Groningen Team F at SemEval-2024 Task 8: Detecting Machine-Generated Text using Feature-Based Machine Learning Models

Rina Donker and **Björn Overbeek** and **Dennis van Thulden** and **Oscar Zwagers**
Rijksuniversiteit Groningen
{t.r.donker,b.b.j.overbeek,d.l.van.thulden,o.y.zwagers}
@student.rug.nl

Abstract

Large language models (LLMs) have shown remarkable capability of creating fluent responses to a wide variety of user queries. However, this also comes with concerns regarding the spread of misinformation and potential misuse within educational context. In this paper we describe our contribution to SemEval-2024 Task 8 (Wang et al., 2024), a shared task created around detecting machine-generated text. We aim to create several feature-based models that can detect whether a text is machine-generated or human-written. In the end, we obtained an accuracy of 0.74 on the binary human-written vs. machine-generated text classification task (Subtask A monolingual) and an accuracy of 0.61 on the multi-way machine-generated text-classification task (Subtask B). For future work, more features and models could be implemented.

1 Introduction

Recent large language models (LLMs), such as ChatGPT, have shown remarkable capability of creating fluent responses to a wide variety of user queries. This, in combination with increased accessibility to these models, has lead to an increase of machine-generated content over various channels. However, these LLMs come with concerns regarding the potential misuse of such tasks, like spreading misinformation and misuse within the education system (Wang et al., 2023). Therefore, detecting whether a text is human-written or machine-generated is extremely important.

Unfortunately, humans perform only slightly better than chance at this task, as found by Gehrmann et al. (2019). This introduces the need for developing automatic systems that can identify machine-generated texts, in order to mitigate their potential misuse (Wang et al., 2023).

While previous work has been done on identifying machine-generated texts, they either focused on only one or two particular languages, or focused on detecting machine-generated text for a specific LLM or a specific domain (Wang et al., 2023).

For instance, Macko et al. (2023) note that most of the research on machine-generated text detection uses systems that were trained on English datasets and that prior works show that detectors fine-tuned on English data fail to generalize to other languages. This can be seen in the results from Mitchell et al. (2023), where a decrease is seen from 0.946 AUC ROC to 0.537 when working with German data. In addition, Macko et al. (2023) get similar results.

Meanwhile, Sarvazyan et al. (2023) address the issue of detection systems not generalizing across different generation models and domains. They mention that most previous works often overlook that detection systems would be applied to a broad variety of domains, writing styles, and generation models.

Thus, the goal of Task 8 of SemEval 2024 (Wang et al., 2024) is to take a first step into creating a mono- or multilingual system that is able to detect machine-generated text created by different LLMs in different domains. The Task is divided into three subtasks, of which we participate in two. These subtasks are described in Section 2.

In order to tackle this task, we have created multiple feature-based models. The features on which the models have been trained will be described in Section 3. Our decision for creating these feature-based models is supported by our beliefs that a model with carefully crafted features is computationally less expensive than fine-tuning a LLM, while it may be able to achieve equal or better performance because of its ability to generalize (Wang et al., 2023), which is something that LLMs tend to struggle with (Lasri et al., 2022; Wilson et al., 2023). We focused on creating monolingual models for subtask A and B.

Eventually, we created a model that was ranked 95th out of 137 teams for the monolingual track of subtask A with an accuracy of 0.69. For subtask B, we ranked 50th out of 77 participating teams with an accuracy score of 0.61.

All of our code for this project can be found on our GitHub repository.[1]

2 Background

The goal of this shared task is to create machine-learning models that are able to detect machine-generated text across multiple languages, generators and domains. The datasets we used to train and test were provided by the task organizers. We took part in two subtasks: A and B.

2.1 Subtask A

The goal of subtask A is to detect whether the text is machine-generated or human-written. Every instance in the dataset contains the text generated by a machine or written by a human and its corresponding label, which is either *human* or *machine*. Besides this, it also includes the model that generated the data and the source of the text.

There is one monolingual dataset, which is made up of English texts only, and one multilingual dataset. We chose to only focus on the monolingual track of this subtask, since we can make language specific feature for this dataset. For the monolingual dataset, the source means the domain of the text, e.g. `reddit` or `wikipedia`. The train set for the monolingual track has 119,757 instances, while the development set contains 5,000 instances.

2.2 Subtask B

The datasets for subtask B are similar to those of subtask A, however, as the goal of this subtask is to detect by which specific text generator the text was created, there are more labels: `human`, `chatGPT`, `cohere`, `davinci`, `bloomz` and `dolly`. The train set consists out of 71,027 instances and the development set includes 3,000 instances. All instances are from sources in English.

3 System Overview

We experimented with four different kind of supervised models and the features that we created ourselves. We will describe each model and feature one by one.

[1] `https://github.com/bbjoverbeek/SamEval-2024_Task-8_M4`

3.1 Models

Support Vector Machine We chose to implement a Support Vector Machine (SVM; Cortes and Vapnik, 1995), because this has been one of the most fundamental models in statistical machine learning. It has become less popular with the uprising of neural networks, but we expect it will perform well on this task.

Naive Bayes Naive Bayes is a statistical machine learning model that is easy to implement and works well on large datasets. It is based on Bayes' Theorem which calculates the probability of something happening based on previous encounters.

K-Nearest Neighbors Lastly, we implemented the K-Nearest Neighbors algorithm (KNN; Fix and Hodges, 1989). This algorithm is also specialized in classification and is simple to implement.

Feed Forward Neural Network In an attempt to find more complex relationship between different features, we also implement simple neural networks whith several different architectures.

3.2 Features

The following section describes the set of features that we created and experimented with.

Personal pronouns vs. proper pames The first feature focuses on the difference in the usage of personal pronouns, like "he" or "she", versus the usage of proper names, like "Michael" or "Karen" within the text. According to Mitrovic et al. (2023), humans will usually switch to pronouns to refer to a person after using a proper name once or twice in a paragraph, while ChatGPT tends to refer to a person by their proper name more often. For this reason, we thought it would be interesting to see if this effect would be prevalent for other LLMs and if this would have an impact on the results.

Sentence tense This feature focuses on the tense that a sentence is written in. The three tenses we consider are *past*, *present* and *future*. Our goal with this feature was to discover if there are patterns in tense usage that are more commonly used in machine-generated text when compared to human-written text and vice versa.

Sentence voice For all of the sentences, we collect the voice that the sentence was written in. This could be either *passive* or *active*. Similarly to the sentence tense, we hoped that we could discover patterns that are distinct to either machine-generated text or human-written text, since Mitrovic et al. (2023) mentioned that ChatGPT writes in the passive voice more often than active.

Sentence similarity The sentence similarity calculates how similar a sentence is to its previous and following sentence. In similar fashion to the sentence tense and sentence voice, we want to discover if there are any distinct patterns.

Sentiment We collect the sentiment on sentence level from the text to use as a feature. We believed that there might be a difference between human-written and machine-generated texts in terms of sentiment.

Domain The domain of the text in the train and development data was provided by the task organizers. However, we suspected that this might not be included in the final test set. Therefore, we also included domain as a feature that we could experiment with. We figured that if the domain had a positive influence on the final score, we could build our own classifier that predicts the domain of a text, which can then be used as a feature for our system.

POS-tags and dependency tags Finally, we also included the POS-tags and dependency tags as features. These features contain information about the structure of the text, which we believed could be helpful in distinguishing between machine-generated and human-written text.

4 Experimental Setup

In this following section, we will describe how we created the models and crafted the features.

4.1 Models

Support Vector Machine In order to run the SVM, we made use of the scikit-learn library (version 1.3.2) (Pedregosa et al., 2011).[2] In particular, we used `LinearSVC`. By using the built-in functions of scikit-learn we could train and test the model.

Naive Bayes To build the Naive Bayes classifier, we used the `GaussianNB` classifier from scikit-learn. This type of Naive Bayes classifier assumes that our data is normally distributed.

[2] https://scikit-learn.org/

K-Nearest Neighbors For the KNN algorithm, we used `KNeighborsClassifier` from scikit-learn. When this model is used for subtask A, the number of neighbors is 5. In other cases, the number of neighbors was 15. It is trained and tested in a similar way to the SVM.

Feed Forward Neural Network With the Keras library we created a simple feedforward neural network (FFNN).[3] We experimented with different setups for the neural networks, ranging from 1 up to 4 hidden layers and giving the hidden layers from 8 up to 1024 nodes. The ones that worked best had two or three hidden layers. The size of the layers ranged from 16 up to 256 (depending on the model). All the models use softmax as their activation and Adam for optimization.

4.2 Features

4.2.1 Token-level

Pre-processing With the use of spaCy (version 3.7, using their trained pipeline called *en_core_web_sm*), we could split the full text into tokens.[4] After that, we used the tokens to create our token-level features which will be described below in the following two paragraphs.

Personal pronouns vs. proper names For this feature, we first count how many personal pronouns and proper names the text contains. In order to find the amount of personal pronouns in a text, we use spaCy to find every token that has the POS-tag *pron* (pronoun) and count the number of occurrences.

To collect the number of proper names, we use spaCy's entity recognizer and count every token that has the label *person*.

POS-tags and dependency tags The POS-tags and dependency tags can be easily extracted with spaCy. We use their built-in function to retrieve these tags and then use them as features. For both of these features, we created a bag-of-trigrams.

4.2.2 Sentence-level

Pre-Processing For the sentence-level features, we split the full text into sentences with spaCy. We then use these sentences to extract the features we describe in the remainder of this subsection.

Sentence tense In order to extract the sentence tense, we used spaCy's token-based matching. We

[3] https://keras.io/
[4] https://spacy.io/

created multiple patterns for each sentence tense by using GitHub Copilot[5]. The patterns are made out of combinations of detailed POS-tags, dependency tags, and in some cases, words. The sentences are matched with these patterns and as a result they either get the label *past*, *present* or *future*.

After we collected all the sentence tense labels, we have created trigrams out of these labels and used these in a bag-of-words. We do this by using the `CountVectorizer` that can be found in the skicit-learn library (Pedregosa et al., 2011). The bag-of-trigrams is the feature we use that represents the sentence tense.

Sentence voice Collecting the sentence voice is done in a similar way as the sentence tense. We again use spaCy's token-based matching to determine if a sentence is written in *active* or *passive* voice. The patterns we used were adapted from an example implementation found on Stack Overflow[6]. We then create a bag-of-trigrams in the same way as for the sentence tense and use this as a feature.

Sentence similarity For the sentence similarity feature, the first thing that is done is that each sentence is compared to its previous and following sentence using a sentence-transformers model[7] from Hugging Face (Wolf et al., 2020). We then get two similarity scores per sentence, after which we check for each sentence whether it is most similar to the previous or following one and then give it the value *previous* or *next*, depending on which combination has the highest score. After that, the process is the same as for the sentence tense and voice: we create a bag-of-trigrams to use as a feature.

Sentiment In order to determine the sentiment of a sentence, we used a RoBERTa model from Hugging Face (Liu et al., 2019).[8] Each sentence was assigned one of the following labels: *positive*, *neutral* or *negative*. Afterwards, we again created a bag-of-trigrams so that we could actually use it as a feature.

4.2.3 Document-level

Domain Since domain was already given with the train and test data, we initially used this to experiment with this feature. We trained the models with and without domain to get an insight in the influence of this feature on the final scores. In some cases, the inclusion of domain as a feature improved the score, however, it was only very little. Therefore, we did not find it fruitful to build our own classifier that could predict the domain, especially considering that this classifier may not have been 100% accurate, which would increase the risk of wrong predictions negatively influencing the results.

Evaluation measures In order to evaluate the performance of the model, we calculate precision and recall, the f1-score, and the accuracy. Since the official metric used for subtask A and B is the accuracy, we considered the models with the highest accuracy our best performing models.

We train our different models with every possible combination of features on the training data and evaluate their performance on the development data given by the task organizers. The models with the highest accuracy are the ones we submitted to the shared task.

5 Results

5.1 Subtask A

For subtask A, we handed in three SVMs and three neural networks, since these models received the highest scores on the development set. We achieved the highest accuracy (0.74) on the final test set with a neural network using the sentence tense, sentence voice, sentence similarity and ratio of pronouns and named entities as features. The final ranking released by the task organizers however was not based on the submitted model with the highest score but on the last submitted model, which in our case was the SVM model using the sentence tense, sentence voice and ratio pronouns/named entities as features. This model ranked 95th out of 137 teams. The results of the models we handed in, including our best performing model, can be seen in Table 1. Even though our best performing model obtained an accuracy of 0.74, it is still lower than the RoBERTa baseline, which achieved an accuracy of 0.88.

5.2 Subtask B

For subtask B, we handed in three models, namely one SVM and two neural networks. Our best scoring model was the SVM with an accuracy of 0.61. This model used the sentence tense, sentence voice,

[5] `https://github.com/features/copilot`
[6] `https://stackoverflow.com/a/74594808`
[7] `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`
[8] `https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest`

Model	Features	Precision	Recall	F1-score	Accuracy
RoBERTa (baseline)	-	-	-	-	*0.88*
SVM	Tense, Voice	0.73	0.64	0.68	0.69
SVM	Tense, Voice, Ratio PRON/NE	0.73	0.66	0.69	0.69
SVM	Tense, Voice, Similarity, Ratio PRON/NE	0.77	0.64	0.70	0.71
NN 12e-b64-l0.0001	Tense, Voice, Ratio PRON/NE	0.72	0.64	0.68	0.68
NN 8e-b32-l0.0001	Tense, Voice, Similarity, Ratio PRON/NE	0.79	**0.68**	**0.73**	**0.74**
NN 10e-b64-l0.0001	Tense, Voice, Similarity	**0.83**	0.57	0.68	0.72

Table 1: The results of our best models on the monolingual test data of subtask A. The numbers behind the neural networks (NN) stand for the number of epochs (e), the batch size (b) and the learning rate (l) respectively. All the other hyperparameters were the same. The highest scoring model is the neural network with four different features. It is only outscored by another neural network model on precision.

Model	Features	Precision	Recall	F1-score	Accuracy
RoBERTa (baseline)	-	-	-	-	*0.75*
SVM	Tense, Voice, Sentiment, POS DEP, Similarity, Ratio PRON/NE	**0.60**	**0.61**	**0.58**	**0.61**
NN 48e-b32-l0.0005	Tense, Voice, POS, DEP, Ratio PRON/NE	0.54	0.56	0.50	0.56
NN 48e-b32-l0.0005	Tense, Voice, POS, DEP, Similarity, Ratio PRON/NE	0.56	0.56	0.51	0.56

Table 2: The results of our best models on the test data of subtask B provided by the creators of the Shared Task. The numbers behind the neural networks (NN) stand for the number of epochs (e), the batch size (b) and the learning rate (l) respectively. The highest scoring model is the SVM with seven different features.

sentiment, POS-tags, dependency tags, sentence similarity and the pronoun/named entity ratio as features. This model scored the 50th place out of 77 models in total. The results of our best models can be seen in Table 2. Again, our best performing model scores lower than the RoBERTa baseline that has an accuracy of 0.75.

5.3 Discussion

We found that the SVMs and the feedforward neural networks gave the best results on the development set, while the KNN and Naive Bayes algorithms did not perform well. This can be seen in Table 3 in Appendix A, which shows the best results of the different models on subtask A monolingual. Because of these results we decided to focus our attention on feedforward neural networks and SVMs, for both subtask A as well as subtask B.

Our submissions for both subtasks ended up being in the bottom 50% of the total submissions. However, we used older techniques for this task, as we believed that carefully crafting our own features and training these on simpler models would still return good results while being less computationally expensive.

For both subtask A and B we can conclude that a simple FFNN or SVM performs well, but it does not outperform the current state-of-the-art models.

Overall, some features contribute more to the detection of machine-generated text than others. The features that perform well are the sentence tense, sentence voice, sentence similarity and the ratio of personal pronouns and proper names. The tense and the voice of the sentences even appear in all of our best scoring models.

We think that the reason of the effect of ratio of personal pronouns and proper names on the performance is due to the fact that machines tend to use named entities more often than humans, as we described in Section 3.2.

The feature that did not seem effective was sentiment, which only occurs in one of our top models. One of the reasons we think that the sentiment feature did not seem helpful is due to the fact that most of the texts come from sources that are naturally written in neutral tone, such as Wikipedia and arXiv.

6 Conclusion

To conclude, while we still gained quite good results, our models do not outperform the state-of-the-art models. We achieved an accuracy of 0.74 on subtask A and an accuracy of 0.61 on subtask B. Both scores are lower than the RoBERTa baseline.

We discovered that sentence tense, sentence voice and the ratio between pronouns and named entities seemed to be effective for the classification task, while sentiment did not have that much influence.

In future research, there could be more experimentation with different kinds of machine learning models and more features could be created to further improve the models.

Acknowledgements

This submission has been carried out as a part of the 2023-2024 edition of the master course Shared Task Information Science (LIX026M05) at the University of Groningen. We want to thank Antonio Toral and Lukas Edman for teaching the course. As Antonio Toral was the teacher assigned to our group, we want to thank him especially for supervising our progress, steering us in the right direction and answering all of our questions.

References

C. Cortes and V. Vapnik. 1995. Support vector networks. *Machine Learning*, 20:273–297.

Evelyn Fix and J. L. Hodges. 1989. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3):238–247.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander Rush. 2019. GLTR: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116, Florence, Italy. Association for Computational Linguistics.

Karim Lasri, Alessandro Lenci, and Thierry Poibeau. 2022. Does BERT really agree ? fine-grained analysis of lexical dependence on a syntactic task. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2309–2315, Dublin, Ireland. Association for Computational Linguistics.

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.

Dominik Macko, Robert Moro, Adaku Uchendu, Jason Lucas, Michiharu Yamashita, Matúš Pikuliak, Ivan Srba, Thai Le, Dongwon Lee, Jakub Simko, and Maria Bielikova. 2023. MULTITuDE: Large-scale multilingual machine-generated text detection benchmark. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9960–9987, Singapore. Association for Computational Linguistics.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature.

Sandra Mitrovic, Davide Andreoletti, and Omran Ayoub. 2023. Chatgpt or human? detect and explain. explaining decisions of machine learning model for detecting short chatgpt-generated text.

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.

Areg Mikael Sarvazyan, José Ángel González, Marc Franco-Salvador, Francisco Rangel, Berta Chulvi, and Paolo Rosso. 2023. Overview of autextification at iberlef 2023: Detection and attribution of machine-generated text in multiple domains.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, jinyan su, Artem Shelmanov, Akim Tsvigun, Osama Mohammed Afzal, Tarek Mahmoud, Giovanni Puccetti, Thomas Arnold, Chenxi Whitehouse, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2024. Semeval-2024 task 8: Multidomain, multimodel and multilingual machine-generated text detection. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 2041–2063, Mexico City, Mexico. Association for Computational Linguistics.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Alham Fikri Aji, and Preslav Nakov. 2023. M4: Multi-generator, multi-domain, and multilingual black-box machine-generated text detection. *arXiv:2305.14902*.

Michael Wilson, Jackson Petty, and Robert Frank. 2023. How Abstract Is Linguistic Generalization in Large Language Models? Experiments with Argument Structure. *Transactions of the Association for Computational Linguistics*, 11:1377–1395.

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.

A Results on Development Set of Subtask A Monolingual

Model	**Features**	**Precision**	**Recall**	**F1-score**	**Accuracy**
RoBERTa (baseline)	-	-	-	-	*0.74*
KNN	Tense, Voice, Named Entities	0.637	0.639	0.638	0.637
NB	Tense, Named Entities, Sentiment	0.575	**0.981**	0.725	0.627
SVM	Tense, Voice, Named Entities	0.644	0.888	0.746	0.698
NN 4e-b12-l0.0001	Tense, Voice, Named Entities	**0.687**	0.835	**0.754**	**0.727**

Table 3: This table shows the best performance of each model on the development set of subtask A during our initial experiments. For the NN, we experimented with the number of epochs, batch size and learning rate. We also focused on optimizing the pronouns/named entity feature by using the ratio of pronouns and named entities instead of using absolute values.