to w^e_1, w^e_2, w^e_n$
$q_i \to w^p_1, w^p_2, w^p_n$
$q_i \to w^r_1, w^r_2, w^r_n$
Q_c, Q^w_1
Q_c, Q^w_j
Q_c, Q_m
word level incorrect candidates
word candidates merged → synthetic query pairs
Spell NMT Model
Output
Encoder
Decoder
Transformer
Positive User Feedback Data
Output [SET 2]
Prod Spell Model
User Queries

Figure 3: Pipeline with synthetic & weakly supervised data

on our platform. Phonetic errors are highest and can have numerous forms like *shirt* spelt *shart*, *sart* etc based on user's regional accent. Contextual spell correction is important too, for example, "pan" in "bol **pan** blue", needs correction to "pen" but not in "tea **pan**".

3.2 Synthetic Data Generation

We generate training data based on the error classes defined in the previous section. From our e-commerce search logs, we extract a seed set of clean (correctly spelt) queries on which we induce all types of errors. The clean queries are selected on the basis of their high volume and query CTR (Click Through Rate on the search result page). Errors are induced at word-level and then subsequently put back in the original query to generate the incorrect-correct training query pairs.
Edit-Distance Error: This is done by using one or more of the operations as discussed in detail in the section 3.1. We want frequent errors made by users to have a higher representation in our training data. Hence, we replace the correct word in query with its incorrect form with a probability proportional to its Brill-Moore Error Model(EM) score (Cucerzan and Brill, 2004).
Error Model (EM):We first acquire training triplets (intended word, observed word, frequency) from the user query logs (Whitelaw et al., 2009). Given a target correct word w and input word s, we then use this noisy-channel word error model to compute the probability $P(s|w)$ as described by Brill and Moore.

Phonetic Error: We leverage the fact that the users do a mental transliteration of the word from their native pronunciation to the English (Suzuki and Gao, 2012; Boyd, 2009). Although we focus on Hindi script here as it is the native language of 57% of the Indian population, this approach generalizes across any language. We first use our multilingual e-commerce product catalogue data that is already translated from English to Hindi and extract the transliterated word-pairs for particular fields like brand names and model names. We next train a character level sequence-to-sequence translation models for the transliteration tasks (English to Hindi and Hindi to English) using Transformer (Vaswani et al., 2017) architecture.
To generate synthetic phonetic word-level mistakes, we first transliterate the user typed English word into a Hindi (Devanagari [3] script) word(s) using the transliteration model described above. We then augment the transliterated Hindi word(s) and generate more noisy Hindi candidates by adding the most commonly occurring Hindi spell mistakes in the Devanagari script. These language-specific mistakes are learned separately from Hindi search queries (user query reformulations in the same user session). These native words are then back transliterated to English. Figure 2 shows the an example of how different misspelt forms of the word "mobile" are obtained via this method. Besides, transliteration we also use Soundex (Knuth, 1973), Metaphone (Atkinson, 2009) to generate phonetic candidates. In only edit-distance based erroneous word generation, the number of incorrect candidates explodes combinatorially as the length of the input word and number of edits increase which prevents generating many potential incorrect user spellings. Also, the random edit distance mistakes can generate a lot of garbage mistakes and negatively impact the model learning. The data generated by our phonetic method thus addresses these problems in a principled manner mimicking the user mistakes.
Spell Query Chain Errors: Many a times users type an incorrect spelling and subsequently correct themselves in just the next query they type. We make use of such query chains to get the spell reformulations. To ensure minimal noise we use the LLR and PMI (Jones et al., 2006) measures

[3]https://en.wikipedia.org/wiki/Devanagari

on co-occurring queries along with other checks like CTR of input and target query. We add edit-distance based guardrails to avoid generating drastically different query. Besides edit-distance query pairs, we also obtain *compounding* [("pendrive","pen drive"), ("iphonepro", "iphone pro")] errors here. Note that this method mines the user logs for the entire pair of *(incorrect, correct query)* which is unlike the earlier methods that induce errors on correct queries.

Edit/Phonetic+Compounding Error: This method combines the edit/phonetic errors that we described earlier and compounds the words. For example, for a correct bi-gram compound word "ball pen" this method generates “bolpan” from the individual mistakes on unigram tokens "ball" and "pen" which intermediately generates mistakes such as “bol" and "pan” respectively. These tokens are then joined to obtain the final incorrect unigram form "bolpan".

We finally use all the correct to incorrect word dictionaries generated above and use them to create noisy/misspelled queries for a given set of clean queries.

We also collate a substantial chunk of (*correct, correct*) query pairs so that the model learns not to overcorrect (Zhu et al., 2019; Movin, 2018) in cases when it is undesired.

3.3 Weakly Supervised Data

Another set of data is collected from the interactive user click feedback. These are user queries where our spell system intervenes to correct the input and generates a potentially correct query to fetch the search results. We collect the *(user query, system corrected query)* to form a training pair based on the CTR of the corrected query. Note that this data is be obtained after deploying one version of the spell model. With regular model retraining, this method also helps us to continuously learn on changing query patterns and mistakes.

4 Model and Training Details

We formulate the problem of query correction as a Neural Machine Translation (NMT) problem (Zhou et al., 2017), where for a source sequence (incorrectly spelt query) the task is to predict the spell corrected query. An NMT model learns the conditional probability $p(y_1, .., y_{T'}|x_1, .., x_T)$ where $(x_1, .., x_T)$ is the input query and $(y_1, .., y_{T'})$ is the target correct query. Note that the input query length T may differ from the output length $T^{'}$ due to splitting mistakes (space as a character) by users. We use Transformer (Vaswani et al., 2017) neural architecture that is superior (as shown in results section) due to bidirectional self-attention, multi-head attention and uses best neural tricks of batch normalization, resnets with encoder and decoder layer. Our choice to use Transformer was also because we found pre-trained (on in-domain data) models like BART (et al, 2019) (combining BERT (Devlin et al., 2018) and GPT) was slightly inferior to simple Transformer model which is most likely due to BART/BERT kind of models requiring longer contexts. Hyperparameter and latency details are included in the Appendix.

4.1 Curriculum Learning

Bengio et al. demonstrate the effectiveness of curriculum learning for neural language modeling tasks. Our results demonstrate that it can also help in the spell correction task. We create a curriculum strategy of first training on noisier but simpler data of synthetic query pairs. This is followed by fine-tuning the model on a smaller number of more complex errors ({edit+phonetic} with compounding). Another curriculum is fine-tuning with weakly supervised query pair data obtained from user click feedback which is also relatively cleaner. After training on noisier synthetic data, as a next step of the curriculum, we fine-tune our model on this cleaner weakly supervised data. We present the improvements brought in by the curriculum learning in the results section as compared to learning a model using all the data at once without needing any human labelled data.

4.2 Training and Eval Datasets

Our e-commerce catalog has ∼50 heterogeneous categories like clothing, electronics etc. For each of these categories, we glean top-20 percent queries (by frequency and click rates) over one month and treat them as clean queries. This created a seed set of 300K clean queries which we fed into the synthetic data generation pipeline described in section 3.2. We generate ∼270M synthetic query pairs with all kinds of mistakes. In theory, user queries may have multiple misspelled words but our analysis showed that ∼90% queries have mistakes in up to two words per query which we incorporated in our synthetic data generation pipeline too. To obtain the weakly supervised data, we first deployed

Model	**Training Data**	**Imp**	**Reg**
GoogleSWBS (statistical web search) (Whitelaw et al., 2009)	in-house synthetic data	33.96	82.45
Seq2Seq-BiRNN (Zhou et al., 2017)	in-house synthetic data	34.86	91.1
NeuSpell-BERT (Jayanthi et al., 2020)	neuspell data	12.19	58.55
NeuSpell [in-domain BERT]	neuspell data generation methods applied on in-house data	25.3	83.6
NeuSpell [in-domain BERT]	in-house synthetic data	32.6	84.2
ReparoS-Base	in-house synthetic data	37.63	90.72
ReparoS-C1(fine-tune ReparoS-Base)	ed/ph + comp	42.81	86.63
ReparoS-C2(fine-tune ReparoS-C1)	user click feedback (weak supervision)	**46.09**	**91.62**

Table 1: Accuracy (%) comparison of different models on **Imp**rovement and **Reg**ression datasets

the first version of the model and logged the user interaction with the model output. Whenever the system spell corrects a given user query and user accepts it by clicking on the products (indicated by query CTR) from the altered query, we consider that as a positive correction. We collected this data for a month and used the same to fine-tune the subsequent versions of our model. This fine tuning process allows for continual learning of the model from user feedback. A more detailed continual learning with knowledge distillation is our future area of work. For evaluation, we obtain 72K human labelled query pairs. This data is created by stratified sampling from unique head/tail queries (queries sorted in descending order of frequencies in a month and split into 2 equal quantiles: head and tail). These are then labeled by the human judges who provide the corrected queries for the input set. We measure the performance separately on 2 types of sub-datasets called Regression and Improvement. The regression set consists of 90% head and 10% tail queries and is hence predominantly clean. This ratio is inverted for the improvement set where 90% are tail (tough queries) and has a lot of spell mistakes.

5 Results and Discussion

We conduct both offline and online evaluation of our models.

Offline evaluation: Table 1 shows the results of multiple baselines on the Improvement and Regression data sets. Row 1 compares the performance of ReparoS with a statistical model GoogleSWBS (Whitelaw et al., 2009) that corrects at a word-level. We improved this model to correct the entire query using beam search (Freitag and Al-Onaizan, 2017) (to avoid combinatorial explosion of candidates) and a language model. In Row 2, Zhou et al. train a bidirectional RNN based sequence to sequence model with encoder/decoder architecture for NMT. Row 3-5 has NeuSpell (Jayanthi et al., 2020) that is bidirectional encoder-only model with sequence labeling task for each input token. Jayanthi et al. provide a toolkit where different models (ELMO, CNN-LSTM, BERT) can be plugged while the last two layers remain constant. We chose their best performing variant with BERT, i.e., Neuspell-BERT for comparison. In Row 3, we evaluate unmodified NeuSpell model on our evaluation sets and observed poor performance. To provide a better domain adaptation, we plugged in our in-house trained BERT model that is pre-trained on e-commerce search queries. Row 4, 5 show the performance of this model. In Row 4, we use the data generation methods suggested by Jayanthi et al. on our data and then train this model. Although we see an increase in the numbers but still not better than the other models. Row 5 shows the results when we train this NeuSpell model plugged in with in-domain BERT with our data generation techniques. This also is the best performing configuration among all the NeuSpell experiments. The results also demonstrate the improvements due to our data generation methods as the seed queries/words for Row 4 and 5 were same. Additionally, we observe that seq-to-seq model (RNN, Transformer) is better than sequence labeling one. Another limitation of NeuSpell due to sequence labeling is that it can't handle compounding errors like *iphonepro* to *iphone pro*. It's evident that our diverse synthetic data generation techniques are effective and lead to significant improvement even with a simple 1 layer enc/dec (ReparoS-Base) transformer compared to deeper pre-trained models like BERT and NeuSpell. Adding candidates from our novel phonetic transliteration model was beneficial and led to a total absolute gain of 2-3% at query level consistently across the models and 7.5% accuracy improvement at the word candidate level.

Effect of curriculum learning: While ReparoS-Base itself is good than the competent baselines,

Treatment	Control	∆Page-CTR	∆Cart-Add	∆Null-Search	∆Corr-ections	∆Click-back
ReparoS-Base	Google-SWBS	0.13%	1.02%	0%	2.92%	-1.56%
ReparoS-C2	ReparoS-Base	0.06%	0%	-6.94%	7.65%	-3.43%

Table 2: A/B results against control @5%significance

our error analysis showed that it still couldn't address the complex edit/phonetic + compounding spell errors. This led us to design a few new curricula to improve. Our first curriculum ReparoS-Base-1, was to simply add more of complex edit/phonetic+compounding training samples. This resulted in marginal improvement over ReparoS-Base. However, with a different curriculum of only fine-tuning ReparoS-Base on tougher mistakes (model ReparoS-C1) we observe that the performance increases significantly on the Improvement Set where most of the mistakes lie, however, there's a drop in the Regression Set performance when compared to the ReparoS-Base and ReparoS-Base-1. On analyzing this further, we observed that this was due to the over-correction problem where the model is aggressively altering correct queries in the regression set. Hence, we change the curriculum and in ReparoS-C2 we find that a further fine-tuning on weakly supervised user feedback data improves upon all the variants significantly on both the data sets. This is intuitively due to relatively more frequent queries in the Regression Set on which receiving implicit user-feedback through clicks is possible at scale. Thus adding this new curriculum helped acheive the best performance.

Online evaluation: In production, we adopt a 2-step architecture by adding an ML ranker (Yang, 2022) that does the final candidate selection (from multiple candidates from the ReparoS/GoogleSWBS). This multi-stage setup empirically produced better results than just fine-tuning the NMT model since top-10 accuracy of ReparoS-C2 is 76.31% on Improvement Set and 98.08% on Regression Set. This helped all the models (and ReparoS more due to their better top-k accuracy) and is removed from results discussion for brevity.

Table 2 reports the online performance of ReparoS-Base against its control bucket of GoogleSWBS (our first deployed in-house production model) and later ReparoS-C2 against the control bucket of ReparoS-Base.

In both the A/B tests, 20% of the users were randomly assigned to each bucket (control and treatment) and the experiments were run for 3 weeks each to achieve statistical and practical significance. ReparoS-C2 is the currently deployed model serving the entire search traffic of our app for more than 9 months. At our scale, 0.01% is quite a significant change in PageCTR. Both ReparoS-Base and ReparoS-C2 show a reduction in click backs while increasing spell system coverage as well as the Search results page CTR. ReparoS-C2 also reduced the Null Searches drastically while ReparoS-Base improved the number of cart adds by the users. These results demonstrate the effectiveness of both the versions of ReparoS across multiple user and business metrics.

6 Conclusion

We presented the generation of large synthetic and weakly supervised labeled data for different types of user spell mistakes including a creative deep learning model to generate the phonetic mistakes. We then presented a sequence-to-sequence deep Transformer based Spell model with curriculum learning on tough spell mistakes and user feedback data that demonstrates superior performance than state-of-the-art statistical and neural spelling correction models. Our solution is currently deployed on an India's e-commerce platform and serves over billions of queries per month from over 450 million users across 100% zip codes of India. With new users with varying literacy rates joining our platform every day, spell correction remains a tough problem to solve. We found that deeper NMT models result in better performance but are impractical (in the current state due to higher latencies) to deploy in production to serve real-time user requests. Hence our future work is to focus on model pruning, quantization, and knowledge distillation and continual learning to reduce latencies for deployment and improve upon our current system.

7 Acknowledgements

We thank our colleagues Samir Shah, Pooja Kushalappa, Shubham Kumar Rastogi, Adit Mittal, Rikkin Majani for stimulating discussions and critical feedback. We also thank our human annotators for their help in labelling and evaluation.

References

K Atkinson. 2009. Gnu aspell.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. ICML '09, page 41–48, New York, NY, USA. Association for Computing Machinery.

Adriane Boyd. 2009. Pronunciation modeling in spelling correction for writers of english as a foreign language. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium*, pages 31–36.

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

Silviu Cucerzan and Eric Brill. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 293–300, Barcelona, Spain. Association for Computational Linguistics.

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

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*.

Jeffrey L. Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition*, 48:71–99.

Mike Lewis et al. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

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

Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60, Vancouver. Association for Computational Linguistics.

Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. 2010. A large scale ranker-based system for search query spelling correction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 358–366, Beijing, China. Coling 2010 Organizing Committee.

Wolf Garbe. 2021. Symspell: Spelling correction and fuzzy search: 1 million times faster through symmetric delete spelling correction algorithm.

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.

Sai Muralidhar Jayanthi, Danish Pruthi, and Graham Neubig. 2020. NeuSpell: A neural spelling correction toolkit. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online. Association for Computational Linguistics.

Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, page 387–396, New York, NY, USA. Association for Computing Machinery.

Mark D. Kernighan, Kenneth W. Church, and William A. Gale. 1990. A spelling correction program based on a noisy channel model. In *Proceedings of the 13th Conference on Computational Linguistics - Volume 2*, COLING '90, page 205–210, USA. Association for Computational Linguistics.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Donald Knuth. 1973. The art of computer programming: Volume 3, sorting and searching. pages 391–392.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*.

Maria Movin. 2018. Spelling correction in a music entity search engine by learning from historical search queries.

Peter Norvig. 2009. Beautiful data. page 234–239.

Hisami Suzuki and Jianfeng Gao. 2012. A unified approach to transliteration-based text input with online spelling correction.

Kristina Toutanova and Robert C. Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 144–151, USA. Association for Computational Linguistics.

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 neural information processing systems*, pages 5998–6008.

Casey Whitelaw, Ben Hutchinson, Grace Y. Chung, and Gerard Ellis. 2009. Using the web for language independent spellchecking and autocorrection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, page 890–899, USA. Association for Computational Linguistics.

Fan et al Yang. 2022. Spelling correction using phonetics in E-commerce search. In *Proceedings of the Fifth Workshop on e-Commerce and NLP (ECNLP 5)*, Dublin, Ireland. Association for Computational Linguistics.

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

Canxiang Zhu, Zhiming Chen, Yang Liu, Juan Hu, Shu qiong Sun, Bixiao Cheng, Zhen-dong, and Yang. 2019. Automatic query correction for poi retrieval using deep and statistical collaborative model.

8 Appendix

8.1 Experimental Setup

For model training, we used GPU set-up (NVIDIA Tesla V100-SXM2) while for inferencing in the user path we used CPU set-up (Intel x_86, 64 bit, 2.1GHz, VM with KVM Hypervisor). After hyper-parameter tuning, we used 8 attention heads, and 128 hidden dimensions with Adam Optimizer with learning rate set to 1.0 and $\beta_1 = 0.8$, $\beta_2 = 0.998$, $\epsilon = 1e - 8$. We used sentencepiece (Devlin et al., 2018) (Kudo, 2018) to generate a subword vocabulary of size 8K. Beam-width was set to 10 in the decoder phase. OpenNMT (Klein et al., 2017) was used for training due to its CTranslate utility [4] for faster inference (brought down CPU inference time for 1 layer model to $< 7ms$ from $25ms$ per query at single concurrency. For fine-tuning we set learning rate to 0.0001 and all the model parameters are updated during the training.

8.2 Latency/Accuracy trade-offs with deeper models

While deeper models improve the accuracy, they also take more time in inference which is critical in live systems like search query spell correction. Table 3 summarizes the comparison of deeper models on time taken for inference on both GPU (NVIDIA Tesla V100-SXM2) and CPU(Intel x_86, 64 bit, 2.1GHz, VM with KVM Hypervisor) setups along with the corresponding model metrics. Please note that for NeuSpell, we have used the 12 layer in-domain BERT to match NeuSpell's in-built out-of-domain BERT which also has 12 layers of encoders. For ReparoS, the number of encoder and decoder layers were always kept equal. Due to higher latencies on CPUs (the typical economical serving infrastructure of choice) observed on deeper models, we eventually chose ReparoS-C2 with one layer of encoder/decoder.

[4]https://github.com/OpenNMT/CTranslate2

Model	**CPU time**	**GPU time**	**Imp**	**Reg set**
NeuSpell[in-domain BERT] (12-encoder layers)	71	20	32.6	84.2
ReparoS-C2 (1-encoder and 1-decoder layer)	**6.67**	**7.85**	46.09	91.62
ReparoS-C2 (4-encoder and 4-decoder layers)	16.43	18.19	51.27	92.83
ReparoS-C2 (6-encoder and 6-decoder layers)	56.08	27.92	**52.6**	**93.14**

Table 3: Accuracy(%) vs latency trade-off as depth of the models increase. Time is reported in ms.

8.3 A/B Metrics

- PageCTR: Click through rate of Search Results Page

$$PageCTR = \frac{\#Pages\ with\ at\ least\ 1\ click}{\#Total\ Pages\ shown}$$

- CartAdd: Number of products per user visit (to the search page) being added by the users to the cart to purchase

$$CartAdd = \frac{\#Search\ Visits\ with\ product\ added\ to\ cart}{\#Total\ user\ visits\ to\ search\ page}$$

- NullSearch: It represents ratio of search result page with no results to total search results pages shown

$$NullSearch = \frac{\#Pages\ with\ no\ results}{\#Total\ Pages\ shown}$$

- Corrections: It represents the number of times spell system changed a user query and has to be looked in conjunction with the other metrics.

$$Coverage = \frac{\#Spell\ system\ changed\ original\ query}{\#Total\ searches}$$

- Clickback: It measures events where user tells the system to show results for his/her original query by clicking 'show results instead for <original query>'. Hence, a reduction in Clickback and NullSearch denotes improvement.

$$Clickback = \frac{\#clicks\ on\ 'results\ for\ originalquery'}{\#Total\ search\ requests}$$