Textagon: Boosting Language Models with Theory-guided Parallel Representations

John P. Lalor,[1,2] **Ruiyang Qin,**[3*] **David Dobolyi,**[4] **Ahmed Abbasi**[1,2]
[1]Human-centered Analytics Lab, University of Notre Dame
[2]IT, Analytics, and Operations, University of Notre Dame
[3]Electrical and Computer Engineering, Villanova University
[4]Leeds School of Business, University of Colorado Boulder
{john.lalor, aabbasi}@nd.edu, rqin@villanova.edu, david.dobolyi@colorado.edu

Abstract

Pretrained language models have significantly advanced the state of the art in generating distributed representations of text. However, they do not account for the wide variety of available expert-generated language resources and lexicons that explicitly encode linguistic/domain knowledge. Such lexicons can be paired with learned embeddings to further enhance NLP prediction and linguistic inquiry. In this work we present `Textagon`, a Python package for generating parallel representations for text based on predefined lexicons and selecting representations that provide the most information. We discuss the motivation behind the software, its implementation, as well as two case studies for its use to demonstrate operational utility.

PyPi: https://pypi.org/project/textagon/
GitHub: https://github.com/nd-hal/textagon
YouTube: https://youtu.be/zUxamCT8mPg

1 Introduction

Learning distributed representations of text via large pretrained language models (PLMs) trained with massive amounts of text data has been a driver of recent progress in NLP. Pretrained, numeric representations of words and sentences encode semantic similarity in a high-dimensional space. While PLMs' performance has been impressive, distributed representations learned from large general corpora are not the only type of representation available.

For decades, linguistic researchers and social scientists have worked with representations of texts that are based on grammatical structure, linguistic theories, or domain-adapted lexicons. These lexicons cover ideational, textual, and interpersonal functions of language (Systemic Functional Linguistic Theory, Halliday and Matthiessen, 2014), the pragmatic dimension of language, including actions and intentions (Language Action Perspective, Searle, 1969), key psychological processes (e.g., Pennebaker et al., 2001), and domain-specific lexicons, which shed light on task- and context-related nuances (e.g., finance, Loughran and McDonald, 2011). This literature recognizes that although text, as a data structure, is 1-dimensional, the meanings embodied in natural language are multi-dimensional.

Increasingly, NLP is being used for computational social science tasks where text is scored (i.e., text sequence classification) or analyzed to predict, explain, or describe phenomena manifesting in user-generated content (Grimmer et al., 2022). In these contexts, the use of PLMs has been impeded by several factors. First, labeled data for many social science use cases—such as examining in-text manifestations of confidence, trust, anxiety, distress, empathy, and personality traits—is insufficient for fine-tuning PLMs (Macanovic, 2022). Consequently, researchers and practitioners are concerned about error rates in text classification, which may statistically bias estimation in downstream descriptions and explanations (Yang et al., 2018; Macanovic, 2022). Moreover, those without sufficient computational resources have concerns about whether smaller PLMs can still provide competitive models (Macanovic, 2022). Second, disciplinary norms often dictate the use of certain linguistic resources for content analysis or expected levels of methodological interpretability.

Recent studies have highlighted the potential of extracting and leveraging features from various linguistic dimensions to boost performance in downstream tasks (Yang et al., 2023; Qin et al., 2024b; Abdi et al., 2019; Ahmad et al., 2020; Qin et al., 2024a) via tailored models. Prior work has shown that combining structured features with PLMs can tackle advanced tasks such as bias correction (Lalor et al., 2022), out-of-domain detec-

*Work performed while at Notre Dame.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 82–92

tion (Duan et al., 2022), and misinformation identification (Lee and Ram, 2024). These models can discern features potentially overlooked by larger transformer-based pretrained models.

In this work we present `Textagon`, a Python package for generating parallel representations for text. We define parallel representations as token-level features extracted from multiple lexicons that, when combined, form a token-lexicon feature matrix. `Textagon` provides functionality to generate parallel representations as well as a grid-based feature weighting module to identify the most informative representations. The package allows practitioners to expand raw text data to multi-dimension data based on linguistic theories to augment PLMs. Our contributions are a) `Textagon`, an open-source Python package for generating and selecting parallel representations for text, b) a detailed description of the software architecture, and c) illustrative examples to facilitate easy use of the software. `Textagon` is available via PyPi.[1]

2 Related Work

Recent work has shown that feature expansion and enrichment can enhance text classification tasks within neural network architectures (Zimbra et al., 2018; Huang et al., 2017). For example, Ahmad et al. (2020) generate diverse representations for use in CNN and Bi-LSTM models for analyzing comprehensive psychometric dimensions. Automated Concatenation of Embeddings (ACE, Wang et al., 2020a) automates the process of finding better concatenations of distributed embeddings for structured prediction tasks using reinforcement learning. Alghanmi et al. (2020) combine BERT with static word embeddings. Wang et al. (2020b) demonstrate that combining distributed representations can benefit the language model. Bollegala (2022) show that weighted concatenation can be seen as a spectrum matching operation between source embeddings and the meta-embedding. To the best of our knowledge, there is no existing package for generating and combining parallel representations.

3 The `Textagon` Package

`Textagon` implements two key components. The first generates the parallel representations based on the available lexicons. The second component scores and ranks the top weighted parallel representations so that an appropriate sub-set of representations can be used for specific tasks. An example to generate and score parallel representations with `Textagon` is shown in Figure 1.

```python
import pandas as pd
from textagon.textagon import Textagon
from textagon.tGBS import tGBS

df = pd.read_csv(
    './sample_data/dvd.txt',
    sep = '\t',
    header = None,
    names = ["classLabels", "corpus"]
)

tgon = Textagon(
    inputFile = df,
    outputFileName = "dvd"
)

tgon.RunFeatureConstruction()
tgon.RunPostFeatureConstruction()
```

(a)

```python
featuresFile = './output/dvd_key.txt'
trainFile = './output/dvd.csv'
weightFile = './output/dvd_weights.txt'

ranker = tGBS(
        featuresFile = featuresFile,
        trainFile = trainFile,
        weightFile = weightFile
)

tGBS.RankRepresentations()
```

(b)

Figure 1: An example of running `Textagon`: First generating representations (1a) followed by ranking the representations based on informativeness (1b).

3.1 Generating Representations

`Textagon` generates and ranks parallel representations of token-level lexical features. By parallel representations, we are referring to a matrix structure for an input string. Each column represents a token, and each row represents a lexicon. Each cell then contains the appropriate lexicon tag for the given token. If there is not a tag, then the token is retained as-is.

As a running example, consider the following text: "hypotension. Massive headaches, bp was still on the low side." `Textagon` can expand this into 20 different representations categorized into five groups (Table 1). The base representation, Word, represents a refined version of the original data. Importantly, the parallel representations (Table 1) are token-aligned and can be considered

[1] https://pypi.org/project/textagon/

Group	Representation	Description	Example
	Word	Baseline	hypotension . massive headaches , bp was still on the low side .
T	Hypernym	Replace a token with its superordinate	**CARDIOVASCULAR_DISEASE** . massive ACHE , bp was still on the low **GEOLOGICAL_FORMATION** .
	NER	Named entity recognition (NER) tags	hypotension . massive headaches , **ORG** was still on the low side .
	LexADR	Adverse drug reaction (ADR) tags	**REACTION** . massive **REACTION** , bp was still on the low side .
	LexSYN	Synonym cluster label tags derived by clustering tokens based on their synsets	hypotension . massive **SYN217** , bp was still on the **SYN23 SYN345** .
	LexGloVeCC	GLoVe Common Crawl labels derived clustering tokens based on embeddings	GLOVECC234 GLOVECC251 GLOVECC312 GLOVECC457 , GLOVECC244 GLOVECC46 GLOVECC46 GLOVECC251 GLOVECC251 GLOVECC312 GLOVECC440 GLOVECC251
	LexGloVeTW	GLoVe Twitter labels derived clustering tokens based on embeddings	GLOVETW23 GLOVETW122 GLOVETW147 GLOVETW165 GLOVETW285 GLOVETW392 GLOVETW119 GLOVETW119 GLOVETW238 GLOVETW238 GLOVETW26 GLOVETW349 GLOVETW122
	LexGloVeWG	GLoVe Wikipedia plus Gigaword labels derived clustering tokens based on embeddings	GLOVEWG279 GLOVEWG436 GLOVEWG364 GLOVEWG329 GLOVEWG414 GLOVEWG145 GLOVEWG436 GLOVEWG436 GLOVEWG436 GLOVEWG436 GLOVEWG18 GLOVEWG353 GLOVEWG436
SA	Sentiment	Positive, negative, or neutral tags	**LPOSMNEG** . **LPOSLNEG LPOSLNEG** , bp was **LPOSMNEG** on the **LPOSLNEG LPOSLNEG** .
	Affect	Affect tags	hypotension . massive headaches , bp was still on the **SADNESS** side .
	LexEMOLEX	NRC Emotion Lexicon	hypotension . **EMOFEARNEGATIVESADNESSSURPRISE** headaches , bp was still on the low side .
	LexAILEXCAT	Affect Intensity Lexical Categorization	hypotension . massive **FEAR** , bp was still on the low side .
	LexAILEXINT	Affect Intensity Lexical Intensity	hypotension massive **MFEAR** bp was still on the low side nan
P	LexLIWC	Linguistic inquiry and word count (LIWC) categories	hypotension . massive **HEALTH** , bp **AUXVB ADVERBS FUNCT ARTICLE SPACE RELATIV** .
	LexSAVLEX	SAVLEX word standardization	hypotension . massive headaches , bp was still on the **WP KA** .
SS	POS	POS tags	**NOUN PUNCT ADJ NOUN PUNCT PROPN AUX ADV ADP DET ADJ NOUN PUNCT**
	Misspelling	Tag for misspellings	hypotension . massive headaches , **MISSPELLING** was still on the low side .
	Legomena	Tag for unique words	hypotension . massive headaches , bp was still on the low side .
S	Word&Sense	Labels based on distinct word senses	**hypotension|_|01 . massive|_|04 headaches|_|02** , bp was **still|_|04** on the **low|_|04 side|_|01** .
	Word&POS	Part-of-speech (POS) tags tupled with their respective word occurrences	**hypotension|_|NOUN .|_|PUNCT massive|_|ADJ headaches|_|NOUN ,|_|PUNCT bp|_|PROPN was|_|AUX still|_|ADV on|_|ADP the|_|DET low|_|ADJ side|_|NOUN .|_|PUNCT**
	Word&NER	Named-entity recognition (NER) tags	hypotension . massive headaches , **bp|_|ORG** was still on the low side .

Table 1: A description of the parallel representations generated by `Textagon` for an illustrative example. T: Topical, SA: Sentiment and affect, P: Psychological and pragmatic, SS: Syntax and style, S: Semantics.

as a token-lexicon matrix representation. This allows for easier integration into convolutional or sequence-based learning representations and for easier content analysis of text or PLM attention mechanisms. Moreover, the included representations are guided by linguistic and social science theories (Searle, 1969; Pennebaker et al., 2001; Mohammad and Turney, 2010) and can be easily extended by users via custom lexicons.

3.2 Representation Ranking with tGBS

`Textagon` first generates and selects representations for feature extraction. As Table 1 shows, twenty representations can be generated for a given dataset (though users can add additional lexicon-based representations as needed). Parallel representations can provide diverse linguistic perspectives; however, they can also introduce redundant information, potentially diminishing their utility. To address this, `Textagon` implements an n-gram Grid-Based Subsumption (GBS) algorithm (Ahmad et al., 2020) to retain key features, making the embedding more effective. Subsumption filters higher-order features to remove redundancy and improve information gain (Riloff et al., 2006; Abbasi et al., 2011).

GBS Calculation. We modify the n-gram GBS algorithm of Ahmad et al. (2020) to fit our token-level parallel representation design. The tokenized GBS algorithm (tGBS) gives each token a GBS weight for each representation (refer to Appendix A for details). tGBS generates token importance weights for each token in every representation.

To select the most informative representations for inclusion, we calculate a score for each representation, S_R, which reflects the information gain of the entire representation compared to the original text data. To calculate S_R we consider the ratio of tokens in a representation with non-zero tGBS weight. Specifically, for a tokenized input x_i and a representation R, we calculate the count of tokens x_i where the tGBS score of x_i in representation R is greater than some (small, non-zero) threshold θ.

$$S_R = \frac{|\{x_i \in X | \text{tGBS}(x_i, R) > \theta\}|}{|X|} \quad (1)$$

This ratio, S_R, offers a quantitative insight into the proportion of significant features retained in each representation, thereby serving as an indicator of the representation's richness or sparsity concerning the underlying dataset. After generation, we rank representations based on S_R. Users can then select the appropriate number of representations based on their use cases.

4 Evaluation

In this section, we evaluate the effectiveness of `Textagon` in three ways. First, we present a case study using representations generated by `Textagon` to compare human- and LLM-generated essays. Second, we analyze the expressive power of the tGBS-based parallel representations generated by `Textagon` on 13 testbeds/tasks covering domains such as health, medicine, and disasters, and tasks including inferring trust, anxiety, confidence, distress, and empathy (Table 2). Third, on the same 13 testbeds, we show how representations generated by `Textagon` can boost predictive performance on encoder-only (e.g., BERT, RoBERTa, DistilBERT) and decoder-only models (e.g., GPT). These cases illustrate how `Textagon` can support context-specific computational social science via direct text analysis as well as analysis of fine-tuned PLMs. Future work using `Textagon` can build on these examples.[2]

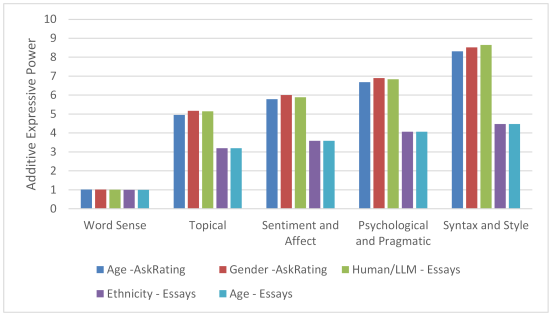
4.1 Content Analysis Case Study

Token-aligned parallel representations can shed light on the important linguistic dimensions of a given token as they relate to a downstream computational social science task of interest. Importantly, `Textagon` can be used for textual content analysis by combining parallel representations and class labels to highlight differences across classes. Because representations are token-aligned, `Textagon` can also surface linguistic dimensions of model attention when fine-tuning a PLM for a target application domain. Here, we present a small case study on automated essay scoring (AES), a problem that is of interest to the NLP community as well as computational social scientists (Taghipour and Ng, 2016; Yang et al., 2020). We use the publicly available human and GPT-generated essay testbed developed by Bevilacqua et al. (2025) and the AskRating drug sentiment dataset (Sharif et al., 2014; Lalor et al., 2022) to explore: (1) linguistic differences between human and GPT essays; (2) BERT attention patterns when fine-tuned to score human versus GPT-generated essays. The essay testbed is comprised of over 15K human-generated essays and approximately 1.5K GPT-generated essays. GPT essays were constructed using the same human essay prompts taken from popular AES testbeds, ASAP (Mathias and Bhattacharyya, 2018) and FCE (Yannakoudakis et al., 2011).

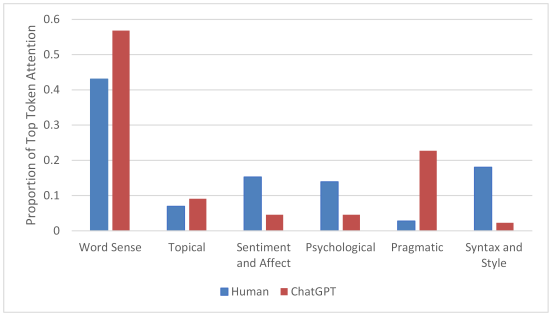
We first extracted parallel representations for human- and GPT-generated essays and used tGBS to score them. Here the label for identifying the most informative representations is the source of the essay (e.g., human or GPT). We then aggregated the expressive power across representations by their linguistic categories. The results appear in Figure 2a as the "Human/LLM - Essays" bar series (middle bars). For comparison, we included two sets of baselines. First, we ran a similar analysis on the AskRating testbed, with two label options for representation ranking: gender (authors self-reported as male/female) and age (above/below the median age). These results are shown in the two leftmost bar series in Figure 2a. For the second baseline we focus on the 15K human essays, and for labels we use ethnicity (self-reported Asian/non-Asian authors) and age (older versus younger authors). These two series appear as the rightmost bars in Figure 2a.

As shown in Figure 2a, we find that the parallel representational composition for human versus GPT-generated essays across dimensions such as topical, sentiment/affect, psychological/pragmatic, and style/syntax differ far more than, say, essays written by different (self-reported) human demographic groups (e.g., Asian versus non-Asian or younger versus older authors). In fact, the parallel representational compositions are akin to those for different demographic groups in the AskRating online health forum testbed (e.g., differences between gender and age of the health forum participants). These results can shed light on the linguistic differences in user-generated content created by differ-

[2]Notebooks for our evaluations are available at `https://github.com/nd-hal/textagon/`.



(a) Parallel representation profiles for human versus GPT-generated essays



(b) Parallel representation profiles for most attended tokens in fine-tuned BERT in human versus GPT-generated essays

Figure 2: Results of our content analysis case study.

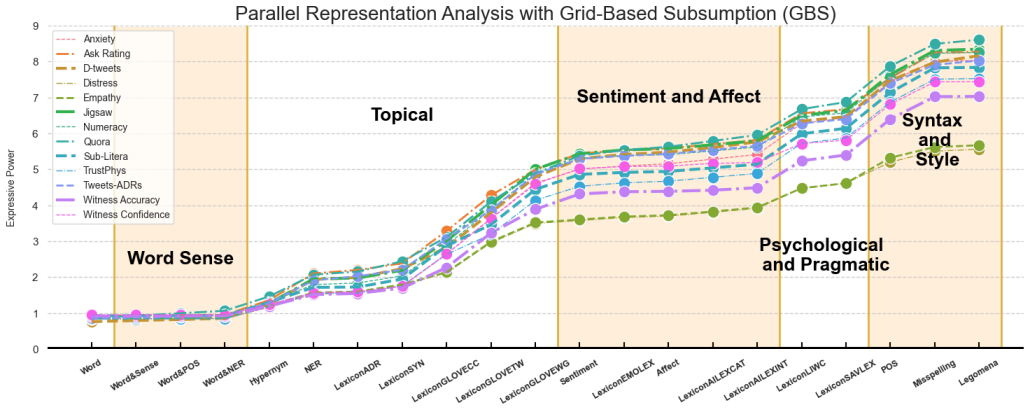


Figure 3: Cumulative expressive power of parallel representations in `Textagon`, across testbeds, by category.

ent user sub-groups, as well as differences between human-LLM content in the era of generative AI.

Next, we fine-tuned a BERT model (bert-base-uncased) on the human-generated essays. We then extracted the top sixty most prevalent tokens in human and GPT-generated essays, respectively, and passed them through the fine-tuned BERT to see how the attention layers were attending to these tokens. For the tokens that BERT was attending to (i.e., where aggregated average attention scores are greater than a predefined threshold t), we then analyzed their tGBS-processed parallel token representations for analysis (Figure 2b).

The bars depict the proportion of the most attended to tokens in the fine-tuned BERT model that have an informative parallel token in that respective language dimension (e.g., word sense, topical, sentiment/affect, etc.). Notably, the results reveal that although the BERT attention for top human/GPT tokens is comparable in terms of its parallel representational composition for word sense and topical tokens, top human texts contain more sentiment/affect, psychological process, and syntax/style information (e.g., once-used/hapax legomenon tokens, misspellings, characters). Conversely, the top GPT tokens attended to are richer in terms of the pragmatic dimensions of language (e.g., actions, intentions, declaratives, etc.). These results, which are made possible via parallel representation generation and token-aligned tGBS scoring via `Textagon`, illustrate deeper PLM content analysis affordances enabled by `Textagon` in an important computational social science context.

4.2 Expressive Power Results

Next, we show the expressive power of the parallel representations produced by `Textagon`, relative to the baseline word token representation, using tGBS (Figure 3). As representations are added across linguistic categories, the amount of information included increases. Looking at the rightmost side of the figure, we note that the total amount of additional information expressed (in terms of potentially informative tokens across the

20 representations) ranges from 4x-7x. These representations are then sorted on a per-dataset basis to identify the top representations for inclusion into downstream tasks (e.g., content analysis, classification). Next, we show how this additional expressive power can translate into enhanced text classification predictive power.

Dataset	N	Reference
Anxiety Numeracy SubjectiveLit TrustPhys	8,502	(Ahmad et al., 2020; Abbasi et al., 2021; Lalor et al., 2022, 2024)
AskRating	20,000	(Sharif et al., 2014; Lalor et al., 2022)
Distress Empathy	1,860	(Buechel et al., 2018)
DisasterTweets	7,613	(Howard et al., 2019; Cloutier and Japkowicz, 2023)
Jigsaw	20,000	(Adams et al., 2017)
Quora20k	20,000	(DataCanary et al., 2017)
TweetsADR	5,009	(Hassan et al., 2013; Zimbra et al., 2018)
WitnessAccuracy WitnessConfidence	2,224	(Dobolyi and Dodson, 2018)

Table 2: Datasets used in our classification example. Please refer to the original citations for further details on data collection, validation, etc.

4.3 Text Classification Performance

We assess the potential lift to PLM classifiers by comparing a directly fine-tuned PLM baseline classifier with one where `Textagon` features extracted from the parallel representations are concatenated with PLMs during the fine-tuning process. Concatenation occurs with the embeddings from the transformer-based models (See Figure 5, panel C in the appendices) and are forwarded into a multilayer perceptron (MLP) to produce the prediction output. The included PLMs were: BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), DistilBERT (Sanh et al., 2019), and GPT-2 (Radford et al., 2019).

Figure 4 shows AUC performance results across a collection of benchmarking datasets (Table 2). Incorporating `Textagon` parallel representations to the classification tasks typically improves predictive performance, with lifts on BERT and RoBERTa ranging from 1%-5% in most cases. Gains on smaller PLMs such as DistilBERT were even more pronounced. `Textagon` enables the identification of more informative parallel representations for each task, which can have important implications for downstream explanatory and descriptive insights (Yang et al., 2018).

5 Conclusion

In this work, we have presented `Textagon`, a Python package for generating and selecting informative, theory-driven parallel representations. `Textagon` implements several key components to facilitate parallel representation generation and selection. Token-level tGBS calculation measures the information gain of each representation compared to the original text data to identify those representations that can improve model performance. The output representations can then be used as standalone features for downstream tasks or can be concatenated with embeddings from PLMs for a richer representation of the input text before classification. We demonstrate use cases of `Textagon` for content analysis and enhancing predictive performance. `Textagon` can facilitate linguistic examinations of which lexicons provide the most information and which are most beneficial to PLMs for classification tasks. In addition, `Textagon` can incorporate new lexicons as future researchers develop them to further enhance predictive power. Our work has important implications for computational social science researchers and practitioners.

There are several limitations for this work. `Textagon` relies on the quality and availability of input lexicons for parallel representation generation. What's more, lexicons are inherently incomplete in that they may only have tags for a subset of tokens. Researchers incorporating `Textagon` should ensure that the lexicons used are appropriate for their use cases. The incorporated lexicons are appropriate for open-domain text, but if needed can be augmented with domain-specific resources as well (e.g., Loughran and McDonald, 2011). Generating and selecting representations can be computationally expensive, in particular for large datasets. While we propose an information-gain heuristic for representation selection (Appendix B), future work on efficient generation and selection can improve processing speed for the overall pipeline.

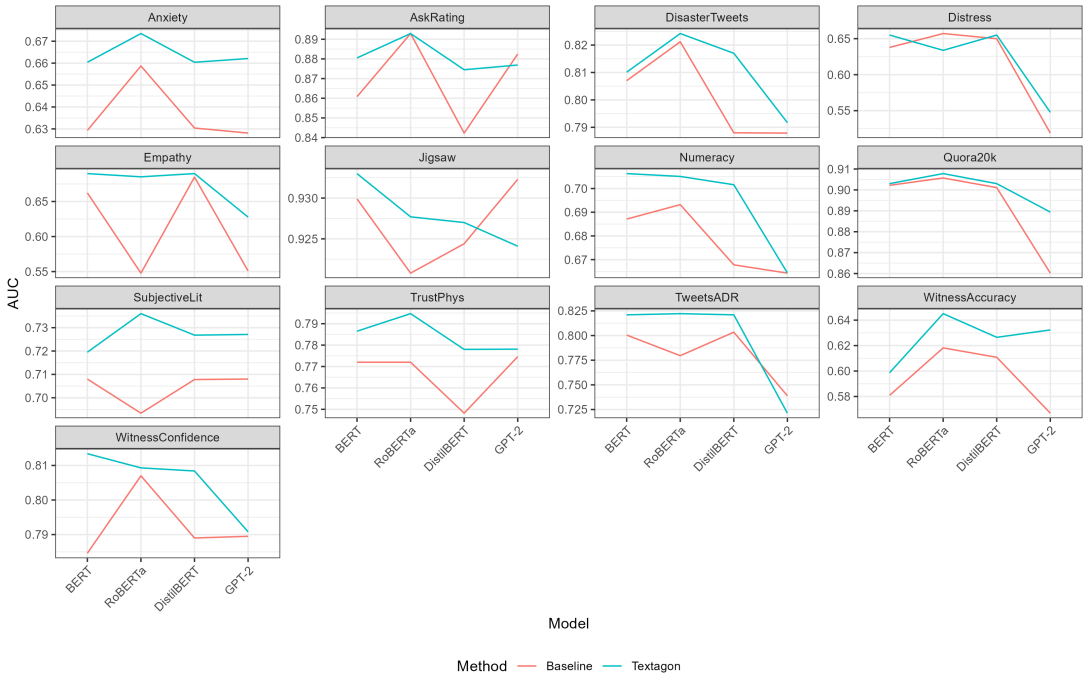


Figure 4: Comparing base PLM models with `Textagon` across benchmarking datasets. `Textagon` improves performance in 46 out of 52 task-model settings (88.5%).

Acknowledgments

This work was supported in part by the following grants from the U.S. National Science Foundation: IIS-1816504, BDS-1636933, and CCF-1629450.

References

Ahmed Abbasi, David Dobolyi, John P Lalor, Richard G Netemeyer, Kendall Smith, and Yi Yang. 2021. Constructing a psychometric testbed for fair natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Ahmed Abbasi, Stephen France, Zhu Zhang, and Hsinchun Chen. 2011. Selecting attributes for sentiment classification using feature relation networks. *IEEE Transactions on Knowledge and Data Engineering*, 23(3):447–462.

Asad Abdi, Siti Mariyam Shamsuddin, Shafaatunnur Hasan, and Jalil Piran. 2019. Deep learning-based sentiment classification of evaluative text based on multi-feature fusion. *Information Processing & Management*, 56(4):1245–1259.

C.J. Adams, Jeffrey Sorensen, Julia Elliott, Lucas Dixon, Mark McDonald, nithum, and Will Cukierski. 2017. Toxic comment classification challenge.

Faizan Ahmad, Ahmed Abbasi, Jingjing Li, David G Dobolyi, Richard G Netemeyer, Gari D Clifford, and Hsinchun Chen. 2020. A deep learning architecture for psychometric natural language processing. *ACM Transactions on Information Systems (TOIS)*, 38(1):1–29.

Israa Alghanmi, Luis Espinosa Anke, and Steven Schockaert. 2020. Combining bert with static word embeddings for categorizing social media. In *Proceedings of the sixth workshop on noisy user-generated text (w-nut 2020)*, pages 28–33.

Marialena Bevilacqua, Kezia Oketch, Ruiyang Qin, Will Stamey, Xinyuan Zhang, Yi Gan, Kai Yang, and Ahmed Abbasi. 2025. When automated assessment meets automated content generation: Examining text quality in the era of gpts. *ACM Transactions on Information Systems*, 43(2):1–36.

Danushka Bollegala. 2022. Learning meta word embeddings by unsupervised weighted concatenation of source embeddings. *arXiv preprint arXiv:2204.12386*.

Sven Buechel, Anneke Buffone, Barry Slaff, Lyle Ungar, and João Sedoc. 2018. Modeling empathy and distress in reaction to news stories. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4758–4765, Brussels, Belgium. Association for Computational Linguistics.

Nicolas Antonio Cloutier and Nathalie Japkowicz. 2023. Fine-tuned generative llm oversampling can improve performance over traditional techniques on multiclass imbalanced text classification. In *2023 IEEE International Conference on Big Data (BigData)*, pages 5181–5186. IEEE.

DataCanary, hilfialkaff, Lili Jiang, Meg Risdal, Nikhil Dandekar, and tomtung. 2017. Quora question pairs.

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*.

David G Dobolyi and Chad S Dodson. 2018. Actual vs. perceived eyewitness accuracy and confidence and the featural justification effect. *Journal of Experimental Psychology: Applied*, 24(4):543.

Hanyu Duan, Yi Yang, Ahmed Abbasi, and Kar Yan Tam. 2022. Barle: Background-aware representation learning for background shift out-of-distribution detection. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 750–764.

Justin Grimmer, Margaret E Roberts, and Brandon M Stewart. 2022. *Text as data: A new framework for machine learning and the social sciences*. Princeton University Press.

Michael Halliday and Christian Matthiessen. 2014. *An Introduction to Functional Grammar*. Routledge.

Ammar Hassan, Ahmed Abbasi, and Daniel Zeng. 2013. Twitter sentiment analysis: A bootstrap ensemble framework. In *2013 international conference on social computing*, pages 357–364. IEEE.

Addison Howard, devrishi, Phil Culliton, and Yufeng Guo. 2019. Natural language processing with disaster tweets.

Minlie Huang, Qiao Qian, and Xiaoyan Zhu. 2017. Encoding syntactic knowledge in neural networks for sentiment classification. *ACM Transactions on Information Systems (TOIS)*, 35(3):1–27.

Masahiro Kaneko and Danushka Bollegala. 2020. Autoencoding improves pre-trained word embeddings. *arXiv preprint arXiv:2010.13094*.

John P Lalor, Ahmed Abbasi, Kezia Oketch, Yi Yang, and Nicole Forsgren. 2024. Should fairness be a metric or a model? a model-based framework for assessing bias in machine learning pipelines. *ACM Transactions on Information Systems*, 42(4):1–41.

John P Lalor, Yi Yang, Kendall Smith, Nicole Forsgren, and Ahmed Abbasi. 2022. Benchmarking intersectional biases in nlp. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3598–3609.

Kyuhan Lee and Sudha Ram. 2024. Explainable deep learning for false information identification: An argumentation theory approach. *Information Systems Research*, 35(2):890–907.

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*.

Tim Loughran and Bill McDonald. 2011. When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of finance*, 66(1):35–65.

Ana Macanovic. 2022. Text mining for social science–the state and the future of computational text analysis in sociology. *Social Science Research*, 108:102784.

Sandeep Mathias and Pushpak Bhattacharyya. 2018. Asap++: Enriching the asap automated essay grading dataset with essay attribute scores. In *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*.

Saif Mohammad and Peter Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, pages 26–34.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001.

Ruiyang Qin, Dancheng Liu, Zheyu Yan, Zhaoxuan Tan, Zixuan Pan, Zhenge Jia, Meng Jiang, Ahmed Abbasi, Jinjun Xiong, and Yiyu Shi. 2024a. Empirical guidelines for deploying llms onto resource-constrained edge devices. *arXiv preprint arXiv:2406.03777*.

Ruiyang Qin, Jun Xia, Zhenge Jia, Meng Jiang, Ahmed Abbasi, Peipei Zhou, Jingtong Hu, and Yiyu Shi. 2024b. Enabling on-device large language model personalization with self-supervised data selection and synthesis. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pages 1–6.

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.

Ellen Riloff, Siddharth Patwardhan, and Janyce Wiebe. 2006. Feature subsumption for opinion analysis. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 440–448.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

John R Searle. 1969. *Speech acts: An essay in the philosophy of language*. Cambridge university press.

Hashim Sharif, Fareed Zaffar, Ahmed Abbasi, and David Zimbra. 2014. Detecting adverse drug reactions using a sentiment classification framework.

Kaveh Taghipour and Hwee Tou Ng. 2016. A neural approach to automated essay scoring. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1882–1891.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2020a. Automated concatenation of embeddings for structured prediction. *arXiv preprint arXiv:2010.05006*.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2020b. More embeddings, better sequence labelers? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3992–4006, Online. Association for Computational Linguistics.

Kai Yang, Raymond YK Lau, and Ahmed Abbasi. 2023. Getting personal: a deep learning artifact for text-based measurement of personality. *Information Systems Research*, 34(1):194–222.

Mochen Yang, Gediminas Adomavicius, Gordon Burtch, and Yuqing Ren. 2018. Mind the gap: Accounting for measurement error and misclassification in variables generated via data mining. *Information Systems Research*, 29(1):4–24.

Ruosong Yang, Jiannong Cao, Zhiyuan Wen, Youzheng Wu, and Xiaodong He. 2020. Enhancing automated essay scoring performance via fine-tuning pre-trained language models with combination of regression and ranking. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1560–1569.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 180–189.

David Zimbra, Ahmed Abbasi, Daniel Zeng, and Hsinchun Chen. 2018. The state-of-the-art in twitter sentiment analysis: A review and benchmark evaluation. *ACM Transactions on Management Information Systems (TMIS)*, 9(2):1–29.

A Token GBS

For parallel representations $R = \{r_1, r_2, ..., r_m\}$, the initial weight of a 1-gram feature f_{ix} from representation r_x is given by:

$$w(f_{ix}) = \max_{c_a, c_b} \left(p(f_{ix}|c_a) \log \frac{p(f_{ix}|c_a)}{p(f_{ix}|c_b)} \right) + s(f_{ix}) \quad (2)$$

where the first term is the log-likelihood ratio that measures discriminatory potential and $s(f_{ix})$ captures the semantic orientation:

$$s(f_{ix}) = \frac{1}{dw} \sum_{y=1}^{d} \sum_{q=1}^{w} [pos(f_{ix}, q) - neg(f_{ix}, q)] \quad (3)$$

This ensures the differentiation of features with opposing orientations. For subsumption within r_x, each 1-gram f_{ix} with $w(f_{ix}) > 0$ is compared to every other 1-gram. If the classification of f_{ix} matches that of another 1-gram, given by:

$$c(f_{ix}) = \underset{c_a, c_b}{\operatorname{argmax}} \left(p(f_{ix}|c_a) \log \frac{p(f_{ix}|c_a)}{p(f_{ix}|c_b)} \right) + s(f_{ix}) \quad (4)$$

subsumption decisions are made based on a threshold t:

$$w(f_{ix}) = \begin{cases} 0 & \text{if } w(f_{ix}) \leq w(f_{ux}) + t \\ w(f_{ix}) & \text{otherwise} \end{cases} . \quad (5)$$

For each pair of representations r_x and r_z, 1-gram features are selected into subsets A and B. Using k-Means clustering with $k = 2$, the result is $G = \{g_1, g_2\}$ clusters. A link between r_x and r_z is based on entropy reduction:

$$L(r_x, r_z) = \begin{cases} 1 & \text{if } \frac{H(G|r)}{H(G)} \leq l \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

The entropy across clusters is denoted as $H(G)$. The entropy $H(G|r)$ considering a specific representation r (either r_x or r_z) is defined as:

$$H(G|r) = -\sum_{r \in \{r_x, r_z\}} P(r) \sum_{\delta \in G} P(\delta|r) \log_2 P(\delta|r). \quad (7)$$

After establishing links, subsumption between r_x and r_z is performed in a similar manner, but bidirectionally.

Here, correlated 1-gram features between linked representations r_x and r_z are addressed. For every pair of representations r_x and r_z with $L(r_x, r_z) = 1$, any remaining feature f_{ijx} in r_x with weight $w(f_{ijx}) > 0$ is compared against all other remaining features f_{uvz} in r_z with weight greater than 0, given $j = v$. If the correlation between f_{ijx} and f_{uvz} surpasses the threshold p, then $w(f_{ijx})$ is set to 0.

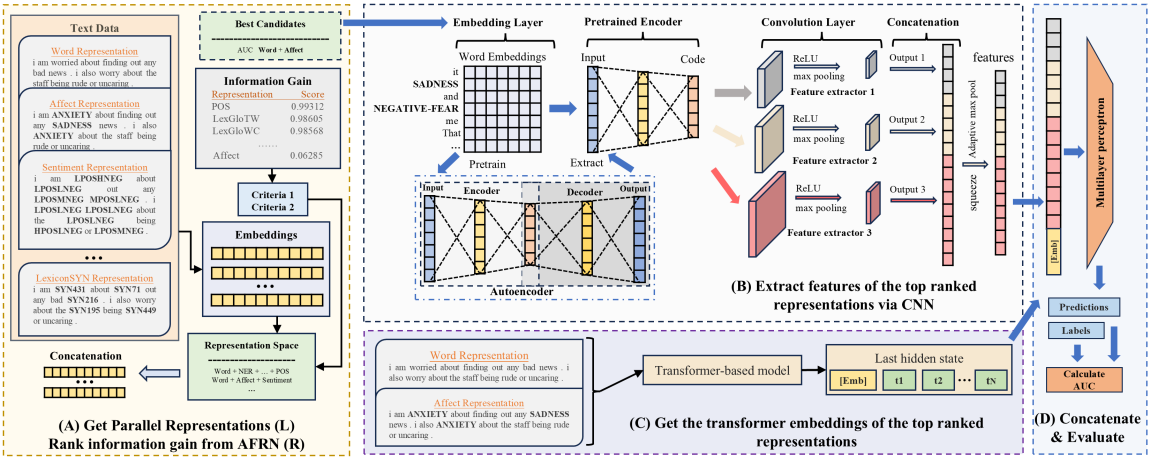


Figure 5: Example of applying `Textagon` to a classification pipeline.

B Classification Details

Figure 5 shows the pipeline for our classification example. We first extract and score representations using `Textagon`. We then extract features from the parallel representations in a high-dimensional space. The third component uses the extracted features either as standalone features or concatenated with embedding outputs of a transformer-based model as input to a downstream prediction model. This component also evaluates the predictions and returns the evaluation to the first component for assessing representation combinations.

B.1 Selecting the Representation Space

Having generated representations and calculated S_R, the next step is to decide which representations to include alongside the word representations. We rely on two selection criteria: treating S_R as information gain and a search space limiting heuristic. We first sort the representations by S_R and select the top n based on S_R. We then search through all three-way representation combinations. This reduces the search complexity from $O(2^n)$ to $O(n + \binom{n}{3}) = O(n^3)$.

B.2 Representation Controller

Having identified the pool of candidate representation combinations, the representation controller iterates over the representation space. Given a combination, the representation controller takes the embedding of each contained representation from the text data and concatenates all embeddings in parallel (Figure 5, panel B). The concatenation is taken as the input data for the end-to-end, CNN feature extraction model. We first process each representation into embeddable data. We then convert each representation text data into aligned, word-index-based numerical data.

B.2.1 Optimal Search of Representations

As discussed, we do not use a greedy algorithm initially because the initial representation space without any constraints is too large to be efficiently searched. When we contain the upper bound of the representation space complexity to $O(n^3)$, we can use a greedy search to identify the best combination of representations.

We evaluate each representation individually and store the best AUC. Then, we perform a greedy search to find the best combination of three representations. We iterate through all possible combinations of three different representations r_1, r_2, r_3 from R, train the model, and update the best AUC and the corresponding combination if a better AUC is found.

B.2.2 End-to-end Feature Extraction

The input data, which the representation controller generates, contains features not only within but also across representation embeddings. Such high-dimensional features can be captured by a 2D CNN. For the embedded data, it will be used to pretrain an autoencoder (Kaneko and Bollegala, 2020), whose parameters and weights will be saved for future usage. We structure the autoencoder as three convolutional layers; each layer is followed by a ReLU layer. We reduce dimensions smoothly in the autoencoder, via the factors of $\frac{4}{5}$, $\frac{3}{4}$, and $\frac{2}{3}$. Then, the encoder is used to construct a CNN model, along with three feature extractors of different sizes, whose output is concate-

nated to formalize the final output. The three feature extractors have the same structure; each contains one 2D convolutional layer (Conv2d), one ReLU layer (ReLU), and one 2D max pooling layer (MaxPool2d). The kernel size of MaxPool2d corresponds with the kernel size of Conv2d. For Conv2d, each of their kernels is resized by factors of $\frac{1}{6}$, $\frac{1}{4}$, and $\frac{1}{3}$.

B.3 Concatenation Features and Finalize Output

The three feature extractors can go through the input data in different views and eventually capture features in different dimensions. To keep all extracted features, we concatenate them in sequence, and then apply an adaptive pooling layer (AdaptiveMaxPool2d) to get the final output representation (Figure 5, panel D).