INF554 - Data Challenge report
Extractive Summarization with Discourse Graphs

Matthieu Souda
Dimitri Delpech de Saint Guilhem

December 2023

Introduction

The goal of this data challenge was to build an extractive summarization system that predicts whether each utterance of a dialogue transcription should be kept in a summary. It opposes to abstractive summarization that uses generation of new sentences and not pure extraction. This challenge allowed us to study and implement various supervised machine learning techniques, we particularly focused on neural networks as it seemed more suited according to us.

1 Feature extraction and selection

The data provided was composed of dialogues. For each dialogue, we had a `JSON` file that gave the list of the utterances and a text file that gave the discourse link between the utterances. Each utterance was composed of the speaker, the text and the index of the utterance.

In the first place, we used different tokenizers to encode the text of the utterances, by replicating the baseline. We used `BERTTokenizer` from transformers to encode the text composed of the dialogue text and the speaker, or just a standard vectorization layer. It was then fed directly to the model whose first layer was an embedding layer.

As we could see an influence of the types of links on the labels in the conversation, we tried to include them. We were expecting a rise of F1 score but it wasn't the case. Each type was encoded by a vector in $\mathbb{R}^{16}$ with each dimension representing a class of type.

We exploited the graph structure as well, each node represented a text line from the inputs and the edges the links of the discourse graph. However in this case we didn't use edge attributes in the graph.We will explain in the model part, how we exploited the graph.

In this case, for the training or the testing we wanted to feed our model, one conversation at a time with the corresponding graph. However, because of technical difficulties, due to limited memory access on CUDA, we had to decompose the texts and graphs of each document into batches. So we had extracts from the conversation, it is a shame because the goal was to feed the whole conversation so the model could acknowledge all of it.

2 Used models

As a major part of the problem concerned text classification, we mostly focused on neural networks models who have proven great results on the matter.

BERT-based models

Looking for great language classification models, we found BERT [1]. It stands for Bidirectional Encoder Representations from Transformers. The transformers have shown great results on language processing tasks.

We tried several models based on BERT. First we tried to add a MLP just after the embedding done by BERT, using the first token of the returned vector which is the classification token of the result of BERT. We also tried to use a LSTM layer on this same layer and tried to combine it with a MLP taking the types of the utterance (link type from the previous sentence) as an input.

Then we considered using BERT as a sentence embedding layer. As advised [1], for each sentence, we averaged the word embedding returned by BERT. We then applied different layers : a LSTM using sequence of the embedded sentences. a Graph Convolutional Network (inputs : the sentence embeddings as node features and the adjacency matrix of the subgraph of the concerned sentences). Inspired by DiscoBERT [2].

Models based on classic Transformers

Apart from BERT, we also tried to re-create the classic transformer encoder structure from scratch, to gain flexibility, based on the Encoder described in [3].

The first tasks of the Encoder is to positionally encode the vectorized text given as an input. It allows us to make semantic links between words inside a sentence. Then it goes through an Embedding layer.

After that, we used a multi-layer encoder, each Encoder layer being composed of 2 MultiHead Attention layers (using self attention for the first) and a point-wise-feed-forward layer, each connected by an Add and Norm regularization.

At the end of it, the embedded sentences are averaged to get one token per sentence, and a classifier with softmax activation is used to actually make the classification.

Bidirectional LSTM

A simpler architecture we used was using an embedding as well, and then a number of stacked Bidirectional LSTM Layers, that are commonly used in NLP tasks. It allows for reading sentences in both directions and trying to get the context of the different words. We can also process their output with a classifier with softmax activation.

Training, tuning and comparison

Training

Datasets

We divided the training dataset we had into a training set and a test set, so we could evaluate the models without Kaggle submission and without the bias it induces.

We tried to train some models on one entire conversation at a time but we were limited in GPU memory so we had to divide them. We also tried to deactivate the shuffle option in the data loaders so that the batches given in input would be sequences of sentences. We also performed some cross validation, but it didn't significantly improve the submissions, as overfitting was not really a problem.

Loss functions

At the very beginning, we tested several classic classification loss functions on our first neural network models. Only the binary cross entropy gave convincing results when we evaluated the F1 score. Furthermore we designed a "continuous version" of F1 score as loss function in order to maximize F1 score. It makes it differentiable and so we used it as well. Here is how it is computed, with $y_{\text{true}} \in \{0,1\}$ discrete and $y_{\text{pred}} \in [0,1]$ continuous. We have :

$$\begin{array}{ll} TP = y_{\text{true}} y_{\text{pred}} & TN = (1 - y_{\text{true}})(1 - y_{\text{pred}}) \\ TP = (1 - y_{\text{true}}) y_{\text{pred}} & TP = y_{\text{true}}(1 - y_{\text{pred}}) \end{array}$$

And we still have the precision and the recall being (continous version):

$$p = \frac{TP}{TP+FP} \quad r = \frac{TP}{TP+FN}$$

Then we have

$$F1 = \frac{2rp}{r+p} r$$

Additional variations on BERT training

We tried some variations with the models based on Bert where we would use a pre-trained version of BertModel and then train the whole model. Then after a few epochs, we would freeze the BertModel layer of our model and continue with the training of the GCN or the LSTM. The first results were not impressing and we faced technical difficulties that prevented us from further testing.

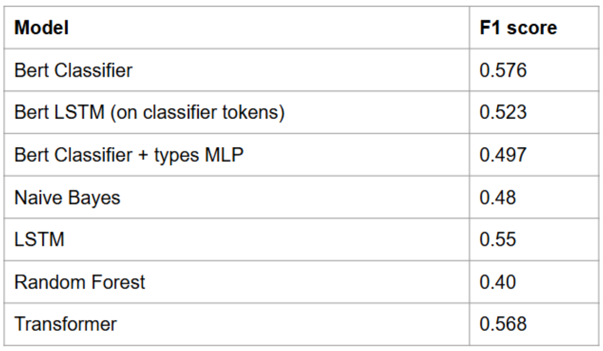
Hyperparameters tuning

The hyperparameters of the models we tuned mostly were the hidden dimensions of the layers, the size of the different layers (heads in multi head attention layers, size of LSTM layers, ...), the number of layers and the embeddings sizes. Regarding the training, we tuned the learning rate, the optimization algorithm, the loss criterion, the batch size and whether we froze some layers or not.

In order to select the best methods, we would train a model with some various parameters settings and keep the ones with higher results.

Results comparison

In the end, we could not test DiscoBERT. We also added some baselines (Random Forest, naive Bayes classifiers, ...).



Model	F1 score
Bert Classifier	0.576
Bert LSTM (on classifier tokens)	0.523
Bert Classifier + types MLP	0.497
Naive Bayes	0.48
LSTM	0.55
Random Forest	0.40
Transformer	0.568

Figure 1: Comparison of different approaches based on F1 score

The selected model for the data challenge submission is then the original BERT classifier, fine-tuned on our data. LSTM and classic Transformer approaches have got around the same level of performance. An idea to improve the performance would be to combine our different approaches (with a majority voting system for example). We could not go further due to the imposed time limit.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

[2] J. Xu, Z. Gan, Y. Cheng, and J. Liu, “Discourse-aware neural extractive text summarization,” 2020. [Online]. Available: https://arxiv.org/abs/1910.14142

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2023. [Online]. Available: https://arxiv.org/abs/1706.03762