Data Analysis and ML with Python Final project
- Toxic comment detector

財金二 柯宥圻 歷史四 陳彤樺
經濟四 楊永晨 土木碩一 林承平

The purpose of our project

Toxic comments are defined as "content that is rude, disrespectful, and unreasonable, making it difficult to engage in rational discussion with the commenter." It can cause psychological harm, elicit negative feelings, and can result in social isolation and physical health consequences, harming both people and society as a whole. Combating toxic comments is critical for establishing a safe and inclusive atmosphere that values the well-being and rights of everyone.

The motivation for this project derives from the recent event of serious "Hate speech" on campus. While supporting those who are discriminated against with mental health support services is definitely important, we believe it is equally crucial to address the underlying causes and put preventative measures in place to minimize the likelihood that such tragedies will occur again.

Our goal is to use the tools learned in this course, as well as BERT's unique concepts of Transformer, Attention, and Word Embedding, to assess the meaning of phrases and texts. This will allow us to develop a "Toxic comment selection" model that will encourage people to think twice before writing a comment. Finally, we anticipate that users will be able to utilize this model along with Chatgpt API to monitor whether their comments are harmful and recommend better sentences. This will encourage users to calm down before submitting potentially deadly language, and text mining techniques will be used to determine whether individuals have posted toxic comments.

Literature Review

We will discuss the models we investigated and explain why we chose the BERT model as the foundation for analysis in the literature review section. Team members carried out comprehensive research on the strengths and shortcomings of these models by reviewing literature and related materials.

(1) Language Model

Language models estimate the probability of a given string of words using Bayesian probability. By evaluating the arrangement of a sentence, we can assess whether it conforms to the expected patterns. For example, if there are two similar results for a given voice input, a well-trained language model can determine which one is more likely. The evaluation of likelihood is performed by analyzing the sentence using a language model, typically using perplexity as a measure of predictive performance. The formula for perplexity is as follows:[2]

$$Perplexity = 2^{-l} \quad \text{where} \quad l = \frac{1}{M}\sum_{i=1}^{m} \log p(w_i)$$

From this formula, we can understand that a lower perplexity indicates that a text is more likely to be accepted by the language model, meaning it resembles the language used in the real world.

[2] The formulas and concepts presented here are derived from Chen tsu pei on Medium: https://medium.com/nlp-tsupei/perplexity%E6%98%AF%E4%BB%80%E9%BA%BC-426f52897513

However, after reviewing relevant literature, our team found that using a language model is not suitable for this type of task. Through collective reading, we discovered that perplexity cannot effectively translate into a measure of "the extent of

toxic comments." The magnitude of perplexity does not provide a clear reference for assessing the degree of emotional blackmail. Additionally, language models are probabilistic models that lack the ability to understand the meaning of each word. They simply rank the likelihood of a particular word appearing. Therefore, they are not helpful in determining the semantics and meanings of words and sentences. We refer to the literature review in Appendix sections 3 to 5 for further details.

(2). recurrent neural network 以及 LSTMs

Most older models for building language models are based on RNNs, but they suffer from the issue of vanishing gradients, making it difficult to model long-term dependencies in the context. LSTMs, a variant of RNNs, address this problem and have the ability to understand longer contexts in a document. The state-of-the-art approach for LSTMs has evolved into the use of bidirectional LSTMs, which read the context from both left to right and right to left. Well-known models in this category include ELMO and ULMFIT.

The purpose of RNN-based model pretraining is to anticipate the next word by utilizing unsupervised learning to learn the representation of natural language. During the embedding output, the RNN considers the contextual information of the sentence . ELMO's pretraining process not only learns word embeddings but also learns a two-layer, bidirectional LSTM network structure.

However, according to our literature review, RNN-based neural network models are not as effective in feature extraction as newer encoder approaches and transformer models. Many studies have shown that the Transformer's capacity to extract features outperforms that of the LSTM. Furthermore, models such as ELMo, which use RNN-based algorithms for bidirectional concatenation and feature fusion, are inferior to BERT's integrated feature fusion approach. Another disadvantage is the long training time. Therefore, we have chosen not to use RNN models and instead opted for the widely recommended Transformers architecture. The review of RNN-related literature can be found in references six to nine in our data.

(3) Gensim+Word2vec

While Word2Vec is quick and easy to use, it has one major drawback: the embedding for each word is fixed. However, as previously said, words that appear the same in different phrases may have different meanings. For example, depending on the context, the term "It" might mean a variety of things. Gensim is a Python package for dealing with learning details. References ten to eleven in our data contain a review of Word2Vec-related literature.

(4) Transformer, attention, and BERT

Our team also investigated the applicability of the BERT model, particularly the Transformer approach, which was not presented in the course but is critical. We also explored the Attention model, which assigns distinct meanings to the same word in different temporal and spatial circumstances. After gaining a deeper understanding of these techniques, we got more sure that using the BERT model for detecting toxic comment semantics is the most accurate strategy currently available. As a result, we used TFBertForSequenceClassification for sentence analysis and classification, and embedding extraction.

The Transformer is a Seq2Seq model based on self-attention that has received a lot of interest in recent years for applications including chatbots, speech recognition, and machine translation. Different models are often required in many NLP applications, and building and testing these models

can be costly. BERT, on the other hand, is trained on a Transformer Encoder with vast volumes of text data and two pre-training objectives (masked language modeling and next sentence prediction). Because of this pre-training, BERT can serve as a basic model for numerous downstream activities, accommodating numerous model structures as needed.

This is the two-stage "transfer learning" process: first, training a general model that understands natural language, then extracting features and fine-tuning parameters for downstream tasks. When reviewing the literature, we deeply appreciated the power of this framework, supported by multiple datasets that confirm it as the best model type. Therefore, for the task of sentiment analysis, we adopted this model architecture for decision-making. The literature review on BERT can be found in the offered materials' references 12 to 15.

Our Solution

Text Classification Based on BERT Model

In summary, this is a binary "classifier" where we take a "pretrained" BERT model and perform "fine-tuning" for text classification tasks. This process is known as "transfer learning," where the model first learns the basic Chinese grammar structure through pretraining. Then, the training data consisting of Chinese sentences is tokenized, and the tokens are fed into the TFBertForSequenceClassification module. Training parameters are specified, and the model undergoes training to generate the final model.

(1)、Collection and Preprocessing of Toxic Comment Texts

Regarding the training data for the model, we used individual malicious comment phrases and conversation messages as text for emotional extortion. Originally, we intended to focus on analyzing emotional extortion phrases between couples. However, this scope was too specialized and lacked relevant data. Therefore, we decided to broaden it to include any sentences containing toxic comments as training text. Although this approach would reduce the specialization of the project, it would allow for analysis of a wider range of scenarios involving toxic comments. Subsequently, it was proven that the model performed remarkably well in terms of toxic comment expressions in the emotional aspect.

(2)、Pre-training

Pre-training is the first stage of constructing the BERT model. It is done in an unsupervised manner and involves two pre-training tasks: masked language modeling (MLM) and next sentence prediction (NSP). If there is a large amount of data and sufficient resources, it is possible to perform pre-training on the training data. However, training a well-performing 12-layer BERT-BASE model with 110 million parameters requires running it on 16 TPU chips for 4 days, costing around $500. On the other hand, the larger 24-layer BERT-LARGE model with 340 million parameters needs to be trained using 64 TPU chips (costing approximately $7,000). (Resources: https://leemeng.tw/attack_on_bert_transfer_learning_in_nlp.html).

Google has uploaded pre-trained BERT models of various sizes on GitHub for download. However, the pre-trained model we use comes from the "bert-base-uncased" file provided by HuggingFace. This project offers transformers models for the English language (including ALBERT, BERT, GPT2) and natural language processing tools (such as tokenization, part-of-

speech tagging, and named entity recognition). The official documentation of this project recommends using BertTokenizerFast as the tokenizer.

(3)、Fine-tuning: Tokenizer, Model

We began fine-tuning the downstream task model (text classification). BERT allows us to embed the pretrained model into a specific task. For text classification, we simply added a simple softmax classifier on top. Combining transfer learning, we transferred the original pretrained model to the new topic of malicious comments, making it a useful module.

The training model (Model) and tokenizer (Tokenizer) are the backbone of training the malicious speech classifier. Additionally, the from_pretrained() function allows for downloading specific pretrained models. The Transformer handles sequential data, and the performance of the encoder is enhanced through the use of an attention mask. After 12 stacked encoders, the embedding of each word is outputted.

Since the model cannot directly recognize Chinese characters, only numbers, we need to map words to numbers. We can create a dictionary for the word vocabulary, where each word represents a corresponding index. Additionally, we specify a "maximum length." For sentences shorter than the maximum length, we pad them with [PAD]. For sentences longer than the maximum length, we truncate them. Therefore, any sentence can be represented as a 200-dimensional vector. In addition, there is a list called the "attention mask." For each sentence, if the corresponding word at a position is [PAD], the element of the attention mask at that position is set to 0; otherwise, it is set to 1.

Next, each word output has an embedding dimension of 768, and there are three types of embeddings: Token embedding, Segment embedding, and Attention mask. There are also special token embeddings such as the [CLS] token at the beginning of the sequence, which can capture the meaning of the entire sentence, and the [SEP] token at the end.

TFBertForSequenceClassification is a pre-packaged classification model specifically designed for sentence classification. It consists of 1 layer of BERT, 1 layer of dropout, and 1 layer of feed-forward network. Its input data is the tokens described earlier, and after deep learning, we obtain the final complete model.

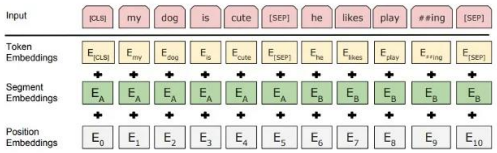

Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

In summary, the format of the training data is (sentence, label). For each sentence, it is tokenized using the BertTokenizerFast, resulting in embeddings. Finally, they are converted into numpy arrays. The dimensions of bert_train are (31000, 3, 200), corresponding to the total number of data points. Each data point has input_ids, token_type_ids, and attention_mask, which are three different lists. Additionally, each of these lists has a length of 200 (specified as 200, padding or truncating as necessary). Lastly, the embedding dimension for each word is 768.

(4)、Model training

First, the data is divided into training data, validation data, and testing data using an 8:1:1 ratio (with 26,400 samples, 3,100 samples, and 3,100 samples, respectively). Each set of data is then

squeezed using .squeeze() to remove unnecessary array dimensions, resulting in the final dataset.

The loss function is used to evaluate the discrepancy between the desired outcome and the predicted outcome. A larger loss indicates a larger gap between the predicted and actual values, indicating poorer performance of the model. Conversely, in a neural network, a smaller loss indicates a smaller gap between the predicted and actual values, indicating a more effective model. During the process of minimizing the loss function, if overfitting does not occur, our goal is to minimize the loss function. Following the recommendations in the Huggingface documentation, we use BinaryCrossentropy as the loss function for binary classification to achieve optimal performance.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

The purpose of the optimizer is to adjust the parameters within the model to find the most accurate values. In a neural network, which is composed of numerous neurons, each neuron has its own weights. The optimizer is responsible for adjusting these parameters to minimize the loss function. Similarly, we have adopted the recommended optimizer: Adam. Adam is a stochastic gradient descent method based on first and second-order moments, and it has advantages such as high computational efficiency and low memory requirements, making it well-suited for problems with large data or parameters. Additionally, Adam performs bias correction and ensures a consistent range for the learning rate at each iteration, resulting in smoother parameter updates.

After compiling the data into the model, we use the model.fit(train_ds, epochs=10, validation_data=valid_ds) function to train the final model. In neural network training, "epochs" refers to the number of times the algorithm has processed each data point in the dataset. In this case, the model's parameters are adjusted 10 times in total.

(5)、Input, processed, and output

We pass toxic comments into the model for prediction. This generates an output that represents the probability of the input being a toxic or non-toxic comment. After applying the Softmax function, the output is transformed into a set of probabilities, with the constraint that their sum is equal to 1. Softmax is a powerful function that takes a set of numbers within the range of positive and negative infinity and converts them into a set of probabilities. It is a normalization function that ensures the probabilities add up to 1. It's important to note that the nature of the data is not affected by the softmax transformation. The original probabilities with higher values will remain higher after the transformation, and vice versa.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

System Outcome

Currently, our presentation model is similar to what was mentioned in the first paragraph. We aim to provide an alerting system that helps us determine whether our spoken statements contain elements of toxic language. To achieve this, we crawled the main posts from the two sections of Reddit and determined if they contained hate speech. Although the format of these posts primarily consists of complaints and differs slightly from chat platforms, the results we obtained generally show low percentages. For example, [1] had only 19% toxicity. Only comments with a more sarcastic tone had higher percentages, such as [2], which reached 99%. Overall, the results are highly accurate, and

we have rarely found non-toxic statements with high percentages of toxicity.

However, there is a small drawback we have observed in our current testing. The model tends to lose accuracy when the text becomes longer. For example, in [3], there were some dialogues discussing toxic comments, but the toxicity score was only 4.5%. The effect was not very significant. We suspect that the main reason for this is that the scores of toxic comment sentences within the article are averaged out by other descriptive sentences. Typically, toxic comments are more limited to short descriptions and chat dialogues, and there are fewer instances of toxic comments in the form of long articles. To achieve better results in these cases, we would need to spend more time building a more comprehensive model. However, our current focus is primarily on the application of short texts, and we have not conducted more detailed analysis. We are also looking forward to future development of the model, where it can be integrated with user-friendly interfaces or applied as a plugin in popular chat platforms. By focusing more on short texts, it will better align with our initial intentions. We will provide a clearer description in the next section.

---

[1] https://www.reddit.com/r/AmItheAsshole/comments/1410ouf/comment/jmy3y6k/?utm_source=share&utm_medium=web2x&context=3

[2]https://www.reddit.com/r/TrueUnpopularOpinion/comments/140yv59/comment/jmyinrj/?utm_source=share&utm_medium=web2x&context=3

[3]https://www.reddit.com/r/AmItheAsshole/comments/140zgec/aita_for_telling_my_wife_i_wont_let_her_name_our/?utm_source=share&utm_medium=web2x&context=3

Further Expectation:

1. With the current text as a foundation, both classification and clustering appear to be relatively straightforward. However, due to the lack of data, we regretfully cannot provide specific numbers at the moment. In the future, we plan to increase the dataset of toxic comments, potentially adding thousands of examples and incorporating different label categories such as "toxic comments about product quality," "toxic comments about service quality," "emotional blackmail related to environmental quality," and so on. We will utilize SVM or KNN models to classify and predict the new input sentences. For example, we can determine that a sentence has a toxicity ratio of 90% and classify it as a toxic comment in the category of service quality.

2. We aim to turn this simulator into a Line Bot, where users can input a sentence and the Line Bot will directly respond whether the sentence contains toxic content or not. Our team has spent a considerable amount of time researching this, but none of us have the necessary expertise, such as knowledge of the Django framework, to implement it. However, in the future, once we acquire the required skills and knowledge, we can unleash the full potential of the Line Bot. It will serve as a helpful tool for anyone using Line for communication, allowing them to ask the Line Bot if a sentence contains elements of emotional blackmail and even receive suggestions for improvement. We believe this would be the best application for our model.

Conclusion

1. We trained our model using BERT, with BinaryCrossentropy as the loss function and Adam as the optimizer.

2. Our model, after undergoing softmax transformation, is capable of effectively distinguishing between emotionally manipulative and non-emotionally manipulative sentences in chat applications and social media platforms.

3. This model can effectively differentiate between shorter toxic and non-toxic comments, but its ability to discern longer texts is relatively weak.

Feelings

財金二 柯宥圻

As the leader of the team, I devoted myself to the project and dedicated most of the time digging into the details of the BERT model and other ML-related tricks that I had not heard of before. What is more, I also learn the basics of front-end development language, such as flask, as the framework of our web-app, HTML, CSS, and so on. It was a hard time learning so much new stuff in a short time, but I was very delighted to have a chance implementing this final project. In the near future, I will try adding new features of the project, and make it more completed as a side project when applying for related intern opportunities.

土木碩一 林承平

From this final report, I believe I have learned a lot about language recognition techniques, particularly through my exposure to the BERT model. This field is something I had never encountered in my previous machine learning studies, and I faced numerous challenges along the way. However, I am delighted to have overcome them with my teammates, and it has given me a great sense of achievement and fulfillment.

經濟四 楊永晨

Reflecting on my work on this project, I have gained valuable experience in data analysis and machine learning. I successfully implemented various techniques and tools, such as Flask, TensorFlow, Javascript, and BERT models. Despite encountering challenges along the way, especially regarding attempting to upload our app onto Amazon's Elastic Beanstalk, I persevered and learned skills in troubleshooting. Overall, this project has been a rewarding journey of growth and learning.

歷史四 陳彤樺

Although our report certainly has room for improvement, I am delighted that our project can serve as a first step towards combating toxic speech on campus. Throughout this process, I have also gained a deeper understanding of machine learning and its practical applications in daily life. Additionally, I am grateful to my team members for their prompt assistance and insightful advice during difficult times. Their assistance has been critical in overcoming challenges, moving forward, and, most importantly, completing this project together.

Reference

Reference documents for the literature review:
[1] 關鍵評論網
https://www.thenewslens.com/article/63074
[2]公式和觀念整理自 Chen tsu pei on Medium:
https://medium.com/nlp-tsupei/perplexity%E6%98%AF%E4%BB%80%E9%BA%BC-426f52897513
[3]https://arxiv.org/abs/2204.06031
[4]https://arxiv.org/pdf/1707.05589.pdf
[5]https://www.techtarget.com/searchenterpriseai/definition/language-modeling
[6]https://arxiv.org/abs/1802.05365
[7]https://asset-pdf.scinapse.io/prod/2944851425/2944851425.pdf
[8]https://hackmd.io/@shaoeChen/Bky0Cnx7L

[9]https://www.cnblogs.com/yifanrensheng/p/13167787.html
[10]https://arxiv.org/abs/1301.3781
[11]https://paperswithcode.com/method/cbow-word2vec
[12]https://arxiv.org/abs/2103.11943
[13]https://arxiv.org/abs/1706.03762
[14]https://leemeng.tw/attack_on_bert_transfer_learning_in_nlp.html
[15]https://aclanthology.org/N19-1242/

Related documents:

1. Transformer on HuggingFace: https://huggingface.co/docs/transformers/
2. Models on HuggingFace: https://huggingface.co/models
3. TFBert on HuggingFace: https://huggingface.co/docs/transformers/main/model_doc/bert#tfbertmodel
4. Google Bert: https://github.com/google-research/bert
5. Oreilly library : https://www.oreilly.com/library/view/getting-started-with/9781838821593/2f41b723-d4d7-4bd9-a7e9-82ecb3d76bb4.xhtml#uuid-a03529e7-7b9c-4770-bb02-9cb564ac3e68
6. Epoch, batch size, learning rate: https://medium.com/%E4%BA%BA%E5%B7%A5%E6%99%BA%E6%85%A7-%E5%80%92%E5%BA%95%E6%9C%89%E5%A4%9A%E6%99%BA%E6%85%A7/epoch-batch-size-iteration-learning-rate-b62bf6334c49
7. https://ithelp.ithome.com.tw/articles/10241789