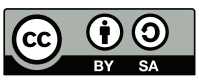
Text Analysis and Retrieval 2023

Course Project Reports

University of Zagreb
Faculty of Electrical Engineering and Computing



Publisher:

University of Zagreb, Faculty of Electrical Engineering and Computing

Organizers & editors:

Josip Jukić
Jan Šnajder

Reviewers:

Filip Karlo Došilović
Nina Drobac
David Dukić
Tin Ferković
Fran Jelenić
Josip Jukić
Laura Majer
Ivan Martinović
Luka Pavlović
Marko Rajnović
Jan Šnajder
Ivan Stresec
Matea Vasilj
Janko Vidaković

ISBN 978-953-184-293-8

Course website:

https://www.fer.unizg.hr/en/course/taar

Powered by:

Text Analysis and Knowledge Engineering Lab
University of Zagreb
Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
https://takelab.fer.hr

Preface

This is the tenth installment in our series of student project reports from the Text Analysis and Retrieval (TAR) course, offered at the esteemed Faculty of Electrical Engineering and Computing (FER), University of Zagreb. TAR is a robust course imparting fundamental knowledge in natural language processing and information retrieval, with a keen emphasis on hands-on applications and industry-recognized best practices.

The course is meticulously crafted to imbue students with both theoretical understanding and practical prowess in data science and machine learning, specializing in the intricacies of text data analysis. The culmination of the course is a practical project, where students are challenged to put the theories learned into practice to tackle real-world issues.

This edition showcases the outcomes of 16 distinct projects, collectively accomplished by 46 diligent students. A majority of the topics were inspired by recent seminars and collective tasks from platforms such as SemEval and CLEF. The scope of projects spanned across sentiment analysis, topic modeling, named entity recognition, to text classification. Several students incorporated deep learning models in their projects, underscoring the continued adoption and growth of this technique in natural language processing.

We strive to ensure that our students are armed with the requisite skills to excel in the rapidly evolving field of data science and machine learning. To this end, the course underscores the vital role of scientific research and academic writing. Students are motivated to format their project reports akin to research papers, thus providing a platform to present their findings and simultaneously hone their scientific writing skills.

We are delighted to have continued the TAR course in the face-to-face format this year, and we take immense pride in the exceptional dedication and zeal exhibited by our students. We express our gratitude for their relentless efforts and are confident that they have acquired both practical and theoretical expertise necessary to thrive in their upcoming careers.

Josip Jukić and Jan Šnajder

Contents

Beyond Turing: A Comparative Analysis of Approaches for Detecting Machine-Generated Text
Muhammad Farid Adilazuarda, Nikolaos Nektarios Arkoulis, Oleksii Chumakov 1

Unleashing Context: Enhancing Tweet Classification with Contextual Insights
Andrija Banić, Marko Jurić, Dario Pavlović 6

LOL (or Not?): Evaluating the Impact of External Context on Humor Assessment in Micro-Edited News Headlines
Marko Barbir, Marin Puharić, Kim Staničić 11

Author Style Change Detection
Luka Družijanić, Tomislav Ćosić, Luka Brečić 15

How Lazy Common Sense Can Be: ComVE Challenge
Chloé Duault, Ellyn Lafontaine, Mallory Taffonneau 19

Exploring the OCEAN Depths: a Look at Personality Trait Classification on Essays
Mateo Hrelja, Dominik Jurinčić, Ivan Linardić 23

Beyond Duplicates: Unleashing Transitivity in Quora Data Augmentation
Renato Jurišić, Mihael Miličević, Josip Srzić 28

Traditional vs. Modern: Comparing Approaches for Offensive Language Detection
Maria Krajči, Ana Vladić, Bjanka Vrljić 33

Named Entity Recognition for Efficient Clinical Concept Mapping in Patient Notes with DeBERTa, RoBERTa, and BioClinical BERT
Tomislav Krog, Tihomir Pavić, Matija Sever 38

The Depth of Deception: Shallow, Deep, and Hybrid Learning Approaches to Unmasking Fake Text
Lea Krsnik, Mislav Perić, Leon Zrnić 42

From Pen to Persona: Essays Give Insight to Who You Are on the Inside
Iva Marić, Lea Grebenar, Nikola Petrović 46

It's No Laughing Matter: Exploring the Quality of the Humicroedit Dataset
Gabriel Marinković 50

A Squeeze of LIME, a Pinch of SHAP: Adding the Flavor of Explainability to Sentence-Level Commonsense Validation
Mirta Moslavac, Roko Grbelja, Matej Ciglenečki 55

Clustering Paragraphs by Author Using Binary Classifiers and Pairwise Probabilities for Being in Same Cluster
Josip Pardon, Marko Damjanić, Valentin Vuk 60

Stress Analysis in Social Media
Jakov Samardžija, Antonio Babić, Mathieu Jehanno 65

Transformers, do we really need them for detecting stress?
Sven Šćekić, Marko Kuzmić, Lovro Bučar 70

Beyond Turing: A Comparative Analysis of Approaches for Detecting Machine-Generated Text

Muhammad Farid Adilazuarda, Nikolaos Nektarios Arkoulis, Oleksii Chumakov

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{faridlazuarda, arkoulis.nick, AlexeyChumakov2001}@gmail.com

Abstract

Significant progress has been made on text generation by pre-trained language models (PLMs), yet distinguishing between human and machine-generated text poses an escalating challenge. This paper offers an in-depth evaluation of three distinct methods used to address this task: traditional shallow learning, Language Model (LM) fine-tuning, and Multilingual Model fine-tuning. These approaches are rigorously tested on a wide range of machine-generated texts, providing a benchmark of their competence in distinguishing between human-authored and machine-authored linguistic constructs. The results reveal considerable differences in performance across methods, thus emphasizing the continued need for advancement in this crucial area of NLP. This study offers valuable insights and paves the way for future research aimed at creating robust and highly discriminative models.

1. Introduction

The drive to discern between human and machine-generated text has been a long-standing pursuit, tracing its origins back to Turing's famous 'Turing Test', which explore a machine's ability to imitate human-like intelligence. With the vast and rapid development of advanced PLMs, the capacity to generate increasingly human-like text has grown, blurring the lines of detectability and bringing this research back into sharp focus.

Addressing this complexity, this paper explores two specific tasks: 1) the differentiation between human and machine-generated text, and 2) the identification of the specific language model that generated a given text. Our exploration extends beyond the traditional shallow learning techniques, exploring into the more robust methodologies of Language Model (LM) fine-tuning and Multilingual Model fine-tuning (Winata et al., 2021). These techniques enable PLMs to specialize in the detection and categorization of machine-generated texts. They adapt pre-existing knowledge to the task at hand, effectively manage language-specific biases, and improve classification performance.

Through an exhaustive examination of a diverse set of machine-generated texts, we deliver insights into the strengths and weaknesses of these methodologies. We illuminate the ongoing necessity for advancement in NLP and the critical importance of developing techniques that can keep pace with the progress of PLMs research. Our paper offers the following contributions:

1. An exhaustive evaluation of the capabilities of PLMs in categorizing machine-generated texts.
2. An investigation into the effectiveness of employing multilingual techniques to mitigate language-specific biases in the detection of machine-generated text.
3. The application of a few-shot multilingual evaluation strategy to measure the adaptability of models in resource-limited scenarios.

2. Related Works

This study's related work falls into three main categories: machine-generated text detection, identification of specific PLMs, and advancements in language model fine-tuning.

Machine-generated Text Detection: Distinguishing human from machine-generated text has become an intricate challenge with recent advancements in language modeling. Prior research (Schwartz et al., 2018; Ippolito et al., 2020) has explored nuances separating human and machine compositions. Our work builds on these explorations by assessing various methodologies for this task.

Language Models Identification: Some studies (Radford et al., 2019) attempt to identify the specific language model generating a text. These efforts, however, are still in growing stages and often rely on model-specific features. Our work evaluates various methods' efficacy for this task, focusing on robustness across a spectrum of PLMs.

Language Model Fine-tuning Advances: Language Model fine-tuning (Howard and Ruder, 2018) and Multilingual Model fine-tuning (Conneau et al., 2020) represent progress in language model customization. They enable model specialization in machine-generated text detection and classification and address language-specific biases, thereby enhancing classification accuracy across diverse languages.

This study intertwines these three research avenues, providing a thorough evaluation of the mentioned methodologies in machine-generated text detection and classification, underscoring the necessity for continuous progress in alignment with the evolving proficiency of PLMs.

3. Methods

3.1. Shallow Learning

We conducted an evaluation of two distinct shallow learning models, specifically Logistic Regression and XGBoost, utilizing Fasttext word embeddings that were trained on our preprocessed training set. Prior to the training process, we implemented a fundamental preprocessing step, which involved the removal of non-ASCII and special characters to refine our dataset and enhance the quality of our results. To

Text Analysis and Retrieval 2023: Course Project Reports (TAR 2023), pages 1–5, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2023.

enrich the training source of the models as showed in Table 1, we propose embedding on four lexical complexity measures aimed at quantifying different aspects of a text:

Average Word Length (AWL): This metric reflects the lexical sophistication of a text, with longer average word lengths potentially suggesting more complex language use. Let $W = \{w_1, w_2, ..., w_n\}$ represent the set of word tokens in the text. The AWL is given by:

$$AWL = \frac{1}{n} \sum_{i=1}^{n} |w_i|$$

Average Sentence Length (ASL): This provides a measure of syntactic complexity, with longer sentences often requiring more complex syntactic structures.Let $S = \{s_1, s_2, ..., s_m\}$ represent the set of sentence tokens in the text. The ASL is defined as:

$$ASL = \frac{1}{m} \sum_{j=1}^{m} |s_j|$$

Vocabulary Richness (VR): This ratio of unique words to the total number of words is a measure of lexical diversity, which can also be indicative of language proficiency and style .If UW represents the set of unique words in the text, the VR is calculated as:

$$VR = \frac{|UW|}{n}$$

Repetition Rate (RR): The ratio of words occurring more than once to the total number of words, indicative of the redundancy of a text. If RW represents the set of words that occur more than once, the RR is computed as:

$$RR = \frac{|RW|}{n}$$

To illustrate our feature engineering process, Table 1 presents a snapshot of our dataset after the application of our feature calculations. The table showcases a selection of texts and their corresponding labels (0 representing machine-generated text and 1 indicating human-generated text), as well as a range of text features derived from these texts. These include Average Word Length **(AWL)**, Average Sentence Length **(ASL)**, Vocabulary Richness **(VR)**, and Repetition Rate **(RR)**. By computing these features, we aimed to capture distinct textual characteristics that could aid our models in accurately discerning between human and machine-generated text.

Table 1: Text feature calculation. Label, AWL: Avg. Word Length, ASL: Avg. Sent. Length, VR: Vocab. Richness, RR: Repetition Rate

Text	Label	AWL	ASL	VR	RR
you need to...	generated	3.12	49.50	0.96	0.04
The Comm...	generated	4.92	62.56	0.69	0.09
I pass my...	human	3.55	90.00	0.90	0.10

3.2. Language Model Finetuning

In this study, we employed multiple models: XLM-RoBERTa, mBERT, DeBERTa-v3, BERT-tiny, DistilBERT, RoBERTa-*Detector*, and ChatGPT-*Detector*. The models were fine-tuned on single and both languages simultaneously using multilingual training (Bai et al., 2021). We found that this setup provides superior performance compared to training separate models for each language.

During evaluation, we employed the F1 score for each class along with the overall accuracy as our primary metrics, given the F1 score's ability to provide a balanced measure in instances of class imbalance. Further bolstering our evaluation approach, we incorporated a Few-Shot learning evaluation to assess our models' capacity to learn effectively from a limited set of examples. This involved using varying seed quantities, specifically `[200, 400, 600, 800, 1000]` instances, applied across both English and Spanish languages, thus ensuring our models' robustness and their practical applicability in real-world scenarios.

4. Experiments

4.1. Dataset

Our experiments utilize two multi-class classification datasets, namely Subtask 1 and Subtask 2, as referenced from the Autextification study (Ángel González et al., 2023). Subtask 1 is a document-level dataset composed of **65,907** samples, designed to differentiate between human and machine-generated text. Each sample is assigned one of two class labels: 'generated' or 'human'. Subtask 2, on the other hand, serves as a Model Attribution dataset consisting of **44,351** samples. This dataset includes six different labels - A, B, C, D, E, and F - representing distinct models of text generation. A detailed overview of the statistics related to both Subtask 1 and Subtask 2 datasets is provided in Table 2.

Table 2: Statistic of the datasets

Language	Subtask	|Train|	|Valid|	|Test|	#Class
English	Subtask 1	27,414	3,046	3,385	2
	Subtask 2	18,156	2,018	2,242	6
Spanish	Subtask 1	25,969	2,886	3,207	2
	Subtask 2	17,766	1,975	2,194	6

4.2. Training and Evaluation Setup

Our approach to fine-tuning PLMs remained consistent across all models under consideration. We utilized HuggingFace's Transformers library[1], which provides both pre-trained models and scripts for fine-tuning. Utilizing a multi-GPU setup, we employed the AdamW optimizer (Loshchilov and Hutter, 2019), configured with a learning rate of `1e-6` and a batch size of 64. To prevent overfitting, we implemented early stopping within 3 epochs patience. The models were trained across a total of 10 epochs.

Multilingual fine-tuning, a key element of our methodology, involves a two-stage approach aimed at optimizing

[1]https://huggingface.co/

the model's performance across multiple languages, specifically English and Spanish. The first stage, pre-training, entails training the model on a large corpus of multilingual data, aimed at learning a universal language representation that captures shared linguistic structures across diverse languages. At this juncture, the model acquires a generalized understanding of linguistic patterns and structures, although it does not specialize in any single language.

The subsequent stage, fine-tuning, significantly differs from pre-training. Contrary to the independent language-specific approach, our method involves fine-tuning the model on a combined dataset comprising both English and Spanish data. This stage aims to further refine the broad language comprehension obtained during pre-training, with an emphasis on enhancing the model's versatility across the two languages. By blending the data in this manner, the model is encouraged to consolidate its shared language comprehension, leveraging the commonalities between the two languages while also gaining exposure to their unique idioms, syntactic patterns, and other nuances. This multilingual fine-tuning approach allows our model to achieve a balanced integration of generalized multilingual comprehension and the capacity to handle the specificities of the individual languages, English and Spanish, in the mixed dataset.

5. Results and Discussion

5.1. Few-Shot Learning

To gauge the performance of our models in few-shot learning scenarios, we systematically increased the count of data points for fine-tuning in increments of 100, ranging from 100 to 500 data points for each language, resulting in a total of 200 to 1000 samples per scenario. The results of the few-shot learning experiments are depicted in Fig. 1. This was computed using the equation:

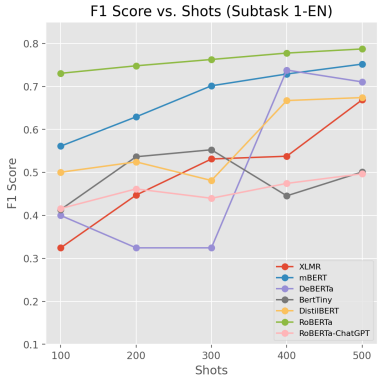
$$L_{\text{few-shot}} = \frac{1}{n} \sum_{i=1}^{n} L_i$$

where n is the total number of instances, and L_i is the loss calculated for each individual data point. The loss function (L) provides a measure of the model's performance in this few-shot scenario.

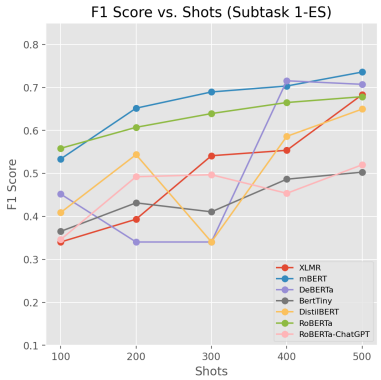
5.2. Distinguishing Capability

From the few-shot learning experiments, the models' performance varied significantly in distinguishing between human and machine-generated text. In the default evaluation, multilingually-finetuned mBERT outperformed the other models in English, and single-language finetuned mBERT exhibited the highest score in Spanish. However, In the few-shot experiment setting, the RoBERTa-*Detector* demonstrated the most robust distinguishing capability, scoring up to 0.787 with 1000 samples.

When comparing these results, we can observe that mBERT maintains strong performance in both the few-shot learning experiments and the single language experiments. It suggests that mBERT could provide a reliable choice across different tasks and experimental settings in both Subtasks.



(a) English



(b) Spanish

Figure 1: Subtask 1 Evaluation on Few-Shot Learning

5.3. Model Generation Capability

Figure 2 illustrates the error rates of the evaluated models, with ***model E*** exhibiting the highest error rate. This indicates that model E demonstrates superior generation capabilities, enabling it to effectively deceive the detector model. On the other hand, ***model F*** exhibits the lowest error rate, indicating a weaker ability to deceive the detector model. However, it is worth noting that these observations may be influenced by the similarity bias in architecture between the text detector and text generator models employed.

5.4. Comparative Analysis of Model Performances

Our analysis from experiments in Table 3 reveals variations in the performance of the models for both tasks: differentiating human and machine-generated text, and identifying the specific language model that generated the given text. For the first task, mBERT emerges as the top performer with English and Spanish F1 scores of 85.18% and 83.25% respectively, in the fine-tuning setup. This performance is closely followed by DistilBERT's English F1 score of 84.97% and Spanish score of 78.77%. In the multilingual fine-tuning configuration, DistilBERT edges out with an English F1 score of 85.22%, but mBERT retains its high Spanish performance with an F1 score of 82.99%.

In the second task, mBERT continues to excel, achieving F1 scores of 44.82% and 45.16% for English and Spanish respectively in the fine-tuning setup. It improves further in the multilingual fine-tuning setup with English and

Table 3: F1 Score for Various Models in English and Spanish for Subtask 1 and 2. **Bold** and underline denote first and second best, respectively

Model	Subtask 1		Subtask 2	
	English-F1	Spanish-F1	English-F1	Spanish-F1
Shallow Learning + Feat. Engineering				
Logistic Regression	65.67%	63.87%	38.39%	42.99%
XGBoost	71.52%	71.53%	38.47%	41.08%
Finetuning				
XLM-RoBERTa	78.80%	76.56%	27.14%	30.66%
mBERT	85.18%	**83.25%**	44.82%	45.16%
DeBERTa-V3	81.52%	72.58%	43.93%	28.28%
TinyBERT	63.75%	57.83%	15.38%	13.02%
DistilBERT	84.97%	78.77%	41.53%	35.61%
RoBERTa-*Detector*	84.01%	75.18%	34.13%	22.10%
ChatGPT-*Detector*	68.33%	64.64%	23.84%	25.45%
Multilingual Finetuning				
mBERT	84.80%	82.99%	**49.24%**	**47.28%**
DistilBERT	**85.22%**	80.49%	41.64%	35.59%

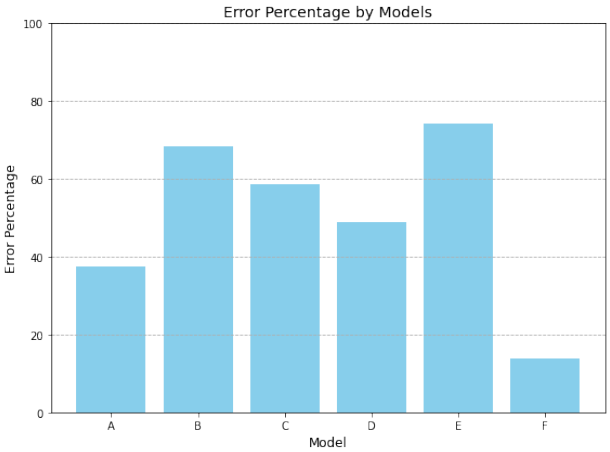


Figure 2: Comparison of Model Error Percentages. The bar chart illustrates the error percentages of multiple models, indicating the proportion of misclassified instances in the dataset. The models, labeled as A, B, C, D, E, and F, were used for prediction. The error rate was computed using mBERT with multilingual fine-tuning.

Spanish scores of 49.24% and 47.28%. However, models such as XLM-RoBERTa and TinyBERT show substantial performance gaps between the tasks. For example, XLM-RoBERTa excels in the first task with English and Spanish F1 scores of 78.8% and 76.56%, but struggles with the second task, with F1 scores dropping to 27.14% and 30.66%. Similarly, TinyBERT shows a notable performance drop in the second task.

The performance disparity suggests that the two tasks require distinct skills: the first relies on detecting patterns unique to machine-generated text, while the second demands recognition of nuanced characteristics of specific models. In conclusion, mBERT demonstrates a consistent and robust performance across both tasks. However, the findings also underscore a need for specialized models or strategies for each task, paving the way for future work in the design and fine-tuning of models for these tasks.

6. Conclusion

This study performed an exhaustive investigation into three distinct methodologies: traditional shallow learning, Language Model fine-tuning, and Multilingual Model fine-tuning, for detecting machine-generated text and identifying the specific language model that generated the text. The analysis revealed significant variations in performance across these techniques, suggesting the need for continued improvements in this evolving field.

Our findings showed that mBERT emerged as a consistently robust performer across different tasks and experimental settings, making it a potentially reliable choice for such tasks. However, other models like XLM-RoBERTa and TinyBERT showed a noticeable performance gap between the tasks, indicating that these two tasks might require different skillsets. This research provides valuable insights into the performance of these methodologies on a diverse set of machine-generated texts. It also highlights the critical importance of developing specialized models or strategies for each task.

Acknowledgements

We express our profound gratitude to our mentors, Professor Jan Šnajder and Teaching Assistant Josip Jukić, for their invaluable guidance, constructive feedback, and unwavering support throughout the duration of this project. Their expertise and dedication have significantly contributed to the advancement of our research and understanding.

References

Junwen Bai, Bo Li, Yu Zhang, Ankur Bapna, Nikhil Siddhartha, Khe Chai Sim, and Tara N. Sainath. 2021. Joint unsupervised and supervised training for multilingual asr.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco

Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 8:264–282.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2020. Discriminating between human-produced and machine-generated text: A survey. *arXiv preprint arXiv:2012.03358*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

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

Roy Schwartz, Oren Tsur, Ari Rappoport, and Eyal Shnarch. 2018. The effect of different writing tasks on linguistic style: A case study of the roc story cloze task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1806–1815.

Genta Indra Winata, Andrea Madotto, Zhaojiang Lin, Rosanne Liu, Jason Yosinski, and Pascale Fung. 2021. Language models are few-shot multilingual learners.

José Ángel González, Areg Sarvazyan, Marc Franco, Francisco Manuel Rangel, María Alberta Chulvi, and Paolo Rosso. 2023. Autextification, March.

Unleashing Context: Enhancing Tweet Classification with Contextual Insights

Andrija Banić, Marko Jurić, Dario Pavlović

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{andrija.banic,marko.juric,dario.pavlovic}@fer.hr

Abstract

In today's media-dominated world, the widespread flow of information increases the risk of encountering dangerous fake news. This alarming issue has attracted the attention of many researchers. However, there is a lack of labeled data available, and the existing labeled data is imbalanced, which poses a major challenge in supervised learning. With numerous research papers focusing on various neural language models and manually created features for classification, this study aims to investigate how the context of an individual tweet affects feature engineering and the interpretability of the results. Specifically, we focus on the task of stance classification, which involves determining the position or attitude expressed in a tweet towards a particular topic. In our investigation, we compare the performance of different feature engineering approaches when considering different levels of context. We explore three scenarios: (1) no context considered, (2) only the context of the tweet being replied to (referred to as „parent" context), and (3) both the replies and the tweet being replied to (referred to as „children" and „parent" context if we think of tweets represented as a tree structure). By examining the impact of context on classification accuracy and incorporating the concept of stance, we aim to understand the potential advantages of incorporating contextual information in tweet classification.

1. Introduction

Rumours have been a part of our lives since the day we were born, representing unofficial and intriguing stories or news that quickly circulate from person to person. In the modern era, these rumours have evolved from harmless gossip about friends and colleagues to potentially dangerous and concerning misinformation. With millions of tweets being posted daily, it is crucial to identify and distinguish those tweets that spread harmful rumours, as they are often convincing and challenging to detect manually. Therefore, it is essential to develop fast, precise, and automated methods to tackle this issue. Our paper concentrates on stance classification, which aims to determine the attitude of the author of a tweet towards a specific target in Twitter conversations. We can visualize these conversations as trees, where the root represents the source post initiating the topic, and the following replies form a thread that branches out. The main goal of stance classificaton is to classify the tweet into one of the four categories which are support, deny, query or comment (**SDQC**).

The objective of our task is to determine the stance towards an underlying rumour, which is not explicitly provided but can often be inferred from the source post or the root of the discussion thread. While several papers have addressed this challenge, as discussed in Section 2., our approach focuses on using simpler machine learning models, such as Random Forest Classifier (RF), Gradient Boosting Classifier (GBC), and Stochastic Gradient Descent Classifier (SGDC). Instead of relying on complex models like GPT, BERT or LSTM-based language models, our emphasis is on feature engineering, described in Subsection 4.2. and, more importantly, context shown in Section 5. By context, we mean incorporating the stance of children or parent tweets, as well as the word embedding difference between the tweet we are classifying and its children or parent tweet. Research has shown that the stance of replies can be highly indicative of the post being classified (Aggarwal and Aker, 2019). Hence, we conducted experiments considering three scenarios:

- No context: We classify the tweet using features extracted solely from the tweet text itself.
- Parent context: We classify the tweet using features extracted from the tweet text itself, along with the word embedding difference between the tweet and its parent, incorporating the stance of the parent as an additional feature.
- Parent and children context: Similar to the parent context, we include the children of the tweet (their stance and word embedding difference) being classified.

In this paper, we aim to present our findings and insights gained from these experiments. Our focus lies in leveraging contextual information and feature engineering rather than relying on more complex approaches. The following sections provide a detailed explanation of our methodology, experimental setup, and results, shedding light on the effectiveness of our approach in tackling the challenge of rumour stance classification.

2. Related Work

The rise of social media and social networking platforms, such as Facebook and Twitter, has led to an explosion of user-generated content, with millions of posts and tweets being shared daily. However, until the 2017 RumourEval competition, little attention was given to the veracity of these tweets. The competition aimed to determine the veracity and support for rumors, marking a turning point in the field (Derczynski et al., 2017). Two years later, another RumourEval competition took place, attracting a significantly larger number of participants. Although the results demonstrated improvements compared to the previous competition, with some teams achieving an F1 score

Text Analysis and Retrieval 2023: Course Project Reports (TAR 2023), pages 6–10, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2023.

as high as 0.6187 in the task of rumour stance classification, the top three teams employed different techniques and models, approaching the problem from multiple angles (Gorrell et al., 2019). Of particular interest is the team that achieved the best performance in stance classification by utilizing a model that has recently revolutionized our lives - the GPT (Generative Pre-trained Transformer) model. This approach involved incorporating feature extraction with the GPT model (Yang et al., 2019). It is worth noting that the most common approach among participants was to create hand-crafted features that would effectively capture the semantic meaning of the tweet (Aker et al., 2017). Conversely, the second-ranked team decided to take a different approach, employing the BERT (Bidirectional Encoder Representations from Transformers) architecture. BERT was a groundbreaking development at the time, as it was introduced in 2018, following the introduction of transformers in 2017 (Devlin et al., 2018). Remarkably, recent studies have revealed that pre-trained language models surpass the performance of LSTM (Long Short-Term Memory)-based language models, which had long been considered the gold standard for stance classification (B. Bharathi and Balaji, 2020).

Expanding our investigation to sentiment analysis, we encounter another difficulty. As indicated by the title of a recent study by (Kenyon-Dean et al., 2018) implies, sentiment analysis is a complex and challenging task in its own right. However, it has been demonstrated that sentiment features can be useful for stance classification, although they are not sufficient on their own (Sobhani et al., 2016). A key distinguishing factor between sentiment analysis and stance classification lies in the consideration of the text's surrounding context. In sentiment analysis, the focus is primarily on analyzing the sentiment or emotional tone expressed within an individual text, without incorporating additional contextual information. This approach seeks to determine whether the sentiment conveyed is positive, negative, or neutral. On the other hand, stance classification involves determining the position or attitude expressed in a text towards a specific topic. Unlike sentiment analysis, stance classification takes into account the surrounding tweets or the broader context in which the text is situated. By considering the tweets preceding or following a particular text, we can gain insights into the stance of the author, such as whether they support, oppose, or are neutral towards the discussed topic.

In our paper, we explore the use of a Polarity Sentiment Analysis feature, which assesses the sentiment polarity of individual tweets, for enhancing stance classification. However, we acknowledge that relying solely on sentiment features is not sufficient for accurate stance classification. Interestingly, in the RumourEval competition, it was observed that the majority of models submitted primarily relied on the stance of the replies while neglecting the parent tweet as a source of context. This paper aims to shed light on the distinction between using no context at all for tweet stance classification, employing only the context of the „parents“ , and finally, using both the „children“ and „parent“ context. By exploring the role of context in determining classification accuracy, we aim to better understand the factors that contribute to effectively categorizing tweets. Our objective is to gain insights into how considering the surrounding information can enhance the accuracy of tweet classification.

3. Dataset Analysis

In our study, we utilized the RumourEval2019 dataset as the primary data source. This dataset offered a valuable resource for our research objectives. It comprised a collection of tweets and Reddit posts, with the Reddit posts modified to match the format of tweet posts. Although the Reddit data was included, it was not used in this paper. The inclusion of both Twitter and Reddit data was particularly advantageous due to the substantial user base, extensive posting activity, and the wealth of information available on these platforms. The dataset was curated by gathering posts from Twitter during various crisis events, including occurrences such as natural disasters or public emergencies. These crisis events serve as topics within the dataset, and each topic consists of a source tweet accompanied by a set of reply tweets. The organization of these tweets follows a tree-like structure, which captures the conversational relationships and hierarchy between the source tweet and its corresponding replies. To ensure the comprehensiveness of the dataset, the tweets were labeled using crowdsourcing platforms. These labels assign one of four categories to each tweet: *supporting*, *denying*, *query*, or *commenting*. This labeling process provides valuable ground truth information that enables the training and evaluation of models for rumor detection and veracity prediction.

- Support: Tweets categorized as „Support“ indicate that the author expresses agreement, endorsement, or validation towards the target under discussion. They share a favorable stance and indicate support for the viewpoint related to the topic.
- Deny: Tweets classified as „Deny“ reflect a contradictory stance or rejection of the target. Authors in this category express disagreement, disbelief, or negation towards the viewpoint associated with the topic of discussion.
- Query: Tweets falling into the „Query“ category signify curiosity, doubt, or a request for further information or clarification about the target. Authors seek more details, explanations, or evidence related to the topic, indicating a neutral or uncertain stance.
- Comment: The „Comment“ category encompasses tweets that do not clearly align with the previously mentioned categories. Tweets in this category serve as general remarks, observations, or opinions on the topic without explicitly expressing support, denial, or inquiry. They might provide additional context, personal experiences, or commentary related to the target.

As depicted in Figure 1, a notable characteristic of the dataset is the significant label imbalance present among the different categories. This label imbalance introduces a challenge due to the disproportionate representation of the

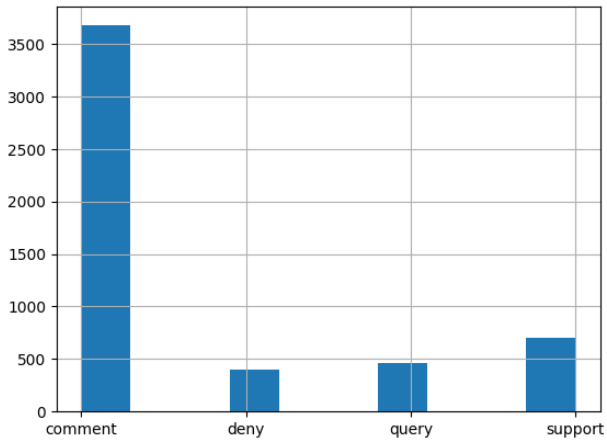


Figure 1: The distribution of labels for the source tweet reveals a significant disparity, with a substantial majority of data being labeled as „comment“. This disproportionate distribution poses a challenge as it can lead to models becoming biased towards predicting the majority label.

comment label, which greatly outnumbers the other labels. Consequently, this label imbalance can potentially lead to biases in the prediction of tweets, favoring the classification of tweets as *comments* due to their prevalence in the dataset. The dataset itself is also relatively small in size, which further impacts label imbalance. With a limited number of instances available for each label category, the model's ability to learn and generalize effectively can be compromised, particularly for the minority classes.

4. Experimental Setup

The setup for the experiment consists of data selection, preprocessing, feature extraction and choosing the optimal models for the analysis. By selecting relevant data and preprocessing the text, the aim is to extract meaningful information and remove unnecessary noise or artifacts that might interfere with the subsequent analysis. These steps help to standardize the text data, make it more manageable, and enable more effective analysis and modeling.

4.1. Data selection and preprocessing

Given the vast amount of data and metadata associated with each tweet, it is necessary to carefully select the relevant information for analysis. In our study, we specifically consider the following data: *favourite count*, *text*, *hashtags*, *retweet count*, and a list of *direct replies*. These selected features are deemed essential for capturing key aspects of the tweet content, engagement, and contextual information. To prepare the text data for embedding feature, we conducted a preprocessing step. This step involved several procedures aimed at enhancing the quality and consistency of the text. Specifically, we performed lemmatization to transform words into their base or dictionary form. Furthermore, we implemented a cleaning process to remove irrelevant elements such as external links, punctuation marks, stop words, mentions of other users, and symbols. Additionally, we replaced numeric values within the text with the hashtag sign as these numeric values were deemed non-informative for our analysis.

It is important to note that while we modified certain aspects of the text during preprocessing, the original text remains intact. This allows us to create additional features based on the original text, such as determining whether the tweet has a link, detecting mentions of news agencies, or quantifying the amount of question and exclamation marks present in the text. These additional features provide valuable insights and contribute to a more comprehensive analysis of the tweet content.

4.2. Features

The extracted features are the following:

- Capital Ratio: This feature calculates the ratio of uppercase characters in the entire text. It can provide insights into the text's writing style or emphasis. For example, a higher capital ratio might indicate a more assertive or emphatic tone, which could be relevant in determining the stance of the reply.
- Length of the Text: The length of the text refers to the number of characters or words in the text. Longer texts might suggest that the reply is attempting to provide more detailed explanations or arguments, potentially indicating a denying stance. Shorter texts, on the other hand, could imply a supporting or commenting stance that doesn't require extensive explanations.
- Polarity Sentiment Analysis: In our study, we utilize the TextBlob library to perform polarity sentiment analysis. This analysis involves assigning a polarity score to the text, which indicates the sentiment expressed in the tweet or reply. The polarity score ranges from -1 to 1, where -1 represents a negative sentiment, 0 indicates a neutral sentiment, and 1 reflects a positive sentiment. For example, tweets with a positive polarity score may indicate support for the target, while those with a negative polarity score might suggest denial or disagreement.
- Amount of Question and Exclamation Marks: This feature counts the number of question marks and exclamation marks in the text. The presence of these punctuation marks can convey different intentions or emotional tones. For example, a higher count of question marks might indicate a questioning or uncertain stance, while exclamation marks could imply strong emotions or emphasis.
- Word Embeddings: They capture the semantic meaning of words by representing them as dense vectors in a high-dimensional space. In this case, the Gensim library is used with pre-trained glove vectors *glove-twitter-25* to obtain word embeddings for the text. These embeddings encode contextual information and can help capture the meaning and relationships between words in the text.
- Has Link: This feature checks whether the post contains a link. The presence of a link might suggest that

Table 1: Accuracy and F1-scores of different models without context, with parent context and with both the parent and the reply context

Model	Macro F1-score	Accuracy	$F1_S$	$F1_D$	$F1_Q$	$F1_C$
Without context						
Random Forest Classifier	0.2624	0.76	0.00	0.00	0.18	0.87
Gradient Boosting Classifier	0.2953	0.71	0.14	0.00	0.21	0.84
Stohastic Gradient Descent Classifier	**0.3735**	0.72	**0.23**	0.00	**0.42**	0.84
With parent context						
Random Forest Classifier	0.2971	0.77	0.00	0.00	0.32	0.87
Gradient Boosting Classifier	**0.3454**	0.69	0.15	0.07	0.35	0.82
Stohastic Gradient Descent Classifier	0.2925	0.72	0.14	0.07	0.41	0.81
With children and parent context						
Random Forest Classifier	0.2964	0.77	0.02	0.00	0.29	0.87
Gradient Boosting Classifier	0.3062	0.69	0.14	0.04	0.23	0.82
Stohastic Gradient Descent Classifier	**0.3641**	0.72	0.21	**0.18**	0.36	0.71

the reply includes a source or reference, potentially indicating a supporting or denying stance by providing external evidence.

- Mentions News Agency: This feature detects whether the post mentions a news agency. This can be relevant in determining the credibility or reliability of the information shared. If a reply mentions a news agency, it might indicate a supporting or denying stance based on the perceived trustworthiness of the news source.

5. Results

Our primary objective was to investigate the impact of context on the performance of different models using three different scenarios: without context, with context from the parent tweet only, and with context from both parent and child tweets. The experimental results are presented in Table 1, where each model is evaluated under the previously mentioned scenarios. To determine the optimal model, we employed the Grid Search algorithm to search for the best hyperparameters. The results indicate that the Stochastic Gradient Descent Classifier (SGDC) outperforms the other models in terms of the overall Macro F1-score, except for when we only consider the parent context. An analysis of the results suggests that the Gradient Boosting Classifier (GBC) outperforms the Stochastic Gradient Descent Classifier (SGDC) which we can conclude on several observations, including the macro F1 score and the F1 scores of other labels, which predominantly favor the GBC model. Notably, when the SGDC model is trained without any context from other tweets, it achieves the best F1-scores for most labels, except for the *deny* label. The *supporting* and *deny* label are particularly significant as they indicate the potential presence of a rumor in the tweet they respond to.

Considering the importance of accurately identifying the *supporting* and *deny* labels, the optimal model for classifying our dataset would be the SGDC model with children and parent context. Although the F1-scores for other labels are not significantly different from the alternative scenarios, the F1-score for the *deny* label improves considerably compared to other models. Thus, based on the evaluation results, we recommend utilizing the SGDC model with context from both parent and child tweets as it offers competitive performance across all labels.

6. Conclusion

In recent years, there has been a significant effort to tackle the challenging problem of rumour veracity. However, this task has proven to be exceptionally difficult due to various factors, including the limited availability of labeled datasets. These datasets are often small in size and untrustworthy, making them inadequate for training accurate models. Another challenge in determining rumour veracity lies in classifying the stance of text, which has also been shown to be problematic due to imbalanced label distributions.

This paper specifically focuses on the stance classification problem using simple machine learning models. Instead of relying on complex models, we employ straightforward machine learning algorithms and explore the effectiveness of different hand-crafted features. However, our primary emphasis lies in investigating the impact of context on stance classification.

To substantiate our claim that context plays a crucial role in classifying stances accurately, it is essential to conduct experiments using more sophisticated models on larger datasets. By doing so, we can further validate the significance of context in enhancing the performance of stance classification models.

References

Piush Aggarwal and Ahmet Aker. 2019. Identification of good and bad news on Twitter. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 9–17, Varna, Bulgaria, September. INCOMA Ltd.

Ahmet Aker, Leon Derczynski, and Kalina Bontcheva. 2017. Simple open stance classification for rumour analysis. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP*

2017, pages 31–39, Varna, Bulgaria, September. INCOMA Ltd.

J. Bhuvana B. Bharathi and Nitin Nikamanth Appiah Balaji. 2020. Textual and contextual stance detection from tweets using machine learning approach.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. SemEval-2017 task 8: RumourEval: Determining rumour veracity and support for rumours. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 69–76, Vancouver, Canada, August. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. volume abs/1810.04805.

Genevieve Gorrell, Elena Kochkina, Maria Liakata, Ahmet Aker, Arkaitz Zubiaga, Kalina Bontcheva, and Leon Derczynski. 2019. SemEval-2019 task 7: RumourEval, determining rumour veracity and support for rumours. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 845–854, Minneapolis, Minnesota, USA, June. Association for Computational Linguistics.

Kian Kenyon-Dean, Eisha Ahmed, Scott Fujimoto, Jeremy Georges-Filteau, Christopher Glasz, Barleen Kaur, Auguste Lalande, Shruti Bhanderi, Robert Belfer, Nirmal Kanagasabai, Roman Sarrazingendron, Rohit Verma, and Derek Ruths. 2018. Sentiment analysis: It's complicated! In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1886–1895, New Orleans, Louisiana, June. Association for Computational Linguistics.

Parinaz Sobhani, Saif Mohammad, and Svetlana Kiritchenko. 2016. Detecting stance in tweets and analyzing its interaction with sentiment. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 159–169, Berlin, Germany, August. Association for Computational Linguistics.

Ruoyao Yang, Wanying Xie, Chunhua Liu, and Dong Yu. 2019. BLCU_NLP at SemEval-2019 task 7: An inference chain-based GPT model for rumour evaluation. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 1090–1096, Minneapolis, Minnesota, USA, June. Association for Computational Linguistics.

LOL (or Not?): Evaluating the Impact of External Context on Humor Assessment in Micro-Edited News Headlines

Marko Barbir, Marin Puharić, Kim Staničić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{marko.barbir,marin.puharic,kim.stanicic}@fer.hr

Abstract

Evaluating humor is a complex and subjective task that poses a significant challenge in computational linguistics. In this paper, we propose adding external context, specifically from Wikipedia, hoping it would improve model performance. We trained and evaluated 5 different models using BERT embeddings (fine-tuned BERT, FFNN, XGBoost, SVR and Ridge regression). In the end, we were unable to improve the model performances with this approach. Still, we believe there is potential in exploiting external context in the task of humor prediction.

1. Introduction

Humor is a unique aspect of human communication. We rely on humor on many different occasions, ranging from personal enjoyment to connecting with others. As intuitive as humor is to us, answering the question "What makes something funny?" isn't trivial. Because of that, humor detection seems to be a much more complicated task for AI systems than it is for humans.

To identify our humor-related research question, we took a look at task 7 in the SemEval 2020 competition, where participants had to estimate the funniness of news headlines that have been modified by humans using a micro-edit to make them funny. Hossain et al. (2020a) analyzed participating system results and compared them to each other to see if they showed any notable trends and similarities. They found that there are many aspects of humor the models seemed to have problems with, such as humor achieved by tension relief, sarcasm, cultural references, lack of world knowledge, etc.

We decided to focus on one of these aspects – lack of world knowledge – and check if including external context for named entities appearing in the headlines would improve the model performance. Our hypothesis is that adding world knowledge will improve the performance. The external context extraction is explained in Section 4, right after Section 3 where we covered the dataset.

The difference between extracting embeddings for the headlines with and without external context is explained in Section 5. Section 6 covers model details, including their training and optimization. Finally, we presented and analyzed our results in Section 7.

2. Related Work

To appropriately add external context to our data, we took inspiration from Wang et al. (2021). One of the main contributions of their paper is the proposal of improving contextual representation of an input sentence through retrieving (external) related text and concatenating the input sentence with the retrieved result for a retrieval-based view. The paper concludes that the contextual representations computed on the retrieval-based view can achieve improved performance compared to the original input view based only on the sentence. We used a similar approach, further explained in Section 4 and Section 5.

3. Dataset

We used the Humicroedit dataset (Hossain et al., 2020b) provided by the competition (SemEval 2020, task 7). The dataset consists of newspaper headlines paired with versions of the same headlines that contain simple replacement edits designed to make them funny. Each edited headline also contains a score from 0 (not funny at all) to 3 (very funny) annotated by 5 judges per headline. Dataset examples are shown in Table 1.

The dataset consists of 15 095 entries split into three sets: training (64%), evaluation (16%) and test set (20%). We only used the edited headlines (and their scores) from all three sets.

It is worth mentioning that there are certain quirks to the dataset. There is a non-uniform score distribution, with extreme values (0 and 3) being less frequent. Some headlines also showed lack of sufficient agreement between judges. Additionally, there is certain bias on the data because there is a considerable number of headlines mentioning "Trump" and "hair", which mostly received high humor scores.

4. External Context Extraction

We chose Wikipedia as the source for external context extraction (using the MediaWiki API[1]) as it offers concise and factual information. Google Search was also considered but proved to be more difficult to extract meaningful contexts from, while also having an API limitation for the free version.

We first extracted named entities from the dataset to identify and isolate specific pieces of information from the headlines, such as people, organizations, locations, and more. This step is crucial for external context extraction because it allows us to focus on the most relevant parts of the headlines and provide additional external context for these named entities, which can help improve the performance of models like BERT (Devlin et al., 2018) in various

[1] https://www.mediawiki.org/wiki/API

Text Analysis and Retrieval 2023: Course Project Reports (TAR 2023), pages 11–14, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2023.

Table 1: Some headline examples, edit and score included, from the Humicroedit dataset. To get the edited headline, the bolded word in the original headline is replaced with the substitute word. The score (0-3) indicates how funny the edited headline is.

Original Headline	Substitute	Score
Kushner to visit **Mexico** following latest Trump tirades	therapist	2.8
London Hit By Suspected **Terror** Attack Days Before Election , PM Says	asthma	0.8
Turkey protests : Erdogan accuses EU of **hypocrisy**	lying	0.0

NLP tasks (Wang et al., 2021). We used spaCy[2] for NER and excluded the named entities with irrelevant tags for our task, such as: cardinal, date, quantity, time, percent, ordinal, money and language.

To retrieve external context, we first searched Wikipedia for the top two article titles of each named entity. We chose the top two articles (instead of just the first one) to avoid missing the correct external context for our named entity. Using more than two articles could, however, introduce too much noise (articles that aren't relevant for the named entity). Afterward, we retrieved the first sentence of the article's summary section for each of the retrieved articles. We only used the first sentence due to computational resource constraints when generating contextualized embeddings using BERT. An example of possible results is given in Table 2.

5. Testing Setup

In order to test the impact of adding context to the edited headlines we decided to adapt an approach inspired by Wang et al. (2021). Similarly to Wang et al. (2021), we used BERT to generate contextualized embeddings, but the exact way in which we used BERT differs slightly. Instead of passing the outputs of BERT into a CRF, our most commonly used approach was taking the outputs of BERT's second to last hidden layer and then either max pooling them into a single "*sentence embedding*", or taking the embedding of only the first token (BERT special token [CLS]), which we then proceeded to use as inputs to regression models.

The other way in which we utilized BERT was fine-tuning it to our task as described in Section 6. This had always been our preferred approach, however due to resource constraints we were able only to slightly dabble in applying it (e.g. the model which utilized external context only ended up running for 5 full epochs before we ran out of GPU time on Google Colab).

5.1. Adding Context

So far we only described the process of retrieving external context for the headlines whose humor level we had to predict, but not how exactly we would use them to provide models with additional information. The way in which we attempted to achieve that was almost exactly the same as in Wang et al. (2021), with one small modification. The process consists of using BERT's sentence separation special token "**[SEP]**" to separate the edited headline from the added context, and additional [SEP] tokens to separate the information retrieved for each of the named entities contained in the headline. The modification was that, due to retrieving 2 possible results for each named entity, we had to slightly modify the separation of the context by inserting additional [SEP] tokens between each of the results for a single name entity. Essentially, this means that the resulting sentence which we tokenized, and fed into BERT had a structure resembling "[CLS]Edited headline. [SEP] 1^{st} result for 1^{st} NE. [SEP] 2^{nd} result for 2^{nd} NE. [SEP] 1^{st} result for 2^{nd} NE [SEP]...".

6. Models

In this paper, we aimed to assess whether incorporating external context could improve humor score predictions. To explore this, we employed a variety of models - including FFNN, fine-tuned BERT, SVR, XGBoost, and Ridge regression - each of which utilized the previously mentioned BERT embeddings. By comparing the performance of each model with and without external context, we were able to directly evaluate the differences in prediction accuracy resulting from the integration of external context, allowing for a clearer understanding of its impact on humor prediction.

We performed hyperparameter tuning for each of the models on the Max Pooled tokens for both the base dataset and the dataset with the added context. For the classic models, we were also able to do hyperparameter tuning using the [CLS] tokens.

6.1. Baseline

We used the same baseline as the one provided in the competition. The baseline always predicts the overall mean funniness grade in the training set which is 0.9356 in this case.

6.2. FFNN

We designed the FFNN using the TensorFlow Keras library[3]. To optimize the model's architecture and hyperparameters, we used the Keras Tuner[4] library. We tested different implementations using 1, 2, 3, 4, 5 and 10 dense layers with a varying number of units (4, 16, 64, 256, 512, 1024). Each layer incorporated dropout, *L2* regularization and a ReLu activation. Possible dropout values were 0.0, 0.1, 0.2 and 0.3. *L2* regularization ranged from 10^{-5} to 1, with a multiplicative step of 10. We optimized the model using Adam (Kingma and Ba, 2014) with the learning rate

[2] https://spacy.io/

[3] https://keras.io/

[4] https://keras.io/keras_tuner/

Table 2: Example of external context retrieval. Kushner and JPMorgan are recognized as named entities

Edited headline	**Kushner** , CIM to Get $600 Million **JPMorgan** Loan for swamp Site
First result for **Kushner**	Jared Corey Kushner (born January 10, 1981) is an American businessman, investor, and former government official
Second result for **Kushner**	David Alan Kushner II is an American singer-songwriter
First result for **JPMorgan**	JPMorgan Chase & Co. is an American multinational financial services company headquartered in New York City and incorporated in Delaware
Second result for **JPMorgan**	JPMorgan Chase Bank, N.A., doing business as Chase, is an American national bank headquartered in New York City, that constitutes the consumer and commercial banking subsidiary of the U.S. multinational banking and financial services holding company, JPMorgan Chase

ranging from 10^{-5} to 10^{-3} with a multiplicative step of 10. For finding the optimal values, we used Bayesian optimization from the Keras Tuner, trying out 20 different combinations and evaluating each model twice.

6.3. Fine-tuned BERT

To use the BERT model for predictions, we fine-tuned it by attaching a simple regression head. We were unable to perform hyperparameter optimization because of resource constraints. For the training process, we used Adam again with learning rate of $2*10^{-5}$ and batch size of 32. We were able to run it for 55 epochs on the dataset without context and for only 5 full epochs on the dataset with context before running out of GPU time on Google Colab.

6.4. SVR, Ridge Regression and XGBoost

For the SVR and Ridge regression models, we used the implementations from the scikit-learn library[5]. The xgboost library[6] was used for the implementation of the XGBoost model. Hyperparameter optimization for each of these models was done using grid search with 5-fold cross validation from the scikit-learn library. SVR was tuned with a parameter grid optimizing *C*, *epsilon* and the kernel choice. Ridge regression was tuned by trying out different values of *alpha*. XGBoost was tuned using a hyperparameter grid optimizing maximum depth, learning rate and the number of estimators.

7. Results and Evaluation

The results for all of the experiments we had performed are displayed in Table 3. All of the experiments were run 10 times with the mean metric reported, except for the fine tuned BERT which is here for completeness and is not statistically relevant. For all models we calculated the root mean squared error (RMSE) and mean average error (MAE) using either the [CLS] embedding, or pooling the embeddings of all the tokens. For the simpler models (Ridge regression, XGBoost, SVR) max pooling seemed to consistently perform better than the [CLS] embedding, especially for the cases without context, which is why we decided to proceed with using max pooling for the remaining 2 models.

At first look, we can see that most of the simple models had similar, yet relatively poor results in both metrics, with Ridge regression and SVR using max pooling being the best of them. We believe this can be explained by the fact that, due to BERT being trained on a semantically different task, there is simply not enough relevant information extracted into the embeddings. One piece of evidence to support this is that our best performing model by far is the BERT model with an attached max pool and output layer fine tuned for 55 full epochs (at which point we ran out of GPU resources, with the validation loss still decreasing at the end).

7.1. Effect of Adding External Context

In our testing, we achieved no improvement by adding external context to the headlines. In fact, every single model using additional context performed significantly worse than the base model which used only the edited headlines. The worst offender is the BERT model fine tuned with additional context, but since it was only able to run for 5 epochs not many conclusions can be drawn from that result.

There could be many reasons as to why the models suffer after adding context. First and foremost we believe that sourcing the context from Wikipedia might not have been the optimal choice. While it does provide factual, and sometimes relevant information, the model most likely does not need it, as the entirety of the English Wikipedia was part of BERT's training corpus, and the named entities appearing in our dataset are quite famous, and information about

[5] https://scikit-learn.org/stable/

[6] https://xgboost.readthedocs.io/en/stable/

Table 3: Model performance results. We list the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) for each model. The labels CLS and MAX refer to the way the BERT output was handled (details in Section 5). The best result for both models with and without external context are bolded, and the overall best result is underlined.

	Model	CLS RMSE	CLS MAE	MAX RMSE	MAX MAE
Without Context	FFNN	-	-	0.5582	0.4577
	Fine-tuned BERT	-	-	**0.5411**	**0.4336**
	SVR	0.5636	0.4555	0.5568	0.4498
	Ridge	0.5605	0.4562	0.5562	0.4523
	XGBoost	0.5656	0.4638	0.5611	0.4590
With Context	FFNN	-	-	0.5753	0.4733
	Fine-tuned BERT	-	-	0.6006	0.4770
	SVR	0.5707	0.4640	0.5707	**0.4639**
	Ridge	**0.5692**	0.4662	0.5695	0.4656
	XGBoost	0.5696	0.4663	0.5695	0.4675
Baseline	RMSE = 0.5747	MAE = 0.4736			

them is widely available. This does not mean, however, that we do not think external context is useless at all for this task. In fact we are hopeful that, with access to Google Search or a similar API, we might be able to improve our results on this task by extracting time relevant opinionated information, for example by searching for news articles and forum posts around the time the headline had been created.

Furthermore, as had already been mentioned, BERT was trained on a task very different from ours. While we speculated that having resulted in general poor results of the simple models, we also hypothesize that it additionally deteriorated the performance with added context, as BERT is not very likely to be able to use the added context we provided, as we'd expect it to. We do not currently have enough data to support this claim, however it would be interesting to, in the future, investigate a similar approach but further experimenting with fine-tuning the model to our task.

8. Conclusion

In this paper, we wanted to test whether the inclusion of external context for named entities would benefit the humor prediction task for edited news headlines. Our hypothesis was that models with external context would perform better than those without. We extracted external context for NE from Wikipedia and used both the original and the combination of the original and extracted data to generate contextualized embeddings using BERT. We tested these on several models and came to a surprising conclusion – the embeddings with external context performed worse than those without, disproving our original hypothesis.

We hypothesize that BERT already had enough context for the named entities present in the dataset (since it also partly uses Wikipedia as a training source) and that the additional external context only confused it. Also, we theorize that topical context, like the general public opinion about named entities present in the dataset, would have been more beneficial than factual information provided by Wikipedia.

Still, we firmly believe that external context inclusion shouldn't be abandoned and could potentially yield better results for other datasets and more sophisticated architectures, specifically focusing on fine-tuning BERT or perhaps a similar language model.

References

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*.

N. Hossain, J. Krumm, M. Gamon, and H. Kautz, 2020a. *SemEval-2020 Task 7: Assessing Humor in Edited News Headlines*. Department of Computer Science, University of Rochester and Microsoft Research AI, Microsoft Corporation, Redmond, WA.

Nabil Hossain, John Krumm, Michael Gamon, and Henry Kautz. 2020b. SemEval-2020 task 7: Assessing humor in edited news headlines. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 746–758, Barcelona (online), December. International Committee for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

X. Wang, Y. Jiang, and N. Bach, 2021. *Improving Named Entity Recognition by External Context Retrieving and Cooperative Learning*. Association for Computational Linguistics.

Author Style Change Detection

Luka Družijanić, Tomislav Ćosić, Luka Brečić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
luka.druzijanic@fer.hr, tomislav.cosic@fer.hr, luka.brecic@fer.hr

Abstract

In this paper, we present our solution for the Style Change Detection problem for PAN at CLEF 2021, which defined three tasks - detecting if a text was written by multiple authors, where the authors interchanged and which paragraphs were written by the same author. With the advances in the field of natural language processing, the tasks of detecting multiple author contribution and authorship attribution have become one of its core areas of research, finding its applications mostly in plagiarism detection. Our approach relies on pretrained BERT to generate embeddings of text, which are then given to binary classifiers which detect style change. We give a brief description of the dataset, the task and our model. Finally, we show our approach outperforms baselines for the first two tasks.

1. Introduction

The task of detecting multiple author contribution and authorship attribution, traditionally known as stylometry, is a complex task with rich history. The field had largely concerned itself with authorship attribution of historical texts, such as the extent of Shakespeare's works (Craig, 2004). Today, solutions to this task find their applications mostly in the area of plagiarism detection, as plagiarism has become easy to take a hand on and, at the same time, really hard to spot and detect.

With the improvements in deep learning techniques, this task has become one of the central areas of research in natural language processing. In search for a satisfying solution, the Style Change Detection task has become frequent at PAN. We present our solution developed for the PAN at CLEF 2021, which asked its participants to detect if a text was written by multiple authors, find the positions of authorship changes and assign all paragraphs of the text to a specific author.

We use a pretrained BERT model proposed in Devlin et al. (2019) for generating embeddings of paragraphs and whole documents, which are then fed into binary classifiers to detect multiple author contribution inside a document or between two paragraphs. For the final task of authorship attribution, we experiment with several clustering algorithms. We compare these BERT models with simpler Bag-of-Words or TF-IDF approaches.

Before explaining the solution in this paper, we outline the dataset structure and explain the task in more detail. Then, we provide a brief overview of related work and state-of-the-art approaches. Next, we explain our approach. Finally, we present our results and comparisons.

2. Related Work

Solutions, shortly described in the following paragraphs, are some of the existing contributions for PAN at CLEF 2021 *Author Style Change Detection* task.

LSTM Powered Attribution Algortihm Solution presented in (Deibel and Löfflad, 2021) makes use of the textual features (commonly used in autorship attribution problem solutions) and pretrained fastText for word embeddings. Approach chosen in order to solve the first task is a multi-layer perceptron (MLP) with word embeddings as input. For second task a two-layered bidirectional LSTM is chosen. In the end, third task is solved using predictions from first two tasks and then iterating over each pair of paragraphs and grouping them together if a style change is not detected.

Solving a Binary Classification Problem Strøm (2021) propose a model using stylistic features provided in (Zlatkova et al., 2018) alongside BERT embedding features. For both first and second task a stacking ensemble of binary classifiers is used. Similarly to previously described LSTM approach, a recursive strategy based on predictions from first two tasks is proposed. While iterating against each of the paragraph pairs a probability that they have the same author is assigned.

Siamese Neural Networks The approach presented in (Nath, 2021) uses Siamese Neural Networks, consisting of a GloVe embedding layer and a bidirectional LSTM, to calculate distances between paragraphs.

Authorship Verification The approach presented in (Singh et al., 2021) relies on an approach for authorship verification contributed to PAN at CLEF 2020 by Weerasinghe and Greenstadt (2020). Authorship verification model extracts features for paragraphs including TF-IDF, n-grams of part of speech tags, and vocabulary richness measures. A logistic regression classifier is used on the differences of these feature vectors between two paragraphs to obtain solutions for the first two tasks. For the third task, hierarchical clustering on the distance matrix obtained from the classifier is used.

Single BERT Zhang et al. (2021) proposes a solution based on Google Open AI's pretrained BERT model. First task is solved by infering results from second and third task solutions. Second task relies on the weights calculated by the binary classificator used to solve third task which computes

Text Analysis and Retrieval 2023: Course Project Reports (TAR 2023), pages 15–18, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2023.

Table 1: Mean, median, min and max of paragraph counts in train and validation data sets.

	Train	Validation
Mean	6.90	6.87
Median	6	6
Min	2	2
Max	47	40

whether a style change occurs between paragraphs.

3. Dataset

The main objective of Style Change Detection is to detect where authors interchange in a multi-authored document. First of all, a model should be able to determine whether the document is written by a single or multiple authors. If the document is written by multiple authors, model should outline the borders between the author changes and assign each paragraph to a different author.

The problem can be stated in three following tasks:

- **Task 1** Given a document, conclude whether the document was written by one or more authors
- **Task 2** In case of a given document having multiple authors, determine the position of changes
- **Task 3** In case of a given document having multiple authors, assign each paragraph to its author

A dataset of documents written in English is provided for development of the solution for the given problem. An important presumption of the given dataset is that author changes can occur only between the paragraphs.

Documents, upon which the provided dataset is built, are user posts scraped from the *StackExchange* network and all are related to technology covering different topics. Dataset is split in three parts - train, validation and test, whereas each subset contains 70%, 15% and 15% of the provided documents respectively. However, the test dataset is unavailable, so we present our scores on the validation dataset. Alongside the documents, ground-truth files are also provided for train and validation subset.

Table 1 contains simple statistics of the data including mean, median, minimum and maximum amount of paragraphs per document from train and validation subsets. Furthermore, Figure 1 outlines paragraph distributions over the train and validation data subsets.

4. Models

For all three tasks, we rely on BERT embeddings (Devlin et al., 2019). The documents are first split into paragraphs, as those are guaranteed to have a single author. Each paragraph is split into sentences using predefined rules, and each sentence is split into tokens using BERT Tokenizer.

Next, tokens are processed by BERT, which produces tensors of size $12 \times l \times 768$ for each sentence, where l is the length of a sentence. To obtain a flattened tensor of constant length 768, we keep only the last 4 of the 12 layers and sum over first two dimensions. To generate an embedding of an entire paragraph, we average the embeddings of its sentences.

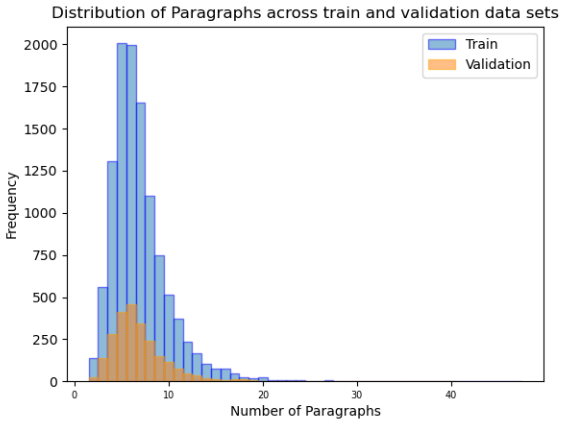


Figure 1: Distribution of paragraphs in train and validation subset.

Summing token embeddings to generate sentence embeddings leads to noticable differences in embeddings between large and short sentences, essentially encoding sentence length information inside the embedding vector. We claim that sentence length is an important stylistic decision by the author, so it makes sense to keep this information inside sentence embeddings. On the other hand, paragraph length is not a feature which is important in differentiating authors, so we decide to average sentence embeddings to obtain paragraph embeddings.

For the first task of detecting multiple authors inside a document, we average the embeddings of all paragraphs to obtain a document embedding. This document embedding is then given to a binary classifier which predicts if the document was written by multiple authors.

The second task asks us to predict if the author changed between paragraphs. For each consecutive pair of paragraphs in the document, we average their embeddings and, again, give them to a binary classifier which predicts if a change in authorship occurred between paragraphs.

The third task is assigning each paragraph to one of the authors. The idea here is to use a clustering algorithm to group paragraph embeddings together. We first obtain a distance matrix by querying our task 2 classifier to give us the probability of each pair of paragraphs being written by two authors. We then perform hierarchical clustering on this distance matrix.

5. Experiments

In all tasks, we compare our model to a simple baseline, denoted as `TF-IDF + CosSim` - we calculate similarity between two paragraphs as cosine similarity between their TF-IDF features. We obtain task 3 results by assigning each paragraph the same author as its most similar paragraph, if it crosses a certain threshold. The threshold is determined experimentally on the validation set. The answers to task 2 and task 1 are then inferred from this grouping. Additionally, we try the same model with Bag-of-Words features,

denoted as `BOW + CosSim`.

We constructed another completely random baseline `Rand` - for each document, a random number of authors is chosen. Each paragraph is then randomly assigned to one of those authors to obtain task 3 results. Tasks 1 and 2 are inferred from task 3.

Wherever we reference hierarchical clustering, we always tested with three linkage criteria - *average*, *maximum* and *minimum*. *Average* always proved the best. All experiments were implemented with the *scikit-learn* library in Python. Classifiers for the first task took a few minutes to train on the i5-10400 CPU, while the classifiers for the second task took up to an hour.

5.1. Task 1

For task 1, we attempted multi-layer perceptron (MLP), Random Forest (RF) and SVM classifiers. We denote these models as `BERT + MLP`, `BERT + RF` and `BERT + SVM` respectively.

By performing a grid search on the validation set, we found these hyperparameters to be the best - for task 1, we train the MLP with 3 hidden layers of sizes 50,20,10 with the Adam optimizer (Kingma and Ba, 2014) in batches of size 200 for 200 epochs with early stopping if loss does not improve in 10 epochs. L2 regularization is set to 0.0001 and learning rate to 0.001. RF is fitted with 1800 trees and gini criterion. For SVM, we use the *rbf* kernel with $C = 10$ regularization parameter. The γ parameter is set to

$$\frac{1}{n \cdot Var(X)}$$

where n is the number of input feautres, and $Var(X)$ is the variance of input data.

5.2. Task 2

For task 2, we tested the same classifiers as in task 1. Here, we train the MLP with 3 hidden layers of sizes 50,10,10 with the Adam optimizer in batches of size 32 for 300 epochs with early stopping if loss does not improve in 10 epochs. L2 regularization and learning rate are both set to 0.001. RF is fitted with 1800 trees and gini criterion. For SVM, we use the *rbf* kernel, $C = 1$ regularization parameter and γ set in the same way as in task 1.

5.3. Task 3

Here we attempt several setups. We will denote the setup described in section 4. as `BERT + MLP + Hierarchical`. For calculating paragraph similarities we used the best classifier from task 2, which proved to be MLP.

Alternatively, we attempted clustering on features themselves instead of relying on similarities from the classifier. Clustering algorithms are known to not work very well on high dimensional data, so we first use Principle Component Analysis (PCA) to reduce the dimensions of embeddings. Finally, we use a clustering algorithm, such as KMeans or hierarchical clustering, to group the paragraphs together.

6. Results

Table 2 shows the F1 scores obtained on the validation dataset for tasks 1 and 2. All the classifiers outperform the baselines. The best model is `BERT + SVM` for task 1, and `BERT + MLP` for task 2.

Table 2: Task 1 and task 2 F1 scores of selected models on validation dataset. Best scores are written in bold.

	Task 1	Task 2
Rand	0.499	0.498
BOW + CosSim	0.557	0.479
TF-IDF + CosSim	0.512	0.485
BERT + MLP	0.708	**0.638**
BERT + RF	0.589	0.604
BERT + SVM	**0.709**	0.626

Table 3: Task 3 F1 scores of selected models on validation dataset. Best scores are written in bold.

	Task 3
Rand	0.304
BOW + CosSim	0.341
TF-IDF + CosSim	**0.357**
BERT + KMeans	0.268
BERT + PCA + KMeans	0.299
BERT + Hierarchical	0.268
BERT + PCA + Hierarchical	0.298
BERT + Task2 + KMeans	0.270
BERT + Task2 + PCA + KMeans	0.288
BERT + Task2 + Hierarchical	0.269
BERT + Task2 + PCA + Hierarchical	0.285
BERT + MLP + Hierarchical	**0.356**

Table 3 shows the F1 scores obtained on the validation dataset for task 3. Perhaps surprisingly, all the models that cluster BERT features, even with PCA, perform worse than random. In each case, PCA does improve the clustering, but not sufficiently. The only model that outperforms the random baseline is the `BERT + MLP + Hierarchical`, although the simple `TF-IDF + CosSim` is still slightly better.

7. Conclusion

In this paper, we presented our approach for solving the Style Change Detection problem for PAN at CLEF 2021. The problem was separated in three tasks - detecting if a document contains multiple authors, determining the positions of authorship changes and assigning each paragraph to its author. Our approaches relied on extracting BERT features from the document and using classifiers to solve all three tasks. For the first two tasks, we outperform benchmarks with SVM and MLP classifiers. However, the third task remains hard to solve. The two best performing approaches were a TF-IDF model with cosine similarity, and a model that used hierarchical clustering on paragraph similarities provided by our best task 2 classifier. Even though our final results do not surpass state of the art, approaches and insights presented in this paper could be useful for fur-

ther improvements in the field of authorship profiling.

References

Hugh Craig. 2004. Stylistic analysis and authorship studies. *A companion to digital humanities*, pages 271–288.

Robert Deibel and Denise Löfflad. 2021. Style change detection on real-world data using an lstm-powered attribution algorithm. In *CLEF (Working Notes)*, pages 1899–1909.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Sukanya Nath. 2021. Style change detection using siamese neural networks. In *CLEF (Working Notes)*, pages 2073–2082.

Rhia Singh, Janith Weerasinghe, and Rachel Greenstadt. 2021. Writing style change detection on multi-author documents. In *CLEF (Working Notes)*, pages 2137–2145.

Eivind Strøm. 2021. Multi-label style change detection by solving a binary classification problem. In *CLEF (Working Notes)*, pages 2146–2157.

Janith Weerasinghe and Rachel Greenstadt. 2020. Feature vector difference based neural network and logistic regression models for authorship verification. In *CEUR workshop proceedings*, volume 2695.

Z Zhang, X Miao, Z Peng, J Zeng, H Cao, J Zhang, Z Xiao, X Peng, and Z Chen. 2021. Using single bert for three tasks of style change detection. In *CLEF*.

Dimitrina Zlatkova, Daniel Kopev, Kristiyan Mitov, Atanas Atanasov, Momchil Hardalov, Ivan Koychev, and Preslav Nakov. 2018. An ensemble-rich multi-aspect approach for robust style change detection. In *CLEF 2018 Evaluation Labs and Workshop–Working Notes Papers, CEUR-WS. org*.

How Lazy Common Sense Can Be: ComVE Challenge

Chloé Duault, Ellyn Lafontaine, Mallory Taffonneau

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{chloe.duault,ellyn.lafontaine,mallory.taffonneau}@fer.hr

Abstract

Common sense has been proved to be quite a challenging task in Natural Language Understanding. It requires going beyond pattern recognition. In this paper, we demonstrate how the original BERT model classifier allows us to get acceptable results and how it is possible to fine-tune the BERT model using spacy lemmatization and NLTK's hypernyms and lemmatized synsets. In addition, we provide instances where our model encountered learning limitations and propose ideas of how those issues could be assessed.

1. Introduction

Common sense Validation and Explanation is the task of deciding whether a sentence makes sense or not. It is a hard task in Natural Language Processing since there is no clear definition of common sense, even if it is trivial for humans. Our aim is to determine if a sentence makes sense, and if not why so.

Figure 1 shows a question asked to chatGPT and its answer. The question contained a sentence and three options to choose from, to explain why the sentence does not make sense. It is a perfect example to illustrate our goal, and what we are trying to achieve.

To achieve our goal we use a pre-trained BERT classifier. In this paper, we try to find what are the best conditions for BERT to perform this task. Then we will compare the obtained result to ChatGPT.

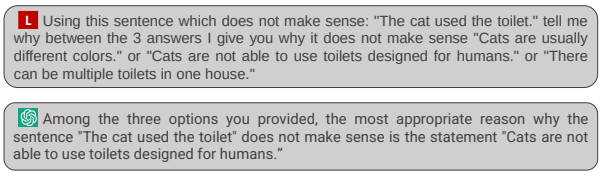


Figure 1: Question asked to ChatGPT and answer given back. The sentence and options came from the dataset used.

2. Related Work

Most of the paper we use as a reference (Shi and Lin, 2019; Yap et al., 2020) focused and tried to create state of art results which was not our case. We mostly focus on finding the best features for each task, but we also want to demonstrate what should not be used in terms of features. However, like us, they mostly use a fine-tuned BERT.

3. ComVE Challenge

The *ComVE challenge*[1], Commonsense Validation and Explanation, is a challenge from CodaLab Challenge. More precisely, it comes from the SemEval 2020 challenge, we will focus on the fourth task. This task is itself composed of three sub-tasks which are described in section 3.1.

3.1. Description

The Validation task, which is the first one, is to choose between two similar wording sentences one which makes sense and another one which does not. Let's take an example, first we have *"The stove was cleaned with a cleaner."* and *"The stove was cleaned with a mop."*, as you can see there, the problem is not about the grammar, it is only about what does and does not make sense. That is why we need a model that will be able to go beyond pattern recognition.

The second task is to find out why one sentence does not make sense between three possibilities. If we take back our previous example, the sentence that did not made sense was *"The stove was cleaned with a mop"* and for this task, you will need to choose between three possible reasons, first *"The mop was with my mom."*, second *"The mop was my hairy dog"* and last *"A mop is too large to clean a stove"*. Here the correct answer would be *"a mop is too large to clean a stove*", because that is the only answer which explains why you can not clean your stove using a mop. The other two sentences are not explaining why it does not make sense, they are just stating random facts that are in no way linked to our original sentence.

The last task consists of generating reasons why one sentence does not make sense. Going back to our example, we could think of *"Mops are one of several tools used for floor care"* as a reason why there is no sense in cleaning the stove using a mop.

In our case, we chose to focus on the first two tasks. We wanted our model to focus on the task of sentence choosing and not on the generation of explanations.

3.2. Our goal

We chose to create one fine-tuned model for each task. Our aim was to be able to determine which features and parameters allowed us to get the best results. Moreover, we also wanted to fathom out what you should absolutely not use in terms of features and parameters.

[1] https://competitions.codalab.org/competitions/21080

Text Analysis and Retrieval 2023: Course Project Reports (TAR 2023), pages 19–22, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2023.

4. Background

4.1. Dataset

The datasets we used to conduct our experiments were those provided on the challenge's github (Wang et al., 2020a). It contains a total of 11 997 examples, 10 000 of them are allocated to the training set, 1000 to the testing set and finally 997 to the development set.

We were only able to use 10% of the training set and the whole development set. In Section 5.1. we will explain why.

We split the train dataset as follows : 90% for training and 10% for validation. The validation set has been of great use to prevent overfitting and to control the learning rate. To test the accuracy of our model, we use the entire development dataset.

4.2. BERT model

The task of our model was to determine if a sentence makes sense, and if not, why so. The best way to do this is to use a classifier. We searched for existing classifiers, and we have been able to find a few interesting ones for us: BERT and ELMo. Thanks to the paper from Wang et al. (2020b), we concluded that BERT is the best model in our case.

We used a pre-trained BERT model classifier provided by the bert-sklearn library from Nainan and Medina (2019), which is based on pytorch and transformers to solve our tasks, we used bert cased and uncased during our experiments.

4.3. Features

We did not know from the start which feature would work out the best for our problem. That is why we have decided to try out some of them, we thought could have an impact (either good or bad) on our model. We thought it would be great to use parsing and more precisely semantic parsing.

We first started by the parsing part, using the spacy library to perform lemmatization, stemming, stop word removals, n-gram retrieving. We also used the similarity method from spacy which is computed as the cosine similarity (1) :

$$similarity = \frac{v1 \cdot v2}{\|v1\|\|v2\|} \tag{1}$$

Where v1 and v2 are vectors composed of an array of tokens computed between two given sentences. We are using this feature in the second task, when we need to choose between three options for one false sentence.

Like Yap et al. (2020) presented in their paper, semantic parsing is an effective approach. We chose to use wordnet with the NLTK library. Wordnet is a large database that contains the semantic relations between words. For a given word, it will create a synset that contains a short explanation of the words, usage examples, it also contains synonyms, hypernyms, hyponyms and meronyms. For us, the important part of wordnet were the synsets and the hypernyms.

5. Realization

5.1. Training

We trained our model using a fine-tuned BERT classifier. We chose to use random state = 42 so that we could have the same results at every training.

Table 1: F1 score of the prediction made on our test dataset. It uses a model trained on the train set sub-sampled with different random states.

Random state	55	42	38	22
F1 score	0.50	**0.65**	0.58	0.54

Table 2: F1-score for a fine-tuned uncased and cased BERT model on 10% of train set for the first task.

Model	Uncased	Cased
Fine tune BERT	**0.645**	0.568

Since the bold number is the higher score, Table 1 shows that 42 gave us better results and it is a reference number used in most cases.

We encountered a problem while training our model, we did not have the computational requirements to train our model on the entire dataset and for more than 3 epochs. That is why we chose to train our model on only 10% of the dataset and for 3 epochs. We were only able to train using 20% of the dataset one time to get better results. However, we were not able to repeat it multiple times since it required a lot from our computers.

We are perfectly aware that it is an issue and that training on a larger part of the dataset with more epochs could have a huge impact on the quality of our model. But we still think it is great to know, even for a limited part of the dataset what results a user could expect. Since not everyone can have access to an appropriate machine to create their own model.

5.2. Test

Our results have been tested using the F1 score, which is commonly used for classification tasks. We used the F1 binary score for the first task. It was of great use there because the results we expect our model to give for this task are binary, they can only be either true or false.

On the other hand, the second task proved to be more complex and we chose to use the F1 macro score. We chose F1 macro (2) over the F1 micro (3) score because we did not want our model to take unbalanced labels into consideration while computing our score.

$$M = \frac{\sum_{i=1}^{N} F1score_i}{N} \tag{2}$$

With $F1score_i$ the F1 score of the i^{th} class and N the number of classes. Which means, two classes for the first task and three for the second.

$$\mu = \frac{TP}{TP + \frac{1}{2}.(FP + FN)} \tag{3}$$

With TP, FP, FN as the number of True Positive, False Positive and False Negative.

Table 3: F1 score results of the **first Task** with BERT model trained on 10% of the dataset and random state = 42. "No treatment" on the *second row* reference to the model predicting on the **test** dataset without any pre-processing done and "Treatment"in the *third row* referenced to the same pre-processing done on the **test** dataset. "No treatment" in the *second column* referenced to BERT model trained without any pre-processing on the **train** dataset. "Synsets" are lemmatized synsets of each words of the sentence.

F1-Score	Classic	Lemmas	Stemming	Stop Word	Bigram	Trigram	Synsets	Hypernyms
No Treatment	**0.654**†	0.445	0.507	0.648	0.648	0.648	0.636	0.647
Treatment	-	0.528	0.497	0.608	0.608	0.608	**0.649**	**0.649**

Table 4: F1-Macro score results of the **second Task** with BERT model trained on 10% of the dataset and random state = 42. "No treatment" on the *second row* reference to the model predicting on the **test** dataset without any pre-processing done and "Treatment"in the *third row* referenced to the same pre-processing done on the **test** dataset. "Classic" in the *second column* referenced to BERT model trained without any pre-processing on the **train** dataset. "Synsets" are lemmatized synsets of each words of the sentence.

F1-Score	Classic	Lemmas	Stemming	Stop Word	Bigram	Trigram	Synsets	Hypernyms	Similarity
No Treatment	**0.488**	**0.468**	0.278	0.171	0.277	0.182	0.171	0.171	0.178
Treatment	-	**0.492**†	0.300	0.171	0.171	0.171	0.171	0.171	0.171

If only one label was chosen during the prediction, the F1 macro score will show a particularly smaller precision compared to the F1 micro score, which computes the score for the entire set instead of computing the score for each class.

6. Results

The results are obtained with the prediction on our test dataset. The training is realized as explained in the Section 5.1., using the same parameters described. In each table, we use bold style to highlight the higher scores and the symbol † to highlight the highest score in the table.

6.1. First task

For the first task, the obtained results in Table 2 showed that the best case is to use the BERT uncased over BERT cased. Following this assumption, all of our experiments for this task have been made using BERT uncased.

When conducting our experiments, we thought about using a multitude of features. In the end, our results show that the best model was the original BERT model, with absolutely no treatment done, either on the train or test set. However, we still found out that hypernyms and lemmatized synsets could show quite some good results.

Furthermore, we discovered that using lemmatization or stemming was a really bad idea, since it gave results close or worse than random.

6.2. Second Task

For the second task, our results show in Table 4 that the best case is to use lemmatization on both the train and test set.

Again, without any treatment on the train and test set we also have good results. In the case of stop word removal, bigram, trigram, synset of lemmas, hypernyms and similarity, the results show that our model is not up to par.

The only time we got some results were for bigram, trigram and similarity without treatment on the test set. In most cases, applying the same treatment and train and test sets are showing an improvement.

6.3. Explanations

In the first task in Table 3, we found out that the best model was BERT without any kind of treatment, which is confirmed by Shi and Lin (2019), closely followed by the synsets of lemmas and the hypernyms. Synsets of lemmas and hypernyms are of great use here, since they are able to feed our model with more information and using this newly added knowledge, our model is better at generalizing. On the other hand, reducing words to simple lemmas or stems has proved to be a really bad idea there.

In the second task, lemmas were one the best results which is in total contradiction with the first task. It led us to think that, in this case, we were able to correctly feed our model in terms of lemmas so that our model could learn efficiently. The classic BERT also obtained some quite good results, as explained by Shi and Lin (2019).

However, we also found out that in the case of hypernyms, synsets, trigram, bigram, stop words removal and similarity our model gave us the same result (e.g. 0.171) over and over again, it is due to the fact that our model failed to learn, each time the model gave us the same answer, (e.g. the label A). We are thinking that for those cases, our model was again underfitted.

As explained before, we were able to train only on 10% of the dataset, since we trained on just a small percentage our model needed to be fed with as much knowledge as possible. All of our experiments tend to show that our

Table 5: Accuracy of our model trained on 20% of the data set and ChatGPT on 50 examples from our test dataset.

Model	Task A	Task B
ChatGPT	**0.92**	**0.94**
Fine-tuned BERT	0.71	0.49

model was underfitted to get better results, and we think that some methods used here could have better results than simple BERT when trained on a larger part of the training set.

6.4. Comparison with another model: ChatGPT

To compare our results we choose ChatGPT since it is a well known state of the art model. We took 50 examples from our test dataset. We then asked chatGPT the questions : "which sentence makes sense?" and "Using this sentence which does not make sense: *false sentence*, tell me between the 3 following answers why it does not make sense: option A or option B or option C", which represent the expectation for the first task and for the second task correspondingly.

The table 5 shows the accuracy obtained on 50 examples for each task and the accuracy of our model. The model is trained with 20% of our dataset for better accuracy. ChatGPT has better results on the second task compared to the first one, which is not the case for our model, for both tasks accuracy is close to 1. Our model has poor results compared to ChatGPT, so maybe BERT was not the best pre-trained model to resolve those tasks. However, let's not forget that our model is trained on only a small part of the dataset and on a small number of epochs. Which can explain why we got poorer results than chatGPT.

7. Conclusion

In this paper we tried to optimize BERT parameters and input treatment for fitting the Common-sense Validation and Explanation challenge (Wang et al., 2020b). We concluded that the easier option, with no input treatment and most common BERT uncased base model, gave us the best results. Additionally, we compared our results with chatGPT, which strongly outperforms our model.

Acknowledgements

This paper was written in the context of a class project for the university of Zagreb. We would like to express our deepest appreciation to Jan Šnajder and Josip Jukić for their advice and guidance throughout this project. Furthermore, this endeavor would not have been possible without Wang et al. (2020b), since they create this challenge. Besides, their paper for the challenge was of great help.

References

Charles Nainan and Ezequiel Medina. 2019. A sklearn wrapper for google's bert model. Available at *https://github.com/charles9n/bert-sklearn*.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling.

Cunxiang Wang, Shuailong Liang, Yili Jin, Yilong Wang, Xiaodan Zhu, and Yue Zhang. 2020a. Github to SemEval-2020 task 4. Available at *https://github.com/wangcunxiang/SemEval2020-Task4-Commonsense-Validation -and-Explanation*.

Cunxiang Wang, Shuailong Liang, Yili Jin, Yilong Wang, Xiaodan Zhu, and Yue Zhang. 2020b. SemEval-2020 task 4: Commonsense validation and explanation.

Boon Peng Yap, Andrew Koh, and Eng Siong Chng. 2020. Adapting bert for word sense disambiguation with gloss selection objective and example sentences.

Exploring the OCEAN Depths: a Look at Personality Trait Classification on Essays

Mateo Hrelja, Dominik Jurinčić, Ivan Linardić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{mateo.hrelja, dominik.jurincic, ivan.linardic}@fer.hr

Abstract

Capturing personality traits from text is a challenging task wherein current available solutions leave a lot to be desired. In this paper, we explore a variety of approaches in tackling this task. We try various text preprocessing methods in tandem with several different machine learning models of varying complexity. Ultimately, we focus on the interpretability of our results and aim to provide better clarity on the reasoning behind them. We use two different approaches when evaluating interpretability in order to get an even better grasp of this difficult, yet interesting task.

1. Introduction

Human personality has been a fascinating area of research for many years now, as it plays a large part in better understanding human behavior and interactions. The Big Five personality traits (McCrae and Costa, 1985), often referred to as OCEAN (Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism) serve as a reliable psychological model for describing the concept of personality for individuals. Over the years, advancements in natural language processing research and methods have made it possible and more reliable to capture human personality traits from text.

In this paper, we dive deeper into understanding how language models infer personality traits from stream-of-consciousness essays. We present several classification methods and models which we used for our results, as well as the effects of various text preprocessing and feature selection methods. As part of our research, we include both simple models which yield highly interpretable results as well as more complex deep models focused on performance.

The main goal of the paper is to delve deeper into the concept of interpretability and better understand certain patterns and insights carried within the writings of individuals, using different methods to interpret the obtained results in tandem with the use of different models.

2. Related Work

Previous work in this field constitutes of the idea that language-based assessments can give valid personality traits measures (Park et al., 2014) and that there exists correlation between language and personality traits which was proven by Schwartz et al. (2013). Also, they have performed experiments which showed that removing stop words harms the performance of their models, which was also useful in our case. Pizzolli and Strapparava (2019) predicted personality traits of characters in stories based on computational linguistics and gave a critic on the Big Five personality traits model we use.

Gjurković et al. (2021) found problems in quality and low number of personality-labeled datasets which induced them to create a new one which included personality and demographics information.

Recent work in deep learning, done on the Big Five personality model using the essays dataset introduced by Pennebaker and King (1999), was done in (Majumder et al., 2017). They used a convolutional neural network (CNN) feature extractor in which sentences were fed to convolution filters to obtain n-gram feature vector. State-of-the-art results on the essays dataset were achieved by Kerz et al. (2022) who took the transformer-based approach. Similar to Kerz et al. (2022), we use the Big Five personality traits and the pre-trained transformer language model BERT (Devlin et al., 2019).

3. Dataset

The essays dataset, introduced by Pennebaker and King (1999), consists of stream-of-consciousness essays written by psychology students. The use of stream-of-consciousness essays for personality trait classification seems appropriate, as the free form nature of the text encourages writers to express their thoughts and use language in a non-restricted way. The dataset contains 2467 essays and each is labeled with the personality scores of the writer for each of the Big Five personality traits. The labels are 'y' and 'n', representing scoring high or low for a given personality trait. The personality scores of students were obtained by filling out the Five Factor Inventory questionnaire (John et al., 1991).

4. Data Preparation and Feature Selection

The first step in the data preparation process was converting the 'y' and 'n' labels into their numerical versions 1 and 0. We then experimented with different text preprocessing techniques and feature extraction methods on the essay texts in order to find the ones which provide the best results. We used spaCy[1] and Natural Language Toolkit (NLTK)[2] to preprocess the raw essay text in several ways. In the end we had 4 versions of the essays: (1) the raw essay text, (2) the

[1] https://spacy.io/

[2] https://www.nltk.org/

Text Analysis and Retrieval 2023: Course Project Reports (TAR 2023), pages 23–27, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2023.

Table 1: Words important for scoring high in a personality trait by coefficient magnitude

Trait	Model	Words
Openness	LR	like, cat, ll, too, maybe, re, love, music, of, you
Openness	SVM Linear	crazy, love, like, maybe, you, ll, cat, of, re, music
Agreeableness	LR	least, would, with, right, on, to, really, so, family, have
Agreeableness	SVM Linear	many, worried, would, least, so, right, family, on, with, have
Conscientiousness	LR	today, it, tonight, hope, party, and, the, my, he, to
Conscientiousness	SVM Linear	student, decision, today, couldn, tonight, my, hope, to, he, party
Neuroticism	LR	don, feel, everything, want, scared, money, me, sex, life, stressed
Neuroticism	SVM Linear	worry, this, scared, me, everything, life, boyfriend, sex, money, stressed
Extraversion	LR	is, its, sorority, and, love, boyfriend, fun, all, am, so
Extraversion	SVM Linear	ready, mean, if, love, its, all, am, boyfriend, fun, so

Table 2: Words important for scoring low in a personality trait by coefficient magnitude

Trait	Model	Words
Openness	LR	college, is, to, my, because, school, home, class, have, classes
Openness	SVM Linear	college, because, is, boyfriend, assignment, class, home, tomorrow, game, confuse
Agreeableness	LR	stupid, girlfriend, don, is, damn, read, more, nothing, same, no
Agreeableness	SVM Linear	stupid, girlfriend, is, read, damn, store, same, don, nothing, wont
Conscientiousness	LR	want, hate, don, this, think, re, wake, god, point, chance
Conscientiousness	SVM Linear	want, hate, this, wake, point, re, think, chance, don, music
Neuroticism	LR	its, would, her, many, semester, already, beat, mind, as, texas
Neuroticism	SVM Linear	its, many, her, would, already, pledge, semester, beat, glad, mind
Extraversion	LR	don, there, in, should, want, something, eyes, perhaps, very, mother
Extraversion	SVM Linear	don, should, in, something, there, want, eyes, very, perhaps, real

lowercase and lemmatized essay text, (3) the lowercase essay text in which we removed stopwords and punctuation, (4) the essay text in which we removed punctuation.

4.1. Feature Extraction

We used a modified grid search approach to determine the best feature extraction method for our bag-of-words models. The grid search was performed with the `TfidfVectorizer` and `CountVectorizer` from scikit-learn[3], with different values of the parameters which control the maximum number of features and the n-gram range. All the versions of the vectorizers were used on all 4 versions of the essays. A support vector machine (SVM) model was trained on 80% of the data and tested on the remaining 20% to determine which combination of essay version and text vectorizer was optimal. The vectorizers were fitted only using the training data to avoid data leakage. The best performance was obtained when using the raw essay text with the default `TfidfVectorizer`, which uses unigrams of words and has no limit on the number of features.

The fact that the best performance came from using unprocessed, raw essay text is not surprising. The specific use of punctuation, capitalization and word forms likely carries information about the personality traits of the writer.

We used average word embeddings from essays, using pretrained GloVe[4] embeddings (Pennington et al., 2014), as inputs in our fully-connected models. The inputs to our transformer models were contextualized embeddings from the `BertTokenizer` module from HuggingFace[5]. The contextualized embeddings were made from 200 word subsections of essays, with word overlapping set to 50 words.

5. Models

The given task is framed as a multi-label binary classification problem. As such, we tried several different models befitting of the task, which use the features we had previously extracted, including a random classifier as a baseline model.

Using a variety of models was done in order to tackle the problem from different angles. The simpler models, which use features extracted from the text directly, are more oriented towards the interpretability of the results. The two deep models are meant to provide better results, as well as display the advantage of using word embeddings as opposed to using features from the text - albeit, at the cost of interpretability.

Random Classifier The random classifier is our baseline model for this task. This simple model randomly assigns labels to data without considering any features or

[3] https://scikit-learn.org/stable/

[4] https://nlp.stanford.edu/projects/glove/

[5] https://huggingface.co/

patterns in the text or overall data.

LR The logistic regression (LR) models use the raw essay text converted into unigrams with the `TfidfVectorizer`. The two hyperparameters which were used in the optimization process are the inverse of regularization strength *C*, and the type of regularization which is applied - *L1* (Lasso) or *L2* (Ridge). The hyperparameters were optimized based on the average F1-score they yielded for all personality traits. We used scikit-learn's `LogisticRegression` implementation for our models.

SVM The approach we used for training the SVM models and their hyperparameter optimization is comparable to the LR models. Once again, the raw essay text was converted into unigrams based on the `TfidfVectorizer`, which were used as the models' features. The hyperparameters we optimized were the inverse of regularization strength *C*, and the type of kernel used in the SVM - Linear, Polynomial or Radial Basis Function (RBF). Equivalent to the approach in the LR models, the optimal hyperparameters were chosen based on the average F1-score in regards to all of the personality traits. We used scikit-learn's `SVC` implementation for our models.

FC The fully-connected deep models (FC) consist of (1) the input layer, which uses a 300-dimensional vector as its input features, representing the average value of the word embeddings for a given essay, obtained from GloVe; the output size of the input layer is 500, (2) a rectified linear unit (ReLU) activation function to induce non-linearity, (3) a fully-connected layer with the input size of 500 and output size of 150, (4) a ReLU activation function to induce non-linearity, (5) the output layer: a fully-connected layer with the input size of 150 and output size of 1. The models were trained on each trait separately, using a 60/20/20 train/validation/test split of the data. We used PyTorch[6] for the implementation of our models. The models were trained for 20 epochs using the `Adam`[7] optimizer with the learning rate of $1 \cdot 10^{-3}$.

Transformer Our transformer models are based on the idea presented in Pappagari et al. (2019). The model primarily relies on BERT, but alleviates BERT's weakness of poor performance on longer texts by splitting long input texts into chunks of 200 words with an overlap of 50 words between each. Each of these chunks goes through the BERT model in order to obtain its embedding. We used the aforementioned `BertTokenizer` module from HuggingFace to obtain the contextualized embeddings, as well as the `BertModel` module in tandem with PyTorch for the implementation of these models. The models were fine-tuned for 10 epochs using the `Adam` optimizer with the learning rate of $2 \cdot 10^{-5}$ and without bias correction.

6. Interpretability

The main benefit of using LR and linear kernel SVM models with the bag-of-words representations of text is the interpretability of these models. We performed experiments to see which features the models deemed the most important for classification.

The importance of a feature for a LR or SVM model with a linear kernel can be determined by the magnitude of the feature coefficient (Chang and Lin, 2008). The largest positive values of coefficients correspond to the features which are the most important for scoring high in a certain personality trait, while the largest negative values correspond to the features which are the most important for scoring low in a certain personality trait.

Another way to determine the importance of features is by using Shapley Additive Explanations (SHAP)[8] values, as proposed by Lundberg and Lee (2017). The values represent the impact each feature had on the output of the model for each example.

7. Results

The models used in this paper were tested for interpretability and performance. Interpretability is explained in detail in Section 6.. The performance results of the LR and SVM models were obtained by 5-fold cross-validation, whereas the fully-connected models and the transformer models were tested on 5 separate training and testing runs using different random seeds. The final results were tested for statistical significance using the t-test at the significance level of 5%. All models showed statistically significant improvement over the random baseline.

7.1. Model Performance

The best results were obtained by the transformer models for all personality traits. The LR and SVM models outperformed the fully-connected models on all personality traits except agreeableness, which is perhaps somewhat surprising at first. We conclude that this is the result of average embeddings being a suboptimal feature for a task such as this. The loss of valuable information such as word order and punctuation ultimately led to worse results in the fully-connected models. Exact results can be seen in Table 3.

7.2. Interpretability Using Coefficient Magnitudes

Table 1 shows the 10 unigrams for which the LR and SVM models had the largest coefficient values, while Table 2 shows the 10 unigrams with the smallest coefficient values.

Openness People who are high in openness tend to be more insightful, curious, and have a broad range of interests, while people who score low tend to be more traditional and struggle with abstract thinking. Words like "fun" and "music" were outlined as important in scoring high by both models, while words like "college", "class", "school" and "home", which could be connected to a more scheduled lifestyle without much time for new experiences, were outlined as important in scoring low.

Agreeableness Being high in agreeableness is often characterized by trust, kindness and affection. The inclusion of the word "family" in Table 1 for both models seems to follow this characterization. On the other hand, being low in agreeableness is characterized by competitiveness, manipulation and lack of caring for others. This often

[6] https://pytorch.org/

[7] https://pytorch.org/docs/stable/generated/torch.optim.Adam.html

[8] https://shap.readthedocs.io/en/latest/index.html

manifests by insulting and belittling others, which may be the reason that the words "stupid" and "damn" are considered important for scoring low in both models.

Conscientiousness Scoring high in conscientiousness implies good impulse control, thoughtfulness and organized and goal-oriented decision making. People who score low in conscientiousness are less structured and organized, and often lack impulse control. The words "today" and "tonight", which were deemed important for scoring high in both models, imply structure and orderliness, while words like "want" and "hate", which were deemed important for scoring low in both models, imply low impulse control. Surprisingly, the word "party" appears in Table 1 for both models.

Neuroticism People who score high in neuroticism often experience sadness, moodiness and emotional instability, while people who score low are more stable, emotionally resilient and relaxed. This is very well reflected in the words which were important in scoring high for both models, as most of them are emotionally charged.

Extraversion Being high in extraversion is characterized by sociability, talkativeness, excitability and emotional expressiveness. The words "love", "boyfriend" and "fun", which appear in Table 1 for both of the models, and the word "sorority", which appears only in the LR model, seem to coincide with this. The words related to scoring low in extraversion do not offer such a clear interpretation.

7.3. Interpretability Using SHAP Values

The second part of the interpretation was conducted using beeswarm plots[9] from SHAP values calculated on the test set, which were ordered by several relevance metrics. The SHAP values analysis was performed on the LR models.

Openness We found that high values of the features "home", "semester" and "college" resulted in small SHAP values, while high values of the features "music" and "love" resulted in large SHAP values.

Agreeableness High SHAP values for the agreeableness trait were obtained by high values of the features "family" and "love", while the lowest SHAP values were obtained by high values of feature "mean" and certain profanities. This is to be expected since low agreeableness is often manifested by using more swear words (Ireland and Mehl, 2014).

Conscientiousness High values of pronoun features, such as "he", "she", "her", "we" and "my", obtained very high SHAP values for this trait. This may be explained by the fact that people high in conscientiousness tend to pay attention to detail, so they avoid using unclear words like "someone" to describe people in their stories.

Neuroticism The highest SHAP values were obtained by high values of features "feel" and "stressed". The high value of the feature "blue" also resulted in high SHAP values, which is possibly explained by the use of the phrase "feeling blue" to indicate negative emotion.

Extraversion High values of the features "sleep", "home" and "typing" had the lowest SHAP values, which is expected since they are indicative of a lower desire for socialization. On the other hand, high values of the features "love" and "sorority" had the highest SHAP values.

[9]The mentioned plots, along with other code, can be found here: https://github.com/ilinardic22/TAR-project-2023

Table 3: Performance results for every personality trait expressed in percentages of F1-score. The best results were obtained by the transformer models.

Model	Ext	Agr	Opn	Con	Neu
Random	50.87	50.73	50.94	50.26	50.28
LR	57.27	53.54	61.68	55.88	56.45
SVM	56.95	53.53	62.17	56.28	57.01
FC	53.14	55.48	58.14	55.51	53.60
BERT	**67.21**	**67.31**	**69.16**	**67.18**	**64.36**

8. Conclusion

Personality trait classification based on text has been a challenging, yet vital problem both in the field of natural language processing, as well as modern psychology for quite a long time now. Text is a fickle medium for such a complex task, and the challenge of processing it properly in order to solve such a task has been tackled from various angles and through various means in existing work over the years. We present various methods and their results at attempting to solve this problem, with varying degrees of complexity and success. This paper's main focus, however, is highlighting the importance of interpretability. Being able to understand not only how people think, but how models think and delving deeper into the "why's" and "how's" behind the results was what we deemed to be the most valuable takeaway from this research.

Acknowledgements

We would like to thank prof. dr. sc. Jan Šnajder and mag. ing. Josip Jukić for the advice and help they gave us while writing this paper.

References

Yin-Wen Chang and Chih-Jen Lin. 2008. Feature ranking using linear svm. In Isabelle Guyon, Constantin Aliferis, Greg Cooper, André Elisseeff, Jean-Philippe Pellet, Peter Spirtes, and Alexander Statnikov, editors, *Proceedings of the Workshop on the Causation and Prediction Challenge at WCCI 2008*, volume 3 of *Proceedings of Machine Learning Research*, pages 53–64, Hong Kong, 03–04 Jun. PMLR.

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

Matej Gjurković, Mladen Karan, Iva Vukojević, Mihaela Bošnjak, and Jan Snajder. 2021. PANDORA talks: Personality and demographics on Reddit. In *Proceedings of the Ninth International Workshop on Natural Language Processing for Social Media*, pages 138–152, Online, June. Association for Computational Linguistics.

Molly E. Ireland and Matthias R. Mehl. 2014. 201Natural Language Use as a Marker of Personality. In *The Oxford Handbook of Language and Social Psychology*. Oxford University Press, 09.

O. P. John, E. M. Donahue, and R. L. Kentle. 1991. *The Big Five Inventory—versions 4a and 5*. University of California, Berkeley, Institute of Personality and Social Research.

Elma Kerz, Yu Qiao, Sourabh Zanwar, and Daniel Wiechmann. 2022. Pushing on personality detection from verbal behavior: A transformer meets text contours of psycholinguistic features.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Navonil Majumder, Soujanya Poria, Alexander Gelbukh, and Erik Cambria. 2017. Deep learning-based document modeling for personality detection from text. *IEEE Intelligent Systems*, 32(2):74–79.

Robert R. McCrae and Paul T. Costa. 1985. Comparison of epi and psychoticism scales with measures of the five-factor model of personality. *Personality and Individual Differences*, 6(5):587–597.

Raghavendra Pappagari, Piotr Żelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. 2019. Hierarchical transformers for long document classification.

Gregory Park, H. Schwartz, Johannes Eichstaedt, Margaret Kern, Michal Kosinski, David Stillwell, Lyle Ungar, and Martin Seligman. 2014. Automatic personality assessment through social media language. *Journal of personality and social psychology*, 108, 11.

J. W. Pennebaker and L. A. King. 1999. Linguistic styles: Language use as an individual differnece. *Journal of Personality and Social Psychology*, 77:1296–1312.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Daniele Pizzolli and Carlo Strapparava. 2019. Personality traits recognition in literary texts. In *Proceedings of the Second Workshop on Storytelling*, pages 107–111, Florence, Italy, August. Association for Computational Linguistics.

HA Schwartz, JC Eichstaedt, ML Kern, L Dziurzynski, SM Ramones, M Agrawal, and et al. 2013. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PLoS ONE 8(9): e73791*, September.

Beyond Duplicates: Unleashing Transitivity in Quora Data Augmentation

Renato Jurišić, Mihael Miličević, Josip Srzić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{renato.jurisic,mihael.milicevic,josip.srzic}@fer.hr

Abstract

This study investigates the effects of transitivity-based data augmentation on the Quora Question Pairs (QQP) dataset. Utilizing BERT embeddings, the dense network was trained on both the original and nine augmented datasets. Results indicate slight performance improvements when including duplicate augmentations up to a certain point, but non-duplicate augmentations consistently reduced model performance. These findings suggest that the augmentation method may inadvertently amplify inherent noise in the dataset, complicating the model's learning process. This study illuminates the complexities of data augmentation while also highlighting the importance of mindful strategy when introducing synthetic data to avoid exacerbating the noise present in the dataset.

1. Introduction

With the rise of the Internet, many popular services have come along which allow people to ask questions on various topics and perhaps answer someone else's in return. What often happens is that people do not care or notice that their question has already been answered, causing many duplicate questions to appear. Consistent identification of such duplicates would help the service organise the information it provides, thereby drastically improving the overall user experience.

Nowadays, Natural Language Processing systems are employed to tackle this problem. However, collecting labelled data used to train these systems is a tedious and expensive venture. That is why it is often beneficial to extract as much value from the data you have at your disposal as you can. One way to achieve that is using dataset augmentations, a set of techniques which expand the dataset by performing various transformations on the existing data.

The goal of this research paper is to explore augmentation on the Quora Question Pairs dataset. Each question pair in the dataset is labelled either as a "duplicate" or "non-duplicate". Noticing that many questions appear in multiple pairs, we take advantage of this to generate additional question pairs by applying the transitivity relation across the dataset. Given duplicate question pairs (A, B) and (B, C), we generate a new pair (A, C) and mark it as duplicate with a distance of one. After expanding the dataset, we repeat this procedure iteratively to obtain transitive pairs with higher distances, using all possible transitive relations in each step. In a similar fashion, we generate additional negative ("non-duplicate") examples. We then train a neural network to evaluate the effectiveness of these augmentations.

In this paper, we address several research questions. Firstly, we analyze the impacts of augmenting the dataset with positive question pairs. Secondly, we explore the effects of different transitivity distances on model performance. Lastly, we assess the effects of augmenting the dataset with negative question pairs, in relation to the number of augmented positive pairs, with the intention of preserving the ratio of positive and negative pairs in the dataset.

2. Related work

The dataset used in this paper is Quora Question Pairs (QQP), a large collection of question pairs from the Quora website covering a broad area of topics (Iyer et al., 2017). The duplicate question detection task has also been addressed by earlier studies using this dataset.

In (Prabowo and Herwanto, 2019), they use the QQP dataset to train a Siamese Neural Network consisting of two identical Convolutional Neural Networks, followed by heuristic matching and a fully-connected layer. Pretrained Glove word embeddings are used as inputs to the network. In our paper, we opted for a simpler architecture, putting more emphasis on the effects of the augmentation and less on the model itself.

Slightly different work has been done in (McCreery et al., 2020), where they focus on a specific domain of duplicate question detection. They make use of the QQP dataset for pretraining the model on a more general domain, before fine-tuning it to their specific domain of medical questions. They use the BERT model to accomplish this. We also use BERT to extract features for our question pairs. However, unlike them, we do not fine-tune it but rather precompute the word embeddings for the sake of simplicity.

The closest work to ours comes from (Chen et al., 2020), where they also represent data from QQP as a graph and explore the same idea to augment the original dataset. They use their findings to fix noisy labels in the dataset. We, on the other hand, explore the effects of different transitivity distances on model performance. We want to see whether generated pairs will be of high quality or contain noise due to the way they are created.

3. Augmentation with transitive relations

Our main goal for this research was to explore how set theory and transitivity relations could be utilized for augmenting datasets for problems such as QQP. In set theory, an equivalence relation is a binary relation that is reflexive, symmetric and transitive. In the QQP problem, the question pairs are reflexive (meaning that question A is semantically equivalent to itself) and symmetrical (meaning that if question A is equivalent to question B, then question B is

Text Analysis and Retrieval 2023: Course Project Reports (TAR 2023), pages 28–32, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2023.

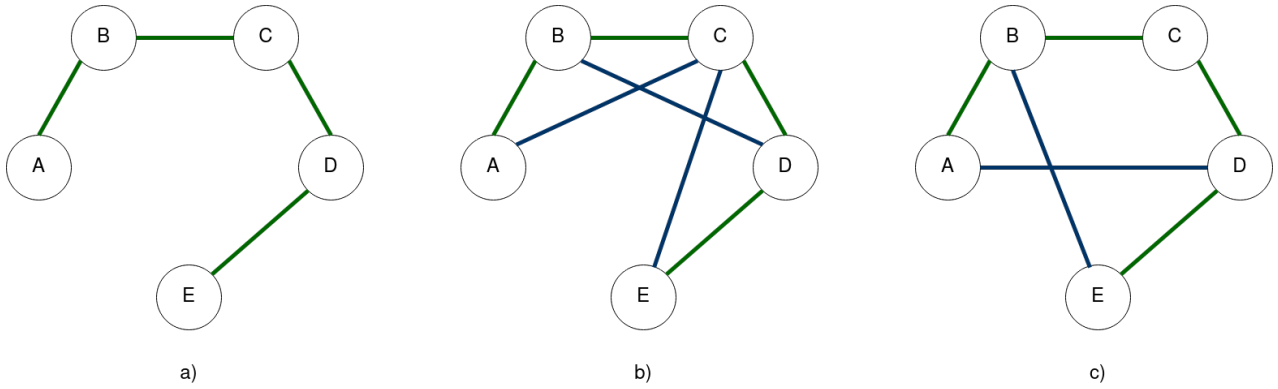


Figure 1: Positive Augmentation. Subfigure a) illustrates the original pairings in the dataset. Subfigure b) shows the potential positive pairs with a distance of one. Subfigure c) further displays the additional positive pairs with a distance of two. Original positive pairs are indicated by green edges, and newly introduced positive pairs are indicated by blue edges.

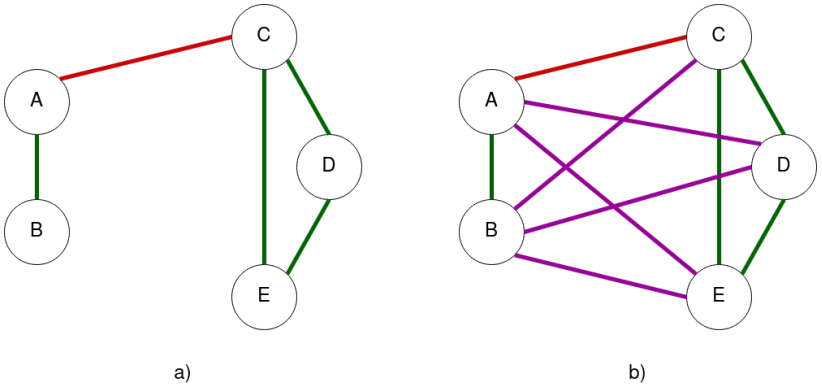


Figure 2: Negative Augmentation. Subfigure a) exhibits the pairs in the original dataset. Subfigure b) highlights the generation of negative pairs. Original positive and negative pairs are indicated by green and red edges respectively, and newly introduced negative pairs are indicated by purple edges.

equivalent to question A) by definition.

Our objective was to approach a partial equivalence relation, rather than seeking to establish a full equivalence. There are two main reasons for this. Firstly, we wanted to explore how different distances affect the model performance. The concept of distances of augmentation can best be explained with a simple example. Let us imagine that in the dataset, we have four questions - A, B, C and D. Question pairs marked as duplicates are (A, B), (B, C) and (C, D). We can apply the transitivity relation, which can be formally expressed as $P(X, Y) \wedge P(Y, Z) \implies P(X, Z)$, where P indicates the duplicates relation, to conclude that $P(A, B) \wedge P(B, C) \implies P(A, C)$ and $P(B, C) \wedge P(C, D) \implies P(B, D)$. So, in the first step of the augmentation, we can extract two new positive pairs, (A, C) and (B, D), with a distance of one. Since questions B and D are now marked as duplicates, in the second step, we can conclude that $P(A, B) \wedge P(B, D) \implies P(A, D)$, with a distance of two. We expect that, with further distances, the quality of the extracted duplicate pairs will decrease, so we wish to examine the effects of the varying number of steps on the model performance. Generating negative examples is conducted in an analogous way - if $P(A, B)$ and $N(B, C)$ hold true, where N denotes the negative (non-duplicate) relation, we can generate a new pair $N(A, C)$. The procedures for generating both positive and negative pairs are visually represented in Figures 1 and 2 respectively.

Secondly, to achieve a formal equivalence relation, each question pair would either have to be marked as duplicate or not duplicate. To comply with this, after the complete augmentation, we would have to label every unlabelled question pair as not duplicate. However, we believe this is not practical, as we admit that the dataset is incomplete, and applying full equivalence would expectedly label some questions with similar intent as not duplicate, thus confusing the model. Another problem with this approach is that the dataset would become very unbalanced, producing many more negative pairs than positive ones, which we also expect would hinder the model performance.

In certain contexts, it is plausible to encounter situations where the transitivity relation may not always hold strictly. For instance, consider three questions A, B and C. Question A could be equivalent to B, and B could be equivalent to C. However, due to nuanced differences in phrasing or context, A might not necessarily be equivalent to C. However, in the QQP problem, we observed that this does not present a significant issue.

The primary objective of our research is to investigate the integration of positive pairs into the dataset, as we be-

Table 1: Examples of augmented question pairs. The first pair represents a positive example with a distance of one. The second pair is also positive, but with a distance of two. The third pair is a positive example with a distance of three or more. Lastly, we have a negative augmented example.

Questions
Q1: How do I develop good sense of humor? Q2: How do I improve sense of humour?
Q1: How can I stop being lazy and useless? Q2: How can I overcome the procrastination problem?
Q1: What are some extremely early signs of pregnancy? Q2: What are the definite pregnancy symptoms?
Q1: Will we ever achieve immortality? Q2: When do you think humanity will become extinct?

lieve this approach holds significant potential for enhancing the model performance. The introduction of negative pairs, while not our main focus, is also considered. The motivation behind this is to examine whether it is necessary to preserve the label ratio of the dataset or if adding negative pairs can provide a beneficial balancing effect.

Several examples of the augmented pairs are presented in Table 1.

4. Experimental setup

To evaluate the effectiveness and potential consequences of our augmentation method, we utilized a BERT model (Devlin et al., 2019) for generating embeddings and subsequently trained a three-layer dense network which takes these embeddings as inputs. This approach involved training the model on both the original Quora Question Pairs dataset and nine additional augmented datasets.

4.1. Datasets

The QQP (Iyer et al., 2017) dataset consists of Quora question pairs labelled as duplicate or non-duplicate. Questions are considered duplicates if they seek identical information. With many sentences appearing in different pairs, this dataset is suitable for our augmentation approach.

The dataset was divided into three subsets, comprising of 344,290 examples for training, 30,000 examples for validation, and 30,000 examples for testing. The validation dataset exclusively served for hyperparameter tuning, while the test set was utilized to evaluate generalization and report the final results.

Augmented examples were generated on the train set, resulting in 48,418 duplicates with a distance of one, 18,990 examples with a distance of two, and all examples with a distance greater than two were grouped together, resulting in 6,621 duplicates in that bucket. The total number of augmented non-duplicate examples was 121,166. Care was taken to eliminate any augmented examples that appeared in the validation or test set to prevent data leakage.

In order to generate additional datasets, positive augmentations were appended to the original train dataset. The first dataset contained positive augmentations with a distance of one, the second included distances one and two, and the third encompassed all positive augmentations. To explore the impact on label balance, six more datasets were created from these initial sets by appending an equal or double amount of negative augmented examples. In total, nine datasets were generated.

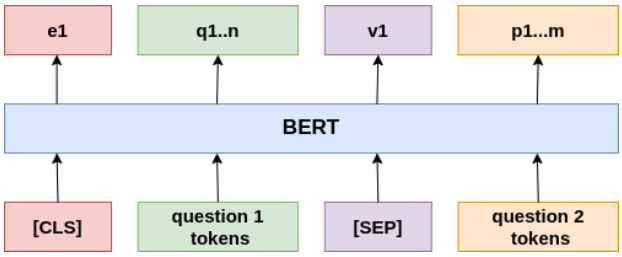


Figure 3: BERT configuration used to get question pair embeddings.

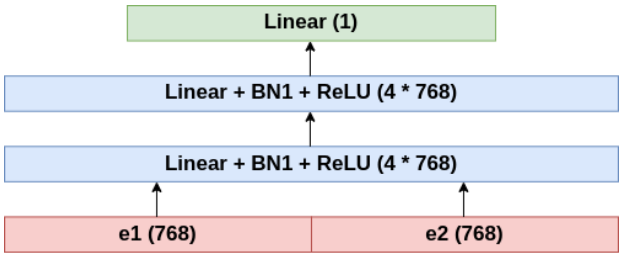


Figure 4: Classifier model trained on top of the question pair embeddings. The dimensions of the output for each layer are indicated in brackets.

4.2. Model

To generate embeddings for question pairs, we employed a BERT model. Following the approach described in the BERT paper (Devlin et al., 2019), we concatenated the two sentences and took the [CLS] embedding **e1** (Figure 3). To mitigate bias related to question ordering, we additionally generated **e2**, where the ordering of the questions was inverted.

The concatenation of two question pair embeddings was passed through a dense network. This network consisted of two sequential layers, each including a Linear layer, Batch Norm 1d layer, and ReLU activation, followed by one last Linear layer with a single output value used for binary class prediction (duplicate or non-duplicate) (Figure 4).

4.3. Training

The model was implemented in PyTorch (Paszke et al., 2019). For training, we utilized binary cross-entropy as the loss function. Hyperparameters were optimized using the validation split and subsequently applied to all 10 training datasets. Tuned hyperparameters together with their final values are: learning rate for Adam optimizer is 0.01, the size of the hidden layers is 4 * 768, the batch size is 32 and the number of epochs is 10.

To optimize computational efficiency, all BERT embeddings were precomputed, and as a result, the BERT model itself was not fine-tuned. In order to mitigate the effects of randomness, we trained the model five times on each of the

Table 2: Average accuracy and weighted F1 score of the model, where the table cell corresponds to the augmented dataset on which the model was trained. The row represents which duplicate augmented examples were added, while the column indicates the ratios of non-duplicate augmented examples with respect to the duplicate augmented examples that were added. The baseline dataset does not contain any augmentations.

Accuracy				Weighted F1			
baseline		.8492		baseline		.8497	
	$\times 0$	$\times 1$	$\times 2$		$\times 0$	$\times 1$	$\times 2$
distance 1	.8495	.8442	.8356	distance 1	.8511	.8443	.8342
distance 1, 2	.8501	.8379	.8331	distance 1, 2	.8509	.8384	.8335
all	.8475	.8419	.8354	all	.8492	.8419	.8346

Table 3: Non-duplicate augmented examples that the model struggled with, together with probabilities assigned by the model.

Questions	Duplicate Probability
Q1: Why does Quora always marks my qustion as needing improvement? Q2: Why does Quora mark my perfectly semantic question as, "Needs Improvement"?	0.9868
Q1: How can I increase in height after 20 years? Q2: What are ways I can increase my height (I'm a ftm Asiam)?	0.9972
Q1: Why do so many people prefer to ask questions on here and wait for an answer rather than type one or two words in a search engine? Q2: Why do people ask questions on Quora when they can easily find the answer for it on Google?	0.9976
Q1: What was the best decision you ever made in life? Q2: What is your most important decision that has made a significant impact on your quality of life today?	0.9880

ten datasets and reported the average model performance on the test set.

The precomputation of embeddings was conducted using an NVIDIA Titan Xp graphics card and required approximately 1.5 hours. Training the model on all 10 datasets was performed 5 times, which took in total around 7 hours on the same GPU.

5. Results

The performance of the model trained on 10 separate datasets is illustrated in Table 2. We observed slight improvements when utilizing only duplicate augmentation examples up to a distance of three. However, including greater distances introduces excessive noise into the dataset, making it less beneficial.

A noteworthy trend evident in the results is the detrimental effect of including non-duplicate augmented examples on the model performance. To delve deeper into this observation, we examined the non-duplicate augmented examples that the model struggled to learn. We sorted these examples based on the probabilities assigned by the model to gain further insights.

Table 3 showcases some examples of question pairs for which the model erroneously assigned a high duplicate probability. A significant portion of the non-duplicate augmented examples consists of edge cases where even humans may find it challenging to determine the appropriate label. These pairs often involve a question that is slightly more specific compared to its counterpart, such as the first example in Table 3: "...my question..." versus "...my perfectly semantic question...". We consider these pairs as noise pairs since their labelling is ambiguous. In inherently noisy datasets like QQP, this method generates a substantial number of such noisy non-duplicate pairs, which provide limited learning opportunities for the model due to the uncertainty surrounding their labelling.

To validate the significance of our results, we conducted unpaired t-tests on the results obtained across five experiments. The results indicate that augmenting the dataset with positive examples yields no statistically significant improvements, while augmenting the dataset with negative examples decreases the model performance.

6. Conclusion

In this paper, we examined the QQP problem and the effects of applying set theory and transitivity relations on the model performance. Building upon the previous research by (Chen et al., 2020), we explored how varying degrees of data augmentation using transitivity relations impact the effectiveness of the model.

Our results show that, on the QQP problem, augmenting the dataset with positive pairs yields no improvements, while augmenting the dataset with negative pairs negatively affects the model performance. Upon investigating the pairs at which our models fail the most, we conclude that the augmentation method magnifies the inherent noise present in the dataset. We suspect this problem to be the main cause of the poor performance of the models trained on the augmented datasets.

For future work, it would be interesting to explore this approach on datasets of different sizes, from different domains. It would also be interesting to compare how different machine learning models would benefit from this augmentation technique.

References

Hannah Chen, Yangfeng Ji, and David Evans. 2020. Finding Friends and flipping frenemies: Automatic paraphrase dataset augmentation using graph theory. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4741–4751, Online, November. Association for Computational Linguistics.

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, June. Association for Computational Linguistics.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First quora dataset release: Question pairs.

Clara H McCreery, Namit Katariya, Anitha Kannan, Manish Chablani, and Xavier Amatriain. 2020. Effective transfer learning for identifying similar questions: matching user questions to covid-19 faqs. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3458–3465.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Damar Adi Prabowo and Guntur Budi Herwanto. 2019. Duplicate question detection in question answer website using convolutional neural network. In *2019 5th International Conference on Science and Technology (ICST)*. IEEE, July.

Traditional vs. Modern: Comparing Approaches for Offensive Language Detection

Maria Krajči, Ana Vladić, Bjanka Vrljić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{maria.krajci, ana.vladic, bjanka.vrljic}@fer.hr

Abstract

Detecting offensive language is an important task when monitoring social media platforms for abusive content. In this paper, we compare traditional and modern approaches to detect offensive language using the OLID dataset, which contains examples of offensive language from social media. For the traditional approach, we used the TF-IDF vectorization technique and other features in combination with Support Vector Machine (SVM). In contrast, for the modern approach, we used transformer-based models, specifically BERT and GPT. To evaluate the performance of these approaches, we measured the macro F1-score as a metric. As expected, the transformer models outperformed the traditional approach, highlighting the superiority of their ability to capture contextual and semantic information. This comparative analysis highlights the advances that transformer models offer in detecting offensive language use and highlights their potential to improve the effectiveness of content moderation on social media platforms.

1. Introduction

Social media platforms are a valuable communication tool but also expose users to offensive content. Offensive language detection is a Natural Language Processing task that aims to automatically identify such language in text. Terms like abusive language, profanity, cyberbullying, and hate speech are used interchangeably, lacking consensus (Waseem et al., 2017). Exposure to offensive language negatively impacts mental health, necessitating efficient detection (Chen et al., 2012). Manual review is impractical, and lexicon-based filtering generates many false positives (Chen et al., 2012). Modern approaches and deep learning models offer solutions, the latter using context and having improved results. On the other hand, traditional approaches with shallow models and manual feature extraction also provide a solution, which has the advantage of faster computation and lower architecture requirements. Our work aims to compare the traditional approach based on feature extraction and a shallow classifier with modern transformer-based approaches (BERT and GPT models). We train and evaluate the models on OLID (Zampieri et al., 2019a), a dataset of annotated Twitter posts, and compare their performance. Our goal is to guide future research by determining the most effective approach for detecting offensive language use based on the OLID dataset.

2. Related Work

OLID was the official dataset for the SemEval-2019 Task 6: Identifying and Categorising Offensive Language in Social Media (Zampieri et al., 2019b). Participants used different pre-processing techniques such as normalizing the Twitter-specific elements, converting emojis to text and removing stopwords. They used traditional machine learning models and deep learning models such as CNN, RNN, BiLSTM, and state-of-the-art model BERT (Devlin et al., 2019). Top five submitted models for the task of detecting offensive tweets used BERT. The top performing team (Liu et al., 2019) fine-tuned the BERT base-uncased with default parameters and achieved the macro-averaged F1-score of 0.829. The SVM baseline introduced in (Zampieri et al., 2019a), which is a linear SVM model trained on word unigrams, achieved the F1-score of 0.69.

Traditional machine learning models have been used in previous work. Davidson et al. (2017) used Logistic Regression and Linear SVM models to categorize tweets by offensiveness. As features they used n-grams, readability scores, counts of Twitter-specific elements such as hashtags, mentions and URLs, and a sentiment lexicon to assign sentiment scores. Posts from other social media have been studied as well. Chen et al. (2012) used features based on a custom offensive word lexicon, syntactic features, and style features such as sentence length and appearance of punctuation or uppercase words to detect offensive YouTube comments.

Caselli et al. (2021) trained a pre-trained BERT model on a dataset collected from the social media platform Reddit. The Reddit Abusive Language English dataset (RAL-E) comprises of 1,478,348 messages posted on banned subreddits. They trained the English BERT base-uncased model by applying the Masked Language Model (MLM) objective. They evaluated the model on three offensive language datasets, including the OLID dataset, on which the model achieved macro-averaged F1-score of 0.809. The model outperformed the base BERT model on all tested datasets.

3. Dataset

The Offensive Language Identification Dataset (OLID) was introduced by Zampieri et al. (2019a). The dataset contains 14,100 English tweets further divided into a train set of 13,240 tweets and a test set of 860 tweets. The tweets were manually annotated using a hierarchical labeling schema with three levels for three subtasks. Level A deals with whether the text is offensive (OFF) or not (NOT). Level B identifies whether the offensive text has a specific target (TIN) or not (UNT). Level C identifies the target, which can be an individual (IND), a group (GRP), or other type of tar-

Text Analysis and Retrieval 2023: Course Project Reports (TAR 2023), pages 33–37, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2023.

Table 1: Number of examples labeled as offensive (OFF) and non-offensive (NOT) in OLID train and test set.

Class	Train	Test
OFF	4400	240
NOT	8840	620

get (OTH). Our work is focused on the subtask A, identifying offensive tweets. The distribution of the level A classes is shown in Table 1. Tweets in the OLID dataset were pre-processed by masking user mentions (@username) and URLs with generic tags "@USER" and "URL".

4. Approach

4.1. Traditional

To pre-process the text we used Ekphrasis (Baziotis et al., 2017), a tool for tokenization, word normalization, hashtag segmentation and spelling correction based on word statistics from a corpus of 330 million English tweets. Using the list of English stopwords provided in the NLTK library (Bird et al., 2009) we removed stopwords from the tweets and then stemmed them with NLTK's `PorterStemmer`. During this process we also removed emoticons. Example of a tweet processed with Ekphrasis and stemmed is shown in Table 2.

We extracted features and trained the model using the Scikit-learn library (Pedregosa et al., 2011). We chose a subset of features from (Davidson et al., 2017). We use the count of characters, words and syllables in the tweet, and measure the quality of each tweet with modified Flesch-Kincaid Grade Level and Flesch Reading Ease scores, where the number of sentences is fixed to 1. We extended the feature set to include the count of punctuation and counts of following elements which were tagged by the Ekphrasis pre-processor: {<url>, <user>, <hashtag>, <repeated>, <elongated>, <censored>, <allcaps>, <emphasis>, <time>, <date>, <number>, <percent>, <money>}. We also used the VADER sentiment analysis tool (Hutto and Gilbert, 2014) to extract positive, negative, neutral and compound sentiment scores for each tweet. We scaled the features using Scikit-learn's `MinMaxScaler`.

We extracted word n-gram features using `TfidfVectorizer`. We experimented with different levels of pre-processing (removing stopwords, stemming, mapping emoticons to words with the Emoji library[1] + stemming), n-gram ranges $ngram\_range = \{(1,2),(1,3)\}$ and values of the SVM's regularization parameter $C = \{0.1, 1, 10\}$. To select the model hyperparameters we split the OLID train set into a smaller validation_train set and a validation set while preserving the percentage of samples for each class in the OLID train set. The distribution of classes after the split is shown in Table 3. We iterated over the hyperparameters, trained a linear SVM model for each combination on the validation_train set and evaluated it on the validation set. We selected the model with the highest macro-averaged F1-score, which was achieved for stemmed text and the following hyperparameters: $C = 1$, $ngram\_range = (1,2)$. We trained two models with those hyperparameters using the entire OLID train set: one with only TF-IDF features, and the other with concatenated TF-IDF and other features. We evaluated both models on the OLID test set.

4.2. Transformers

4.2.1. BERT

BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) is a modern NLP model developed by Google. It has revolutionized various NLP tasks through its transformer architecture (Vaswani et al., 2017) and its innovative pre-training and fine-tuning approach. The main feature of the model is its bidirectional encoding, which allows it to consider both preceding and following words to capture deep contextual information. The transformer-based architecture and self-attention mechanisms enable BERT to model word relationships and understand the semantics of texts.

4.2.2. GPT

GPT (Generative Pre-trained Transformer) (Radford et al., 2018) is a series of transformer-based language models developed by OpenAI. The GPT models are designed to generate human-like text based on the input they receive. As with BERT, GPT models are trained using a two-step process: pre-training and fine-tuning. Unlike BERT, GPT models are unidirectional and generate text based solely on the preceding words.

4.2.3. Implementation

Both transformer models were implemented using the Transformers[2] library, an open-source Python library developed by Hugging Face that supports framework interoperability between PyTorch, TensorFlow, and JAX. We used it for tokenization, loading pre-trained models and fine-tuning them on the OLID dataset.

Both transformer models were trained using Google Colab server with architecture comprising of 2 CPUs, 12 GB of RAM and a Tesla T4 GPU with 16GB of GPU RAM. Training time for one epoch of BERT model was approximately 20 minutes and training time for one epoch of GPT model was approximately 25 minutes.

4.2.4. Optimization

Hyperparameter optimization is critical for transformer training. Hyperparameters such as learning rate and training epochs have a significant impact on model performance and convergence. A range of 1e-5 to 5e-5 was investigated for learning rate optimization, while taking into account dataset, model complexity, and training dynamics. Random seed selection affected model reproducibility and generalization, with a range of 1 to 41 used in the optimization process. An early termination procedure was used to determine the optimal number of epochs, stopping when performance on validation set did not improve from last epoch or

[1] `https://github.com/carpedm20/emoji`

[2] `https://huggingface.co/docs/transformers`

Table 2: Example of a pre-processed tweet using Ekphrasis pre-processor and NLTK's `PorterStemmer`.

original	Go home you're drunk!!! @USER #MAGA URL
processed	go home you are drunk ! <repeated> <user> <hashtag> maga </hashtag> <url>
stemmed	go home drunk maga

Table 3: Number of examples labeled as offensive (OFF) and non-offensive (NOT) in validation_train and validation sets used for hyperparameter selection in the traditional approach.

Class	Validation_train	Validation
OFF	3666	734
NOT	7367	1473

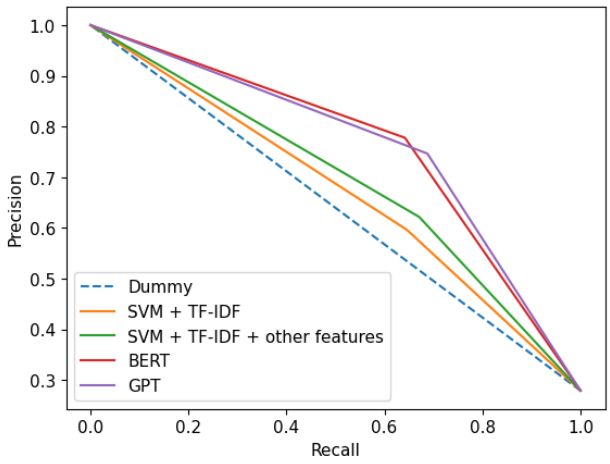


Figure 1: Precision-recall curve for all models.

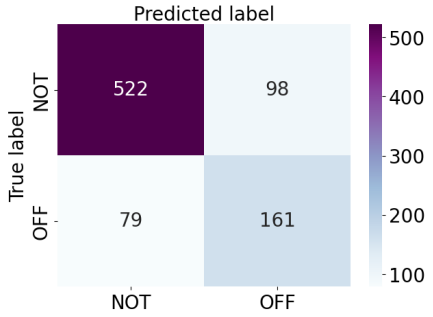


Figure 2: Confusion matrix for SVM + TF-IDF + other features.

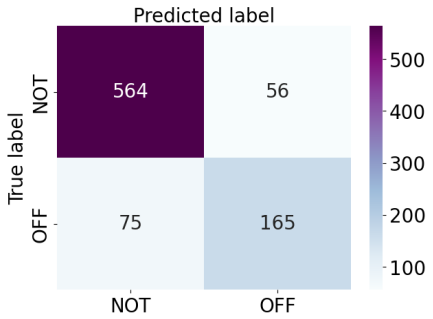


Figure 3: Confusion matrix for GPT.

after a maximum of 5 epochs. This prevented overfitting and found a balance between training progress and model performance.

Three trials were performed, with learning rates uniformly within the specified range and seeds randomly selected for each trial. A 90:10 split between training and validation was used to evaluate the optimized models and it is different split than the split used in Section 4.1. The validation set served as the basis for early stopping and hyperparameter selection. The unbiased evaluation of the final model performance came from the test set, which remained unseen during optimization.

5. Results and Discussion

All models were evaluated on the test set using both macro-averaged and weighted-averaged metrics (precision, recall, F1-score), as well as accuracy. Macro-averaged metrics give equal importance to all classes when calculating the average, while weighted-averaged metrics weigh each class's contribution to the average based on its size (prevalence in the dataset). As per the official competition (Zampieri et al., 2019b), we used the macro-averaged F1-score as the deciding metric. Metrics are shown in Table 4. Figure 1 shows the precision-recall curve for all models. It shows the trade-off between precision and recall for different classification thresholds. For the baseline model, we implemented a dummy classifier[3] which always predicts the most frequent class.

The results clearly demonstrate that each model outperformed the dummy classifier and that transformer-based models outperformed the SVM models. We further compared the best SVM model (SVM + TF-IDF + other features) with the best transformer model (GPT) according to the macro-averaged F1-score. Figure 2 and Figure 3 show the confusion matrix for SVM and GPT, respectively. We observe that the SVM model has more false positives than the GPT model. That being said, our SVM model has better macro-averaged F1-score than the SVM baseline introduced in (Zampieri et al., 2019a), which achieved the macro-averaged F1-score of 0.69.

[3] `sklearn.dummy.DummyClassifier`

Table 4: Macro-averaged and weighted-averaged metrics evaluated on the test set for different models. The best results for each metric are in bold. Using macro F1-score for comparison it is shown that the best model is GPT.

		Macro			Weighted		
Model	Accuracy	Precision	Recall	F1-score	Precision	Recall	F1-score
dummy	0.7209	0.3605	0.5000	0.4189	0.5197	0.7209	0.6040
SVM + TF-IDF	0.7791	0.7272	0.7382	0.7321	0.7852	0.7791	0.7817
SVM + TF-IDF + other features	0.7942	0.7451	0.7564	0.7502	0.7996	0.7942	0.7965
BERT	**0.8488**	**0.8239**	0.7853	0.8009	0.8443	**0.8488**	0.8441
GPT	0.8477	0.8146	**0.7986**	**0.8059**	**0.8447**	0.8477	**0.8457**

Table 5: Examples of test tweets and predictions made by the models GPT and SVM + TF-IDF + other features. Examples show cases where one model correctly predicted while the other model incorrectly predicted the label.

Text	True	GPT	SVM
@USER @USER You do not like the Jewish people because you ALL claims they are holding all the money. Sound familiar? Your left wing libtard family is the family of the KKK. You do not hate Jews? That's funny. Gun control; even funnier. Good luck	OFF	NOT	OFF
#SerenaWilliams is so full of herself...she is just as painful to watch as to listen to...	OFF	NOT	OFF
@USER #FakeNewsMedia is the #EnemyOfThePeople! They want to control YOUR minds by controlling what information you get and do not get. The #1stAmendment may protect #elitist #propaganda but it does not legitimize it or make it palatable to an engaged public. URL	NOT	NOT	OFF
Aparently my computer doesn't want to operate at the moment. So I will upload the episodes Individually. Sorry for the inconvenience.	NOT	NOT	OFF

Table 5 provides examples of test tweets and the corresponding conflicting predictions of the models GPT and SVM + TF-IDF + other features. The table shows the cases in which the two models disagreed in their predictions, highlighting both offensive and non-offensive tweets. The first two examples were misclassified by the GPT model, the reason for which is difficult to explain due to the complexity of deep models. The third example is mislabeled by SMV because of inference based on the n-gram that the label is offensive. The fourth example shows that the SVM model misclassifies completely innocuous tweets as offensive because SVM is not able to understand the context, whereas transformer-based models are.

Table 6 shows the distribution of predicted tweets for both models. It shows in how many cases the models gave the same prediction, and how often one model outperformed the other. There are almost twice as many instances in which the GPT model gave the correct prediction while SVM did not than vice versa.

Table 6: Number of examples in predicted results (GPT and SVM + TF-IDF + other features) for different cases (e.g. both models predicted the correct label).

Description	Count
all test data	860
both correct	631
both incorrect	79
GPT correct and SVM incorrect	98
GPT incorrect and SVM correct	52

6. Conclusion

This work compared traditional and modern approaches for detecting offensive language. Traditional methods involve manual feature extraction and shallow classifiers, which offer faster computation and lower architectural requirements. Modern approaches use transformer-based models such as BERT and GPT, which leverage context and produce better results.

We trained and evaluated our models on the OLID dataset. Results show that all of our models perform better than the baseline. We compared the best SVM model (SVM + TF-IDF + other features) to the best transformer model (GPT) using the macro-averaged F1-score and analyzed the cases in which their predictions disagree. The results show that the transformers outperform the traditional approach and provide the direction for future research in offensive language detection.

References

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. DataStories at SemEval-2017 Task 4: Deep LSTM with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada, August. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Tommaso Caselli, Valerio Basile, Jelena Mitrović, and Michael Granitzer. 2021. HateBERT: Retraining BERT for abusive language detection in English. In *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, pages 17–25, Online, August. Association for Computational Linguistics.

Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 71–80.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515.

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, June. Association for Computational Linguistics.

C. Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. *Proceedings of the International AAAI Conference on Web and Social Media*, 8(1):216–225, May.

Ping Liu, Wen Li, and Liang Zou. 2019. NULI at SemEval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 87–91, Minneapolis, Minnesota, USA, June. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Vaswani, Shazeer, Parmar, Uszkoreit, Jones, N. Gomez, Kaiser, and Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008. IEEE Transactions on Neural Networks, December.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding abuse: A typology of abusive language detection subtasks. In *Proceedings of the First Workshop on Abusive Language Online*, pages 78–84, Vancouver, BC, Canada, August. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the type and target of offensive posts in social media. 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 1415–1420, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA, June. Association for Computational Linguistics.

Named Entity Recognition for Efficient Clinical Concept Mapping in Patient Notes with DeBERTa, RoBERTa, and BioClinical BERT

Tomislav Krog, Tihomir Pavić, Matija Sever

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
tomislav.krog@fer.hr, tihomir.pavic@fer.hr, matija.sever@fer.hr

Abstract

In this study, we conduct a comparative analysis of DeBERTa, RoBERTa, and BioClinical BERT for keyphrase detection in patient notes. It is important to note that the term 'keyphrase detection' in this context is synonymous with Named Entity Recognition (NER), a critical aspect of our research focus. The task at hand involves mapping clinical concepts from exam rubrics to their diverse expressions within patient notes, a crucial step towards enhancing the transparency and ease of administration in patient note scoring. In a fresh and innovative approach, we give DeBERTa a head start by pretraining it on patient notes, boosting its understanding of the clinical context. Our hands-on findings show that this pretrained DeBERTa outshines its counterparts, proving the strength of its advanced architecture and our pretraining strategy. This study shines a spotlight on the potential of DeBERTa pretraining for Named Entity Recognition tasks and nudges us towards more exploration in this exciting field.

1. Introduction

In the ever-evolving landscape of medical text analysis, the task of keyphrase detection from clinical notes has emerged as a critical challenge. This task, which is essentially a form of Named Entity Recognition (NER), involves mapping clinical concepts from exam rubrics to their diverse expressions in patient notes. This is a crucial step towards making patient note scoring more transparent and easier to administer, thereby enhancing the overall quality of healthcare delivery.

Recent advancements in the performance of large language models (LLMs) such as DeBERTa, RoBERTa, and BioClinical BERT have opened up new avenues for research in this field. However, these models may not always accurately capture the intended clinical concepts, raising questions about the validity of their outputs (Gu et al., 2020).

In this study, we propose a novel approach to address this issue. We pretrain DeBERTa on patient notes, thereby enhancing its understanding of clinical context. Our findings reveal that DeBERTa outperforms its counterparts, underscoring its potential in the field of medical text analysis and keyphrase detection. This study invites further exploration in this promising field, with the hope of paving the way for more accurate and reliable methods of keyphrase detection in clinical notes.

2. Related Work

The field of medical text analysis, particularly keyphrase detection from clinical notes, has seen significant advancements in recent years. Several studies have explored the use of large language models (LLMs) such as DeBERTa, RoBERTa, and BioClinical BERT for this task (Gu et al., 2020). However, these models often struggle with accurately capturing the intended clinical concepts, leading to questions about the validity of their outputs.

Named Entity Recognition (NER), a critical component of keyphrase detection, has been extensively studied since its introduction by Roche and Schabes (1995). Various approaches have been proposed to improve the performance of NER systems, including rule-based methods, machine learning techniques, and more recently, transformer-based models. In the realm of healthcare, NER has been used for extracting medical concepts from clinical notes, with models often trained on annotated medical corpora (Jagannatha and Yu, 2016).

However, these models typically struggle with the variability in the way clinical concepts are expressed in patient notes. Our work differentiates itself by focusing on the comparative analysis of DeBERTa, RoBERTa, and BioClinical BERT for the task of keyphrase detection from patient notes. Furthermore, we propose a novel approach of pretraining DeBERTa on patient notes to enhance its understanding of clinical context, a strategy not explored in previous studies.

3. Dataset

Our dataset is comprised of text data derived from the USMLE Step 2 Clinical Skills examination,[1] a licensure exam designed to assess a trainee's capability of recognizing essential clinical facts during encounters with standardized patients.

During the examination, each test taker interacts with a Standardized Patient, an individual trained to simulate a clinical case. Following the interaction, the test taker documents the significant facts of the encounter in a patient note. Each of these notes is scored by a trained physician, who seeks the presence of certain key concepts or features pertaining to the case as outlined in a rubric.

In the context of this study, a 'Feature' signifies a clinically relevant concept associated with each case. The 'Patient note' is a text document created during an interview

[1] Kaggle. (2022). NBME Clinical Patient Notes Competition. Retrieved from https://www.kaggle.com/competitions/nbme-score-clinical-patient-notes/data.

Text Analysis and Retrieval 2023: Course Project Reports (TAR 2023), pages 38–41, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2023.

with a standardized patient, containing key phrases that are relevant to the feature. 'Annotation' refers to the words or key phrases our system detects based on the feature, while 'Location' indicates the position of these key phrases within the patient note. These components of the patient note and their interactions are visually represented in Figure 1.

The primary goal of this study is to develop an automated approach for identifying the relevant features within each patient note, particularly concentrating on the patient history sections where the information from the interview with the standardized patient is documented.

The training dataset comprises roughly 40,000 portions of Patient Note history, only a subset of which have annotated features. This dataset also includes a file of the relevant features for each clinical case. Following the integration of each note with its respective annotated feature, a training dataset of 14,300 instances was curated and then divided into separate subsets for training, validation, and testing.

The test set accounted for 15% of the data, while from the remainder, 10% was allocated for the validation set. This strategy ensured a robust framework for model evaluation and fine-tuning. Furthermore, each feature was accompanied by a description, with annotations within each note denoting a feature. This included the text and character spans indicating each annotation's location within the note.

For the dataset of 14,300 data entries, several challenges were encountered. Among these, 4,399 instances did not have feature annotations, a situation that could potentially cause class imbalance and predispose the model to predicting 'no feature' frequently. Moreover, with 131 unique entities in the dataset, some entities could be underrepresented, potentially reducing the model's accuracy in recognizing them.

This study obtained approval from an academic institutional review board, ensuring compliance with ethical guidelines for scientific research. All participants consented to the use of their patient notes for research purposes.

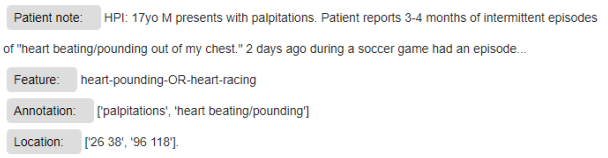


Figure 1: Illustration of the Key Components of the Dataset.

4. Experimental Setup

The preprocessing of data for training the model was a multi-step process, each designed to optimize the data for our specific task. Initially, the locations in the dataset, which correspond to the positions of words representing the feature in patient notes, were transformed into a list of tuples. Each tuple encapsulated the start and end positions of a word representing a feature within the note.

Subsequently, patient notes and associated words representing the feature were tokenized, adhering to a maximum length constraint. This process employed a tokenizer, which generated token sequences and attention masks. The token sequences are numerical representations of the input text, while the attention masks serve to guide the model's focus towards the relevant parts of the input.

In addition, labels were prepared for each word representing a feature in the patient note. These labels were vectors composed of zeros and ones, indicating the absence or presence of a word representing a feature respectively. This process involved mapping each character's location in the text to its corresponding token. If a word representing a feature fell within a token's span, the token was labeled as one, otherwise, it was labeled as zero. Figure 2 visually represents how key phrases are found in a patient note based on the given feature.

Finally, all these processed elements were consolidated into a data structure that included the token ID sequences, attention masks, and labels if applicable. This formed the ready-to-use data for the model. This comprehensive process was performed on each entry in the input dataframe, ensuring uniform preprocessing across the entire dataset, thereby enhancing the consistency and reliability of our model's training.

5. Architecture

The architecture of our model is specifically designed to cater to our task of keyphrase detection. It begins with an input layer that accepts the tokens in their numerical representation, followed by another input layer designed for attention masks. These layers are crucial for processing the tokenized patient notes and directing the model's focus to the relevant parts of the input.

The next component of our architecture is a pre-trained language model, which could be RoBERTa, DeBERTa, or BioMedicalBERT, depending on the model under consideration. These models, downloaded from HuggingFace, serve as the backbone of our architecture, providing a robust foundation for our task.

To prevent overfitting, a Dropout layer with a probability of 0.2 is included in the architecture. This layer helps in creating a more generalized model by randomly setting a fraction of the input units to zero during each update in the training phase.

The final component of our architecture is a Dense layer with a single neuron and a Sigmoid activation function. This layer assigns a probability of being labeled as zero or one to each token, aligning with the format of our labels.

This architecture allows us to evaluate our models effectively by comparing the predicted labels with the true labels of tokens for each patient note.

6. System Overview

In our study, we focused on refining several large transformer models using a provided training set. We relied on the HuggingFace library (Wolf et al., 2020), to download and train these models.

Our selection included DeBERTa base, RoBERTa base, BioClinicalBERT, and we also took an extra step to pretrain the DeBERTa base model. After the training phase,

we put each model to the test and thoroughly compared the outcomes.

This approach, combining fine-tuning with careful performance comparison, is the core of our study, providing a versatile and effective way to examine and boost transformer models' effectiveness in real-world scenarios.

6.1. BioClinicalBERT

BioClinical BERT is a domain-specific model designed for clinical text (Alsentzer et al., 2019). It has been demonstrated that using a domain-specific model like BioClinical BERT can yield performance improvements on several common clinical NLP tasks.

6.2. RoBERTa base

RoBERTa, a robustly optimized BERT pretraining approach, was also employed in our study (Liu et al., 2019). RoBERTa modifies key hyperparameters in BERT, including removing BERT's next-sentence prediction (NSP) pretraining objective, and training with much larger mini-batches and learning rates.

6.3. DeBERTa base

Our first approach involved training a DeBERTa model on the provided data (He et al., 2021). We utilized a DeBERTa-base model and trained it for ten epochs at a learning rate of 2e-5 with Adam optimizer on the competition training data. This model achieved a precision of 0.874, a recall of 0.856, and an F1-score of 0.865.

6.4. Pretrained DeBERTa base

To improve the performance of the DeBERTa model, we pre-trained it on patient notes using a Masked Language Modeling (MLM) approach to enhance its understanding of the clinical context. MLMs, by their design, learn to understand the context of a word in a sentence. This is particularly useful in NER, where the meaning and type of an entity often depend on its context. We employed a distinct set of patient notes, not utilized in the testing phase, as the foundation for pre-training our model. This approach ensured an unbiased refinement of the model, further enhancing its capability to generalize and perform effectively on unseen data. This strategy is novel and has not been explored in previous studies but has proven successful.

7. Results

In our evaluation, we primarily relied on the micro F1 score as the main metric. As described in the Section 4, the output of our model is essentially a binary classification of tokens. For a token that the model deems relevant in patient notes for a given feature, that token is assigned a label of 1, while all other tokens are assigned a label of 0. Notably, there can be multiple tokens with a label of 1, and these can be located at various positions. A more detailed description of token labeling is also provided in the Section 4.

We performed token labeling of patient notes based on the actual positions given in the dataset, and compared this labeling with the labels assigned by our model for that patient note. For each token, we then examined whether the correct label was assigned. Given that for a single labeled patient note, there are many more labels of 0, we concluded that accuracy is not a good metric for evaluation.

This is where the micro F1 score comes into play. The micro F1 score aggregates the contributions of all classes to compute the average metric. It is calculated by counting the total true positives, false negatives, and false positives.

Our results, as shown in Table 1, revealed that the pretrained DeBERTa model outperformed the other models. This underscores the effectiveness of our strategy of pretraining DeBERTa on patient notes to enhance its understanding of the clinical context. The superior performance of DeBERTa indicates its potential for keyphrase detection tasks in the medical domain, and invites further exploration of this approach.

Table 1: Summary of the scores of all the models tested in this paper.

Model	Precision	Recall	micro-F1
DeBERTa base	**0.874**	0.856	0.865
Pretrained DeBERTa base	0.857	**0.898**	**0.877**
RoBERTa base	0.849	0.863	0.856
BioClinicalBERT	0.863	0.820	0.841

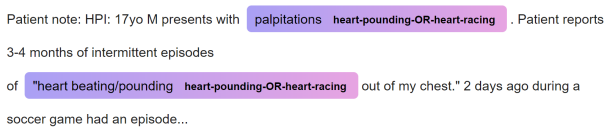


Figure 2: Based on the patient note and the feature shown in Figure 1 as an input to our model, Figure 2 visually represents how key phrases are found in a patient note based on the given feature 'heart-pounding-OR-heart-racing'.

8. Potential Improvements

This section outlines potential advancements for future research in the application of Named Entity Recognition (NER) models, particularly in the context of patient notes. The current study employs BioClinical BERT, RoBERTa, and DeBERTa, as well as DeBERTa pretrained with Masked Language Modeling (MLM). We aim to identify several areas where our models could be further improved (Chen et al., 2021).

Context Augmentation: To address the challenge of semantically ambiguous entities in patient notes, a knowledge-based system can be used. This system, potentially built on resources like Wikipedia, can provide related context information, improving token representations and entity discernment.

Adversarial Training: This approach leverages unlabeled data and enhances results. It involves an encoder learning entity knowledge from labeled data, and a discriminator selecting less language-dependent data, leading to better generalization across diverse datasets.

Error-Aware Training: Given the potential for typos and errors in patient notes, introducing noise into the training data can enhance the model's robustness against such imperfections.

Ensemble Models: Combining predictions from several models can enhance the robustness and overall performance of the NER system.

9. Conclusion

In this study, we presented a comparative analysis of DeBERTa, RoBERTa, and BioClinical BERT, three transformer models, for keyphrase detection from patient notes. Our approach involved fine-tuning these models on the provided training data, with a particular focus on enhancing the understanding of clinical context.

Our findings revealed that DeBERTa, especially when pretrained on patient notes, outperformed its counterparts. This outcome underscores the potential of DeBERTa in the field of medical text analysis and keyphrase detection, and highlights the benefits of our pretraining strategy.

In conclusion, this study contributes to the rapidly evolving field of medical text analysis by providing insights into the performance of different transformer models on a critical task. It invites further exploration and underscores the potential of advanced transformer models, particularly DeBERTa, in improving the transparency and administration of patient note scoring.

References

E. Alsentzer, J. R. Murphy, W. Boag, W. H. Weng, D. Jin, T. Naumann, and M. McDermott. 2019. Publicly available clinical bert embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78.

Weile Chen, Huiqiang Jiang, Qianhui Wu, Börje F. Karlsson, and Yi Guan. 2021. Advpicker: Effectively leveraging unlabeled data via adversarial discriminator for cross-lingual ner. *arXiv preprint arXiv:2106.02300*.

Yu Gu, Rishi Tinn, Huanrui Cheng, Mike Lucas, Naoto Usuyama, Xiaodong Liu, Tobias Naumann, Jianfeng Gao, and Hoifung Poon. 2020. Domain-specific language model pretraining for biomedical natural language processing.

P. He, X. Liu, W. Chen, and J. Gao. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2970–2978.

A. Jagannatha and H. Yu. 2016. Structured prediction models for rnn based sequence labeling in clinical text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 856–865. Conference on Empirical Methods in Natural Language Processing.

Kaggle. 2022. Nbme clinical patient notes competition.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Emmanuel Roche and Yves Schabes. 1995. Deterministic part-of-speech tagging with finite-state transducers. *Computational Linguistics*, 21(2).

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, October. Association for Computational Linguistics.

The Depth of Deception: Shallow, Deep, and Hybrid Learning Approaches to Unmasking Fake Text

Lea Krsnik, Mislav Perić, Leon Zrnić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{lea.krsnik, mislav.peric, leon.zrnic}@fer.hr

Abstract

Text generation has become extremely popular in the last couple of months with the rise of ChatGPT and similar models. Because of this rise, automatic generated text detection has become more important as well. However, recent surveys do not focus on the difference in types of models for text classification. In this paper, we analyze the different feature extraction methods and types of models that can be used for generated text classification on the AuTexTification dataset. We find that a hybrid model which uses RoBERTa to extract text representations with additional text features yields the best results when compared to regular shallow and deep models. We conclude that additional research should further explore hybrid models and that more heuristically picked features should be added to future papers.

1. Introduction

Recent advancements in automatic text generators have sparked fear and wonder alike in the world. Specifically, the sudden rise of ChatGPT[1] has caused a surge in use of generative models for broad text generation. From helping programmers code quicker to fake news and student essay writing, the field's use has exploded with the general public. Because of this widespread use, for both good and malicious usage, the need for automatic generated text detection has become great. In the field of natural language processing (NLP), many classifiers for text have already been developed. However, few classifiers have been specialized for the task of detecting generated text, let alone for the recent generative models. Furthermore, the surveys that summarize generated text classifiers do not focus on comparing shallow and deep text detection models.

As such, the goal of this paper is to explore a wide variety of feature extraction methods and different models while also comparing shallow and deep text detection. It is important to note that the focus is not to produce the best results for one model but rather to analyze the differences between feature extraction methods and types of models with the goal of binary text classification.

The paper is organized as follows. Section 2. summarizes recent work regarding automatic detection models for generated texts. In Section 3. we give a description of the AuTexTification dataset and what was used for the scope of this paper. Afterward, Section 4. goes into detail about the different feature extraction methods and models that were used in the paper. We show the binary classification results of the models in Section 5. and discuss them in Section 6.. Lastly, we conclude the findings of our paper in Section 7..

2. Related Work

Research regarding automatic detection of machine-generated text is lacking. Few authors have done reviews of automatic detection models and their methods (Beresneva, 2016; Jawahar et al., 2020) and even fewer have done so recently (Crothers et al., 2023). Furthermore, these surveys do not focus on comparing shallow and deep text detection models, which is the focus of this paper. Besides this, new research on automatic detection models is required given the recent advances in text generation models such as OpenAI's ChatGPT.

3. Dataset

The AuTexTification: Automated Text Identification[2] is a shared task in which contestants make automatic generated text detection models for two subtasks. The first subtask is binary classification between human and machine-generated text while the second subtask is attributing each text to the model that generated it. The texts come from five different domains, three of which are used in the train dataset ('legal', 'tweets', and 'wiki') and two for the test dataset ('news' and 'review'). The train and test datasets are given for English and Spanish since the task is also to do cross-lingual detection.

The scope of this paper encompasses the datasets for binary classification of English texts. In contrast to the contestants, our goal is to explore the difference between shallow and deep classification models whereas they focus on making a cross-lingual detection model with a high F1-score. The train set contains 33845 texts, 17046 of which are human and 16799 are generated. The test set contains 21832 texts with 10642 being human and 11190 generated.

4. Methods

4.1. Preprocessing

For preprocessing, we started by expanding common abbreviations from their short form. The list of common abbreviations was given to us by ChatGPT. Afterward, we used standard NLP preprocessing methods such as replacing URLs with special URL tokens using regular expressions, removing punctuations, numbers, emojis, and stopwords. We also expanded common language contractions and used NLTK's `wordnet`[3] to replace elongated words

[1] https://openai.com/blog/chatgpt

[2] https://sites.google.com/view/autextification/home

[3] https://www.nltk.org/howto/wordnet.html

Text Analysis and Retrieval 2023: Course Project Reports (TAR 2023), pages 42–45, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2023.

with their basic form (e.g. 'happpyyyyyyy' to 'happy'). Lastly, we used Spacy's `en_core_web_sm` tokenizer[4] to get the token lists for each text input.

4.2. Feature Extraction

Bag of Words

For Bag of Words (BoW) extraction, we used `CountVectorizer` implemented by Sklearn[5]. Additionally, when working with each model we used Sklearn's `GridSearchCV` to further optimize the hyperparameters of `CountVectorizer`. These hyperparameters define what length the n-grams that the BoW should include are (`ngram_range`) and how frequent they have to be in the dataset (`min_df`). In our paper, BoW serves as a feature for the baseline model with which we compare more advanced models.

Tf-idf

Similarly as for BoW, `TfidfVectorizer` from Sklearn was used for term frequency-inverse document frequency (tf-idf) feature extraction. The vectorizer uses the same hyperparameters as `CountVetorizer` which we also optimized for each model. Since tf-idf solves the issue of rare words in BoW extraction, we assume it will provide better information for detecting generated texts.

Additional Features

Besides counts and frequencies, we added functions to extract additional features from the text inputs. These additional features were seen during our preliminary analysis of the dataset and this motivated us to extract them as new information for the model inputs.

- **Specific word counts**: Counting the number of words in a text, uppercase and lowercase letters, capitalized and lowercase words, punctuation, stopwords, unique words, large numbers (e.g. using *1 000 000* instead of *1M*), expressive language (e.g, writing *sleeeeeepy* instead of *sleepy*), and multiple instances of expressive punctuation in a single text. Additionally, we use Textstat[6] for sentence counts and Emoji[7] for emoticon counts.
- **Difficulty to read**: Metrics that define how hard a text is to read. For this we used Textstat's `flesch_reading_ease`, `reading_time`, `syllable_count`, `polysyllabcount`, and `monosyllabcount` functions. For `reading_time`, the `ms_per_char` parameter was set to 14.69.
- **Sentiment**: Spacy's `eng_spacysentiment` provides the `spacytextblob` method which can be added to its pipeline for sentiment analysis. Here we extracted the positivity, polarity, and subjectivity of each text.
- **Repeating phrases**: In some cases, text generators repeat phrases or sentences in a loop during generation. Because of this, we implemented a function that counted when phrases longer than three words were repeated for more than five times in a single text. These thresholds were heuristically chosen based on the analysis of the train dataset.
- **Profanity use**: `Profanity_check`[8] uses a linear SVM trained on 200k human-labeled samples to check for use of profanity. We used the `predict` function which returns a one if the text contains profanity and a zero if it does not.
- **Syntax errors**: Using a Python wrapper for LanguageTool[9], we checked the number of syntax errors that a text contains using the `check` method.

4.3. Shallow Models

Logistic Regression

For logistic regression (LR) we used the model implemented by Sklearn in `sklearn.linear_model`. The five combinations of features used for LR are as follows:

- BoW (+ Additional Features)
- Tf-idf (+ Additional Features)
- Additional Features

GridSearchCV was used to optimize `model__max_iter` and `model__C` (alongside the vectorizer as described in Subsection 4.2.) when using BoW and tf-idf. When using Additional Features a Sklearn StandardScaler was used to normalize the features before training.

LGBM

The light gradient boosting machine (LGBM)(Ke et al., 2017) is Microsoft's open-source framework for machine learning. From the LGBM repository[10] we used the decision tree-based model `LGBMClassifier`. The combinations of features used to train the classifier are the same as for the LR model. The hyperparameters were optimized using Optuna[11] an open-source hyperparameter optimization framework. Optuna works quicker than GridSearchCV for models with many hyperparameters such as LGBM. The hyperparameters we optimized are as follows: `lambda_l1`, `lambda_l2`, `num_leaves`, `feature_fraction`, `bagging_fraction`, `bagging_freq`, `min_child_samples`, `max_depth`, `max_bin`, `num_iterations`.

4.4. Deep Models

BERT

The Bidirectional Encoder Representations from Transformers model (BERT)(Devlin et al., 2019) is a pre-trained deep transformer model which can be used for a variety of different NLP tasks. HuggingFace's transformer library features a BertTokenizer and BertModel implementation that we used for this paper. Specifically, we used the `bert-base-uncased` model. The model was trained for ten epochs with a batch size of 32 and a maximum token

[4] https://spacy.io/models/en
[5] https://scikit-learn.org
[6] https://pypi.org/project/textstat/
[7] https://pypi.org/project/emoji/
[8] https://pypi.org/project/profanity-check/
[9] https://pypi.org/project/language-tool-python/
[10] https://github.com/microsoft/LightGBM
[11] https://optuna.org/

length of 128. For optimization, we used HuggingFace's AdamW with a learning rate of 2e-5 and an early stopping mechanism implemented on a validation set made out of 20% of the train dataset. Besides the default BERT model architecture, we added a classification head to the hidden layer before the output. The PyTorch library[12] was used when needed for these additions to the model architecture.

We trained the BERT classifier in two different ways. First, we let the model train only on the embeddings that it generated from the input texts. Second, we added the Additional Features, described in Subsection 4.2. as additional information to the model.

RoBERTa

The Robustly optimized BERT (RoBERTa)(Liu et al., 2019) is a reimplementation of BERT which uses a different tokenizer and pretraining scheme. This difference brings better results than the regular BERT model, making RoBERTa the better model in general. We used the implementation from HuggingFace for RobertaTokenizer and RobertaModel, with the weights from `roberta-base` as a starting point. The rest of the training and model architecture is set as with the previously described BERT model.

4.5. Hybrid models

Shallow models rely more on the sheer syntactical features of a text while deep models can extract the semantics and meaning from text. Since these two groups of features are quite different, we decided to combine deep and shallow models so that more diverse information can be extracted for text classification. Specifically, we combined the word embeddings from RoBERTa with the LGBM classifier to produce better results than each model by itself.

First, we trained RoBERTa on all texts from the train set. Afterward, RoBERTa was used to create text representations for the dataset by taking the representations from RoBERTa. These representations were concatenated to the additional features and then fed to the LGBM as input. LGBM was optimized using Optuna in the same way as in Subsection 4.3..

5. Results

In this section, we report the results for each type of model and the combinations of features used for them. Table 1 has the F1 scores for the shallow, Table 2 for the deep, and Table 3 for the hybrid models. We used macro F1 scores from Sklearn's `classification_report` implementation. To statistically test the significance of the difference between different feature combinations and models, we calculated the macro F1 score for different subsets of the test data. When we calculated these F1 samples for each feature combination, we used a t-test to check the difference in mean F1 score between model samples.

For training the shallow models we used a computer with an Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz, NVIDIA GeForce GTX 1650, and 16 GB of RAM. for the deep and hybrid model training, we used the servers from Kaggle[13] and Google Colab[14] based on their availability.

[12]https://pytorch.org/

[13]https://www.kaggle.com/

[14]https://colab.research.google.com/

Table 1: Reported F1 score for shallow models

Model	LR	LGBM
BoW	0.57	0.58
Tf-idf	0.59	0.58
Features	0.60	0.58
BoW + Feat.	**0.65**	**0.68**
Tf-idf + Feat.	0.65	0.66

Table 2: Reported F1 score for deep models

Model	Base	Base + Additional Features
BERT	0.61	0.60
RoBERTa	0.67	0.56

6. Discussion

The statistical analysis proved a significant difference between the scores for the baseline models (BoW for shallow models, Base for deep) and the other models inside the same category ($p > 0.05$). Besides this, there was also a significant difference when looking at the F1 scores between the base models and the models with additional features.

When looking at the F1 score, the hybrid approach proved to be the best at classification, utilizing the strengths of both approaches. What's more curious is that the shallow models had a better score than the deep ones. One reason for the low F1 scores could be the difference in domains for the train and test datasets. This difference can be further amplified in the classification for deep models which rely upon semantics and deeper text meaning, unlike the shallow models. Since the deep models did not see these domains before they cannot understand the domain-specific errors that can happen in the test texts.

All models struggled to correctly classify short texts. These short texts contain little information that can be used to differentiate between texts, usually comprising less than ten words. A few examples are provided in Table 4. While some are indistinguishable from human ("I love it!"), other short-generated texts can be spotted for their lack of information ("The New York Times.") or their misuse ("It's really a shame."). However, the models require additional information to pinpoint these mistakes in the generated text which is why they classify longer-generated text better. Figure 1 drives this point further. We can see that despite the train set containing both long and short texts, the LR

Table 3: Reported F1 score for hybrid models

Model	F1
LGBM + RoBERTa	0.71

Table 4: Example of wrongly classified short texts. One stands for generated text and zero for human text.

Message	Domain	True label	Predicted label
The New York Times.	`news`	Generated	Human
I love it!	`review`	Generated	Human
It's really a shame.	`review`	Generated	Human
I love it and hope everyone gets one	`review`	Generated	Human
Keep your stupid comments to yourself.	`review`	Generated	Human
No wonder the American public is nuts.	`review`	Human	Generated
I got it as gift to bring up friends	`review`	Human	Generated

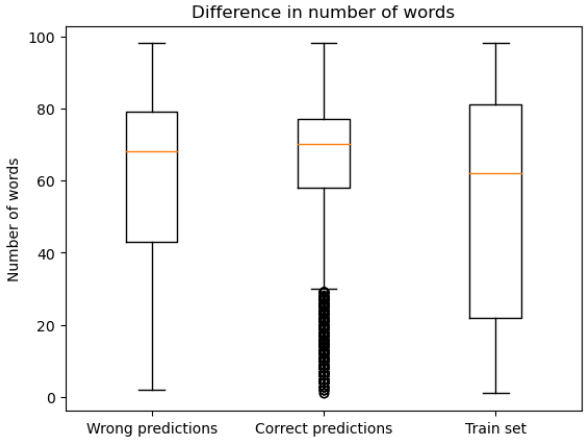


Figure 1: Boxplots visualizing the word lengths of the LR BoW + Feat. predictions and train set.

BoW + Feat. classifier struggled mostly with the shorter texts.

7. Conclusion

Despite the results not being SOTA, there is a clear difference between feature extraction methods and types of models. Heuristic features which were chosen during preliminary dataset analysis should be used as additional features for the model inputs alongside the classic BoW, tf-idf, and embedding features. Besides this, the combination of using deep models, such as RoBERTa, for text representation embeddings and shallow models for generated text classification works best. Another interesting finding is that the shallow models worked better than the deep ones.

Further research in automated text prediction should be explored, especially the effect of the difference in text domains and the number of words in texts on classification. Besides this, the difference in performance between shallow and deep models requires further investigation since our paper focused more on the combinations for the shallow models because of the computational constraints imposed on us.

References

Daria Beresneva. 2016. Computer-generated text detection using machine learning: A systematic review. In Elisabeth Métais, Farid Meziane, Mohamad Saraee, Vijayan Sugumaran, and Sunil Vadera, editors, *Natural Language Processing and Information Systems*, pages 421–426, Cham. Springer International Publishing.

Evan Crothers, Nathalie Japkowicz, and Herna Viktor. 2023. Machine generated text: A comprehensive survey of threat models and detection methods.

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

Ganesh Jawahar, Muhammad Abdul-Mageed, and Laks Lakshmanan, V.S. 2020. Automatic detection of machine generated text: A critical survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2296–2309, Barcelona, Spain (Online), December. International Committee on Computational Linguistics.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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.

From Pen to Persona: Essays Give Insight to Who You Are on the Inside

Iva Marić, Lea Grebenar, Nikola Petrović

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{iva.maric, lea.grebenar, nikola.petrovic}@fer.hr

Abstract

This research paper explores the relationship between language use and personality traits by analyzing a dataset of essays. The objective was to develop a classification model that accurately predicts the polarity of the Big Five personality traits: extroversion, neuroticism, agreeableness, conscientiousness, and openness to experience. By training the model with the provided data, the paper aims to gain insights into how language patterns can indicate specific personality characteristics. Additionally, the paper investigates the properties of the dataset and personality model, such as the frequency of co-occurring traits and potential exclusions.

1. Introduction

Language is a powerful tool that not only conveys information, but also reflects the characteristics and personality traits of individuals. Uncovering and understanding these traits can be of great interest in fields like psychology, social sciences and, nowadays, even marketing. Recent advances in machine learning, particularly in the development of NLP models, have been a major step in enabling these fields to get the wanted information more quickly and easily.

The Big Five model, comprising extroversion (EXT), neuroticism (NEU), agreeableness (AGR), conscientiousness (CON) and openness to experience (OPN) is a widely recognized framework for describing core personality traits (Roccas et al., 2002). Using this model, we can attempt to determine personality traits of people based on the texts they wrote.

In this research paper, we aim to explore the relationship between language use and personality traits by leveraging a dataset consisting of essays. Each essay within the dataset is accompanied by labels representing the individual's Big Five personality traits. Our goal is to develop a classification model that can sufficiently predict the polarity of the Big Five personality traits. By doing so, we can gain insights into how language patterns are indicative of certain personality characteristics.

Besides training the model with the given data, our second research question is to investigate the properties of the given database and personality model, for example how often do some personality traits appear together or what traits potentially exclude each other. Very little or no existing research focuses on extracting that information from this database, but we think such information can be very significant to sociolinguistics experts. It can help them derive information about the frequency and relation of some personality traits, and maybe even make them reconsider the overall quality of the Big Five model.

2. Related Work

Several previous studies have investigated the connection between personality traits and language use. We will briefly discuss some articles that explore personality trait classification based on language analysis.

The authors of the "essays" dataset conducted prior research on text-based personality trait classification. They utilized Linguistic Inquiry and Word Count (LIWC) features to establish a correlation between the content of written essays and the Big Five personality traits of the authors, as outlined in the study by Pennebaker and King (1999).

Another research worth mentioning was written by Pizzolli and Strapparava (2019). We find this research interesting because Pizzolli and Strapparava used the same essays dataset we used, but their models were employed to categorize the personality traits of characters in literature written by Shakespeare. It is crucial to acknowledge that in this task, there are no predetermined correct answers, and therefore the performance evaluation was conducted through manual subjective assessment.

Gjurković et al. (2021) analyzed large-scale textual data from various subreddits to investigate how language use and demographic information correlate with personality traits. Their findings emphasized the influence of demographics and online environments on the expression of personality. Their work partly explains why our demographic-less database resulted in less accurate models.

The last research we consider relevant is Park et al. (2015) which focuses on automatic personality assessment using language patterns extracted from social media platforms. This research, like ours, resulted in developing a model for predicting the Big Five personality traits. That model utilized linguistic features and machine learning algorithms.

3. Dataset

The dataset used in this paper was originally developed by Pennebaker and King (1999). It consists of 2467 essays, each labeled with five binary labels corresponding to the Big Five personality traits (whether the author possesses a given trait or not). The essays, which were written as stream-of-consciousness texts by psychology students, were collected between 1997 and 2004 (Deilami et al., 2022). The labels were derived from the authors' self-assessments. The derivation of these labels involves the computation of z-scores by Mairesse et al. (2007), followed

Text Analysis and Retrieval 2023: Course Project Reports (TAR 2023), pages 46–49, University of Zagreb,
Faculty of Electrical Engineering and Computing, Zagreb, July 2023.

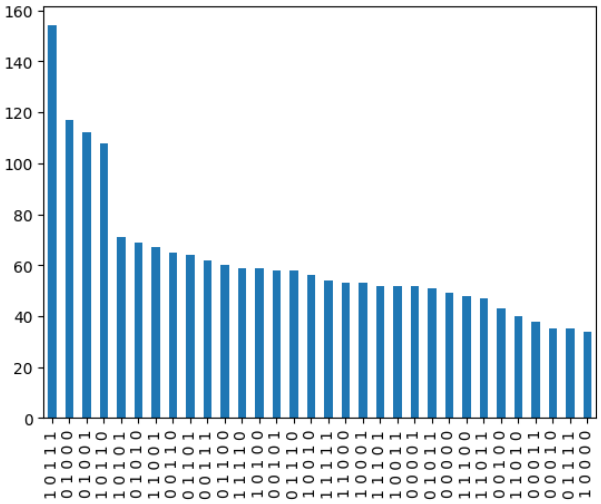


Figure 1: Bar chart displaying number of occurrences of each combination of traits in the training set.

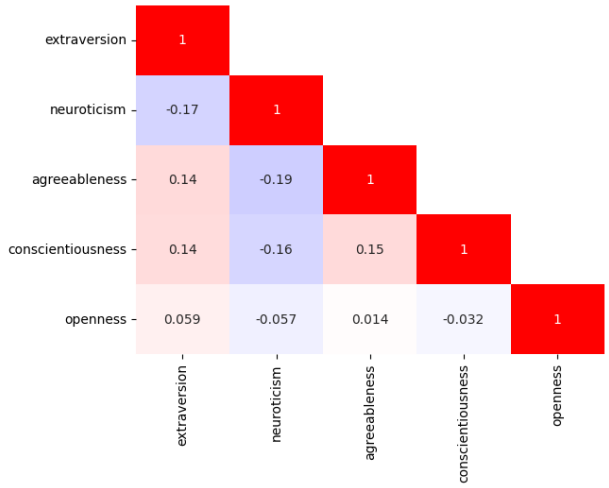


Figure 2: Correlations between personality traits.

by a conversion from scores to nominal classes using a median split technique as performed by Celli et al. (2013).

Collecting labels this way can be problematic for use in NLP because of several reasons. Firstly, subjectivity, which can introduce noise and uncertainty into the training data because the accuracy and consistency of self-assessments may vary among individuals. Next, limited representativeness, since the dataset consists of essays from psychology students, which may not fully represent the general population. Also, stream-of-consciousness writing style allows for unfiltered, unstructured and spontaneous expression of thoughts which makes it more challenging for models to capture meaningful patterns and relationships. It doesn't help that the essays were collected over a span of several years, which can cause the data to be influenced by temporal factors, such as changes in cultural norms, language usage, or the psychological state of individuals.

Each entry in the dataset consists of the following features: author id, essay, and binary labels for traits (ext, neu, agr, con, opn) which are denoted by 'y' or 'n'.

The essays in the dataset vary widely in length, ranging from 39 to 2964 words and 1 to 324 sentences. On average, essays contain around 742.4 words and 48.6 sentences. However, these deviations in length can easily create challenges for machine learning models. Additionally, some essays are incomplete, with some abruptly ending mid-word and other ones lacking punctuation despite being a single sentence.

Data analysis was performed on a subset of the dataset, later to be used as the training set. Figure 1 shows all the combinations of personality traits present in the set (of $2^5 = 32$ possible combinations, all were present) sorted from most to least common. Combinations are written as bit vectors denoting the presence of each trait in the following order: extraversion, neuroticism, agreeableness, conscientiousness, openness.

Figure 2 shows the correlation of traits; agreeableness and openness exhibit the least correlation, while agreeableness and neuroticism exhibit the strongest correlation (whether negative or positive).

The strongest negative correlations discovered between the traits are as follows: agreeableness and neuroticism, neuroticism and extraversion, and conscientiousness and neuroticism. That kind of correlation means that those labels often exclude each other. We find these conclusions expected since individuals who are empathetic (strong agreeableness) or outgoing (strong extraversion) or even responsible (strong conscientiousness) are less prone to experiencing emotional instability or intense negative emotions.

4. Model Building and Evaluation

4.1. Preprocessing Data

We put the text data under several preprocessing steps to prepare it for further analysis. By performing these steps, the raw text data is transformed into a cleaner and more standardized representation.

Firstly, we removed punctuation marks to eliminate any extraneous symbols that do not contribute to the meaning. This step puts the focus solely on the textual content. Next, we removed any numeric digits present, as they typically do not carry significant semantic information in this context. To maintain consistency, we converted the entire text to lowercase. This standardizes the text and prevents variations in capitalization from affecting subsequent analyses.

The process of lemmatization is then applied to the text. Lemmatization reduces words to their base or root form, considering their contextual meaning and grammatical role.

4.2. Modeling

Considering that each label belongs to either a positive or negative class and corresponds to a particular personality trait, it is clear that the objective of this study is to perform multi-label classification.

For vectorization, we employed two commonly used approaches, CountVectorizer and TfidfVectorizer. CountVectorizer creates a matrix that captures the frequency of words in the text, enabling the models to learn from the presence

and absence of specific words within documents. TfidfVectorizer calculates the TF-IDF value (Das et al., 2021) of each word, emphasizing words that are informative and distinctive across the dataset.

In terms of the classification models, we utilized Random Forest, Logistic Regression, and Support Vector Machines (SVM). Random Forest is an ensemble learning method that constructs multiple decision trees and combines their predictions for accurate classification. Logistic Regression is a linear model that estimates the probability of each class based on the input features. SVM finds the optimal hyperplane to separate data points of different classes effectively. To handle multi-label classification, these models can be extended using the one-versus-rest approach, where separate classifiers are trained for each label, treating each classification task independently.

To train the models we used the ClassifierChain and One-vs-Rest strategies. ClassifierChain is an ensemble method that chains multiple binary classifiers together, considering the dependencies among the labels. It utilizes the predictions from previous classifiers as additional features for subsequent classifiers. One-vs-Rest, also known as one-vs-all, trains separate classifiers for each label, treating each classification task independently.

The chosen models were selected based on their ability to handle multi-class classification, their compatibility with the multi-label classification setting, their effectiveness in capturing complex relationships, and, last but not least, their interpretability.

4.3. Implementation Details

The code used for the research is written using python and it is organized into a jupyter notebook. Other than commonly used python libraries, we also used those intended for machine learning. Pandas[1] came in handy for managing and manipulating structured data, such as loading the dataset and preprocessing the textual data. We used spacy[2] because it simply provides NLP functionalities like tokenization, lemmatization, and stop word removal.

Scikit-learn (sklearn) by Pedregosa et al. (2011) provided vectorizers for text vectorization, models for multi-class classification and, finally, scores for evaluation metrics. The optimization of these libraries resulted in short training times for the models, typically lasting under a minute. The longest process was the lemmatization of long essay texts which lasted several minutes.

4.4. Training and Evaluation

Before training, we divided the dataset into a training set and a test set. It is worth noting that the test set comprises 20% of the data, hence the models were trained on 80% of the provided texts.

After training, we calculated the evaluation metrics based on the model's predictions and the actual target values. Our most important metric computed is F1-macro because it provides a balanced assessment of the model's performance across all classes.

[1] https://pandas.pydata.org/

[2] https://spacy.io/

Table 1: F1 scores for each classifier. SVC is Support Vector Classifier, LR is Logistic Regression, and RFC is Random Forest Classifier

	SVC	LR	RFC
all	**0.594987**	0.575431	**0.595216**
ext	*0.496491*	*0.502798*	*0.496491*
neu	0.540439	0.568682	0.540439
agr	0.532396	0.513243	0.532396
con	0.537704	**0.599836**	0.537704
opn	**0.582679**	**0.585931**	**0.582679**

5. Results and Discussion

Given that our goal was not to achieve state-of-the-art results as demonstrated by Majumder et al. (2017), the obtained outcomes are reasonably anticipated. Our models achieve a higher accuracy than the state-of-the-art specifically for conscientiousness (with our best F1 score at 0.59 compared to their best at 0.56). However, it is important to note that Majumder et al. (2017) evaluated their results using cross-validation, while we only performed a single split of the dataset into train and test subsets.

We ran various combinations of features and models, kept and displayed only the best ones, yet our models didn't manage to outperform random classifiers. Several factors, all related to the dataset, can contribute to this outcome. As mentioned in Section 3., the essays consist of noisy and unstructured data, comprising informal language and grammatical errors. This poses a challenge for the model to identify meaningful patterns. Additionally, stream-of-consciousness essays often lack adequate contextual information, which is crucial for precise personality trait prediction. Furthermore, the subjective labeling of personality traits in the essays can introduce noise and impact the model's performance.

In Table 1, we put all F1 macro scores of one-vs-rest model for each personality and F1 macro score of classifier chain for all personalities together. Taking into consideration F1-scores of all other models and these results, we can conclude that conscientiousness and openness are easiest for some models to predict, while all models perform poorly when predicting extraversion. We will leave these results to be explained by sociolinguistic experts, but we can try to explain it this way: conscientiousness and openness are relatively easier for some NLP models to predict because these traits can be expressed through written language, which is abundant in stream of consciousness essays. However, extraversion, being a multifaceted trait that extends beyond written language and is influenced by various factors, poses challenges for NLP models in accurately predicting it.

While analyzing the dataset in Section 3., we saw that agreeableness and neuroticism exhibit the strongest negative correlation. When we trained the models only on those two labels, we got slightly better F1 scores (0.611491 when trained only on agr and neu, as opposed to 0.575431 when trained on the entire set of labels). This is expected since

the high negative correlation implies that the occurrence of one label is likely to indicate the absence of another label, which makes it easier for models to predict accurately.

We only highlighted the most obvious and interesting points that we could conclude using the extracted information about the database and the calculated models' evaluation metrics. We leave it to other researchers, sociolinguistics experts or curious readers to uncover or explain more about the relationships of personality traits and their role in creating the stream-of-consciousness essays.

6. Conclusion

The aim of this research paper was to explore the relationship between language use and personality traits by leveraging a dataset of essays. We successfully developed several different classification models to predict the polarity of the Big Five personality traits. Although we did not achieve state-of-the-art results, the models achieved decent accuracy. It is important to note that challenges such as the noisy and unstructured nature of the dataset, informal language, and subjective labeling of personality traits in the essays influenced the model's performance. To answer our second research question, we investigated the properties of the dataset and personality model. This gave us valuable information about the frequency and relationships of different personality traits. The analysis revealed correlations between traits, with agreeableness and openness exhibiting the least correlation, while agreeableness and neuroticism exhibited the strongest negative correlation. Our findings provide a deeper understanding of the Big Five model's quality and contribute to the improvement of our classification models.

Overall, by conducting this research, we contributed to the field of NLP and provided valuable insights into the connection between personality traits and language use. Our findings highlight the importance of considering contextual information, dataset quality, and the complexities of personality traits when developing NLP models for personality trait classification.

References

Fabio Celli, Fabio Pianesi, David Stillwell, and Michal Kosinski. 2013. Workshop on computational personality recognition: Shared task. *Proceedings of the International AAAI Conference on Web and Social Media*, 7(2):2–5, August.

Mamata Das, Selvakumar Kamalanathan, and PJA Alphonse. 2021. A comparative study on tf-idf feature weighting method and its analysis using unstructured dataset. In *COLINS*, pages 98–107.

Fatemeh Mohades Deilami, Hossein Sadr, and Mojdeh Nazari. 2022. Using machine learning based models for personality recognition. *CoRR*, abs/2201.06248.

Matej Gjurković, Mladen Karan, Iva Vukojević, Mihaela Bošnjak, and Jan Snajder. 2021. PANDORA talks: Personality and demographics on Reddit. In *Proceedings of the Ninth International Workshop on Natural Language Processing for Social Media*, pages 138–152, Online, June. Association for Computational Linguistics.

François Mairesse, Marilyn A Walker, Matthias R Mehl, and Roger K Moore. 2007. Using linguistic cues for the automatic recognition of personality in conversation and text. *Journal of artificial intelligence research*, 30:457–500.

Navonil Majumder, Soujanya Poria, Alexander Gelbukh, and Erik Cambria. 2017. Deep learning-based document modeling for personality detection from text. *IEEE Intelligent Systems*, 32(2):74–79.

Gregory Park, H. Andrew Schwartz, Johannes C. Eichstaedt, Margaret L. Kern, Michal Kosinski, David J. Stillwell, Lyle H. Ungar, and Martin E. P. Seligman. 2015. Automatic personality assessment through social media language. *Journal of Personality and Social Psychology*, 108(6):934–952, June.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

James W Pennebaker and Laura A King. 1999. Linguistic styles: language use as an individual difference. *Journal of personality and social psychology*, 77(6):1296.

Daniele Pizzolli and Carlo Strapparava. 2019. Personality traits recognition in literary texts. In *Proceedings of the Second Workshop on Storytelling*, pages 107–111.

Sonia Roccas, Lilach Sagiv, Shalom H. Schwartz, and Ariel Knafo. 2002. The big five personality factors and personal values. *Personality and Social Psychology Bulletin*, 28(6):789–801, June.

It's No Laughing Matter: Exploring the Quality of the Humicroedit Dataset

Gabriel Marinković

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
gabriel.marinkovic@fer.hr

Abstract

This paper presents a critical assessment of Humicroedit, a humor analysis dataset encompassing roughly 16,000 news headlines subjected to "micro-edits" for humor induction. Despite the intentionality behind the annotation process, our evaluation reveals potential quality issues stemming from crowd-sourced data, notably the forced humor potentially skewing grading. We illustrate this concern qualitatively through the high level of annotator disagreement for headlines with an overall low grade, which comprise the majority of the dataset. To substantiate our argument, we utilized a BERT-based model on what we believe to be a less biased dataset subset, presenting modest results that allude to the necessity for further research. Consequently, we advocate for additional examination of Humicroedit's quality, striving for improvements that can refine humor comprehension models.

1. Introduction

Humor analysis continues to be one of the most challenging problems in NLP. The comprehension of humor hinges upon a complex relation of common sense knowledge, awareness of relevant real-world occurrences, and often, the personal stance on controversial topics such as political decisions. Another key obstacle in humor analysis is its inherent subjectivity. Sense of humor is incredibly individualistic, and the appreciation of a joke rarely overlaps completely between two individuals. This reliance on multiple diverse layers of knowledge, coupled with subjective interpretations of humor, makes humor analysis a very attractive field of research in NLP.

Every dataset necessarily carries some bias, but the subjective nature of humor makes this a bigger issue than usual. One approach to reducing bias is to focus on one domain of jokes, embracing this subjectivity bias. Reddit is an effective way to collect jokes sharing a common theme, which are already organically pre-annotated by the community (Tang et al., 2022). Jokes collected in this way are "standalone" and miss real-world context present in everyday humor. Other datasets are created from scratch in order to better focus on a specific research topic. Humicroedit (Hossain et al., 2019) is one such dataset, consisting of a large number of news headlines which were "micro-edited" — one or two words were replaced in the hopes of inciting a humorous response. By focusing on one-word edits, the dataset tries to facilitate research on the boundary of what makes a piece of text funny or not. Humicroedit was created and annotated by a large number of professional annotators, which has the potential to significantly affect the quality of the data compared to organically created jokes. This paper will try to present a critique of the creation process of Humicroedit, with an emphasis on using crowd-sourcing to create humorous edits, instead of collecting them organically from existing communities.

2. Related Work

Humicroedit authors themselves paid a lot of attention to the annotation process and data quality of this dataset. Annotator selection was strict with specially designed tests to qualify both editors and judges. Many easily detectable data quality issues were filtered by them, like cheap use of profanity to invoke humor. However, the authors fail to comment on the skewed distribution towards unfunny mean grades, and the psychological effect this might have had on the judges. Hossain et al. (2019) also did an overview of theories of humor and their applications on Humicroedit, like the relief and incongruity theories of humor (Ritchie, 2004). Humicroedit was a featured dataset in the *SemEval 2020 Task 7* (Hossain et al., 2020) which further explored the effectiveness of current state-of-the-art models in detecting humor. While some complex approaches achieved a significant improvement over baseline (Morishita et al., 2020), applications of these models are not practical nor do they reveal deeper insights into understanding humor.

Humicroedit is not the only dataset focused on humor analysis. Larger datasets like rJokes (Weller and Seppi, 2020) consists of over 550,000 jokes collected from Reddit. The main differences between existing datasets and Humicroedit is (i) their origin: most datasets contain organically sourced jokes from multiple communities, while the edits in Humicroedit were crowd-sourced, and (ii) their scope: Humicroedit intentionally focuses on small word-level edits to allow examining the boundary between funny and unfunny text.

3. The Humicroedit Dataset

The Humicroedit dataset was designed with the intentional purpose of advancing research in computational humor. It consists of news headlines with their respective "micro-edits", which are short edits of the original headline, usually one or two words long, which in theory renders the edited headline funny, see Table 1 for examples. The headlines have been collected from news media posted on Reddit in the time period between January 2017 and May 2018. Most of the headlines were collected from world news and politically oriented sub-communities, meaning that the headlines require a significant amount of real-world knowledge for proper understanding. Five distinct annotators graded the funniness of each headline-edit pair. The grades range from 0 (not funny) to 3 (funny). The full dataset consists of

Text Analysis and Retrieval 2023: Course Project Reports (TAR 2023), pages 50–54, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2023.

Table 1: Example headlines and their respective edits, along with the mean funniness grade. Highlighted word is to be replaced by the edit. Over 50% of data is graded below $0.8/3$, while only 3% of data is graded above $2.0/3$.

Headline	Edit	Mean Grade
Scientists appeal for more people to donate their brains	use	$2.5/3$
Rocks falling into ocean, not climate change, causes rising sea levels, claims congressman	toddler	$2.6/3$
Comey: Trump has "an emptiness inside of him and a hunger for affirmation"	ham	$2.8/3$
Hamas makes demands as UN chief arrives in Gaza for visit	dinner	$0.0/3$
Will Sweden Become the First Country to Go Cash-Free	reward	$0.0/3$
Netflix says it now has 104 million subscribers worldwide	toothpicks	$0.0/3$

15,095 graded headline-edit pairs.

The authors deliberately chose to base their dataset on headlines and small edits. News headlines are short summaries of longer information-dense articles, and as such carry a lot of information for their length (most headlines are under 20 words long) — a model of humor would need a deep understanding of both relevant world knowledge and common-sense reasoning to reason about the data effectively.

3.1. Edit Creation and Annotation

The authors used Amazon Mechanical Turk to create and grade the edits. Editors were instructed to make the headline amusing to a broad audience. Certain types of humor, such as profanity, crude references, and compound words, were explicitly forbidden. Authors selected 100 editors who have proven to create acceptable edits with a mean funniness grade above 0.8. Each editor edited roughly 1,500 headlines. Potential humor judges were asked to grade headlines edited by the authors themselves. While grading the headlines potential judges had to reach a level of agreement with the author's predetermined grades. This process qualified 150 judges. Inter-annotator agreement was expressed using Krippendorff's α interval metric (Krippendorff, 1970). The α for qualified judges was 0.64, where 0 means no consensus, and 1 means complete agreement.

In an attempt to reduce annotator bias judges were instructed to grade the edits "objectively", without taking their personal values into account — an edit with a high grade should be funny to a broad audience. Tasks were separated into smaller batches; annotators needed to wait least 24 hours between batches in an attempt to reduce annotator exhaustion. Editors and judges assigned to each batch were randomized so that edits made by a single editor are not always labeled by the same set of judges.

3.2. Concerns About the Annotation Process

The authors took great care to minimize annotator bias whenever possible. However, we argue that despite their measures the dataset is of a relatively poor quality. While Turkers have often been credited with producing high-quality data (Callison-Burch, 2009), recent years have seen the reliability of such data, especially in the context of psychological research, come under scrutiny (Rouse, 2015). Turkers are financially incentivized to complete their tasks as fast as possible (Moss et al., 2023), which does not create a good environment to come up with funny jokes. It is unreasonable to think that editors will perform at a high level for all $1,500$ headlines, even if such tasks are spread over multiple days. Large number of headlines caused editors to resort to using quick but ineffective techniques, like reusing a very small vocabulary of "funny" words, or excessively using profanity. Creation of Humicroedit ignored the fact that humor can not be forced. News parodies are a popular entertainment genre with mass appeal (Baym and Jones, 2012), but not all headlines hold the potential for humor.

Unreliable editing which produces unfunny edits also strongly degrades the quality of grading. As can be seen in Figure 1, annotator consensus is rare, and only ever achieved for unfunny edits. Over 50% of data has a mean grade under 0.9. This validates our concerns about the quality of the data. We believe that this distribution of grades and annotator consensus can be explained by (i) annotators becoming more lenient to the large chunk of unfunny headlines as they progressed through the tasks, and (ii) annotators not wanting to repeatedly grade headlines as 0 due to response bias (Attali and Bar-Hillel, 2003). We hypothesise that the majority of averagely graded headlines with high annotator disagreement represent noisy data.

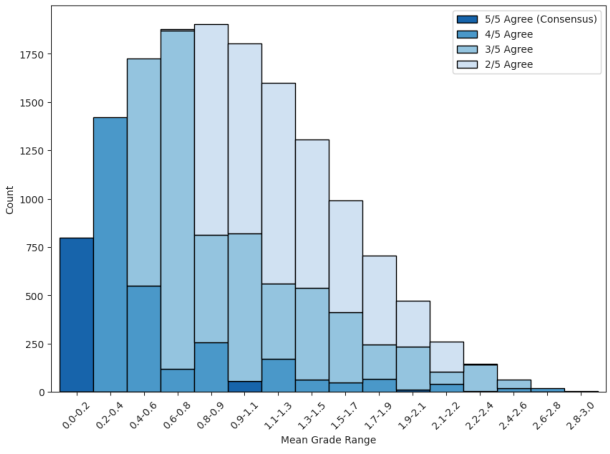


Figure 1: Distribution of mean grades. Color represent annotator agreement for each headline. Notice that annotators only reach consensus for mean grades under 0.5.

4. Experiments

In order to verify our claims of extremely graded (meaning close to 0 or 3) headlines carrying most of the signal for humor detection we trained four regression models on different subsets of the dataset. The models were trained to minimize the Mean Square Error (MSE) when predicting the mean grade of a headline-edit pair. The subtask 1 of the *SemEval 2022 Task 7* also focuses on predicting mean grades and uses RMSE; we opted for MSE instead, which allows us to conduct a better statistical analysis.

4.1. Training Data

We use the standard train-validation-test split as outlined in the subtask 1 of the *SemEval 2022 Task 7*. Train, validation, and test splits contain 64%, 16%, and 20% of the data respectively. To test our hypothesis, we further split the training and validation datasets in three parts. These datasets are sorted by their mean grade and split at the first and third quantile, so that each section contains 25%, 50%, and 25% of the original set. The baseline model is trained on the full training set, while the other models are trained on their respective subsets. All models are always evaluated on the full test set.

4.2. Model Architecture

We use a standard pretrained BERT Transformer network (Devlin et al., 2018) with a fully connected layer as the output layer, illustrated in Figure 2. The fully connected layer down-projects the `[CLS]` token embedding into a single real-valued output, the predicted mean grade. Our model is a simplified version of Jin et al. (2020)'s work - instead of including a separate feature representation layer we simply rely on the flexibility of the Transformer to adapt to our data during finetuning. This allows us to achieve acceptable performance while substantially simplifying our implementation. Four our base model we are using `bert-base-uncased` ($L = 12$, $d = 768$), from the HuggingFace Transformers library (Wolf et al., 2020). The dimensions of the fully connected layer are 768×1, and include a single scalar bias parameter.

The input $\mathbf{s}$ to our model is a concatenation of tokens of the original $\mathbf{x}$ and edited $\tilde{\mathbf{x}}$ headline, separated by a `[SEP]` token, as is standard in BERT based models. The special `[CLS]` token is prepended to the input, and its output is fed into the fully connected layer.

$$\begin{aligned} \mathbf{x} &= (x_1 ... x_{T_o}),\ \tilde{\mathbf{x}} = ((\tilde{x}_1 ... \tilde{x}_{T_e})) \\ \mathbf{s} &= (\texttt{[CLS]};\ \tilde{\mathbf{x}};\ \texttt{[SEP]};\ \mathbf{x}) \end{aligned} \tag{1}$$

The input is padded to 512 tokens. Due to all of our headlines being fairly short, the input is never truncated. During experimentation we did not find a significant difference in encoding when using this or other ways of encoding the headline-edit pair.

4.3. Training and Hyperparameters

We used a fairly typical setup for our training and hyperparameter tuning. Each model was trained for 10 epochs, with a batch size of 32. We used a learning rate of 5×10^{-5} and a linear warm-up schedule with a warm-up ratio of 0.1. After every 100 batches the model was evaluated on the validation set, and after 10 epochs the checkpoint with the lowest MSE on the validation set was chosen, which usually happened around epoch 4. Training of each model took around 15 minutes on an `RTX 3060` GPU.

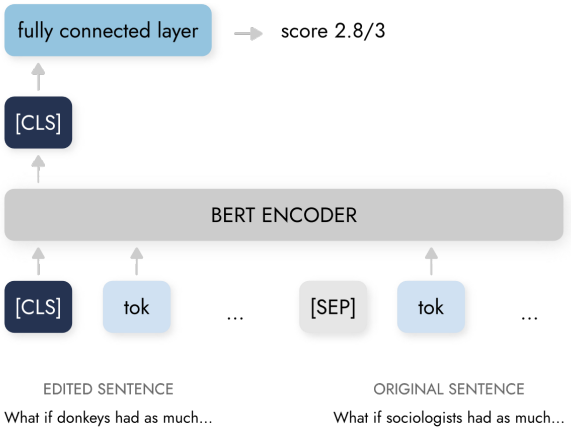


Figure 2: Schema of the BERT-based regression model.

5. Results

The obtained results mostly align with our initial hypothesis and provide a degree of support for it. In Table 2 we present the Mean Squared Error (MSE) for each model. The test set is divided into different subsets based on mean grade percentiles. Each column in the result table represents the MSE computed on one of these subsets. In order to determine whether there is a statistically significant difference in performance between each model and the baseline model, we conducted a two-sided t-test on the square errors. Each of the three models was compared against the baseline, using their respective square errors. The significance level for the t-test was set at 0.05. A statistically significant difference, should one be found, would suggest a meaningful divergence in the MSE scores between the given model and the baseline.

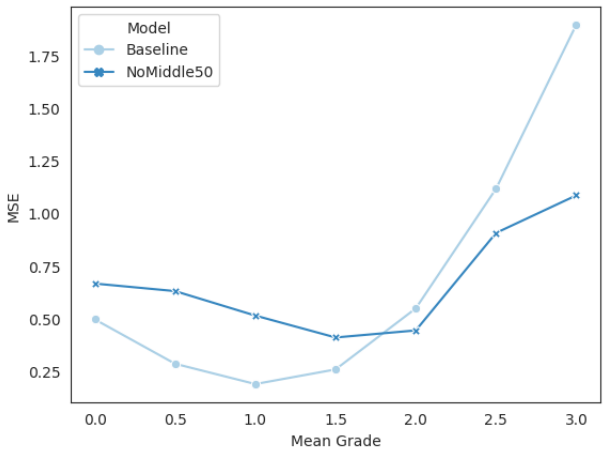


Figure 3: MSE evaluated on the test set, comparing the model trained without "noisy" data against baseline.

Table 2: Achieved MSE metric for different portions of training set used. The column denotes the part of the test set on which the metric was computed - *Bottom 5%* metric was computed on the 5% of the test set with the lowest mean grade. Best results for each dataset subset are denoted in bold. Results which **do not** significantly differ from baseline (two-sided t-test, $p < 0.05$) are denoted with a superscript†.

Model	Train Set % Used	Entire Set	Bottom 5%	Bottom 25%	Middle 50%	Top 25%
`Baseline`	100%	0.314	0.582	0.495	0.227	0.501
`NoMiddle50`	50%	0.545	0.591†	0.665	0.534	0.474†
`NoBottom25`	75%	0.386	1.237	0.991	0.277	**0.328**
`NoTop25`	75%	**0.294**†	**0.449**	**0.361**	**0.141**	0.818

When removing a part of the training set, the error in that section of the test increases, while error in adjacent sections falls. The most interesting model is `NoMiddle50`, whose MSE is compared with the baseline in Figure 3 — even though it was trained on 50% less data, the performance only suffers in the mean grade range of 0.5 to 1.5. This model never achieves the best performance in any subset of the dataset, which is likely explained by leaving out too much data during its training (including some legitimate, non-noisy examples). We also notice that by eliminating the "middle" portion of training data the model can better discern between the funniest and least funny headlines, which we think are the valuable parts of the dataset.

5.1. Limitations of the Analysis

While the results somewhat support out hypothesis, more research is needed to reach a conclusive results. Even though we think that most of the examples between the grades of 0.5 and 1.5 are noisy, there are definitely some headlines which "truly" belong into this grade class. We have no way of discerning these headlines from the "noisy" ones, so we are removing all of them, which impacts our results. Our analysis also does not take into account possible *controversial* headlines where annotators are in extreme disagreement with each other; these headlines are rare, but they do exist. It would be worthwhile to repeat the analysis using more complex models with better scores on the *SemEval 2020 Task 7* challenge, like the Hitachi system (Morishita et al., 2020).

Our initial results seem promising, but because of the nature of the dataset outlined in Section 3., any kind of evaluation on this dataset is difficult, partly because of the questionable annotation process, partly because of the poor performance of industry-standard NLP models, and partly because the data (especially the grades) is difficult to interpret for each individual researcher.

6. Future Work

Our results on their own are not enough to conclusively discredit the use of Humicroedit for further humor research. More research is needed, and the first step would be to evaluate our findings on more complex models which achieve better performance at this task (Morishita et al., 2020). Most of these high performing models are complex ensembles of computationally-intensive models, and as such were not in the scope of our paper.

Our model shows promising ability to discern headlines with very low grades from the ones with very high grades. This property should be explored for more robust mean grade prediction. Ordinal regression based on an ensemble of binary classificators (Frank and Hall, 2001) could perform better than a simple regression. Using the MSE loss to model headline grades is not necessarily the best choice, and modelling the probability of the most common grade might yield more accurate and interpretable results.

Lastly, in the future we hope to focus on the individual grades assigned by annotators, instead of only their mean. Different methods of modeling subjective annotations, such as ensembles of classifiers with a separate voting model, have shown promising results (Reidsma and op den Akker, 2008). Our analysis does not pay special attention to controversial headlines with extreme annotator disagreement, which could yield insightful results.

7. Conclusion

This paper critically evaluates Humicroedit, a humor analysis dataset comprising around 15,000 news headlines with "micro-edits" intended to create humor. Five annotators graded the funniness of each headline-edit pair. Both the edits and the grades were collected through Amazon Mechanical Turk. Despite careful annotation, we highlight concerns about the quality of this crowd-sourced data. Our primary issue is the enforced humor in every headline, leading to possible bias in grading where annotators give higher grades to unfunny edits due to a lack of truly funny examples. We qualitatively illustrate this through annotator disagreement concerning mean grades.

To reinforce our argument, we trained a standard BERT-based model on a subset of the dataset that, we suspect, carries less bias, and compared its performance with a model trained on the full dataset. The results modestly back our claims, hinting at superior model performance for very poorly and highly graded headlines. The low overall score and inability to test more advanced models warrant further research to validate our assertions.

In conclusion, Humicroedit could enable novel research yet its potential quality challenges may hamper its utility. It is crucial to investigate the quality of crowd-sourced datasets such as Humicroedit and develop strategies to overcome their complications. This would pave the way for more sophisticated models capable of grasping humor, a unique and complex component of human language.

References

Yigal Attali and Maya Bar-Hillel. 2003. Guess where: The position of correct answers in multiple-choice test items as a psychometric variable. *Journal of Educational Measurement*, 40(2):109–128.

Geoffrey Baym and Jeffrey P Jones. 2012. News parody in global perspective: Politics, power, and resistance. *Popular Communication*, 10(1-2):2–13.

Chris Callison-Burch. 2009. Fast, cheap, and creative: Evaluating translation quality using amazon's mechanical turk. In *Proceedings of the 2009 conference on empirical methods in natural language processing*, pages 286–295.

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*.

Eibe Frank and Mark Hall. 2001. A simple approach to ordinal classification. In *Machine Learning: ECML 2001: 12th European Conference on Machine Learning Freiburg, Germany, September 5–7, 2001 Proceedings 12*, pages 145–156. Springer.

Nabil Hossain, John Krumm, and Michael Gamon. 2019. " president vows to cut¡ taxes¿ hair": Dataset and analysis of creative text editing for humorous headlines. *arXiv preprint arXiv:1906.00274*.

Nabil Hossain, John Krumm, Michael Gamon, and Henry Kautz. 2020. SemEval-2020 task 7: Assessing humor in edited news headlines. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 746–758, Barcelona (online), December. International Committee for Computational Linguistics.

Shuning Jin, Yue Yin, XianE Tang, and Ted Pedersen. 2020. Duluth at SemEval-2020 task 7: Using surprise as a key to unlock humorous headlines. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 986–994, Barcelona (online), December. International Committee for Computational Linguistics.

Klaus Krippendorff. 1970. Estimating the reliability, systematic error and random error of interval data. *Educational and Psychological Measurement*, 30(1):61–70.

Terufumi Morishita, Gaku Morio, Hiroaki Ozaki, and Toshinori Miyoshi. 2020. Hitachi at SemEval-2020 task 7: Stacking at scale with heterogeneous language models for humor recognition. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 791–803, Barcelona (online), December. International Committee for Computational Linguistics.

Aaron J Moss, Cheskie Rosenzweig, Jonathan Robinson, Shalom N Jaffe, and Leib Litman. 2023. Is it ethical to use mechanical turk for behavioral research? relevant data from a representative survey of mturk participants and wages. *Behavior Research Methods*, pages 1–20.

Dennis Reidsma and Rieks op den Akker. 2008. Exploiting 'subjective'annotations. In *Coling 2008: Proceedings of the workshop on Human Judgements in Computational Linguistics*, pages 8–16.

Graeme Ritchie. 2004. *The linguistic analysis of jokes*, volume 2. Routledge.

Steven V. Rouse. 2015. A reliability analysis of mechanical turk data. *Computers in Human Behavior*, 43:304–307.

Leonard Tang, Alexander Cai, Steve Li, and Jason Wang. 2022. The naughtyformer: A transformer understands offensive humor.

Orion Weller and Kevin Seppi. 2020. The rJokes dataset: a large scale humor collection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 6136–6141, Marseille, France, May. European Language Resources Association.

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, October. Association for Computational Linguistics.

A Squeeze of LIME, a Pinch of SHAP: Adding the Flavor of Explainability to Sentence-Level Commonsense Validation

Mirta Moslavac, Roko Grbelja, Matej Ciglenečki

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{mirta.moslavac,roko.grbelja,matej.ciglenecki}@fer.hr

Abstract

Addressing the lack of transparency in AI systems' decision-making regarding commonsense knowledge, we propose leveraging explainable artificial intelligence (XAI) techniques. Inspired by SemEval-2020 Task 4, we refine the subtask of commonsense validation to quantify the probability of a single sentence exhibiting common sense. Our approach involves fine-tuning a pretrained transformer-based model and using Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP) for model explainability. Our study investigates the impact of individual words on commonsense validation decisions and contributes to the broader objective of improving the reliability and trustworthiness of commonsense models. We observe that the two different explainability techniques yield different explanations as to how individual words attribute to the decisions on the commonsense nature of a sentence.

1. Introduction

Commonsense knowledge is crucial for human intelligence, allowing us to understand the world, infer intentions, and predict outcomes. However, incorporating commonsense knowledge into artificial intelligence (AI) remains challenging (Davis and Marcus, 2015). Previous research has made progress in tasks like question-answering (Talmor et al., 2018), natural language inference (Zellers et al., 2018), and visual commonsense reasoning (Zellers et al., 2019). Although transformer-based models have been successful in addressing these challenges, they are opaque in their decision-making process, unlike humans who possess individual and inherent common sense (Smith, 1995). Therefore, it is important for humans to understand the reasoning path of these models.

Despite the advancements, there is a research gap in explicitly validating and quantifying the common sense of individual sentences. Inspired by SemEval-2020 Task 4: Commonsense Validation and Explanation (ComVE) (Wang et al., 2020), specifically subtask A (Validation), we build upon the task organizers' work in commonsense validation. However, we modify the original task formulation by focusing on quantifying the probability of an individual sentence being commonsense or not. We also adapt the provided dataset for a better fit with our task requirements.

Using eXplainable artificial intelligence (XAI) techniques, our goal is to examine the impact of individual words on the predicted degree of common sense of a sentence. To achieve this, we fine-tuned a pretrained transformer-based neural language model called Decoding-enhanced BERT with Disentangled Attention (DeBERTa) to create a commonsense validation model. To gain insights into the model's explainability, we utilized two techniques: Local Interpretable Model-agnostic Explanations (LIME) (Ribeiro et al., 2016) and SHapley Additive exPlanations (SHAP) (Lundberg and Lee, 2017). By pursuing this approach, our aim is to contribute to the broader objective of enhancing model explainability through the understanding and application of commonsense knowledge.

2. Background

2.1. Commonsense Knowledge

Incorporating commonsense knowledge into AI tasks is essential for bridging the gap between human understanding and machine intelligence. It serves as a foundation for reasoning, decision-making, and effective communication, enabling AI systems to gain a deeper comprehension of the world, improve predictive accuracy, and engage in intuitive interactions. However, challenges arise due to the implicit nature, contextual reliance, and variability of commonsense knowledge across time and cultures (Smith, 1995). Traditional machine learning models struggle to grasp intuitive concepts like cause-and-effect relationships and social norms, as formalizing and representing these concepts is complex. While previous approaches employ external knowledge graphs (Speer et al., 2017), they face limitations in coverage and scalability. Recent advancements have explored contextualized embeddings to enhance the integration of common sense into ML models (Bouraoui, 2022).

2.2. Explainability

Explainability is crucial in XAI for human understanding and trust in AI system decision-making. A similar term, interpretability, involves presenting model outputs in a human-understandable manner, emphasizing cause-and-effect connections (Doshi-Velez and Kim, 2017). In contrast, explainability focuses on understanding the internal mechanisms of an AI system and the processes influencing its decision-making (Linardatos et al., 2020). While an interpretable model enhances understanding, it does not provide insight into the model's inner logic. This paper specifically investigates explainability to analyze the impact of input features on commonsense validation decisions.

XAI techniques can be classified based on their emphasis and model compatibility (Linardatos et al., 2020). Two categories are model-specific methods designed for specific model families and model-agnostic methods applicable to any model. Techniques can also be categorized as local or global, with local methods providing explanations for in-

Text Analysis and Retrieval 2023: Course Project Reports (TAR 2023), pages 55–59, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2023.

dividual model decisions using simpler interpretable models based on neighboring context (Ras et al., 2022). In our study, we focus specifically on model-agnostic and local techniques, namely LIME and SHAP, to explore explainability in a black-box model for sentence-level commonsense validation. These posthoc methods analyze pretrained models, as opposed to intrinsic methods integrated into the explained model. Note that we do not investigate the combination of LIME and SHAP known as Kernel SHAP in this paper (Lundberg and Lee, 2017).

2.2.1. LIME

Local Interpretable Model-agnostic Explanations (LIME) is an explanation technique proposed by Ribeiro et al. (2016). It provides local interpretability for ML models by sampling neighboring instances of input and generating perturbations to observe the impact of input features on the output. A simpler model, such as a linear one, is trained using these perturbations to approximate the predictions of the original model. By analyzing the weights of this simpler model, LIME showcases the contribution of each feature, even for textual input where it can be a single word in a sentence, to the output of the original model by comparing distances from the perturbations to the input instance.

2.2.2. SHAP

The SHapley Additive exPlanations (SHAP) framework, introduced by Lundberg and Lee (2017), provides an explanation mechanism similar to LIME, attributing contributions to each feature of input towards a specific prediction. By leveraging SHAP values based on Shapley values from cooperative game theory, SHAP constructs a simplified local approximation model. SHAP values quantify the change in the expected model prediction when a specific feature is considered, offering a measure of the feature's impact on the prediction. Unlike LIME, SHAP values ensure desirable properties such as local accuracy and consistency in the explanations provided.

3. Related Work

The previously mentioned SemEval-2020 Task 4 (ComVE) (Wang et al., 2020) consists of three subtasks, two discriminative (A, B) and one generative (C). For our approach, we are inspired by subtask A, but other tasks also tackle similar commonsense issues. Transformer-based pre-trained language models (PLMs), such as BERT, have become the standard for solving ComVE subtasks. They have demonstrated state-of-the-art performance in SemEval 2020 Task 4 (Fadel et al., 2020). Zhang et al. (2020) achieved 97% accuracy on subtask A using K-BERT and knowledge graph triples. Huy et al. (2022) recently surpassed the state-of-the-art results using ensemble learning with RoBERTa, DeBERTa, and ELECTRA, highlighting the effectiveness of output aggregation from individual models.

While Liu et al. (2023) addressed a related problem, there is a lack of work specifically focused on predicting the degree of sentence commonsense. Most recent research has concentrated on SemEval 2020 Task 4, approaching the problem as binary classification. In this paper, inspired by subtask A, we propose a novel approach that brings explainability to the task of commonsense validation.

Table 1: Example of the original dataset format. Label 0 indicates that the sentence with index 0 is non-commonsense.

Index	Instance	Label
0	*He walked his fish.,He walked his dog.*	0
1	*She eat a lot of food.,Food eat her a lot.*	1

Table 2: Example of the modified dataset format. Label 1 indicates that the sentence is commonsense, 0 not.

Index	Instance	Label
0	*He walked his fish.*	0
1	*He walked his dog.*	1
2	*She eat a lot of food.*	1
3	*Food eat her a lot.*	0

4. Method

4.1. Model

For the model, we chose a pre-trained DeBERTaV3 trained on 160GB data (He et al., 2021). We opted for the smallest version of the pre-trained model (DeBERTaV3$_{\text{small}}$) which has 44M parameters. To perform the downstream task of binary classification, we use DeBERTaV3's features in combination with a linear head classifier.

4.2. Dataset

The original dataset contains 11,997 sentence pairs with binary labels indicating which sentence violates common sense. Each pair consists of two sentences with the same structure but differ in a few words (Table 1). The dataset is divided into training (10,000 pairs), test (1,000 pairs), and validation (997 pairs) subsets.

To align with our modified problem formulation, we made changes to the original dataset while maintaining the predefined train/test/validation split. Each sentence pair was split into two separate instances: one representing the non-commonsense sentence labeled as 0, and the other representing the remaining sentence as 1, indicating common sense alignment. This modification effectively doubled the dataset size, resulting in 20,000 instances for training, 2,000 for testing, and 1,994 for validation (Table 2).

5. Experiments

5.1. Packages

To train the model and perform the analysis, we used the following Python packages: PyTorch [1] and transformers [2] for training the model, NLPAug [3] for sentence augmentations, shap [4] and lime [5] for commonsense analysis, and tensorboard [6] for experiment tracking.

[1] https://github.com/pytorch/pytorch
[2] https://github.com/huggingface/transformers
[3] https://github.com/makcedward/nlpaug
[4] https://github.com/slundberg/shap
[5] https://github.com/marcotcr/lime
[6] https://github.com/tensorflow/tensorboard

5.2. Experiment Details

We used the dataset and split defined in 4.2.. We fine-tuned the model with binary cross entropy as the loss function, AdamW optimizer (Loshchilov and Hutter, 2017) with default PyTorch parameters, and a cosine learning rate scheduler accompanied by a linear warm-up scheduler with a 0.1 ratio. Models were trained between 3 and 5 epochs (60,000-100,000 steps). During the training, we evaluate the model on the validation set every 500 steps and save the model with the highest Macro F1. Every model was trained on a single RTX 3060 mobile. It took us a total of 9 hours to train 28 different models.

5.3. Augmentation

We used NLPAug to randomly swap words during training. To avoid turning common sense into non-common sense, this augmentation was applied only to sentences that were already non-commonsense, as it is likely that they would remain so after the augmentation. Each example in the batch had a 50% chance of being augmented. If applied, it swapped 30% of the words in the sentence. For instance, if a sentence had 14 words, 4 of them would be swapped.

5.4. Hyperparameter Optimization

Multiple experiments were conducted to find the best hyperparameter combination. The optimal set was determined based on the maximum Macro F1 metric on the validation set. Augmentation, learning rate, and model freezing strategy had the most impact on Macro F1, while other hyperparameters remained constant during training.

Among the 28 trained models, the 12 best-performing models had BERT freezing disabled, while the rest had it enabled. Out of the 5 best models, only one (2nd best) utilized data augmentation described in 5.3.. Attempting to fine-tune only the linear classification head by freezing the parameters of DeBERTaV3 resulted in inferior results. Therefore, the final model was fine-tuned without freezing any parameters. The hyperparameters for our best model are as follows: BERT freezing set to FALSE, no data augmentation applied (Augmenter = None), the learning rate of 5×10^{-6}, and a training duration of 5 epochs while the metrics on the validation set are the following: Accuracy = 0.862, Macro F1 = 0.863, ROC-AUC = 0.862.

The word swapping augmentation did not improve the model's performance. The augmented sentences could further compromise the sentence's grammar, making them even less meaningful. This may cause the model to learn both correct and incorrect grammar instead of learning the overall semantics of the sentence. Augmented sentences that lack sense and potentially have grammatical errors may inadvertently push the model to focus more on distinguishing between correct and incorrect grammar.

5.5. Baselines

In evaluating our model, we compared it to two baselines: random labeling and a logistic regression model using TF-IDF vectorization. Results for the baselines and our model are shown in Table 3.

Table 3: A baseline model comparison of metrics obtained on the validation set.

Baseline	Accuracy	Macro F1	ROC-AUC
Random	0.500	0.500	0.500
Logistic Regression	0.574	0.583	0.574
Our Model	0.862	0.863	0.862

6. Explainability Analysis and Discussion

Due to the short form of the paper and the local nature of the XAI techniques in question, meaning that an individual sentence can be analyzed at a time, we will showcase an explanation analysis on a pair of sentences. This pair was randomly selected from the original dataset and mapped to our modified dataset, where each sentence is treated as a separate instance. The utilization of paired sentences ensures that one sentence aligns with common sense expectations and that the other deviates from them.

To determine the commonsense probability of each example, we utilize our commonsense validation model. The rounded-off probabilities are also provided in the LIME visualizations, as depicted in Figures 1 and 2.

During LIME explanation generation, up to top 10 most influential features, corresponding to words in the sentence, are considered. The neighborhood size for learning the linear model associated with a specific sentence is limited to 150 due to hardware constraints. It is crucial to acknowledge that these explanations may not fully capture the intricate complexity and variability of the underlying model due to the limited neighborhood size. As a result, the reliability of the explanations might be somewhat compromised.

In the process of SHAP explanation generation, the computation of SHAP values is performed for each sentence individually. Subsequently, the resulting positive and negative feature contributions are visually presented for both possible classifications: LABEL_0 (or NCS, representing non-commonsense) or LABEL_1 (or CS, representing commonsense). The base value in the visualizations represents the prediction which would occur if if we did not know any features to the current output, while the output value $f_{LABEL0/1}(inputs)$ refers to the actual prediction (Lund-

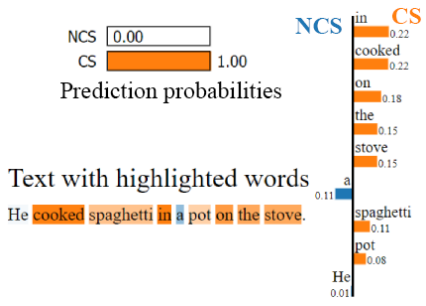


Figure 1: LIME commonsense input explanation. NCS indicates the non-commonsense class, CS commonsense.

berg and Lee, 2017). The individual word SHAP values add up into the final output value, the SHAP value of the input. The positive (red) SHAP values attributed to the words indicate that the presence of the specific word increases the likelihood of the observed label, whereas the negative (blue) suggests the opposite.

6.1. Commonsense Example

The provided sentence, *He cooked spaghetti in a pot on the stove*. is categorized as CS in the modified validation dataset. The commonsense validation model assigns a prediction probability of 0.9986 for the CS class.

The LIME explanation in Figure 1 highlights the relevance of words like *in*, *cooked*, *on*, *the*, *stove*, *spaghetti*, and *pot* for the commonsense interpretation. Among these words, *cooked* and *in* have the highest coefficient value of 0.22, indicating their crucial role in conveying the emerging commonsense nature of the sentence. These words emphasize the act of preparing spaghetti and its specific positioning in relation to the pot. While *spaghetti* and *pot* have slightly less significance, they still contribute to the overall context understood by the model. Conversely, *a* and *He* have weaker associations with the non-commonsense interpretation, as reflected by their coefficients of 0.01 and 0.11, respectively.

The SHAP analysis, shown in Figure 3, reveals a different explainability perspective on the sentence's commonsense nature. The base and the output value for both classes are the same, which suggests that the sentence features captured by the SHAP values have minimal impact on the model's final prediction. Nonetheless, we can observe that the words that positively contribute the most to the NCS label are *spaghetti* (0.374) and *cooked* (0.373), serving as strong indicators of the sentence being classified as NCS. Somewhat expected, the same words carry the most contribution in the possibility of CS classification, but now in a negative manner. The opposing logic applies to the words *in*, *He*, etc., due to yielding the most negative contributions for NCS classification and positive for CS.

6.2. Non-Commonsense Example

The provided sentence *He cooked a pot inside of spaghetti*. is categorized as NCS in the modified validation dataset. The commonsense validation model assigns a prediction probability of 0.9997 for the NCS class.

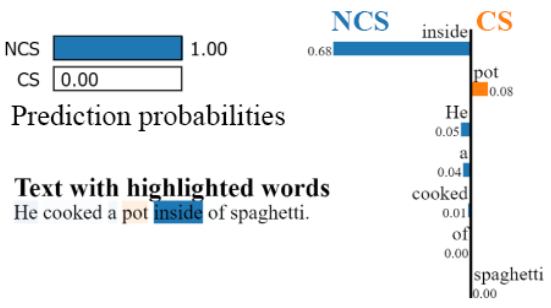


Figure 2: LIME non-commonsense input explanation. NCS means the non-commonsense class, CS commonsense.

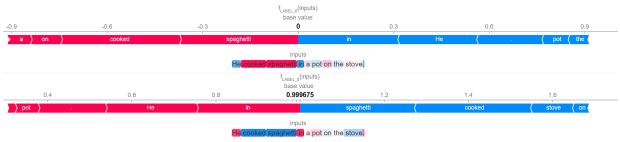


Figure 3: SHAP explanation of a commonsense sentence. The top representation indicates contributions to non-commonsense classification, bottom commonsense.

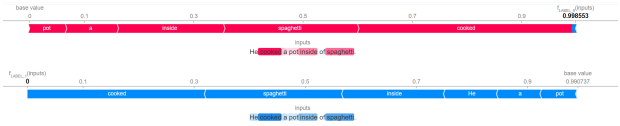


Figure 4: SHAP explanation of a non-commonsense sentence. The top representation indicates contributions to non-commonsense classification, bottom commonsense.

The LIME explanation is given in Figure 2. The word *inside* has a coefficient of 0.68, indicating its strong contribution to the non-commonsense interpretation. This suggests that the idea of pot placement inside spaghetti is a key factor in determining the non-commonsense nature of the sentence. Other words in the sentence do not show strong contributions to the non-commonsense classification. The word *pot* demonstrates a slight connection to the commonsense interpretation.

The SHAP analysis, depicted in Figure 4, demonstrates that all words in the sentence strongly contribute to the NCS classification while having a negative impact on the CS classification. This aligns with the expected classification of the sentence, indicating that the presence of these words serves as reliable indicators of the NCS label.

7. Conclusion

In conclusion, this paper addresses the challenge of commonsense validation at the sentence level, aiming to quantify the probability of an individual sentence being commonsense or not. Our approach builds upon existing research on commonsense knowledge and XAI, addressing a research gap in explicitly quantifying the common sense of individual sentences.

To achieve this, we fine-tune a transformer-based PLM and employed XAI techniques, namely LIME and SHAP, to analyze the model's explainability. By focusing on the explainability of our model and analyzing the impact of individual words on commonsense validation, we contribute to the broader objective of enhancing commonsense model transparency and trustworthiness. Our findings reveal that employing different explainability techniques to our proposed model results in distinct explanations regarding the contribution of individual words to the determination of a sentence's commonsense nature.

References

Zied Bouraoui. 2022. *Inducing Commonsense Knowledge Using Vector Space Embeddings*. Ph.D. thesis, Univer-

sité d'Artois.

Ernest Davis and Gary Marcus. 2015. Commonsense Reasoning and Commonsense Knowledge in Artificial Intelligence. *Communications of the ACM*, 58(9):92–103.

Finale Doshi-Velez and Been Kim. 2017. Towards a Rigorous Science of Interpretable Machine Learning. *arXiv preprint arXiv:1702.08608*.

Ali Fadel, Mahmoud Al-Ayyoub, and Erik Cambria. 2020. JUSTers at SemEval-2020 Task 4: Evaluating Transformer Models against Commonsense Validation and Explanation. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 535–542, Barcelona (online), December. International Committee for Computational Linguistics.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving Deberta Using Electra-Style Pretraining with Gradient-Disentangled Embedding Sharing. *arXiv preprint arXiv:2111.09543*.

Ngo Quang Huy, Tu Minh Phuong, and Ngo Xuan Bach. 2022. Autoencoding Language Model Based Ensemble Learning for Commonsense Validation and Explanation. *arXiv preprint arXiv:2204.03324*.

Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. 2020. Explainable AI: A Review of Machine Learning Interpretability Methods. *Entropy*, 23(1):18.

Jiacheng Liu, Wenya Wang, Dianzhuo Wang, Noah A. Smith, Yejin Choi, and Hannaneh Hajishirzi. 2023. Vera: A General-Purpose Plausibility Estimation Model for Commonsense Statements. *arXiv preprint arXiv:2305.03695*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101*.

Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. *Advances in neural information processing systems*, 30.

Gabrielle Ras, Ning Xie, Marcel Van Gerven, and Derek Doran. 2022. Explainable Deep Learning: A Field Guide for the Uninitiated. *Journal of Artificial Intelligence Research*, 73:329–397.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?" Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Barry Smith. 1995. Common Sense. In Barry Smith and David Woodruff Smith, editors, *The Cambridge Companion to Husserl*, pages 394–437. New York: Cambridge University Press.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. *arXiv preprint arXiv:1811.00937*.

Cunxiang Wang, Shuailong Liang, Yili Jin, Yilong Wang, Xiaodan Zhu, and Yue Zhang. 2020. SemEval-2020 Task 4: Commonsense Validation and Explanation. *arXiv preprint arXiv:2007.00236*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. *arXiv preprint arXiv:1808.05326*.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. From Recognition to Cognition: Visual Commonsense Reasoning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6720–6731.

Yice Zhang, Jiaxuan Lin, Yang Fan, Peng Jin, Yuanchao Liu, and Bingquan Liu. 2020. CN-HIT-IT.NLP at SemEval-2020 Task 4: Enhanced Language Representation with Multiple Knowledge Triples. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 494–500, Barcelona (online), December. International Committee for Computational Linguistics.

Clustering Paragraphs by Author Using Binary Classifiers and Pairwise Probabilities for Being in Same Cluster

Josip Pardon, Marko Damjanić, Valentin Vuk

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{josip.pardon,marko.damjanic,valentin.vuk}@fer.hr

Abstract

This research investigated a novel clustering method that utilizes binary classifiers to generate pairwise probabilities of each element being in the same cluster. Several binary classifiers were trained to predict whether two paragraphs were written by the same author or not. However, the combination of these classifiers with the proposed method did not yield results better than a random classifier. Further experimentation with our method and more advanced classifiers is a promising avenue for future work.

1. Introduction

The research was initially based on the PAN style change detection in 2021, but it deviated from this framework after conducting a series of experiments and engaging in brainstorming sessions. We leveraged their dataset to construct a more suitable one for binary classification training. This modified dataset consisted of two paragraphs which served as **x**, with a corresponding label of 0 or 1, denoting whether the paragraphs were written by the same author or not. Rather than solely focusing on participating in the competition and striving to surpass the best previous scores, our objective shifted towards investigating methods for clustering paragraphs. Consequently, we devised a novel clustering method, which exhibits applicability beyond the realm of texts and authors. We refer to this approach as "clustering using binary classifiers and pairwise probabilities."

2. Related work

The most frequently employed methods for clustering data typically include k-means clustering, Gaussian mixtures, and others. Conversely, clustering based on the pairwise connection strength between each pair of elements remains an unpopular research topic, with only a handful of papers addressing it (Alush et al., 2016; Chehreghani, 2022; Duan and Dunson, 2021; Bulò et al., 2010). Surprisingly, despite its simplicity and intuitiveness, we could not locate any resource that even mentions this approach, except for a solitary question on StackExchange with inadequate answers. In the following sections, we will provide a comprehensive and detailed exposition of this method, utilizing a simple example to illustrate it.

3. Our idea

3.1. Example

Consider three elements: A, B, and C, which need to be clustered. We have acquired probabilities from a binary classifier indicating the likelihood of each pair belonging to the same cluster: $P(A, B) = 0.6$, $P(A, C) = 0.7$, and $P(B, C) = 0.3$. The number of possible clusterings for the set $\{A, B, C\}$ can be determined using Bell's numbers. However, as the set size increases, the computation becomes increasingly complex.

3.2. Encoding

Let us now examine one possible clustering of our set: $\{\{A, B\}, \{C\}\}$. In order to facilitate computations and other operations, it is advantageous to encode this clustering in a more convenient format. The underlying concept is rather straightforward: we create a dictionary that includes all possible pairs of elements as keys, with values of either 0 or 1 depending on whether the elements are clustered together or not. Consequently, the clustering $\{\{A, B\}, \{C\}\}$ can be encoded as $\{\{A, B\} : 1, \{A, C\} : 0, \{B, C\} : 0\}$, which can also be visualized as a binary sequence $[1, 0, 0]$.

3.3. Probability of Clustering

The encoding of a clustering can be utilized to calculate its probability by considering the probabilities associated with each pair of elements being in the same cluster or not. For elements X and Y being in the same cluster, we incorporate the probability $P(X, Y)$ in the final formula, and $1 - P(X, Y)$ if they are not. Therefore, the probability for the clustering $\{\{A, B\} : 1, \{A, C\} : 0, \{B, C\} : 0\}$ can be expressed as $P(A, B) \cdot [1 - P(A, C)] \cdot [1 - P(B, C)]$.

3.4. Algorithm

Our clustering algorithm explores the space of possible clusterings to identify the most probable configuration. This can be formally expressed as the maximization of the following function:

$$L(\mathbf{c}) = \prod_{i \neq j} P_{ij} * 1(e_i, e_j) + (1 - P_{ij}) * (1 - 1(e_i, e_j)) \quad (1)$$

where **c** is clustering, e_i is element, and $1(e_i, e_j)$ is 1 if e_i and e_j belong to same cluster, 0 otherwise. P_{ij} is short form of $P(e_i, e_j)$.

3.5. Greedy Method

One could argue that it is unnecessary to iterate through all possible combinations and instead propose constructing an encoding for the most probable clustering by greedily examining pairwise probabilities. If $P(X, Y) < 0.5$, then $1 - P(X, Y) > 0.5$, indicating that X and Y should not be in the same cluster (thus, $\{X, Y\} : 0$ is placed in the encoding). Conversely, if $P(X, Y) > 0.5$, X and Y should

Text Analysis and Retrieval 2023: Course Project Reports (TAR 2023), pages 60–64, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2023.

Table 1: Average bitwise overlap percentage (BOP) between greedily obtained clustering and most probable feasible one.

BOP	no. of elements		
88.50%	3	72.93%	7
80.33%	4	72.00%	8
78.70%	5	70.47%	9
76.46%	6	70.22%	10
		67.09%	11

be chosen to be in the same cluster (resulting in $\{X, Y\} : 1$ being added to the encoding). While this approach may yield the largest multiple of probabilities (as each part will be larger than 0.5), it can also result in an infeasible clustering. Let us apply this method to our exemplary probabilities to illustrate the potential issues it may introduce.

$P(A, B) = 0.6$, therefore A and B should be in same cluster - $\{A, B\} : 1$. $P(A, C) = 0.7$, therefore A and C should be in same cluster - $\{A, C\} : 1$. $P(B, C) = 0.3$, therefore $1 - P(B, C) = 0.7$ so B and C should not be in same cluster - $\{B, C\} : 0$. Obtained clustering is $\{\{A, B\} : 1, \{A, C\} : 1, \{B, C\} : 0\}$ which is not feasible: if A and B are in same cluster, and A and C are in same cluster, B and C can not be in different clusters.

3.5.1. Performance of Greedy Method

However, a greedily constructed clustering can serve as a useful starting point in the search for the most probable feasible clustering. To support this claim, we conducted a comparison of the bitwise overlap percentage between these two clusterings. We executed the algorithm 200 times, randomly selecting $P(X, Y)$ for each pair of elements and determining the most probable clustering by iterating through all feasible configurations and constructing it greedily. For a set of 10 elements, the average bitwise overlap percentage between the greedily obtained clustering and the most probable feasible one was 70.22%. With 9 elements, this value increased to 70.47%. The complete results of the experimentation are presented in Table 1.

As anticipated, the bitwise overlap percentage decreases as the number of elements increases. This is because the number of pairs increases, thereby increasing the probability of selecting an incorrect number (0 or 1). Still, greedy method can be utilized as heuristic.

3.6. Combinatorial Explosion and Solutions

The number of possible clusterings or partitions, quantified by Bell's number, becomes staggering for sets with more than 14 elements, reaching billions. To mitigate this challenge, a clever function for generating all clusterings or partitions can be employed. Instead of generating all configurations and subsequently discarding those that violate specified constraints (such as a maximum number of authors behind paragraphs in a document, which could be set at 5 in our dataset), it is more efficient to avoid generating such infeasible configurations altogether. However, even with this improvement, the algorithm's performance may still be hindered by the significant number of iterations required. Consequently, employing an optimization algorithm such as a genetic algorithm (GA) proves to be a more favorable approach.

3.6.1. Genetic Algorithm and Louvain Partitions

We conducted an extensive literature search but found no previous work that utilized the genetic algorithm or any other optimization algorithm in the manner we presented for finding the most probable clustering. Most of the papers we reviewed focused on employing the genetic algorithm for clustering in a similar fashion to k-means (Gesú et al., 2005; Maulik and Bandyopadhyay, 2000). In order to compare the genetic algorithm with the brute force method, we developed three models. In these models, individuals (or more precisely, their chromosomes) were represented as binary sequences encoding clusterings. The fitness function returned the probability of a clustering if it was feasible, and 0 otherwise. As mentioned earlier, the greedily constructed clustering encoding exhibited an average 70.47% bitwise overlap with the encoding of the most probable feasible clustering (in the case of 9 elements). Consequently, we utilized the greedily constructed clustering as the "Adam and Eve" for the initial population in the genetic algorithm. Specifically, all starting individuals were generated by randomly switching 30% of the bits in the greedily constructed clustering encoding. For comparison, another genetic model employed randomly generated clusterings as the starting population. Each starting individual in this model was created as follows: a random number of brackets (up to a specified upper limit) were chosen, each element was placed in a random bracket, empty brackets were removed, and the final result was encoded as a binary sequence. It is important to note that starting populations generated in this manner could not have infeasible individuals, whereas greedily created populations could. The third model was based on Louvain partitions, where the elements to be clustered were treated as nodes in a network, and the pairwise probabilities served as weights between the nodes. The parameters for both genetic models were set as follows: $num\_generations = 1000$, $num\_parents\_mating = 2$, $gene\_type = int$, $crossover\_probability = 0.3$, and $mutation\_probability = 0.1$, while the remaining parameters were left at their default values using PyGAD 3.0.1. The parameters for the Louvain partition model were all left at their default values using NetworkX 3.1.

3.6.2. Testing

We conducted a total of 200 runs of the algorithm, where we randomly selected $P(X, Y)$ for each pair of elements. We then found the most probable clustering by iterating through all feasible ones. Clusterings were also generated with aforementioned models. In this experiment, we used 9 elements. We calculated the average bitwise overlap percentages between the most probable feasible clustering and the clusterings outputted by the models. The results are presented in Table 2.

It was quite surprising to observe that the model with randomly generated feasible clusterings as the starting population performed the best, while the greedy model performed the worst. Upon analyzing the best solutions in each gen-

Table 2: Average bitwise overlap percentage between clustering obtained by model and most probable feasible one.

Bitwise overlap percentage	algorithm
66.07%	GA greedy
77.37%	GA random
74.94%	Louvain partitions

Table 3: Percentage of Feasible Encodings (PFE) for Clustering of 5 Elements With Each Possible Number of Ones

no. of ones	PFE		
0	100%	5	0%
1	100%	6	2.38%
2	33.33%	7	0%
3	8.33%	8	0%
4	4.76%	9	0%
		10	100%

eration, it was discovered that the issue lies in the fact that the greedily created starting population often lacked feasible individuals/solutions. Furthermore, it was observed that the first feasible solution that appeared in the population was consistently (in approximately 25 test runs) a sequence predominantly consisting of zeros, with very few ones. This encoding represents a clustering where practically every element is in a separate cluster. This observation is not unexpected, as the fewer ones the encoding has, the higher the likelihood of it being feasible. To support this claim, a test was conducted using a list of 5 elements to be clustered. The encoding for clustering of 5 elements is a binary sequence with 10 digits. For each possible number of ones in the encoding (ranging from 0 to 10), all possible encodings were generated and the percentage of feasible encodings was measured. The results are presented in Table 3.

Furthermore, it is worth noting that out of all possible encodings for clustering of 5 elements, only 5.08% of them were found to be feasible. This percentage drops significantly to a mere 0.04% for encodings of clustering with 7 elements. These findings shed light on another interesting observation: for both genetic algorithms (GAs), the best solution across all generations (referred to as the algorithm's final output) was often found within the first 20-30 generations, despite a total of 1000 generations being executed. Moreover, in many cases, the GA with randomly generated feasible clusterings had its best solution within the starting population (or the 0-th generation). This suggests that only a few instances of evolution (mating and offspring reproduction phases, generation switches) led to improved solutions. The reason behind this phenomenon lies in the fact that very few encodings or solutions are feasible. Consequently, the probability of producing a feasible child through mating and reproduction becomes exceedingly small. As a result, the vast majority of offspring generated are inferior to the current best solution. Additionally, this explains why running the algorithm with a "keep_elitism" parameter of 1 yields slightly better results on average. These findings indicate that the proposed encoding for clusterings (using 0 or 1 for each pair of elements) is only suitable when brute force combinations exploration can be applied, which is typically the case when dealing with a small number of elements to be clustered. However, as the number of elements increases, the low probability of encountering a feasible encoding becomes a significant obstacle for combinatorial optimization algorithms, such as GAs with binary sequences as individuals. Nevertheless, it is important to emphasize that although the proposed encoding may not be suitable as the sole input for such algorithms, it remains valuable for calculating clustering probabilities, as explained in Section 3.3.. Therefore, it should not be completely disregarded, but rather utilized in conjunction with another encoding scheme that is more suitable for combinatorial optimization purposes.

3.7. Additional

3.7.1. Computer Friendly Version

It is worth noting that summing logarithm of probabilities in the formula introduced in Section 3.3. is more convenient than multiplying, as multiplying many small numbers may lead to the underflow problem.

3.7.2. Bitwise Overlap Percentage (BOP) Issue

Let's say that we have list of 4 elements that need to be clustered: $\{0,\ 1,\ 2,\ 3\}$ for which correct clustering is: $\{\{0,2,3\},\{1\}\} - \{\{0,1\} : 0,\{0,2\} : 1,\{0,3\} : 1,\{1,2\} : 0,\{1,3\} : 0,\{2,3\} : 1\}$. Algorithm outputed: $\{\{0\},\{2\},\{3\},\{1\}\} - \{\{0,1\} : 0,\{0,2\} : 0,\{0,3\} : 0,\{1,2\} : 0,\{1,3\} : 0,\{2,3\} : 0\}$. BOP between outputed clustering and correct one is 0.5. Let's now look at another example where correct clustering is: $\{\{0,1,2,3\}\} - \{\{0,1\} : 1,\{0,2\} : 1,\{0,3\} : 1,\{1,2\} : 1,\{1,3\} : 1,\{2,3\} : 1\}$ and outputed one is: $\{\{0\},\{1,2,3\}\} - \{\{0,1\} : 0,\{0,2\} : 0,\{0,3\} : 0,\{1,2\} : 1,\{1,3\} : 1,\{2,3\} : 1\}$. Here, BOP is also 0.5. So, in both examples, BOP between encoding of correct clustering and outputed one is 0.5, even tough in second example outputed clustering is much closer to correct (only 0 was in wrong cluster).

4. Classification of same author paragraphs

4.1. Baseline model

For the baseline, we employed a straightforward logistic regression model that utilized the difference of readability features extracted from paragraph pairs. These features were computed using the Textstat (Ward, 2023) package, allowing us to analyze and assess the readability characteristics of the text.

4.2. Our approach

Recent research has demonstrated that Transformer-based models, despite their primary focus on sentence-level semantics, can also be effectively employed for tasks requiring greater emphasis on syntactic and stylistic information (Tzu-Mi Lin, 2022). Therefore, we will leverage the power of Facebook AI's pretrained RoBERTa (Yinhan Liu, 2019)

model to generate our embeddings. To align the dataset with our requirements, we perform preprocessing steps. Subsequently, we employ multiple supervised models, including logistic regression, SGDClassifier with logistic regression loss, and Random Forest Classifier. Additionally, we experimented with cosine similarity and L2 distance as similarity measures. However, these classification models yielded results no better than random chance.

4.3. Preprocessing

We initially created a new dataset by pairing all possible paragraphs within each document. This resulted in 279,761 data points in our training set and a total of 60,365 data points in our test set. Due to the nature of the data, a slight class imbalance occurred, with the training set consisting of 125,679 same-author data points and 154,082 different-author data points, while the test set had 27,727 same-author data points and 32,638 different-author data points.

To address the class imbalance issue, we incorporated class weighting into our supervised models during subsequent stages. With our updated dataset, we proceeded to preprocess the paragraph pairs. Our first step involved splitting each paragraph into sentences. However, this process posed some challenges as the text contained various prefixes, websites, acronyms, and dates, making it difficult to accurately split sentences based solely on punctuation marks such as periods, exclamation marks, or question marks. To overcome this, we employed regular expressions to replace these anomalies with a special token, minimizing the chances of incorrect sentence separations.

Next, we employed the RoBERTa tokenizer to tokenize the modified sentences. These tokenized sentences were then processed by RoBERTa, resulting in an embedding tensor of dimensions s x l x 768, where s represents the number of sentences and l represents the length of the longest sentence.

Given the variability in sentence length, we made the decision to sum up the embeddings along the length dimension, resulting in a 1x768 tensor for each sentence. While averaging the embeddings is an alternative approach, summing them preserves the information related to sentence length, which can be highly valuable, as demonstrated in previous works (Aarish Iyer, 2020). To obtain the final embeddings for each paragraph pair, we took the average of all the sentence embeddings, resulting in a final tensor comprising 768 values.

4.4. Tested models

We experimented with several supervised models, namely logistic regression, SGDClassifier with logistic regression loss, and Random Forest Classifier. Grid search was employed to determine the optimal hyperparameters for the Random Forest Classifier, while the other models were trained using their default parameters. The accuracies and F1-scores of these models are presented in Table 4. It is crucial to highlight that these results are obtained from our custom-created dataset, which may account for the relatively lower performance observed. All models were trained using Python's ML library Scikit-learn (Pedregosa, 2011)

Table 4: Accuracy and F1-scores of Tested Models

Model	Accuracy (%)	F1-score
Log_Reg(TextStat)	56	0,081
Log_Reg(RoBERTa)	59,1	0,578
SGD(RoBERTa)	58,4	0,505
Random Forest(RoBERTa)	61,1	0,545

4.5. Results

Based on the findings presented in Table 4, several important observations can be made. Firstly, the Textstat representation alone is insufficient for capturing the essence of the author within a paragraph. This is understandable since different authors can have similar text readability levels. Additionally, the models trained on our preprocessed datasets yielded subpar results. There could be various reasons for this outcome. Firstly, it is worth noting that the top-performing models in the competition, from which our dataset was derived, achieved F1-scores ranging from 0.65 to 0.75 for the author change task. These models were significantly more complex, yet still fell short of achieving excellent results. It is plausible that our preprocessing steps can be refined, and it would be beneficial to rerun our model experiments after implementing these improvements. Furthermore, incorporating additional features alongside our RoBERTa representations could help capture more stylistic and syntactic information from the paragraphs.

5. Integrating Classifiers With Our Method

Unfortunately, the integration of our proposed clustering method with the binary classifiers for Author Change Detection did not yield satisfactory results. Despite the strong potential demonstrated by our clustering concepts, a highly capable binary classifier is required to truly assess their effectiveness. However, building such a classifier would be exceptionally challenging or even impossible, as indicated by the performance of top competitors in the competition. Furthermore, the nature of our clustering method necessitates the ability to handle unreliable data, as an inaccurate probability estimation can significantly undermine the model's capacity to achieve accurate clustering.

6. Conclusion

Our method does not require setting a predefined value for the number of clusters. Instead, our algorithm finds the optimal number of clusters implicitly. This feature is particularly advantageous for problems such as the one presented in the competition, where a small number of elements needs to be clustered. Moreover, our method has the potential to outperform mainstream clustering methods when combined with a powerful binary classifier. Suggested encoding in Section 3.2. is useful for calculating the probability of a clustering but it is not suitable for combinatorial optimization functions like the binary genetic algorithm. Utilizing out method requires powerful classifiers to be effective.

References

Soroush Vosoughi Aarish Iyer. 2020. Style Change Detection Using BERT Notebook for PAN at CLEF 2020.

Amir Alush, Avishay Friedman, and Jacob Goldberger. 2016. Pairwise clustering based on the mutual-information criterion. *Neurocomputing*, 182:284–293.

Samuel Rota Bulò, André Lourenço, Ana Fred, and Marcello Pelillo. 2010. Pairwise probabilistic clustering using evidence accumulation. In Edwin R. Hancock, Richard C. Wilson, Terry Windeatt, Ilkay Ulusoy, and Francisco Escolano, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, pages 395–404, Berlin, Heidelberg. Springer Berlin Heidelberg.

Morteza Haghir Chehreghani. 2022. Shift of pairwise similarities for data clustering. *Springeg Link*.

Leo L Duan and David B Dunson. 2021. Bayesian distance clustering. *HHS Author Manuscripts*.

Vito Di Gesú, Raffaele Giancarlo, Giosué Lo Bosco, Alessandra Raimondi, and Davide Scaturro. 2005. Genclust: A genetic algorithm for clustering gene expression data. *BMC Bioinformatics*.

Ujjwal Maulik and Sanghamitra Bandyopadhyay. 2000. Genetic algorithm-based clustering technique. *Pattern Recognition*, 33(9):1455–1465.

Varoquaux G. Gramfort A. Michel V. Thirion B.-Grisel O. Blondel M. Prettenhofer P. Weiss R. Dubourg V. Vanderplas J. Passos A. Cournapeau D. Brucher M. Perrot M. Duchesnay E. Pedregosa, F. 2011. Scikit-learn: Machine learning in python.

Yu-Wen Tzeng Lung-Hao Lee Tzu-Mi Lin, Chao-Yi Chen. 2022. Ensemble Pre-trained Transformer Models for Writing Style Change Detection.

Alex Ward. 2023. Textstat: python package to calculate readability statistics of a text object - paragraphs, sentences, articles.

Naman Goyal Jingfei Du Mandar Joshi-Danqi Chen Omer Levy Mike Lewis Luke Zettlemoyer Veselin Stoyanov Yinhan Liu, Myle Ott. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach.

Stress Analysis in Social Media

Jakov Samardžija, Antonio Babić, Mathieu Jehanno

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{jakov.samardzija,antonio.babic,mathieu.jehanno}@fer.hr

Abstract

Stress is one of the most prevalent issues that people face these days. People often express signs of stress on social media since it provides an outlet for individuals to share their thoughts, emotions, and experiences. In this paper, we focus on stress classification performance, that is, a comparison between three different models on a Dreaddit dataset. We demonstrate that there are no significant differences in models' performances. However, we unexpectedly found that LSTM performed worse than SVM and Naive Bayes. SVM with tf-idf vectors achieved the highest accuracy of 72%. Performances can be increased by acquiring more training data and experimenting with different approaches when constructing the models. We believe that the findings of this paper can serve as an effective benchmark for future work on this topic.

1. Introduction

In today's digital era, social media has become a hub of user activity, with new content being posted daily. Simultaneously, stress has emerged as a prevalent and easily observable phenomenon (Turcan and McKeown, 2019). In this context, the study of stress analysis in social media presents a valuable opportunity to understand and recognize states of mental or emotional tension. This report focuses on building systems capable of identifying posts on Reddit that exhibit stress markers and comparing their performances.

The nature of Reddit as a platform introduces a diverse range of statements across multiple domains, necessitating a broad understanding of natural language. The primary objective is to develop a system that can classify whether a post displays aspects of stress or not. To achieve this, we leverage the data collected from Reddit, which provides a rich source for studying the worries and stresses experienced by people worldwide.

To address the challenge of stress analysis in social media, we propose the utilization of the "Dreaddit" dataset, which consists of lengthy social media posts across various categories. The dataset includes both stressful and non-stressful texts, along with different ways of expressing stress. Additionally, a subset of the data has been annotated by human annotators to provide ground truth labels, explained in Section 3..

In our study, we employ three models to predict stress in the collected dataset: Naive Bayes, Long Short-Term Memory (LSTM), and Support Vector Machine (SVM) with tf-idf vectors. These models were selected based on their relevance to stress understanding and their performance characteristics.

We have certain assumptions and expectations regarding the performance of these models. The LSTM model, known for capturing long-range dependencies and context in text, is expected to excel in stress understanding, considering its ability to capture the temporal nature of text (Huang et al., 2015). The SVM model with tf-idf vectors is effective at handling well-separated classes, although this assumption may not hold true in stress analysis scenarios (Silva et al., 2010). Lastly, the Naive Bayes model, while simple and assuming independence among features, may struggle with sarcasm, negation, and other linguistic nuances (Bhattu and Somayajulu, 2012).

We evaluate the performance of the three models in terms of accuracy, precision, recall, and F1 score. Accuracy is chosen as an evaluation metric to assess overall correctness; precision is used to measure correct positive predictions; recall is used to gauge the ability to correctly identify positive instances; and the F1 score combines precision and recall into a single metric, providing a balanced measure of the model's performance by considering both false positives and false negatives. These metrics provide valuable insights into different aspects of model performance and help assess the trade-offs between correct classifications and false positives or negatives.

The objective of this paper is to compare the performance of three different models on a stress analysis task. In the subsequent sections of this report, we will review the dataset, describe our methodology, and present and discuss the results. Through this study, we aim to contribute to the field of stress analysis in social media and provide valuable insights into the problem of stress detection in social media posts.

2. Related Work

One of the prominent papers on stress analysis was done by Lin et al. (2014). The paper proposes a deep neural network model for automatically detecting psychological stress from social media data. It explores the correlations between users' stress and their social media content and behavior patterns. The model combines low-level content attributes with user-scope statistical attributes to effectively detect psychological stress. Results demonstrate the model's effectiveness and efficiency in detecting stress from micro-blog data. Rather than defining specific attributes, we use only text and labels to train our models.

3. Dataset

The dataset we use was created (Turcan and McKeown, 2019) for the purpose of analyzing stress in text data. It

Text Analysis and Retrieval 2023: Course Project Reports (TAR 2023), pages 65–69, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2023.

consists of data collected from posts on Reddit. Reddit is a social media website, very similar to a forum, on which people share their thoughts and content on specific topics. People can search for and join different subreddits that are focused on a certain topic. To collect expressions of stress, categories of subreddits were selected where members are likely to discuss stressful topics. These topics involved interpersonal conflict, mental illness, and financial need. Out of almost 200 000 posts that were scraped, only 3553 segments of those posts were assigned a binary stress label. The posts' average length is 420 tokens, providing more comprehensive data compared to other social media platforms.

Selected data was annotated using Amazon Mechanical Turk to characterize stress. Each annotator was given five randomly selected text segments and instructed to label them with the following categories: "Stress", "Not stress" or "Can't tell". Segments labeled with "Can't tell" were removed from the selected data, so the remaining data is binary labeled. Annotators labeled 3553 segments from 2929 posts, 52.33 percent of which were labeled stressful. Agreement on all labeled data was 0.47, measured with Fleiss Kappa, suggesting moderate agreement between annotators in how stress is expressed. The dataset was split into two parts: a training set and a test set. We were given both sets by the University of Zagreb. The train set consists of 2838 posts, while the test set consists of 715 posts. Due to the limited size of the dataset, we do not perform optimization of hyperparameters on a validation dataset because it would require taking a certain number of examples out of a training set, which might result in underfitting of the model.

4. Methodology

In order to correctly label Reddit posts from our dataset, we chose four models: Random Classifier, Naive-Bayes, LSTM, and SVM with tf-idf vectors. The models were implemented in Python using *sklearn*[1], *nltk*[2], and *Tensorflow*[3] libraries.

Random Classifier. The first model is the Random Classifier; just like the name suggests, this model will randomly label Reddit posts without any previous training on the data. We used this model as a baseline to compare other models' performance on the data.

Naive Bayes. Our second model is another simplistic model, that should give efficient results. Naive Bayes classifier applies Bayes' theorem with the assumption of independence among the features. It calculates the conditional probability of posts having a certain label based on the features of that post. The model is suitable for datasets with a large number of features, which makes it the perfect model for our second baseline. We expect better performance than the one from the random classifier because it does calculate probabilities of label, which can be very accurate in cases where dependencies between the features are not strong. However, in our case, we expect noticeable dependencies; therefore, we use this model as the second threshold for more complex models to outperform.

LSTM. LSTM is a recurrent neural network (RNN) designed to model sequential data, such as posts on Reddit. One of the major strengths of LSTM is its ability to capture context in large texts and effectively handle dependencies between words that are mentioned far apart in the text. This effect is realized with a hidden state that updates at each time step. The hidden state in LSTM acts as a memory cell, allowing the RNN to store and propagate information throughout the sequential data. This way, context is retained over long sequences. Using gating mechanisms (forget, input, and output gates), LSTM effectively models long-term dependencies and connects distant words in a sequence. Considering that locating stress in Reddit users' posts is a complex task and requires capturing the nuanced expressions that could point to stress, this model has the potential to be a very accurate and efficient solution. We expect this model to be the best-performing; however, that still is not a guarantee considering LSTM requires a large amount of labeled training data for tuning.

SVM (tf-idf). SVM (support vector machine) is a popular machine learning algorithm used for classification tasks, and in this model, it is combined with the TF-IDF representation for our text classification. SVM aims to find an optimal hyperplane that separates data points belonging to different classes. Tf-idf is an information retrieval metric that numerates terms within a document or a text. More information is placed on a word that appears frequently in a text while appearing less frequently in other texts. We vectorize the tfidf of the posts to represent them in vector space, which can be fit in an SVM model. We expect this model to perform better than Naive Bayes but not as well as LSTM, considering that it does not capture context as well as LSTM. This model was used as a baseline in similar work regarding positive movie reviews (Martineau and Finin, 2009) and got an accuracy of 82.85% which gives high expectations for good results.

5. Results

It took less than 5 minutes for each model to train on a CPU. We present our results in Table 1.

The random classifier achieved a 53% accuracy. It was used as a benchmark in our evaluation. All three models have performed better than the random classifier, suggesting that they managed to capture some underlying patterns or features that are indicative of stress in social media posts.

The Naive Bayes classifier achieves 70% accuracy. It correctly identifies stress-related posts about two thirds of the time. The high recall value of 0.84 implies that the Naive Bayes classifier effectively captures a substantial number of stress-related posts.

Moving on to the LSTM model, an accuracy of 66 percent was achieved. Even though LSTM is quite popular in NLP tasks such as stress analysis, its precision, recall, and F1 score fall short compared to Naive Bayes and SVM.

Finally, the SVM model utilizing the tf-idf representation stands out with an accuracy of 72% and the highest evaluation metrics among all evaluated models.

[1] https://scikit-learn.org/stable/

[2] https://www.nltk.org/

[3] https://www.tensorflow.org/

Considering these findings, all of the evaluated models have some room for improvement.

5.1. Performance differences

Since the results show certain differences in performance among the models, it is important to test the statistical significance of these differences. For that purpose, we decided to use a parametric independent t-test and a non-parametric Friedman test. The independent t-test allows us to assess whether there is a significant difference in performance between two models or algorithms. It helps determine if the observed difference in their means is statistically significant or simply due to random variation. The Friedman test is a statistical test used to compare the performance of multiple models or algorithms. It is commonly used when the data is not normally distributed or when the assumptions for parametric tests, such as the t-test or ANOVA, are violated. We do not know if these assumptions are true, which is why we decided to perform both statistical tests. Note that both tests set a null hypothesis that there are no differences in performance. Then, we calculated the p-values based on the values of the following statistics:

$$t = \frac{x_1 - x_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \quad (1)$$

where x, s, and n represent the mean, the standard deviation, and the sample size, respectively. The sample in this case is the list containing the values of the four metrics we used. For the Friedman test, the following statistic was used:

$$\chi^2 = \frac{12}{k \cdot n \cdot (k+1)} (\sum_{i=1}^{N} R_i^2 - \frac{k \cdot (k+1)^2}{4}) \quad (2)$$

where k is the number of models, n is the number of samples per model, and R is the rank assigned to each model (ranked from highest to lowest performance).

Both the t-test and Friedman test yielded p-values that are much higher than the usual 0.05 threshold, meaning that the differences in performance among the models are not significant. The lack of statistical significance in the performance difference among the models could be a result of various factors related to the dataset, models, evaluation metrics, and task characteristics. It highlights the need for careful consideration of these factors and potentially exploring alternative approaches to gain further insights into the performance differences.

6. Discussion

6.1. Expectations and results

As stated earlier, we expected LSTM to achieve the best performance. Similarly, we believed that the Naive Bayes classifier should perform the worst among the three models. Despite our assumptions and expectations, our models achieved unexpected results.

6.1.1. Naive Bayes

Since Naive Bayes is quite a simple model, we did not set high expectations for it. The 70% accuracy is quite satisfying for such a model. Plus, it is astonishing that it managed to achieve a recall of 0.84. However, the overall performance may have been affected by the assumption of independence, which may not hold true for stress-related texts on social media. We hypothesize that loosening up the independence assumption might lead to an increase in performance. This will be dealt with in the upcoming subsection.

6.1.2. LSTM

Since LSTM is widely used in many NLP tasks, it was reasonable to expect relatively high performance on stress analysis in social media posts. Considering its ability to capture the sequential nature of text and to model complex relationships between words, we expected the best overall performance from LSTM. Contrary to our expectations, the LSTM had the worst performance among the three models. It is important to note that LSTMs still have issues with learning long-term dependencies, i.e., memory capacity and uncertainty.

6.1.3. SVM with tf-idf vectors

When it comes to binary NLP tasks like this one, SVM with tf-idf vectors can perform quite well, especially if there is a clear separation between positive and negative stress-related posts. In spite of high expectations for it, it was a bit far-fetched to assume that SVM would perform better than LSTM. Nevertheless, SVM outperformed initial expectations and achieved 72 percent accuracy. Despite achieving the unexpected, SVM may have struggled in cases of class overlap.

6.2. Error analysis and improvements

In order to thoroughly understand the performance of each model, we conduct an error analysis of the models. Furthermore, we introduce potential improvements for the models that might lead to an increase in their overall performance.

6.2.1. Naive Bayes

In Table 1, one can realize that the recall is much higher than the precision when it comes to Naive Bayes. This indicates that the evaluation yielded a much higher number of false positives than false negatives. The reason for such an imbalance is the independence assumption. The Dreaddit dataset consists of posts written in context-dependent language, which is filled with sarcasm, irony, negation, and other complex sentence structures that might have created dependencies between words. In such a case, treating each word independently is an unreasonable approach that inevitably leads to lower performance.

Here is an example of a post that was falsely labeled as positive for stress: *Child Sexual Abuse is a huge problem which needs ending. Children never build a true sense of self and emotions if someone takes it away from them. I don't want to be seen as a victim. I just want people to know I understand them and hope this will make them happier. Thank you for reading and have a good day*. In this and many similar cases, the classifier viewed certain words such as "abuse" or "victim" independently and concluded that the post is stress-positive. When looking at the bigger picture, it can be easily seen that the overall tone of the message does not display stress; instead, it simply points out a societal issue.

Table 1: The performance of the models

Model	Accuracy (%)	Precision	Recall	F1 score
Random Classifier	53	0.54	0.55	0.55
Naive Bayes	70	0.67	**0.84**	**0.74**
LSTM	66	0.67	0.66	0.66
SVM (tf-idf)	**72**	**0.71**	0.77	**0.74**

The classifier could be significantly improved by mitigating the effect of independence assumption and using more efficient feature selection. This could be done by selecting an auxiliary feature in order to reclassify the text space aimed at the selected features (Zhang and Gao, 2011) or using semi-supervised learning techniques (Bhattu and Somayajulu, 2012).

6.2.2. LSTM

LSTM achieved 66% accuracy with roughly equal precision, recall, and F1 score. This is the most unexpected segment of the results since the overall performance is worse than both Naive Bayes and SVM. There are a couple of possible causes for this mediocre performance.

First, the dataset is simply too small. With 3553 examples, 2838 of which are used for training, the amount of data is not sufficient for a complex classification task such as stress analysis. LSTMs typically require a substantial amount of labeled training data to learn effectively. Second, although LSTMs are efficient at capturing long-range dependencies and contextual information in sequential data, they are not immune to certain challenges. Social media posts can involve subtle differences, sarcastic phrases, and other expressions that require a deep understanding of the underlying meaning. LSTMs might be less effective at capturing such differences. Here is an example of a post falsely labeled as negative: *I've always hated nail files. Somehow that's a part of this. God. I'm confused by it all. It's a feeling to recall it that I've carried my whole life but never understood like a cloud.* In this hard-to-read post, the stress can be noticed in the first four sentences, that is, in some words associated with stress such as "hated" or "God", which in this case is used as an interjection. However, LSTM was not capable of capturing these phrases since they represent specific context-dependent expressions.

The overall performance could be increased by obtaining more training data and fine-tuning the model architecture and hyperparameters. A vast majority of posts from the Dreaddit dataset are not labeled. Labeling these posts might cost researchers time and money, but it will almost certainly lead to more promising performances in LSTM-based models. Furthermore, experimenting with different architectures, such as stacking multiple layers or using bidirectional LSTMs could help improve the performance of the model (Huang et al., 2015).

6.2.3. SVM with tf-idf vectors

The best among the three models was SVM with tf-idf vector representation. The accuracy it achieved was 72%. The most likely reason for such a result is the use of tf-idf vectors, which assign weights to words based on their importance. This approach helped the model make better-informed decisions and achieve higher accuracy than LSTM and Naive Bayes. However, 72 % is far from satisfying. This is likely a consequence of training data insufficiency, as it was with LSTM, and the linear inseparability of classes. Moreover, in stress analysis, the boundary of the classes is not clear, which is why SVMs may not perform as expected.

The following post was falsely labeled as positive: *I was younger than 13 during my period of abuse, and one thing I'm noticing now is that I can't wrap my head around consent. People tell me that I couldn't possibly consent to anything that happened, even if I asked for it. That I didn't even know what was happening to me. It doesn't feel that way to me. On paper, I get that.* The post itself may be hard to classify, not only for SVM but for people as well. On one hand, the author of the post talks about a stressful experience. On the other hand, the overall tone of the post is rehearsed rather than tensed.

When it comes to improvements, in a complex NLP task such as stress analysis, it is crucial to have a large and diverse dataset in order to adequately train the SVM model and achieve satisfactory results. In addition, other approaches, such as non-linear SVM kernels, can lead to an increase in accuracy. Plus, ensemble methods could be explored to combine multiple SVM models and leverage their collective predictive power, leading to improved performance (Silva et al., 2010).

7. Conclusion

Stress analysis from social media is a challenging but worthy task that has huge potential for future scientific work, especially in the social science and psychology fields. The goal of our paper was to see how different models perform binary classification of the Reddit posts into positive-on-stress or negative-on-stress classes. The results showed that performance differences between the models were insignificant. This is primarily due to the small size of the evaluation set and the complexity of such a task. SVM with tf-idf vectors achieved the highest performance with a 72% accuracy. The models' performances could be improved by increasing the size of the dataset and using different approaches or architectures. Finally, this work highlights the challenges in stress analysis and presents an adequate baseline for future work on stress analysis and detection in social media posts.

References

N. Bhattu and D.V.L.N. Somayajulu. 2012. Semi-supervised learning of naive bayes classifier with feature constraints. In *Proceedings of the First International Workshop on Optimization Techniques for Human Language Technology*, Mumbai. COLING.

Z. Huang, W. Xu, and K. Yu. 2015. Bidirectional lstm-crf models for sequence tagging.

H. Lin, J. Jia, Q. Guo, Y. Xue, Q. Li, J. Huang, L. Cai, and L. Feng. 2014. User-level psychological stress detection from social media using deep neural network. In *Proceedings of the 22nd ACM International Conference on Multimedia*, page 507–516, New York, NY, USA. Association for Computing Machinery.

J. Martineau and T. Finin. 2009. Delta tfidf: An improved feature space for sentiment analysis. *Proceedings of the International AAAI Conference on Web and Social Media*, 3(1):258–261.

C. Silva, U. Lotric, B. Ribeiro, and A. Dobnikar. 2010. Distributed text classification with an ensemble kernel-based learning approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40:287–297.

E. Turcan and K. McKeown. 2019. Dreaddit: A reddit dataset for stress analysis in social media. *Columbia University Department of Computer Science*.

W. Zhang and F. Gao. 2011. An improvement to naive bayes for text classification. *Procedia Engineering*, 15:2160–2164.

Transformers, do we really need them for detecting stress?

Sven Šćekić, Marko Kuzmić, Lovro Bučar

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{sven.scekic, marko.kuzmic, lovro.bucar}@fer.hr

Abstract

In the modern day and age, where negative emotions such as stress are present in everyday life of most people, social media allows them to share those emotions with the world. Using posts from the social media site Reddit, the dataset Dreaddit was created. In this paper we will explore different approaches for stress detection using some simple and some more advanced models. Models that we worked with were logistic regression along with two transformer approaches: DistilBERT and Chat GPT 3.5 Turbo. Our goal was to compare these different approaches to stress detection and find out which model would produce the best results while also not being too expensive and complex.

1. Introduction

Stress is a very common feeling everyone experiences in a smaller or larger amount during their life. It negatively impacts our health and detecting it could enable us to help the person who is experiencing it and improve their existing situation. This paper will try to find the optimal approach of detecting stressful situations from user descriptions. For this task we will be using the Dreaddit dataset (Turcan and McKeown, 2019) which contains user posts from the Reddit website. Throughout this paper we will be referencing the original one where the dataset was introduced.

In the first part of the paper we are going to describe the dataset, followed by the models that we decided to use to tackle the problem. We tried to continue the research of the authors of the dataset and use their best performing model to compare it with some other currently popular solutions. In the end we are going to try to answer the question if the more complex and costly solutions were really justified or should this problem be solved with a more simple approach.

2. Related Work

Stress detection was researched in a couple of other approaches in the last couple of years. One of the approaches was detecting stress using deep neural networks and psychological signals (Li and Liu, 2020). This work however focused more on the medical aspects of stress while our aim was to detect stress using persons description of an event. The approach that we looked at the most was by the authors of the dataset (Turcan and McKeown, 2019). They focused on developing the dataset as well as testing how different models solved the given task.

3. Dataset

The dataset that is used in this paper was introduced in a paper which collected posts from a webside Reddit (Turcan and McKeown, 2019). On that website users can create posts in communities which focus on specific topics called subreddits.

The dataset consists of posts from subreddits where stressful topics are likely to be discussed in between January 1, 2017 and November 19, 2018. These subreddits fall into one of these categories:

- **Social**: The posts from this category are from the subreddit called `r/relationships`. Posts from this subreddit talk about problems in relationships, romantic or non-romantic.
- **Abuse**: posts from this category describe topics related to abuse in relationships and users share their experiences and advices. Subreddits that fall into this category are `r/domesticviolence` and `r/survivorofabuse`.
- **Anxiety**: subreddits from this category are `r/anxiety` and `r/stress`. Users in posts from this category talk about these mental illnesses, their symptoms, share advices and stories regarding these illnesses.
- **PTSD**: Same as the Anxiety category, posts from this category talk about mental illness, but focus on Post-Traumatic Stress Disorder. The only subreddit from this category is `r/ptsd` and same as with posts from the Anxiety category, users share advices and stories and ask questions about the illness.
- **Financial**: posts from subreddits that fall into this category talk about difficult financial situations, share stories about homelessness and other stressful financial topics. Subreddits from this category are `r/almosthomeless`, `r/assistance`, `r/food_pantry` and `r/homeless`.

In the dataset there are 187,444 posts in total from these 10 subreddits. The distribution of these posts can be seen in the Table 1.

Text Analysis and Retrieval 2023: Course Project Reports (TAR 2023), pages 70–74, University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, July 2023.

Table 1: Distribution of collected posts

Topic	Subreddit Name	Total Posts	Avg Tokens/Post	Labeled Segments
Social	r/relationships	107,908	578	694
Abuse	r/domesticviolence	1,529	365	388
	r/survivorsofabuse	1,372	444	315
Anxiety	r/anxiety	58,130	193	650
	r/stress	1,078	107	78
PTSD	r/ptsd	4,910	265	711
Financial	r/almosthomeless	547	261	99
	r/assistance	9,243	209	355
	r/food_pantry	343	187	43
	r/homeless	2,384	143	220

3.1. Data Annotation

A portion of the data was annotated using Amazon Mechanical Turk, a crowdsourcing marketplace that allows individuals to outsource jobs.

Primary job for annotators was to determine if there was stress present in the sentence that they were presented. The definition for stress which was taken from the Oxford English Dictionary states that stress is 'a state of mental or emotional strain or tension resulting from adverse or demanding circumstance'. Each annotator first had to take a qualification test to ensure that they would label the sentences correctly. In the qualification test annotators were given instructions on how to correctly label post segments. After that, each annotator was given 5 segments which they had to annotate with one of the following labels: "Stress", "Not Stress" or "Can't Tell". In addition to the qualification test, each annotator was given one of the 50 'check questions' which were labeled by creators of the dataset. If they did not label these 'check questions' correctly their annotations were not included in the final dataset. Since posts can often be longer, which would prove to be complicated for the annotators to label, they were divided into five-sentence chunks. Added bonus of this technique is that the data can be used to determine the specific location of stress in the post.

In total, 3,553 labeled data points were collected out of which 39% had perfect agreement. With 52.3% of the data being labeled as stressful, the dataset is nearly perfectly balanced. Labeled data was split into two subsets: train and test subset. The train subset contains 2,838 data points and the test subset consists of 715 data points.

4. Models

Motivated by the results of the aforementioned paper, our approach focused on three different models which solve the given classification problem.

Firstly we will introduce the **logistic regression** (Peng et al., 2002) model which the authors of the dataset presented as the best solution, excluding state-of-the-art (SOTA) solutions such as BERT (Devlin et al., 2019). This model is going to represent a simple model approach to the problem without using more complex solutions such as transformers. Then we are going to introduce two transformer approaches, one being **DistilBERT** (Sanh et al., 2020) and the other one, currently very popular, **Chat GPT 3.5 Turbo** (OpenAI, 2023).

We performed our training on the whole dataset and our focus was not to get the best possible results for each model but to compare them and find out which is best suited for the given classification task (taking into account the cost of the model). Because of that reason, we also did not remove lower agreement rate examples from our training dataset to enable our models to handle even the harder examples which are inevitable in the real applications.

4.1. Preprocessing

Before we trained our models we needed to do some form of preprocessing on our data. This step is important because it enables adequate learning from the training set. Given the dataset already contained pretty clean and structured data, we did not need to do a lot of preprocessing to get the desired results. We started with the basic stop word removal, followed by digit and punctuation removal. Then we performed lemmatization which enabled us to properly train the word embeddings for the logistic regression model and tokenizer for the DistilBERT model.

4.2. Logistic regression

Logistic regression (Peng et al., 2002) is a commonly used machine learning model which can be used for binary classification problem. This is a simple model which does not require long and expensive training, so, inspired by the great results that this model showed in the original paper (F-score of 79.8 (Turcan and McKeown, 2019)), we have also decided to include it in our own research.

Logistic regression expects a fixed size input but words and sentences in natural language can be of any size. Today, this problem is solved by using word embeddings which represent words from our dataset as multidimensional vectors. Their advantage over some other approaches that were used in the past, such as one-hot vector representation, is

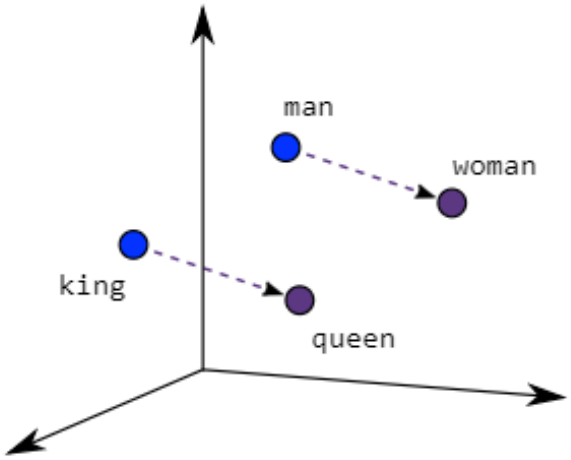


Figure 1: Relations between similar word vectors

that they are able to find relations between words and produce similar vectors for similar words. The relation between similar words can be seen in figure 1. In our approach we used **word2Vec** (Mikolov et al., 2013) embeddings which were trained on the whole training dataset and produced 300-dimensional representations. Learned word embeddings were then used for calculating the final representation for each training example. Here we experimented with 2 approaches:

1. **Averaging of word embeddings:** summing up each representation in the input sequence and dividing it with the number of words in the sequence

2. **TF-IDF averaging:** Term Frequency - Inverse Document Frequency is an algorithm that is used to determine an importance of the given word in the document. Importance is represented as a scalar value. This scalar value for each word is then multiplied with the word's corresponding representation. The final representation is then calculated in the same way as described in approach 1.

4.3. DistilBERT

The second model we used was DistilBERT (Sanh et al., 2020). This model comes from the family of deep learning models called transformers (Vaswani et al., 2017). The main advantage of these types of models is the self-attention mechanism. Self-attention allows the model to find the importance of each word in a sequence and build more effective contextual representations.

DistilBERT is a model similar to the more popular BERT (Devlin et al., 2019), but faster and more lightweight while still preserving almost all the capabilities of the larger model. Because of its features and our limited hardware we decided it would be the best option for representing current SOTA solution in the field of transformers. Model and tokenizer were trained simultaneously on the whole training set using the implementations from the *HuggingFace* library (Wolf et al., 2020).

"If we describe stress as a negative emotion, how would you classify the following sentence:

...example sentence from the test set...

Please answer 1 if the above sentence is describing stress or a stressful situation and 0 if it's not describing it. Just answer with 1 or 0 please."

Figure 2: Text input for Chat GPT 3.5 Turbo API

4.4. Chat GPT 3.5 Turbo

The final model that we used in our research was Chat GPT 3.5 Turbo (OpenAI, 2023). This model was developed by OpenAI and it aims to generate human like responses based on the provided input text. We could not perform direct training on this model, but instead we used the provided API to prompt the model to give us the classification result for each example in the test set. You can see how we formulated our prompt in figure 2.

It must be mentioned that there was some manual data cleanup needed after we got the responses from the API, such as explanations behind the given results that were contained in the labels.

5. Results

The results of our four different models are present in table 2. We can see that our logistic regression (LR) model with basic word embedding averaging performed the worst, what was expected given its simplicity. This simple model, however, performs much better when we introduce TF-IDF averaging.

5.1. LR(TF-IDF averaging) vs DistilBERT

LR model with TF-IDF averaging performed very similarly to the DistilBERT approach. Transformers are currently SOTA approach in NLP and we expect them to perform much better than the simpler models. However, classifying stress, using this pretty clean dataset, is not a particularly hard task and our logistic regression model deals very well with the problem, which is also what authors of the original dataset pointed out.

Here we compared only the results of the two models not taking into account the cost of the more complex model. Training of the DistilBERT model took longer and required more hardware just to get similar results. This leads us to the conclusion that sometimes the solution to the problem could be a simple model without the need for more complex and expensive solutions.

5.2. DistilBERT vs Chat GPT

In our two transformer approaches Chat GPT showed much better results. What shows the true power of this transformer model is that, even though it was not strictly trained on the training data, it managed to outperform DistilBERT

Table 2: Different model results

Model	P	R	F1
LR + averaged word2Vec embeddings	0.6441	0.7995	0.7134
LR + TF-IDF averaged word2Vec embeddings	0.6859	0.8049	0.7406
DistilBERT	**0.7105**	0.7913	0.7487
Chat GPT 3.5 Turbo	0.6905	**0.9431**	**0.7972**

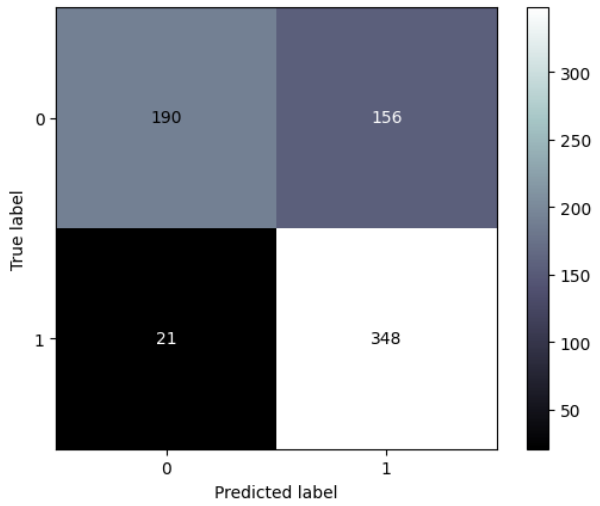


Figure 3: Confusion matrix for Chat GPT 3.5 Turbo. 0 represents data labeled not stressed while 1 represents data labeled stressed.

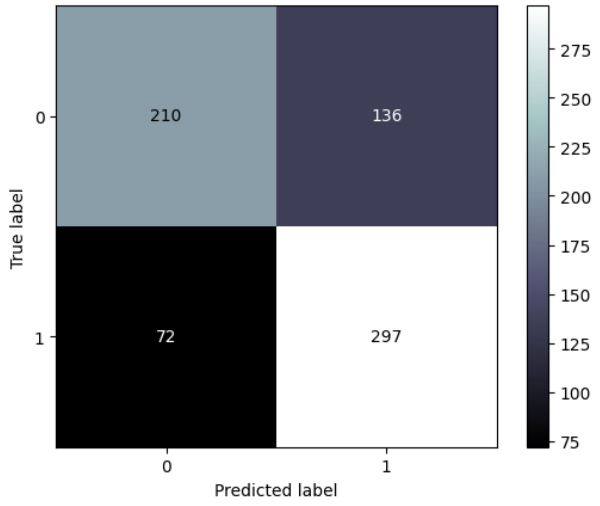


Figure 4: Confusion matrix for LR with TF-IDF averaging.

model. As we can see in figure 3 this model managed to have an extremely low number of false negative (FN) predictions. In certain cases we would want to prefer this behaviour where not detecting stress represents a big problem.

The downside of this approach is the cost behind it. Chat GPT API is a paid service and if we want to use this approach in for stress detection this might not be the best option if our budget is limited.

5.3. LR (TF-IDF) vs Chat GPT

We can see that Chat GPT performed better than our LR model when we compare their F1 scores. The change comes primarily from the higher recall score which Chat GPT manages to achieve (less false negative examples). While we can look at the scores by themselves and conclude that Chat GPT is a better model for the given task, this approach does not take into account the cost of the model.

Our simpler LR model achieves comparable results to the SOTA solution and we claim that it is the optimal approach for solving stress classification problem from the given dataset. Model is easy to learn and could even get better results if it was trained on the dataset which contained examples with higher agreement factor, as it was shown from the authors of the dataset.

If we have more resources and a bigger budget, Chat GPT with its powerful API service could be a better choice, mainly because it rarely does not detect stress which could definitely help in certain situations.

6. Conclusion

In this paper we looked at different approaches for detecting stress from the Dreaddit dataset. We continued our research on findings of the authors and tried to find out if more complex models and solutions gave better results.

We believe that the simple model solves this problem well enough and that more complex and expensive transformer solutions are not worth the extra cost. In some rare cases, where not detecting stress could pose a problem, we can suggest trying the Chat GPT API service if the demand is not high and the budget is not limited. Transformers may be a better option if we just look at their capabilities, but at what cost? Sometimes the simpler solution gets the job done and we should stick to it.

References

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

Russell Li and Zhandong Liu. 2020. Stress detection using deep neural networks. *BMC Medical Informatics and Decision Making*, 20, 12.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

OpenAI. 2023. ChatGPT (Version 3.5). https://www.openai.com/. [Software].

Joanne Peng, Kuk Lee, and Gary Ingersoll. 2002. An introduction to logistic regression analysis and reporting. *Journal of Educational Research - J EDUC RES*, 96:3–14, 09.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Elsbeth Turcan and Kathy McKeown. 2019. Dreaddit: A Reddit dataset for stress analysis in social media. In *Proceedings of the Tenth International Workshop on Health Text Mining and Information Analysis (LOUHI 2019)*, pages 97–107, Hong Kong, November. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Author Index

Adilazuarda, Muhammad Farid, 1
Arkoulis, Nikolaos Nektarios, 1

Babić, Antonio, 65
Banić, Andija, 6
Barbir, Marko, 11
Brečić, Luka, 15
Bučar, Lovro, 70

Chumakov, Oleksii, 1
Ciglenečki, Matej, 55
Ćosić, Tomislav, 15

Damjanić, Marko, 60
Duault, Chloé, 19

Grbelja, Roko, 55
Grebenar, Lea, 46

Hrelja, Mateo, 23

Jehanno, Mathieu, 65
Jurinčić, Dominik, 23
Jurić, Marko, 6
Jurišić, Renato, 28

Krajči, Maria, 33
Krog, Tomislav, 38
Krsnik, Lea, 42
Kuzmić, Marko, 70

Lafontaine, Ellyn, 19
Linardić, Ivan, 23

Marić, Iva, 46
Marinković, Gabriel, 50
Miličević, Mihael, 28
Moslavac, Mirta, 55

Pardon, Josip, 60
Pavić, Tihomir, 38
Pavlović, Dario, 6
Perić, Mislav, 42
Petrović, Nikola, 46
Puharić, Marin, 11

Samardžija, Jakov, 65
Šćekić, Sven, 70
Sever, Matija, 38
Srzić, Josip, 28
Staničić, Kim, 11

Taffonneau, Mallory, 19

Vladić, Ana, 33
Vrljić, Bjanka, 33
Vuk, Valentin, 60

Zrnić, Leon, 42