ATCS - PROJECT
Transformer architectures for fine-grained sentiment analysis

Michele Bortone

July 9, 2021

Abstract

In the last years, a lot of different Transformer architectures have been published, and they became very popular language models. Exploiting transfer learning, it is possible to finetune a pretrained Transformer to learn a specific domain task, such as sentiment analysis, text summarization and question-answering.
In this project, I examined the fine-grained sentiment analysis task, but in a special case in which sentences are represented using a parse tree, instead a common sequence. I applied transfer learning starting from different Transformer-based architectures, BERT and GPT2, and compared results with previous approaches.

1 Introduction

Sentiment analysis is a task of text classification in which a piece of text has to be classified into one of the predefined sentiment classes. Typical approaches to solve this task are rule-based systems or supervised machine learning algorithms.
Normally, we deal with a binary sentiment analysis, in which the piece of text can be only "Positive" or "Negative". In some contexts, this approach would be very polarizing, so we can increase the number of labels to convert it in a Fine-grained sentiment analysis.
In the last ten years researchers have published lot of machine learning techniques for fine-grained sentiment analysis, with some common features.
The first common step in sentiment classification of a text is the embedding, where a piece of text is tokenized at different possible levels (e.g. words, phrases, sentences) and then each token is converted in a numerical vector. After the embedding phase it's possible to use different approaches to perform the classification.
In this project I formulated the hypothesis to obtain an improvement of performances using Transformers architectures. This Hypothesis is already partially confirmed with related works that I will describe in the next section, so I went more in depth testing different architectures didn't tried yet.
I performed the embedding and the classification phases trough transfer learning (pretraining and finetuning), starting from pretrained Transformer architectures and finetuning them for the fine-grained sentiment analysis task on the Stanford Sentiment Treebank (SST) dataset.
The Transformer [1] was proposed in 2017, as an alternative of recursive neural networks for the machine translation task. It exploits the Attention mechanism to force the network focusing in particular tokens.
Starting from the original Transformer architecture, a lot of languange models have been published based on this architecture. A trained language model is able to predict the next word or piece of text from a specific vocabulary, given a text in input. In this project I analyzed performances of two Transformer language models (finetuned for sentiment analysis), that represent the state of the art for many tasks in natural language processing. The first is BERT [2] (Bidirectional Encoder Representations from

Transformers) and then I compared it with an other famous and more recent language model, GPT2 [3]. About these models, the next sections go more on depth.

2 Related work

In 2013 researchers has published RNTN [4] (Recursive Neural Tensor Network), a recursive model trained with the Stanford Sentiment Treebank that pushed the state of the art for binary sentiment classification in terms of accuracy to 85.4% and 45.7% in the fine grained variant. This study showed that there are a lot of difficulties to predict the fine grained sentiment of the entire sentence (the root node of the parse tree), due to its length and words that completely change the sentence meaning such as negations. The tree representation of the sentence helps in these specific cases.
Recently Trasformer architectures outperformed RNNs in most of NLP tasks reaching the state of the art. One of the first works that followed this approach is a classifier based on BERT [5].
At the time of writing, the state of the art for the fine grained sentiment analysis tested on the Sentiment Treebank is a model based on RoBERTa [6]. This solution reached 59% of accuracy considering only the entire sentence.

3 Dataset

To deal with the fine-grained sentiment analysis I trained my models with the Stanford Sentiment Treebank dataset. It's a dataset containing 11855 sentences extracted from movie reviews.
These sentences have been parsed by the Stanford Parser, generating 215154 phrases stored in the tree structure of the sentence. With phrase it's intended a group of words with a grammatical structure, but it doesn't represent a complete thought, so a sentence is composed from many phrases.

The parse tree is labelled by humans not only in the root node, that represent the entire sentence, but in internal nodes as well. In this way we have every subsentence with a grammatical sense (phrase) that is labelled with its sentiment.
It's possible to see in Figure 1 the structure of a sentence and how it's labelled. Each leaf represent a word and this particular node is labelled with the sentiment of the single word.
In this project I worked on a five classes Sentiment analysis, so the dataset I used is the five labels version (SST-5), in which the label is represented by an integer from 0 to 4 corresponding to the following list of sentiment: Very negative, Negative, Neutral, Positive, Very positive. Also there is the binary version (SST-2) in which neutral sentences are deleted and we have only positive or negative sentences.
The tree structure of SST dataset is useful to analyze the context of the words, instead scoring each single word to assign the sentiment of the entire sentence.
There are cases in which predict the sentiment is tricky. An example of borderline cases could be negations of the sentence or some words. For instance the sentence "This movie was actually neither that funny, nor super witty." is clearly negative but its structure could trick a Sentiment analysis model.
In order to do some experiments using a bigger dataset, I worked with another similar dataset, Yelp-5. This dataset contains about 6 GB of reviews and the reviewer assigned 1 to 5 stars. In this case we don't have the tree structure but only a sequence and the mean length is bigger than SST-5 because reviews are simply text and not sentences or phrases. From this dataset I extracted 100000 labelled reviews.

4 Method

To deal with Sentiment analysis I've choosen to apply transfer learning, so the general strategy is to finetune my models with SST-5, but starting from a pretrained version.
As described before, I trained and compared two different transformers architectures, BERT and GPT2.

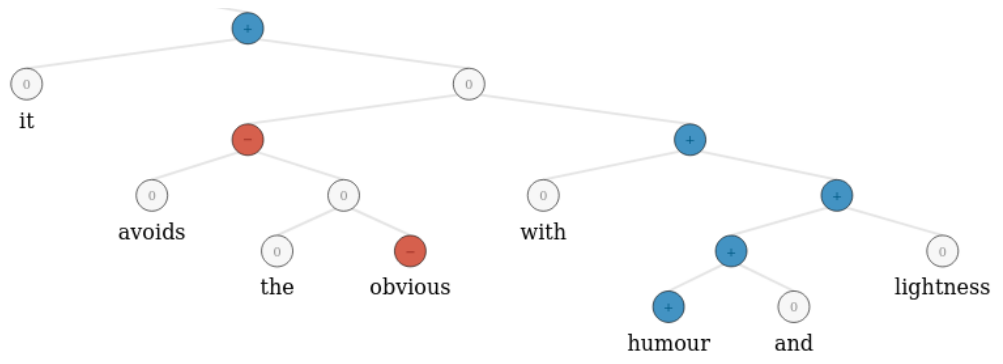

Figure 1: An instance of a parse tree generated and labelled

4.1 BERT vs GPT2 - Comparison

Both of these architectures are a derivation of The Transformer, a sequence to sequence neural network developed for the machine translation task.

Its architecture is shown in Figure 2, forming an

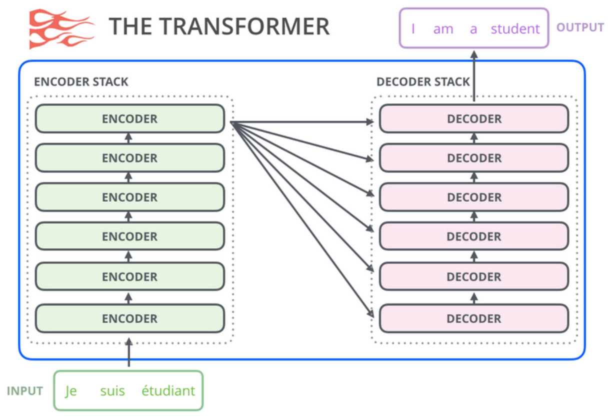

Figure 2: High level architecture of the Transformer

Encoder-Decoder structure. In particular it is a stack of Encoders followed by a stack of Decoders. Each Encoder and decoder are composed by sublayers, one or more for the Attention mechanism and then a feedforward neural network to compute the output. BERT is a language model built using a stack of Transformer encoders, instead GPT2 is a stack of Transformer Decoders, as it's possible to see in Figure 3.

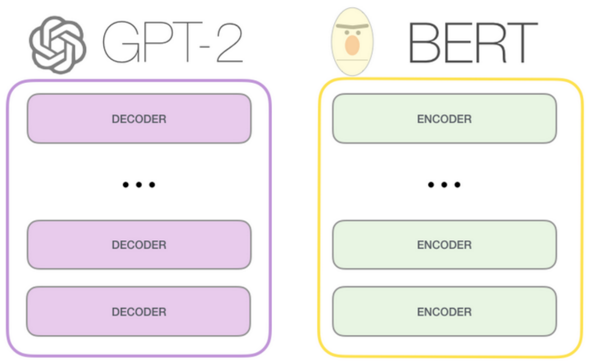

Figure 3: Difference between the GPT2 and BERT

Encoders and Decoder differ in how they implement the Attention mechanism.

4.2 Architecture for Sentiment Analysis

I built simple architectures starting from pretrained BERT and GPT2. For the two classifiers the pipeline is quite similar, and it's shown in Figure 4.

Preprocessing First of all, the text of each sentence needs to be preprocessed. Exploiting the tree structure of SST-5, I generated all the subsentences represented by each internal node and obviously the root node. The result is a bigger dataset containing

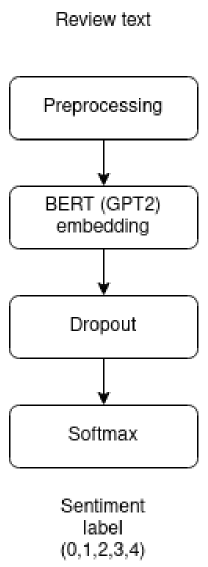

Figure 4: Architecture of my classifiers

all the subsentences and respective node labels.

Embedding Entries in the dataset are tokenized and transformed in numerical vectors using pre-trained BERT or GPT2. This phase is different in the two models, due to their features.
In the BERT model the sequence is padded to the left with the special token *[PAD]* until it reaches the same length of the longest sequence.
In GPT2 model I've chosen to pad at the right of the sequence because of its Masked Self-attention layer. This is one of the key differences in the Attention mechanism between the two models. The masked Self-Attention layer hides information from tokens that are to the right of the position of the current token in input. Instead, the common Self-Attention layer in BERT doesn't operate in this way.
Due to the fact that the GPT2 vocabulary doesn't provide a standard padding token, I replicated the end-of-sequence token until the sequence reaches the right length.
Together with the sequence, it's provided in input an Attention Mask vector. it's a simple binary vector with the same length that indicates to the model which tokens belong to the padding and which to the sequence.

Classification To perform the sentiment classification two layers have been added. These layers are a fully connected layer for the classification with Softmax activation function and a dropout layer to prevent overfitting.
This block of layers is not pretrained and weights are initialized randomly.

5 Experiments

Above I cited the most important works on this task and some of these confirmed partially my hypothesis of an improvement of performances with Transformer architectures.
Starting from the BERT classifier and the GPT2 classifier, I compared one to each other in accuracy performances on SST-5 and SST-2, both for all nodes and only root node. Moreover, I reported results published in the related works.
To make a fair comparison between different Transformer architectures, I tried to mantain a similar number of parameters for both GPT2 and BERT. I tested the small version of GPT2 that has similar number of weights to the base version of BERT (about 110M).
Huge models like BERT and GPT2 are very difficult to train and slower to produce the output than other models, so I tried to observe the variation of performances in relation to the number of layers and parameters. For this reason I tested the "distilled" version of these 2 models, a smaller architecture that has 40% less parameters than base version.
The last experiment I tried, is a training phase with the Yelp-5 dataset cited above. I trained GPT2 first with 100000 reviews extracted form Yelp-5 and then I finetuned the model with SST-5.
In Table 1 I reported performances of all models tested. For each specific task it's marked in red the best performance, in orange the best alternative and in yellow the third highest result.

Model	SST-5		SST-2	
	All	Root	All	Root
RNTN	80.7	45.7	87.6	85.4
BERT$_{BASE}$	83.9	53.2	94.0	91.2
BERT$_{LARGE}$	84.2	55.5	94.7	93.1
RoBERTa$_{LARGE}$	-	59.1	-	-
myBERT$_{BASE}$	83.6	55.5	87.3	92.7
myGPT2$_{SMALL}$	83.1	56.2	85.1	93.2
myGPT2$_{SMALL}$(yelp)	82.5	56.7	85.7	92.9
myDistilBERT	82.3	53.8	84.4	91.1
myDistilGPT2	82.2	54.2	84.8	90.5

Table 1: Accuracy performance in % for each model tested

As it's possible to see each model performance is very close to the others. Starting from the accuracy in SST-5 considering only root nodes, the large version of RoBERTa is the current state of the art with 59.1%. My GPT2 models are the best alternative but we have to consider the difference in size between the two models. RoBERTa large has a triple number of weights, so I can assume that a bigger version of GPT2 could be closer to RoBERTa performances.
We can also notice that GPT2 seems performing better than my BERT and the BERT published in 2019, indeed myGPT2$_{SMALL}$ outperforms BERT$_{LARGE}$ with less than 70% of parameters. Continuing considerations about the number of weights, it's possible to see that distilled versions are very close to the base version, so they are a really good alternative in case of low computational power to train models or to compute outputs faster.
As I mentioned before, I tried to train GPT2 adding 100000 reviews from another dataset (Yelp-5) before the finetuning with SST-5: This training method has improved performances only in the root case. We can interpret this result as an improvement due to length of Yelp-5 reviews. It seems that with this training method GPT2 is more able to predict labels on longer sequences but this not helps when we analyze all tree nodes, because a lot of sequences are very short.
Considering all BERT architectures and all GPT2 architectures we can notice that BERT performs quite well in short sequences like when we consider all nodes, instead GPT2 outperforms BERT in long sequences such as the root node.
Last considerations are about SST-2. Performance of my models in SST-2 considering all nodes are quite lower than others. This result can be explainable due to the model structure. These classifiers are not built for a binary classification but I simply converted predicted labels to their natural binary representation. We can assume that it's possible to obtain better results with a different output layer for binary classification and training with SST-2 directly.

6 Conclusion

The fine-grained Sentiment Analysis is another task in which Transformer architectures reached the state of the art. We can see that all of these models outperformed RNTN, the original recursive neural network made by authors of SST-5 dataset. All of Transformer architectures perform well, so the natural conclusion is to underline the common feature of these models, the Attention mechanism to perform this task. Attention permits model to focus on particular tokens in each during each step, so these architectures are capable to capture context and linguistic properties. Huge language models trained on big corpus can be finetuned for various tasks so they are the starting point for a lot of NLP tasks. Also distilled models showed their scalability in terms of size, indeed we can obtain similar results with smaller models. Instead, if we consider only performances we can repeat Transformer blocks creating bigger stacked models able to slightly increase accuracy. Another important conclusion from the Yelp-5 experiment is that we don't need a huge dataset for the finetuning phase. Transfer learning has the advantage to teach the model linguistic rules and embeddings in the pretraining phase, in which is sufficient a normal and easy to find text provided in input, with the condition of a large size. Then, it's possible to finetune with a pretty small dataset compared to the one used before.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[3] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[4] J. Wu J. Chuang C. D. Manning A. Ng C. Potts RR. Socher, A. Perelygin. Recursive deep models for semantic compositionality over a sentiment treebank. 2013.

[5] Manish Munikar, Sushil Shakya, and Aakash Shrestha. Fine-grained sentiment classification using bert, 2019.

[6] Zijun Sun, Chun Fan, Qinghong Han, Xiaofei Sun, Yuxian Meng, Fei Wu, and Jiwei Li. Self-explaining structures improve nlp models, 2020.