Can Part-of-Speech Embeddings Help Us Identify Generative AI Text?

Ioana-Daria Vasile iova@itu.dk
Niclas Claßen niclc@itu.dk
Frederik Bachmann Faarup frfa@itu.dk
Steinar Slette stsl@itu.dk

Abstract

This study investigates the relationship between POS tagging and the detection of human versus AI-generated text. Using an LSTM architecture, we aim to identify differences in the structuring of text generated by humans and AI. We hypothesize that the structuring of POS tags differs between human and AI-generated text and can be used to improve the accuracy of generated text detection. Our results show a small increase in performance with the POS model, achieving 82% accuracy on the test set. However, the improvement is not significant enough to draw conclusive results. Additionally, potential biases in the data and methodology may have influenced the findings, highlighting the need for more investigation in this area. The code for this project is available on GitHub: https://github.com/niclasclassen/danish-ai-text-discriminator

1 Introduction

With increasingly complex generative AI and ease of access to these models in everyday life, it has become extremely hard to determine whether the texts we are reading are in fact written by humans or by generative AI. This challenge has serious implications across various domains, including phishing (Baki et al., 2017), academic dishonesty (Carnino et al., 2024), or spread of misinformation (Zellers et al., 2019).

To tackle this problem, there have been many proposals that have attempted to discern between human and AI generated texts. However, this remains a challenging task. For example, in January 2023 openAI released an AI classifier for indicating AI-written text which discontinued in July 2023, due to low accuracy (OpenAI, 2023). Furthermore, many of these implementations focus on English-language texts, leaving other languages underrepresented.

In this project, we propose an AI-generated text discriminator specifically designed to handle texts in Danish. It is important to note that our focus is not the development of a novel state-of-the-art model. Instead, we investigate whether incorporating part of speech (POS) tagging as an extra feature in our discriminator can enhance its prediction performance. Therefore, our research question is defined as follows:

- **RQ:** Can part of speech tagging be used as an extra feature to enhance the performance in distinguishing between text created by humans or generative AI?

2 Related Work

Detecting AI generated text has become increasingly important, as advanced language models like GPT and LLaMA continue to evolve. Existing research has often struggled with distinguishing human written from AI generated text, especially when dealing with unseen models or out-of-domain content. MAGE (Li et al., 2024) addresses this challenge by building a comprehensive testbed, and it achieved 86.54% accuracy on out-of-distribution texts, thus proving that improved detection methods are feasible. However, current detection methods still have significant limitations. Some of the standard techniques such as watermarking, neural network-based classifiers, and zero-shot models can be bypassed using attacks like recursive paraphrasing, which alters text structure while preserving the meaning (Sadasivan et al., 2024). This is a vulnerability that shows the need for more robust detection features that go beyond surface-level text patterns. To strengthen detection, researchers have also explored various linguistic features. In educational contexts, models using n-gram bag-of-words representations have proven to be very effective. For example, an SVM-based language model successfully distinguished human-written essays with

100% accuracy by focusing on text structure and linguistic patterns (Cingillioglu, 2023). This suggests that integrating deeper linguistic features can improve detection models.

To the best of our knowledge, no existing work properly explains why a model identifies a specific text as AI-generated. In this project, we hypothesize that AI-generated text tends to be more structured and follows a stricter POS order compared to human-written text. Therefore, we propose enhancing our discriminator by adding POS tagging as an additional feature.

3 Data

For this task, we utilize two datasets. The first dataset is the Danish subreddit corpus from the Convokit dataset (University, 2024), and the second is a dataset generated by us with the use of a LLM prompted to rewrite the existing Reddit data contained in the first dataset. We specifically selected the Reddit dataset as our source for human text as we believe it to be genuinely written by humans to a high degree. Other datasets may contain more text generated by AI, posing a risk for data contamination.

For the human-generated Reddit data, we apply several filtering criteria in order to ensure good data quality and reduce noise. We only extract posts that have a minimum of 10 upvotes and a word count between 50 and 1000, in order to filter out potential spam or low quality content. From the resulting data after these procedures, we include both the title and the content of the Reddit posts.

4 Methodology

This section describes our approach, including how we generated and prepared the data, as well as the design of our model.

4.1 Data Generation & Preprocessing

We first preprocessed the extracted posts to ensure the text is consistent and to reduce noise. This involved unescaping HTML entities to restore original characters, removing extra whitespaces, and stripping common markdown formatting such as bold and underline. This represents the human text in our analysis.

To generate the AI text, we used a LLaMA model from Hugging Face (Meta-Llama-3-8B-Instruct) (AI, 2024). We opted for two different approaches in the AI data generation: first was to prompt the model to rephrase the given post, and the second involved generation of text based on the post title. The prompts used for both approaches can be found in the Appendix under C, and D.

After the AI text generation, post-processing of the text was performed to reduce the potential of shortcuts the model may exploit. For example, some of the rewritten text contained several llama-specific references, such as "here is the rewritten text in Danish". All such instances were removed to prevent the discriminators from being influenced by irrelevant or model specific text. Furthermore, we manually eliminated rows where the model could not provide an output due to the content being harmful or inappropriate.

With the first approach of AI generation, the final data consisted of 2,082 rows, where each row contains both the human post and its AI pair. Similarly, with the title based approach we obtained 2,076 rows. This difference suggests that the model identified more inappropriate posts when processing the content using titles, and had to be deleted.

The data was further preprocessed using a simple tokenizer that lowercases tokens and splits them on space. Samples were then cut off to have a maximum length of 2,000 tokens, and if too short, a padding token was added repeatedly.

4.2 POS Tags as Indicators

As stated in the introduction, our study is based around determining whether POS-tags can be beneficial in determining whether a text is generated or human written. For this purpose we utilize the library *spaCy* (Honnibal et al., 2020), where we use the model *da_core_news_sm*, which is specifically trained to handle POS tagging in danish. The POS-tagger has a total of 18 different POS-tags possible, including an unknown tag. These tags are applied to each post in our data where possible, and are subsequently used in model training as a means to investigate the structuring of POS-tags in a sequence of text and its effect on predictions. We further added a 19th POS-tag to account for padding.

4.3 Model Architecture

To build the AI text discriminator, we create two models. First, we train a baseline Long Short-Term Memory (LSTM) and next, we enhance the model by incorporating POS tags as additional features. An LSTM layer consists of an LSTM unit (see Figure 1) that for each time step t computes an

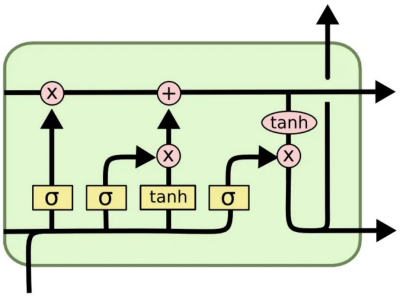

Figure 1: Visual representation of one LSTM unit at time step t.

output o_t, a context c_t and a hidden state h_t vector using an input gate, forget gate and output gate. Mathematically, the LSTM unit can be described by equation 1-6:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (1)$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (2)$$

$$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \quad (3)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (4)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t \odot \tanh(c_t) \quad (6)$$

Here, $\odot$ is element-wise multiplication. The weight matrices W_f, W_i, W_c, W_o and bias vectors b_f, b_i, b_c, b_o are learnable parameters during training. In short, the unit takes in the input at each time step and uses the gates to compute how much should be kept and how much should be forgotten from the input and from the hidden state at the previous time step, i.e. h_{i-1}, to produce the hidden state h_t at time step t.

Two models were created using LSTM. First, a baseline, which consists of an embedding layer, a single, bidirectional LSTM layer that takes as input the word embeddings to produce a hidden state, followed by a linear layer to produce the final output. To mitigate the problem of overfitting, a dropout layer was added between the LSTM and the linear layer. Finally, the outputs were fed through the sigmoid function to push the outputs between 0 and 1, i.e. get the probably that the given sample input is AI-generated.

Second, an identical architecture was made but with an additional embedding and LSTM layer, which concurrently with the word embedding LSTM, takes in POS embeddings of the input. The hidden states of the two LSTMs were concatenated before being fed through the linear layer.

The word embeddings had a lookup table of 221,242 words with vector lengths of 400, which were pretrained on Danish Twitter data using Word2Vec[1]. The POS embeddings contained 19 POS-tags with sparse, binary vectors of length 19 due to the nature of one-hot encoding. Both of the embedding layers were finetuned during training.

4.4 Experimental Design

For the experiment, we chose to combine the two different datasets we generated into one. This gave us one dataframe containing the human text, the AI paraphrased text, as well as the title based AI text. This resulted in a class imbalance (twice as many AI posts), however, the imbalance was mitigated during the training process. The data was split into 80% train and 20% test, stratified by the output label. The train set was further split into 20% validation, again stratified. This allowed for hyperparameter tuning on the train-validation sets.

Table 1 shows the hyperparameters that were tuned. Both of the models were tuned on the same hyperparameters. Some parameters were frozen to limit the search space. For example, the model with two LSTM layers always had the same hidden size in both layers. Further, a batch size of 32, a dropout of 0.3, 1 LSTM layer in each model (beside the bidirectional part) and default parameters of `PyTorch` ADAM optimizer were chosen. For both models, a hidden size of 64, learning rate of 0.0001 and roughly 50 epochs seemed to be optimal.

Hyperparameter	Search Values
Hidden size	{16, 32, 64}
Learning rate	{0.0001, 0.0005, 0.001}

Table 1: Hyperparameter search space used for both models.

For training, weighted binary cross entropy loss (BCE) was used as loss function (equation 7). Briefly, BCE computes how much the output probability diverges from the actual label. True labels were weighted by 0.5 because data consisted of $2/3$ true labels and $1/3$ negative labels, i.e. double the amount of true labels.

[1] https://robvanderg.github.io/modeling/twit-embeds/

$$l_n = -w_n[y_n \cdot log\sigma(x_n) + (1-y_n) \cdot log(1-\sigma(x_n))] \quad (7)$$

Finally, both of the models were trained on the whole train set with the approximately best hyper-parameters. Then, they were evaluated on the test set.

4.5 Human Annotation

To evaluate the model's performance further, a human annotation was conducted. This task involved blindly distinguishing between human-written and AI generated texts from a random sample of the dataset. In total we have 600 annotated Reddit posts after this procedure, which can be compared with the performance of our neural network based models.

5 Results

In this section, we present the evaluation results obtained from both models, followed by an analysis of human annotation performance on a subset of the data.

To compare the two discriminator models, we evaluate their performance using accuracy and F1 score (Table 2). The overall performance achieved by the Baseline LSTM is an accuracy of 81.29% and an F1 score of 85%, while the LSTM+POS model achieved an accuracy of 82.01% and an F1 score of 86.34%.

Model	F1 Score (%)	Acc. (%)
Human Annotation	55.74	54.76
Baseline LSTM	85.00	81.29
LSTM+POS	**86.34**	**82.01**

Table 2: Models - F1 Scores and Accuracy

In the human annotation evaluation, we observed that annotators were significantly more likely to mistake AI generated text for human-written content than the other way around. The human annotation resulted in a F1 score of 55.74% and an accuracy of 54.76%. Figure 2 illustrates the classification performance, while figure 3-4 shows the same for the models.

6 Analysis

For our analysis, we mainly examine performance by class in order to validate the results. The first method investigates LSTM models performance

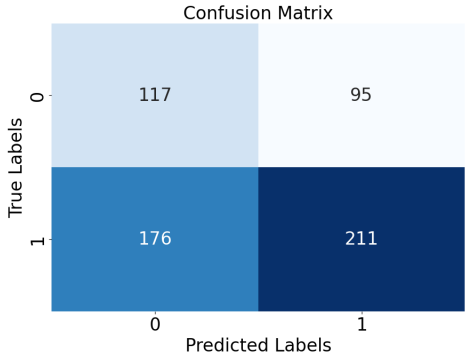

Figure 2: Confusion Matrix for Human Annotation Results

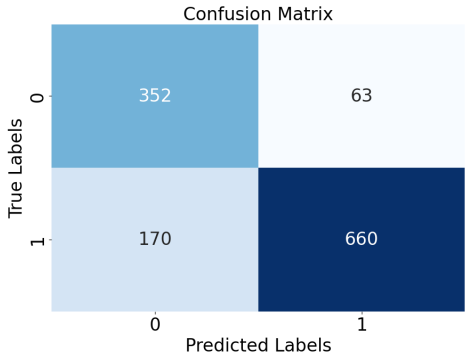

Figure 3: Confusion Matrix for Baseline LSTM model Results

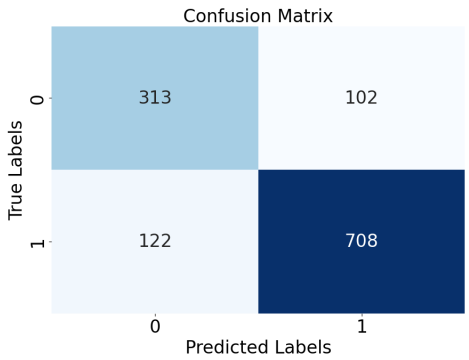

Figure 4: Confusion Matrix for LSTM + POS model Results

based on the source of the data (AI or human), while the second investigates the human annotation.

6.1 LSTM Results

To get a better understanding of how the model performs, we calculated its performance within each class for both models. Although we theoretically have only two classes (AI or human), we further divided the AI category into rephrased and generated. The results show that both models are best at predicting AI generated text. When it comes to the other classes, the POS model is slightly better at AI-generated text detection and the Baseline is better at human text detection. Results are shown in Table 3.

Individually, our models show a high accuracy for the groups containing human written and fully AI generated text, while achieving a slightly lower accuracy for the text that is rephrased. As we expect the rephrased text to have a less clear divide between the underlying patterns of AI/human generated text and to be harder for the model to classify correctly. Moreover, while the POS model gets a substantial increase in the accuracy for AI rephrased text, it achieves a lower accuracy than the baseline in predicting human written texts. This difference might come from the fact that AI generated text is more ordered/structured with POS tags, and these patterns are easier to learn by the model. On the other hand, human text tends to be more variable and less predictable.

Baseline LSTM	
Data Source	**Accuracy/Recall (%)**
Human	84.82
Rephrased (AI)	63.13
Generated (AI)	**95.90**
LSTM+POS	
Data Source	**Accuracy/Recall (%)**
Human	75.42
Rephrased (AI)	72.53
Generated (AI)	**98.07**

Table 3: Accuracy for different classes in the dataset for both models.

6.2 Human Annotation

The accuracy values (Table 4) show how well humans could tell different types of text apart. They were best at spotting AI generated text, but found rephrased text the hardest to identify. This outcome aligns with our expectations, as rephrased texts likely blur the distinction between human and AI generated content.

Data Source	**Accuracy/Recall (%)**
Human	55.19
Rephrased (AI)	45.36
Generated (AI)	**63.73**

Table 4: Human Annotation Evaluation - Accuracy

7 Discussion

One possible reason why POS tagging did not seem to improve performance by a lot could be the way we encoded POS tags as one-hot vectors. This representation might have limited the model's ability to capture the sequential and contextual relationships between tags. Future work could explore more advanced encoding methods, such as embeddings or contextualized representations.

For the AI-generated text, we utilized Meta-Llama-3-8B-Instruct, a relatively lightweight model compared to the large language models (LLMs) commonly used for text generation tasks, such as GPT or Claude. Consequently, our results may not fully represent the outcomes achievable with more advanced LLMs. As a result, the quality of the AI-generated text might be worse, which is something that could have effected the performance of our models. However, we tried to find a good trade off between available computational resources and performance by utilizing ITU's HPC cluster. Furthermore, we ensured a sufficient quality of the AI generated text by manually annotating a subset.

We hypothesize that the performance of our LSTM models is influenced by the specific LLM used for text generation. Different generative models produce outputs with varying structural characteristics, which is what our method leveraging POS-tagging aims to detect. This likely limits the generalizability of our approach across different LLMs.

We also assume that the size and diversity of the datasets significantly impact model performance. Capturing the subtle nuances of human language, and distinguishing these from AI-generated text, likely requires larger and more diverse datasets. Improving the generalizability of our model to detect AI-generated text across various sources remains a

key area for future work. This limitation is partly due to the small size and limited diversity of our training data, which consists solely of Reddit posts.

During our review following training, we discovered a few instances of AI-generated data leaks. For example, in some cases, the model appended an English phrase like "here's your Danish Reddit post" at the end of the text. Furthermore, for the text generated from the title, the resulting text contained parts in english. We tried to limit this by defining very specific prompts and an automated and manual post-processing to remove not valid data.

For longer posts, in the rephrased text there was a cut off at the end, due to the token limitation that was set. This may introduce bias in the form that it gives our model very clear indicators of AI written text, which could in turn hurt our models ability to generalize across the data.

8 Conclusion

In conclusion, while our results show a slight improvement in performance with the addition of POS tagging, it is not significant enough to confidently say that POS tags substantially aid in AI-generated text detection. This indicates that the role of grammatical structure in improving detection needs further exploration with larger datasets and more advanced models. Nonetheless, this study provides a starting point for investigating the potential of incorporating grammatical features in AI text detection systems.

References

Meta AI. 2024. Meta-llama-3-8b-instruct. https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct. Accessed: 2024-12-17.

Shahryar Baki, Rakesh Verma, Arjun Mukherjee, and Omprakash Gnawali. 2017. Scaling and effectiveness of email masquerade attacks: Exploiting natural language generation. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, page 469–482, New York, NY, USA. Association for Computing Machinery.

J.M. Carnino, N.Y.K. Chong, H. Bayly, et al. 2024. Ai-generated text in otolaryngology publications: a comparative analysis before and after the release of chatgpt. *European Archives of Oto-Rhino-Laryngology*, 281:6141–6146.

I. Cingillioglu. 2023. Detecting ai-generated essays: the chatgpt challenge. *International Journal of Information and Learning Technology*, 40(3):259–268.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spacy: Industrial-strength natural language processing in python.

Yafu Li, Qintong Li, Leyang Cui, Wei Bi, Zhilin Wang, Longyue Wang, Linyi Yang, Shuming Shi, and Yue Zhang. 2024. Mage: Machine-generated text detection in the wild. *Preprint*, arXiv:2305.13242.

OpenAI. 2023. New ai classifier for indicating ai-written text. https://openai.com/index/new-ai-classifier-for-indicating-ai-written-text/. Accessed: 2024-12-19.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2024. Can ai-generated text be reliably detected? *Preprint*, arXiv:2303.11156.

Cornell University. 2024. Convokit subreddit corpus documentation. https://convokit.cornell.edu/documentation/subreddit.html. Accessed: 2024-12-17.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *Advances in Neural Information Processing Systems*, 32:9054–9065.

A Statement of Contribution

All of us contributed the same amount of work to all parts (code and report) of this project.

B Usage of chatbots

- **Assistance purely with the language of the paper:** -
- **Short-form input assistance:** We used GitHub Copilot for auto-completion for some parts of our code.
- **Literature search:** -
- **Low-novelty text:** -
- **New ideas:** -
- **New ideas + new text:** -

C Prompt to rewrite Reddit posts

```
messages = [
    {
        "role": "system",
        "content": (
            "You are a Danish language assistant specialized in rewriting text. "
            "Your task is to rephrase the input text for clarity, maintaining the same meaning, tone, and context. "
            "The rewritten text should be roughly the same length as the input. "
        "You **must preserve all formatting** such as bold, italic, or other styles, as well as links, hashtags, and any special characters.\n\n"
            "Your output **must strictly follow this format**:\n\n"
            "<<Title>>\n[Rewritten title here]\n\n"
            "<<Text>>\n[Rewritten text here]\n\n"
            "Do not deviate from this format under any circumstances."
        ),
    },
    {
        "role": "user",
        "content": (
            "Please rephrase the following **title and text** in Danish for a Reddit post. "
            "Ensure the meaning, links, formatting, and approximate length are preserved. "
            "Return the output **strictly** in the following format:\n\n"
            "<<Title>>\n[Rewritten title here]\n\n"
            "<<Text>>\n[Rewritten text here]\n\n"
            "Here is the input:\n\n"
            "<<Title>> " + title + "\n\n"
            "<<Text>> " + message
        ),
    },
]
```

D Prompt to generate Reddit posts based on title

```
messages = [
    {
        "role": "system",
        "content": (
            "You are a Danish language assistant specialized in writing and rephrasing Reddit posts. "
            "Your task is to generate a well-written Reddit post based **only on the input title** provided. "
            "The post must maintain the same tone and context as the title and be clear, engaging, and appropriate for Reddit. "
            "The generated text should align with the typical length of Reddit posts (a short paragraph or two).\\n\\n"
            "You **must preserve all formatting**, such as bold, italic, links, hashtags, or other styles, if relevant.\\n\\n"
            "Your output **must strictly follow this format**:\\n\\n"
            "<<Title>>\\n[Rewritten title here]\\n\\n"
            "<<Text>>\\n[Generated Reddit post text here]\\n\\n"
            "Do not deviate from this format under any circumstances."
        ),
    },
    {
        "role": "user",
        "content": (
            "Please write a Reddit post in Danish based on the following title. "
            "Ensure the meaning, tone, and formatting are preserved. "
            "Return the output **strictly** in the following format:\\n\\n"
            "<<Title>>\\n[Rewritten title here]\\n\\n"
            "<<Text>>\\n[Generated Reddit post text here]\\n\\n"
            "Here is the input title:\\n\\n"
            "<<Title>> " + title
        ),
    },
]
```