Innuendo Discovery with BERT

Devin Xiao and Vincent Xiao

Introduction

An innuendo is a euphemism for an explicit or sexual action or idea that may or may not be obvious in purpose. We want to test the hypothesis that innuendos appear in similar contexts as the explicit words they represent. If this is the case, we want to determine if language models can learn sufficiently distinct representations of words such that it can identify the suggestive nature of texts even in documents without overtly explicit content. There are many ways to motivate innuendo detection. One such motivation of this project is content filtering and moderation, an ongoing issue that changes as language develops. Social media platforms or other products that market themselves to an underage audience must monitor posts and interactions to protect vulnerable audiences.

1 Data and Resources

We used multiple sources for this project. Our analysis mainly focused on comments from Reddit, where we found a Kaggle dataset containing 25000 comments for each of the 40 most visited subreddits in May 2019. In particular, the subreddit on which the comments were posted was of interest to us, with some being explicit and others being more general in nature. We combined this with a Kaggle dataset of another 10000 scraped comments from the subreddit r/gonewild specifically and the Jigsaw Unintended Bias in Toxicity Classification dataset from Kaggle, which scores comments on several metrics, including the sexual nature of the comment, based on human-given ratings. Since not all the Jigsaw comments received sufficiently high sexually explicit ratings, we only included a subset of the comments with a rating of about 0.5 (at least half of the raters categorized the comment as sexually explicit) for our analysis. Lastly, we obtained a list of obscene words, including those used in a sexual context, from Kaggle for evaluation purposes.

Since we had fewer sexually explicit comments than general comments, we downsampled the general comments to match the number of explicit comments. In the end, we had 9662 training examples, 4831 validation examples, and 4831 testing examples. We also set aside an additional 19325 explicit comments for masked language modeling (MLM) fine-tuning.

After gathering our data, we could classify the comments as either coming from an explicit source (r/gonewild or the Jigsaw dataset) or coming from a general source (the other non-explicit subreddits). However, we wanted an even finer distinction between outwardly explicit and sexually suggestive comments for evaluation purposes. More specifically, we were interested in our model's ability to correctly classify comments as coming from an explicit source even when there are no explicit words. Similarly, we also wanted to evaluate our model's performance on correctly predicting comments as not coming from an explicit source even when there are explicit words. As such, we further labeled the comments as either containing explicit words (or not) for testing only. To identify explicit words, we used a subset of the list of obscene words that were labeled as having to do with 'sexual anatomy / sexual acts', 'bodily fluids / excrement', or 'sexual orientation / gender', and further modified the list by removing inappropriate or unrelated words and adding some missing words based on a scan of the explicit source comments.

2 Methodology

2.1 Models

We decided to fine-tune a BERT model for classification. One useful output of BERT is the embedding for the CLS token, or some transformation of it given by the pooler output. The CLS embedding is a useful representation of a sentence that can be passed through a linear layer to get a sentence-level classification. Additionally, similar research in classifying sexually explicit text has shown that BERT works well, especially with its attention mechanism and contextually dependent embeddings (Qiu, 2024).

2.2 Masked Language Modeling

It is worth investigating if additional MLM pretraining would improve classification accuracy. We follow the standard procedure for masking. That is, for each batch, 15% of the tokens are selected for masking. For each selected token, there is an 80% chance to replace it with the mask token, 10% chance to replace it with a random token, and 10% chance to remain unchanged. The training procedure is treated as a classification task. The model is to predict the masked tokens, and a cross-entropy loss is calculated from the resulting probability distribution and ground truth token. MLM pretraining was accomplished with huggingface's Trainer API[1].

Sequence Classification

After the MLM fine-tuning, we loaded the fine-tuned model for sequence classification. The model is to predict if the document came from an explicit source or non-explicit source. We trained the model for 3 epochs and saved the model after each epoch if the f1-score on the validation set improved.

2.3 Model Evaluation

To evaluate our results, we compared our model with multiple baselines, First, to test the effect of MLM fine-tuning, we compared our model to a BERT model without additional MLM pretraining (in other words, it immediately started the classification task). Additionally, we assessed the importance of the contextualized embeddings by comparing it with [1]classical models like logistic regression and naïve Bayes. For logistic regression, we tested with vectorized counts and averaged GloVe embeddings as inputs.

3 Results

Table 1 summarizes the performance of the models across different metrics below. We also summarize accuracy across the different test sets: explicit source explicit words (ESEW), explicit source no explicit words (ESNW), control source explicit words (CSEW), and control source no explicit words (CSNW).

From our evaluations using the test set, we found that the fine-tuned BERT models performed the best based on overall accuracy and F1 scores. The logistic regression model using averaged GloVe embeddings performed slightly worse than the count vectorizer models. The non-BERT models all had different strengths and weaknesses in terms of test set performance. In particular, for the innuendo set (explicit source, no explicit words), logistic regression with vectorized counts had the highest accuracy, though we also see that it performed worse on the control source test sets. In this case, it seems like the model is predicting comments as coming from an explicit source more often. There are also differences in test set performance between the fine-tuned BERT models. The MLM fine-tuned BERT has higher accuracy than the classification-only BERT across the explicit source test sets, but lower accuracy across the non-explicit source test sets. Even though the overall accuracy is not affected much, it seems like there is an effect of MLM fine-tuning on the classification task. Specifically, MLM fine-tuning seems to have pushed the model towards predicting comments as coming from an explicit source. In cases where

Model	ACC	F1	ESEW	ESNW	CSEW	CSNW
Naïve Bayes	0.809	0.806	0.915	0.734	0.732	0.836
Logistic Regression (vectorized counts)	0.805	0.812	0.912	0.809	0.696	0.775
Logistic Regression (averaged GloVe)	0.795	0.789	0.828	0.739	0.774	0.828
BERT – classification only	0.893	0.893	0.965	0.841	0.786	0.919
BERT – MLM + classification	0.892	0.892	0.978	0.875	0.728	0.893

Table 1: Model metrics.

[1]https://github.com/huggingface/transformers/blob/main/src/transformers/trainer.py

detecting potential sexual innuendo and intent is of greater concern than anything else, such as in content filtering, the MLM fine-tuned BERT may be preferred.

To study how the MLM task affected the final embeddings, we employ principal component analysis (PCA) to find the directions of largest variance and visualize our fine-tuned embeddings in two dimensions. We first create sentences containing innuendos and identify the innuendo term within the document. For example, one such sentence is "It's like a hundred 99 degrees When you're doing it with me, doing it with me," where the suspected innuendo is "doing it". Next, we sample some words from the explicit words list and some random words from the tokenizer's vocabulary as baseline comparisons. We replace the innuendo in the sentence with each of these sampled words and extract the embeddings from our MLM fine-tuned BERT model. If the tokenizer separates a word into subwords, the embeddings of its subwords are averaged. Finally, we execute PCA on the resulting matrix of embeddings. Figure 1 shows the results of this procedure with the sentence mentioned previously. We observe a clear dividing line between the explicit and non-explicit words. Sampling different words using the same template sentence yields similar results, suggesting that these representations have learned a distinction between the two types of terms. The location of the suspected innuendo is also interesting. Notably, in some cases, the position of its embedding is in between the clusters of non-explicit and explicit words, which suggests the double entendre nature of innuendos.

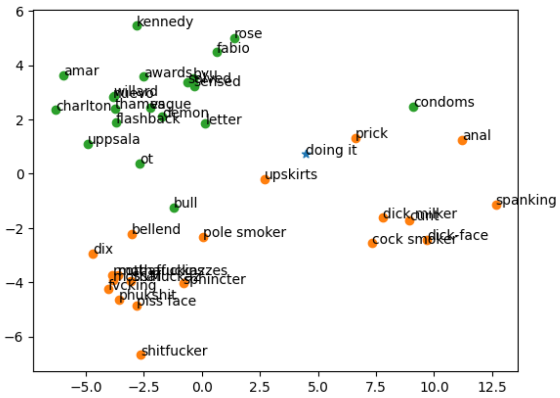

Figure 1

To try to get a deeper understanding of why our fine-tuned BERT models make the predictions that they do, we looked at some model explanation methods. The first that we looked at was the simple L2 gradient, which looks at the gradients of the model output (class logits) with respect to the inputs (input embeddings), then takes the L2 norm to reduce the gradients to a singular value. Here, we define the input embeddings to be the sum of the token, position, and segment embeddings passed as input to the BERT encoder layers. Let k be the class of interest and $\mathbf{X}$ be a vector of input embeddings.

$$\mathrm{grad}_1(i, k, \mathbf{X}) = \frac{\partial o(k, \mathbf{X})}{\partial i}$$

Equation 1

Equation 1 gives the L2 gradient of the output with respect to the i^{th} embedding vector.

Focusing on a particular input embedding, if the L2 norm of the gradient with respect to that embedding is relatively large, then a small change in the embedding could result in a relatively large change in the predicted class probabilities. One drawback of the simple L2 gradient is that it does not give a sense of direction since taking the L2 norm forces the value to be positive. As a result, we cannot tell which embeddings influenced the models to predict the explicit class over the non-explicit class and vice-versa.

An alternative to the simple L2 gradient is integrated gradients. One problem with the simple L2 gradient is that the gradient will diminish for inputs in the saturated regions of an activation function. To solve this, we integrate over all gradients on a linear interpolation between a baseline input $\mathbf{X}_{\mathrm{bar}}$ and $\mathbf{X}$. In this case, $\mathbf{X}_{\mathrm{bar}}$ is a vector of all-zero embeddings.

$$\mathrm{grad}_{\int}(i, k, \mathbf{X}) = \int_{\alpha=0}^{1} \frac{\partial o(k, \bar{\mathbf{X}} + \alpha(\mathbf{X} - \bar{\mathbf{X}}))}{\partial i} \partial\alpha$$
$$\approx \frac{1}{M} \sum_{m=1}^{M} \frac{\partial o(k, \bar{\mathbf{X}} + \frac{m}{M}(\mathbf{X} - \bar{\mathbf{X}}))}{\partial i}$$

Equation 2

We can similarly take the L2 norm of the gradients to get a singular value representation, though the Captum documentation[2] suggests an alternative method. If we summed up the elements of each integrated gradient vector and normalized them, then we would retain some directional

[2] https://github.com/pytorch/captum

information since the values are not forced to be positive. For our two-class problem, focusing on the gradient of the logit for the explicit class, a positive integrated gradient suggests the embedding pushes the prediction toward the explicit class, while a negative integrated gradient suggests the embedding pushes the prediction away from it. One drawback is that some understanding of overall importance is lost, since large positive and negative components of the integrated gradient vector cancel each other out when we sum across the vector.

The last model explanation method we tested was a token perturbation method. The idea is that if a token is particularly important in the classification, then if we perturb the token by masking it, then we may see a significant change in the output probabilities. To implement this method, we took a sentence and passed it through our models to obtain outputs (class logits and probabilities). Then, in turn, we took the tokenized sentence, replaced a singular token with the mask token, and obtained model outputs for the new sentence. We then identified important tokens by comparing changes in the class probabilities from the baseline unmasked sentence. If the probability of the non-sexually explicit class increased after masking the token, then the token contributed to the positive class prediction. Else, if the probability of the non-sexually explicit class increased after masking the token, then the token contributed to the negative class prediction. One drawback to this approach is that it may be less effective if there are multiple tokens that are strongly contributing to a particular class prediction, causing the probabilities with or without masking to all be close to 1 or 0. In this case, differences between probabilities are diminished, making it difficult to pinpoint important tokens.

Below, we compare the model explanation methods on example comments from our test set. The scale is relative to the largest score (in absolute

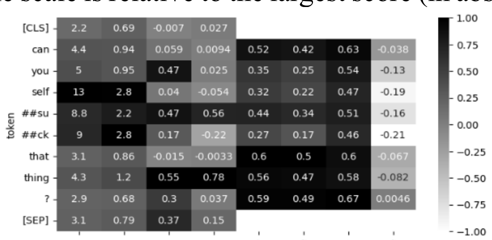

Figure 3

value) in each column. We focused on comments without any explicit words as our primary interest is in assessing our model's ability to detect innuendos. Let's consider two cases where the model predictions differ. In each case, the comment was from an explicit source.

In this case, the non-MLM fine-tuned model predicted the explicit class while the MLM fine-tuned model predicted the non-explicit class. One interesting thing to note is that the simple L2 scores are similar in scale between both models. There are some large differences in the integrated gradient scores for the two models, such as in the scores for "you", "##ck", and "?". Since "suck" is not in the vocabulary, it gets tokenized into subwords. As a result, the explanation scores for "suck" may not be as high as we would expect if we had a proper fine-tuned embedding for it. One idea is to add new words to the vocabulary, though this requires further training data. Based on the perturbation columns, it seems like the tokens that most influenced the MLM fine-tuned model's misclassification were "self" and "##ck".

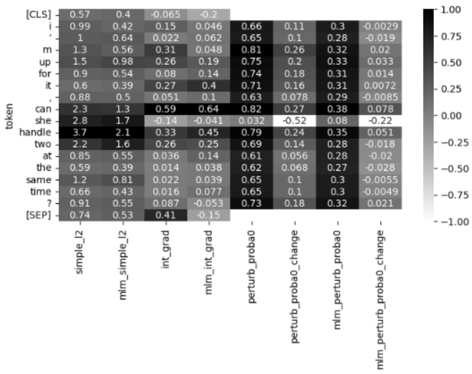

Figure 2

In this case, the non-MLM fine-tuned model predicted the non-explicit class while the MLM fine-tuned model predicted the explicit class. Based on the perturbation columns, it seems like the token "she" most influenced the non-MLM fine-tuned model's misclassification. This is also reflected in the integrated gradients, with the embedding relating to "she" having the most negative score.

4 Team structure

Everyone on the team contributed equally to this project. This includes data gathering and cleaning, MLM pretraining, classification finetuning, model evaluation, and this final report.

References

Huachuan Qiu, Shuai Zhang, Hongliang He, Anqi Li, and Zhenzhong Lan. 2024. Facilitating Pornographic Text Detection for Open-Domain Dialogue Systems via Knowledge Distillation of Large Language Models. In *27th International Conference on Computer Supported Cooperative Work in Design*.

Nina Poerner, Benjamin Roth, and Hinrich Schutze. 2018. Evaluating neural network explanation methods using hybrid documents and morphological agreement. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics.*

Mathurin Aché. 2021. The Obscenity List. https://www.kaggle.com/datasets/mathurinache/the-obscenity-list.

Jigsaw. 2019. Jigsaw Unintended Bias in Toxicity Classification. https://www.kaggle.com/competitions/jigsaw-unintended-bias-in-toxicity-classification.

Samuel Magnan. 2020. 1 million Reddit comments from 40 subreddits. https://www.kaggle.com/datasets/smagnan/1-million-reddit-comments-from-40-subreddits.

Harsh Pandey. 2020. Sexually explicit comments. https://www.kaggle.com/datasets/harsh03/sexually-explicit-comments.