DATS6312 Natural Language Processing

Professor Amir Jafari

Final Project Report

Akinci Sertan

KONG Deyu

May. 1, 2022

Table of Contents

Abstract

Neural Networks have changed the way data is processed and analyzed by creating more advanced methods and architectures that not only drastically improved the older results but also allowed us to solve more complex problems that were not possible with classical models. Natural Language Processing is one of the fields that benefited from these new network structures the most. Especially with the development of transformers a new area of cutting-edge architecture was born.

Introduction

Natural Language Processing has been evolving for around 50 years. It started with simple models like Bag-of-Words which keeps track of occurrences of each word in a document. However, as the size of the documents grew this became a costly method keep up. This led to the introduction of TF-IDF model that used frequency of the terms as well as the importance of these words to other documents. Common words like ‘are’, ‘a’, ‘the’, ‘of’… etc. have also been removed from the calculations to reduce noise. This method though still fell short of predicting semantic relationships between words. This was followed by Co-occurrence Matrix and eventually Word2Vec which used embeddings to solve this problem. These models were powered by Recurring Neural Networks (RNN). This was particularly popular because of its recurring nature allowed to consider the order of

the words and their effect on their neighboring tokens. Like the others this network too had a weakness, the vanishing gradient. The solution to this problem came with Long Term Short Term Memory structure. This allowed long-term dependencies to be learned by retaining information from an earlier sequence and applying them to the much later sequences. Even with this advancement certain tasks like language translation, text summarization, and question answering were still challenging due to the required deep understanding of the context of the words. With the 2017 paper *'Attention Is All You Need'*, authors introduced an attention mechanism for which more complex networks like recurrent or convolution in encoder-decoder structure could be dispensed for a simpler architecture based only on attention which was the birth of Transformers. Transformers are currently the state-of-the-art model architectures in NLP. Their recent development also the reason why they are not extremely common yet. This paper will demonstrate the use of one of the best performing Transforms called Electra. The model will label Wikipedia comments based on the toxicity of the person's language. It is important to prevent harassment and abuse that often impacts information sharing, open discussions, and spread of reliable information. In addition, it will also provide us with useful insight on how censorship works, something in our everyday life from e-mails to social media.

Model: Electra

Transformers come in different shape and sizes. Although a good number of them use the encoder-decoder structure, depending on the task some use only the encoder part (including most BERT-based or inspired variants), while others use only the decoder part, like GT3 and Electra. Another big difference with Electra compared to its predecessors is that instead of masking words during pre-training, like masked language models (MLM), Electra replaces certain words with random ones that come from a small generator model (In our case a smaller version of BERT). Therefore instead of training the model to predict masked tokens, Electra trains a discriminator that identifies which tokens are original and which are generated. This way Electra processes all the tokens instead of just the masked ones which is shown to have a better contextual representation of tokens, outperforming MLMs with less computation required.

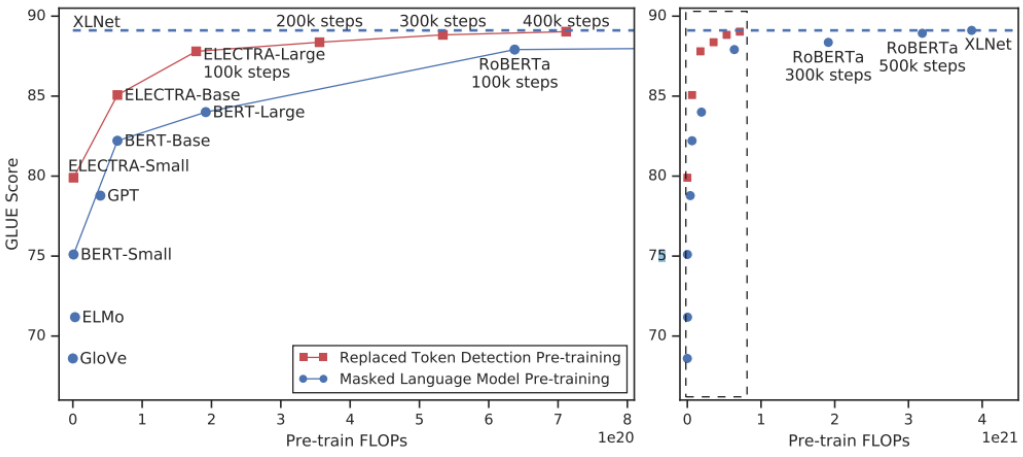

Fig.1 Electra Outcompetes MLMs in terms of performance/compute

Pretraining of Electra is very similar to GAN type architecture where two networks, a generator and a discriminator, compete against each other to better fake and better differentiate the input respectively. However with Electra, after the initial pretraining the generator model is disregarded and only the discriminator part becomes the main model. This model can handle anything from classification to question answering and text generation.

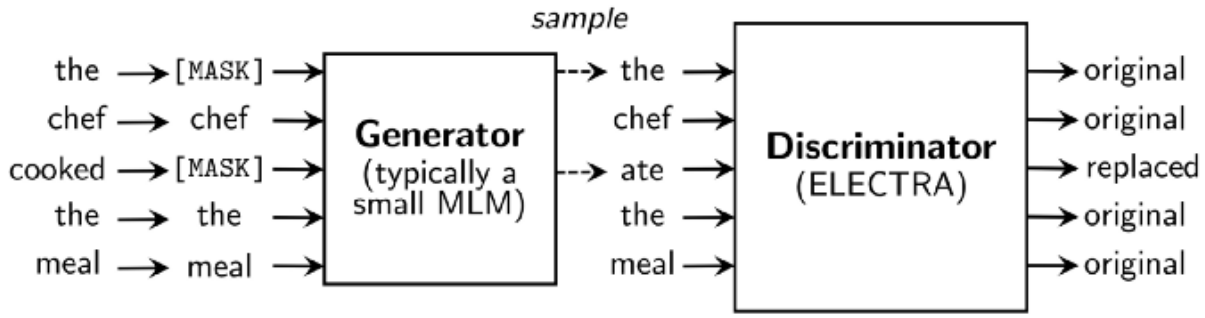

Fig.2 Training layout of Electra

A modest model can be trained from scratch but for advanced and useful models, they require an immense amount of data along with extreme computational power. This is where transfer learning comes handy where one can take advantage of already pretrained models by big companies for the public use.

Fig.3 Different heads could be attached to pretrained Electra based on task requirement

We used Electra's 'google/electra-small-discriminator' pretrained model along with its own tokenizer from 'transformers' library for our task.

Dataset

Our dataset is retrieved from Kaggle's *Toxic Comment Classification Challenge*[1] The dataset is a collection of comments from Wikipedia's talk page edits. There are approximately 159,500 comments with varying levels of foul language among neutral, harmless comments in the training set. The goal is to flag each comment with any of the six labels; toxic, severe_toxic, obscene, threat, insult, and identity_hate, making it a multi label classification problem. The distribution of the labels as follows:

Label	# of Comments containing this label
Toxic	15,294
severe_toxic	1,595
obscene	8,449
threat	478
insult	7,877
identity_hate	1,405

Table 1 Label distribution in training set

As can be seen from the table the classes are quite unbalanced. If no precaution is taken the results might be misleading or unexpected: the model would prone to classify comments as neutral or simply 'toxic' (The largest minority label) while failing to discern sentiments in other less represented minority classes.

[1] https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/overview

Training and Evaluation

During training we experimented with a variety of parameters. We tried batch sizes 8, 16, 32, and learning rate between 1.e-4 and 1e-6. The scheduler ReduceLROnPlateau was also used to schedule learning rate descent, while early stopping with patience of 5 epochs was implemented to prevent overfitting and exiting the training. Due to long training time for transformers in nature (And Electra is already a smaller one among them!), we save the model checkpoint after hours of training, usually 5 or 10 epochs, lower the learning rate and train for another 5 or 10 epochs. The final model was saved for evaluation on the heldout set provided by the competition organizer. Since the competition ended 4 years ago, full label and scoring details were released. The heldout set contains 153,164 entries while disabled a large portion of neutral comments (89,619 of them) during scoring. The final label distribution in heldout is shown below:

Label	# of Comments that contains this label
toxic	6,090
severe_toxic	367
obscene	3,691
threat	211
insult	3,427
indentity_hate	712

Table 2 Label distribution in heldout set

Results

Model	Epoch	Learning Rate	Test Acc	Eval Acc	Eval hlm
unbalanced	10 + 10	1e-4, 1e-5	0.96842	0.86508	0.03197
sampled(least_class)	5, early stopped	1e-5	0.15704	0.35379	0.16621
sampled_unsampled(least_sampled)	5 + 10	1e-5,1e-6	0.93753	0.89815	0.03842
sampled,unsampled,sampled(cycle)	5+10+5	1e-5,1e-6,1e-6	0.92907	0.89673	0.03880

Table 3 Results from different experiments

The most promising results came with batch size of 16 with initial learning rate of 1e-5 and 1e-6, and a fixed epoch size of 10 for each run. After the first training, we achieved high test accuracy scores. However, this was due to the unbalanced test set that was dominated by the same majority class as the training set resulting in high accuracy in that class but low accuracy in others. In order to have a more balanced training set we used a class called MultlabelBalancedSampler[1] by Issa Memari. This class had multiple options to balance the data, including cycling through classes or picking a certain sample with uniformly random amount. We used two of these options and trained the model three separate times, twice when we balanced the sample and once when we didn't balance it. This helped improve our overall as well as the minority class accuracy by a few percentages.

A Closer Look

To get a taste of how the model prediction would look like in deployment, we used models we used best performers of our previously trained model on several entries we picked from heldout set, as well as cuss word we wrote ourselves with expected labels, being as follows:

```
input1 = ":Dear god this site is horrible."  # 0,0,0,0,0,0
input2 = ":If you have a look back at the source, the information I updated was the correct form. " \
         "I can only guess the source hadn't updated. " \
         "I shall update the information once again but thank you for your message."  # 0,0,0,0,0,0
input3 = "DJ Robinson is gay as hell! he sucks his dick so much!!!!!"  # 1,0,1,0,1,1
input4 = "::No, he is an arrogant, self serving, immature idiot. Get it right."  # 1,0,1,0,1,0
input5 = "== All admins must DIE! == "  # 1,0,0,1,0,0
input6 = ", you stupid retard, or I will kill you!!!   ]]"  # 1,1,1,1,1,0
input7 = "a wild ass contains dick in the pussy"  # 1,1,1,0,0,0
input8 = "There is a special place reserved for you in hell, you nazi piece of shit"  # Original, Toxic, insult
input9 = "Sorry, I shouldn't argue with you. I'm outta here... to sleep with your MAMA!"  # Original, Obscene Sarcasm
```

Fig. 4 Comments for inference, with their labels

The unbalanced model using original dataloader could distinguish general toxic speech from neutral ones well, but often makes mistakes or failed to identify minority labels including ‘severe_toxic’, ‘threat’ and ‘identity_hate’. The two models using balanced dataloader(under different sampling options) are more aggressive, tends to classify the slightest disagreement and constructive criticism as toxic and insult. Furthermore, even these best models could not detect sarcasm in the tone. Taking ‘input9’ as example, we had to replace ‘sleep’ into cuss word to have the model flag this comment, otherwise it was regarded as neutral. In terms of deployment usability, we would recommend to go with the unbalanced model,

since too many false positives would lead to a flood of user appeals and complaints, thus defeating the purpose of applying machine learning techniques.

Limitations and Possible Improvements

The data is highly unbalanced therefore it is important to use some statistical balancing method to reduce the inconsistency. Another thing that may help is to clean the data. Since it is informal writing, some minimal manual work may be beneficial along with a casual tokenizer that could tidy up some of the messy words. We haven't had a chance to investigate the inner workings of the electra pretrained tokenizer.

We could apply data augmentation on misclassified classes as well, after each training loop, collect all entries with at least one label being misclassified and train on this subset. This is expected to boost performance as well.

For the dataloader, we could implement another mode that collect each label at least once to each batch. In this way, the model performance shouldn’t be as aggressive as the current balancing sampler does.

Conclusion

Transformers have become the most cutting edge tool for many areas of data science especially in NLP. Since they are relatively new we wanted to demonstrate the set up and the use of one of the best performing transformers, Electra. We decided to take on a multi label text classification problem with over 150,000 samples that turned out to be highly unbalanced. We over/under sampled minority and majority classes with MultlabelBalancedSampler and trained our model while fine tuning parameters like learning rate and batch size. Initially we attained a promising accuracy however a closer look revealed the true reason for the high score, the dominance of the majority class in both training and validation set. Then we adjusted for the unbalance and trained again on the balanced set as well as the unbalanced set sequentially.

In terms of accuracy score on heldout test set, we could not compete with top runners on the leaderboard. But we still learnt a good deal on how to implement finetuning of a transformer to a real dataset, practised working on custom dataloader, and made effort to cope with class imbalance issue. From the classification result, we also obtained better understanding on how these classifiers work, and gained some insights into censorship that is everywhere nowadays.

References

[1] Issamemari. Issamemari/Pytorch-multilabel-balanced-sampler: Pytorch sampler that outputs roughly balanced batches with support for multilabel datasets. GitHub. Retrieved May 1, 2022, from https://github.com/issamemari/pytorch-multilabel-balanced-sampler

[2] Clark, K., Luong, M.-T., Le, Q. V., & Manning, C. D. (2020, March 23). *Electra: Pre-training text encoders as discriminators rather than generators.* arXiv.org. Retrieved May 1, 2022, from https://arxiv.org/abs/2003.10555