March 31, 2024

INF582 KAGGLE REPORT

Auguste CRABEIL, Valentin DORSEUIL and Théo GABORIAUD

SUMMARY

INTRODUCTION

In today's digital era, the important volume of news articles published daily poses a challenge for readers to navigate through relevant content efficiently. Compelling headlines play a crucial role in capturing readers' attention and influencing their decision to engage with an article. This challenge addresses this need by tasking participants with developing advanced NLP models to automatically generate informative titles for news articles.

In this report, we outline our approach to this challenge, detailing the methods used and insights gained from our exploration of this problem space. As requested, we did not use any external data or models except for embeddings. Our approach was organized around this constraint and the excessive difficulty of training a language model from scratch with so little data (and the computing time it would require).

We tried different methods for this challenge and we decided to present them in separated notebooks, one for each method.

1 CONCEPTUAL FRAMEWORK

1.1 Relevant NLP Domains

As explained in the introduction, the problem is finding a title for an article. The first idea is to find a sentence that sums up the text. This brings us to a branch of NLP called "text summarization".

The first method presented in the baseline is to take the first sentence, which is supposed to contain the most important information in the text. The score of this method is 0.15. But the baseline also shows that if you select the right sentence in the text you can get 0.31. Most of our next ideas were inspired by this result, and we tried to select a sentence from the text that gave a good score. In this sub-section, we will present some simple NLP methods to find the best sentence with unsupervised algorithms and the code can be found on **unsupervised_methods.ipynb**. The first idea is to use tf-idf. In text summarization, TF-IDF can be employed to identify significant terms or phrases within a document. Sentences containing terms with high TF-IDF scores are selected as they encapsulate the essential information of the document. This gave better results than choosing the first sentence but with this method, only the words are taken into account, not the sentences as a whole.

We then looked at other methods such as textrank, which this time considers text as a graph and therefore better represents the concept of sentences. TextRank is an unsupervised algorithm used for automatic text summarisation and keyword extraction. It represents text as a graph, where phrases or words are nodes and edges indicate relationships such as co-occurrence or semantic similarity. By applying graph-based ranking algorithms such as PageRank, TextRank assigns importance scores to each node, indicating their importance in the document. These scores are then used to identify phrases or keywords that represent the main ideas or topics of the text. To vectorise the sentences we used Hugging Face's 'distiluse-base-multilingual-cased' embedding because this embedding is trained on many languages, including French. See the reference.

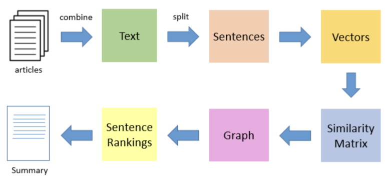

Figure 1: How textrank works.

1.2 The chosen pipeline

After reviewing the methods that might have been useful to us, we had to come up with a pipeline that would take us from the article to the predicted title. The figure below shows our idea.

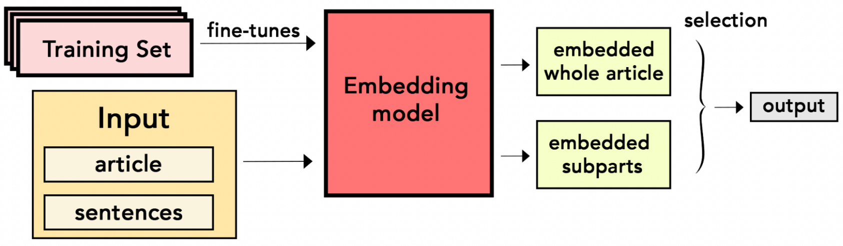

Figure 2: Our pipeline.

The idea is to first use the training set to find an embedding model, and then use this embedding model to predict what the title is going to look like. Then, the most relevant part of the article should be extracted and then recombined in our prediction for the title.

We present in the next section what we chose for the embedding model, for its finetuning and for the selection of the best subparts of the article text.

2 Our model

2.1 Our trials for the embedding model

2.1.1 • Pre-trained models

As we were authorized to use pre-trained embedding models, we tried to use some powerful ones to help us predict a good title. After a deeper look into the validation dataset, we found that often the titles were quite long and very descriptive, almost like a small summary. Given that observation, our main idea was to find the sentence in the article that summarizes the best the whole article. To do that we used embeddings model to embed first the whole text and then each sentence separately to find which one was "nearest" to the text, using the cosine similarity on embedded vectors.

By testing different embedding models like BERT or GPT, we finally reached a score of 0.18 on the validation dataset with the model MiniLM-L12-v2. This pipeline was not using the train dataset.

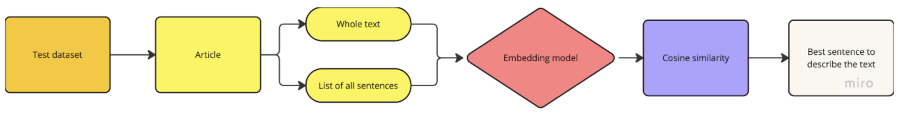

Figure 3: Pretrained model pipeline.

2.1.2 • Fine-tuning the model : in-place method

After that we did the same thing but before computing the embedded sentences, we tried to fine-tune our embedding model. For that we used the train dataset where we have already generated titles. We updated the weights of our embedding model with the objective of the embedding of the whole text to be nearer the embedding of the titles associated. With this idea we were hoping that the embedding model would be more accurate in choosing the right sentence to describe the text “like the article”. For that we used the “multiple negatives ranking loss” that is suited for fine tuning when we don’t have contradiction labels. This is the method that worked the best for us, it helped us improve the score on the validation dataset to 0.195.

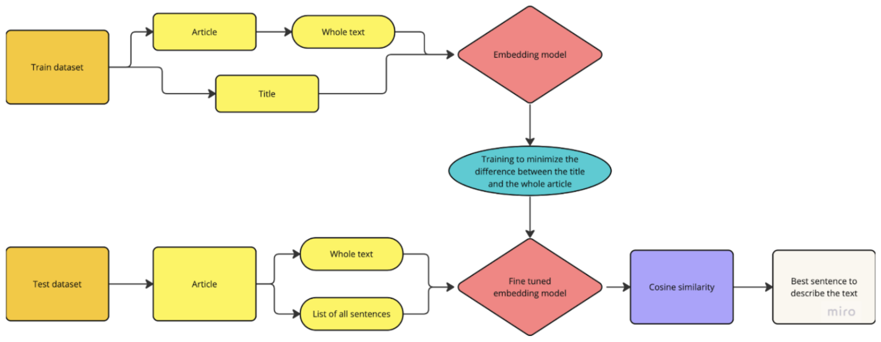
Train dataset
Article
Whole text
Title
Embedding model
Training to minimize the difference between the title and the whole article
Test dataset
Article
Whole text
List of all sentences
Fine tuned embedding model
Cosine similarity
Best sentence to describe the text
miro

Figure 4: Inplace model pipeline.

2.1.3 • Fine-tuning the model : adding layers

We also tried to do the same thing but instead of changing the already trained weights of the embedding model (which was very long in computation due to the large number of parameters), we tried to add some layers (a siamese network) after the embedding to do the same task. With that approach we need to calculate the embedding of all the articles only once and we have fewer parameters to train. But it worked a little less well for us with an improvement only to 0.17 on the validation dataset.

These 3 methods are presented in the **embedding_methods.ipynb** file.

2.2 The selection method

Now that we can predict an embedding for what the title of our article would likely look like, the question is what to do with it. First, as we are not able to use text generation methods, the most natural idea is to search for the whole sentence that maximizes its cosine similarity with the predicted embedding. This is the method with which we got the best result and using which we submitted our data. However, we’ve tried another approach, and we feel like it is not optimal to search for a whole sentence, as only subparts may be relevant as a title.

The other idea is to look for the k most relevant n-grams among the whole article text, and then to concatenate them to make our title. This idea comes from the fact that the aim metric is a F1-score, thus to some extent considering the precision. Building our title, we want the least possible words not to be relevant. We ran a grid search searching for the best (k, n) to make this work. What the result tells us is that using 1-grams is the best for this technique. However, we face the problem that this method is not adaptive to the distribution of score that we get. One way to improve this idea would be to select n-grams of different sizes, according to the distribution of the best similarity scores. See **selection_method_choosing.ipynb** for the detail.

3
SOME OTHER METHODS

3.1 Sec2sec models

In this section we will present the models that are currently at the forefront of all these text summarization tasks: the Seq2seq models. Seq2seq models summarize texts by capturing the essence of the text's meaning. These models encode and decode texts into output sequences, generating concise summaries that retain key information. By taking advantage of attention mechanisms, seq2seq models can prioritize important parts of the input text for inclusion in the summary, improving its coherence and relevance. In addition, training seq2seq models on large corpora of textual data allows them to learn the patterns and structures inherent in summarisation tasks, improving their ability to produce accurate and informative summaries.

Today these models are the best for these tasks, so we've tried to see what we can do with these complex models. We took the code from TD5 (see file **seq2seq.ipynb**) and we also tried to adapt other simpler models (such as here). We managed to get these models running. But as you might expect, our computing power and computer memory are not sufficient to train such complex models. We did, however, try to optimize the RAM and code a little, but the maximum results were repetitions of the same few words. Despite these inconclusive results, it has been possible to work on these complex models and to gain a deep understanding of their mechanisms, which today have produced remarkable results for text summarization and translation for example.

3.2 Rephrasing the problem into a classification task

In this part all the texts and all the sentences of these texts have been embedded using Hugging Face's 'distiluse-base-multilingual-cased' embedding because this embedding is trained on many languages, including French.

Knowing that our computers are not powerful enough for complex text generation models, we tried to transform this generation problem into a classification problem. We tried two main ideas (which can be found in the file **classification_methods.ipynb**). Firstly, say that a word is 'important' if it is in the title. We can then label all the words in the training texts. And we train a model which, based on the texts, tries to predict the "important" words. The idea afterwards is to reform a sentence from these words. This model requires a vector of the size "number of words" and was therefore very complicated to train and stabilize. We then tried a much more promising method. We labelled 1 the sentence in the text that was closest to the title (in terms of red score) and all the others 0. The second model then took a text and a sentence from this text and had to indicate whether this sentence was labelled 1 or 0. So we built a model that indicates how close a sentence is to the title. The models tested here are mainly MLPs and CNNs. More complex models containing RNN took too much time even if they would have been interesting in this context of sentences and text. This second method trains very well and allows you to go from a Rouge score of 0.14 to 16.5 in about a hundred epochs. But the training started to get really long. We think that by optimizing this model and combining it with other methods we can obtain good results for finding sentences close to titles.

CONCLUSION

In conclusion, our exploration of methods for automating news article title generation has provided valuable insights into the complexities of Natural Language Processing (NLP) tasks. From simple approaches like TF-IDF and TextRank to more advanced techniques such as fine-tuning embeddings and exploring Seq2seq architectures, we have experimented with a diverse array of methodologies. While some methods showed promise, others presented challenges, particularly in terms of computational resources and model complexity.

Moving forward, we believe that integrating insights from various approaches and leveraging advancements in NLP technologies will be key to developing more effective and efficient automated title generation systems. We are grateful for the opportunity to participate in this endeavor and look forward to contributing further to the advancement of NLP in real-world applications.