NLP in Industry — Final Project
Statement Attribution in News Articles

Lina Conti and Isaac Murphy

December 4, 2022

1 Introduction

One of the goals of news analysis is to find out what politicians and other public figures are saying in newspaper articles or other similar texts. To do this, it would be helpful, as a first step, to have a program which, given a paragraph, could identify who (if anyone) is expressing an opinion in it. Humans can do this fairly easily, but manual annotation of the paragraphs is expensive and takes a long time. Given these incentives to automate the task, we tried to train a model for it.

2 Reference corpus

2.1 Original task

The first step in order to achieve this was to establish a reference corpus, with examples of news paragraphs and the expected output for each of them. This corpus could then be used to train and evaluate the models.

We used the corpus of French news articles and the script to split them into paragraphs that were provided. NLP systems are generally not very good at handling long sequences of text such as a full article. By looking at the data, we can also notice that, when someone expresses an opinion, this is often done over multiple sentences — which means looking for the speaker at sentence level would not work. So, using paragraphs as a basic unit seems like a good compromise.

Paragraphs were then annotated manually according to the following schema: if a person was identified as presenting an opinion in the paragraph, the paragraph was annotated with their full name. Multiple speakers could be attributed to the same paragraph, and likewise it was possible that no speaker was attributed to a paragraph.

The annotation task was shared across the class. 871 paragraphs were annotated in total. Some annotations contained mistakes, for example, definite descriptions such as 'un proche de Marine Le Pen' were added as speakers and had to be removed. Duplicate paragraphs were discarded. This left us with 790 paragraphs. [1] As mentioned before, NLP systems generally struggle with long texts. Left-to-right LSTMs for example tend to 'forget' the beginning of the input if it is too long. To avoid running into this problem, we removed paragraphs longer than 200 words. In the end, we had 688 paragraphs in total.

[1] The cleaned version of our reference corpus data can be found here.

2.2 Simplified task

The initial goal of the project was to identify the speaker of a paragraph whether they were directly mentioned or not. When manually annotating articles, we attributed the statement to a speaker even if his identity was only stated in a previous paragraph and was unobtainable from the paragraph itself. This approach was chosen because the goal of the reference corpus was to show what a human would do for this task. However, if the goal had been to provide the best possible training data for our models, it would probably have been best to annotate the paragraphs out of order and not to consider those speakers.

Another case in which a paragraph could be attributed to a person whose name did not appear in the text was when a definite description and some basic knowledge of French politics made it possible to guess who the speaker was. For example 'le locataire de l'Elysée' indicates Emmanuel Macron. In one paragraph, 'Le président de l'Assemblée Nationale' was annotated as referring to Richard Ferrand. This last example shows just how slippery this approach is, since, in June 2022, this became Yaël Braun-Pivet.

For our automatic system to identify the correct speaker even in these cases would require it to have more than a knowledge of Natural Language and would greatly complicate the task. For some of the models we trained, we decided to simplify the task, only identifying speakers which were directly mentioned in the paragraph. We believe this to be a defensible choice because keeping our expectations realistic makes it a lot more likely that they can be achieved by a relatively simple NLP system and comes at only a small cost. Indeed, there are only a few paragraphs were this makes a difference. In the 688 paragraphs we worked with, only 39 were attributed to someone whose name was not present in the text. Upon closer look, we can even see that the vast majority of them fit in the first case: the speaker was mentioned in a previous paragraph. Identifying the speaker in this case would require a change in the way articles are split into paragraphs, which was not in the scope of this project. So by simplifying the task we are mostly leaving out examples that were unsolvable in any case.

3 Models

Due to the complexity of the problem, we decided to avoid a rule-based approach. Natural language is complex and designing effective rule-based algorithms takes time and skill. Instead, based off the successes achieved by neural networks for other NLP tasks, this is the path we decided to follow.

3.1 Bi-LSTM Mono-label Classifier

Since the input to our model would be paragraphs, we chose to use an LSTM, a good system for handling sequential data. LSTMs can compute contextual representations of the whole sequence and of each token in the sequence. In both cases, the representations combine information from all tokens in the sequence as well as from the order in which they appear, which is important in Natural Language Understanding.

We decided to start by building the simplest possible system using an LSTM. This system could then be used as a baseline to check that further modifications really improved performance. Given that our examples consist in pairs of paragraphs and speakers, the most obvious strategy was to use a classifier that takes paragraphs as input and outputs the predicted speakers. Given that multi-label classification is more difficult than single-label, we decided to try to predict a single speaker for each paragraph. This first coarse approximation would allow us to assess whether the classifier path was promising or not.

Our first attempt was therefore a many-to-one classification system using an LSTM. [2] In order to train such a system, we modified our data such that paragraphs with multiple attributions were turned into multiple examples, one with each attribution as gold class. We then split our annotated data into train and test sets. Again, with the goal of keeping our system as elementary as possible, we tokenized the paragraphs simply into words using the `polyglot` library. The input vocabulary was that of all words present in the training paragraphs and the output vocabulary was that of all speakers present in the training examples, along with the label `None` for when no one was expressing an opinion. At inference time, words or speakers not present in the training data were replaced with the special `<unk>` token.

The model contains a dictionary of randomly initialized word embeddings for representing the input vocabulary. These representations are trained along with the model. Tokenized and embedded paragraphs go through a bidirectional LSTM. The encoded representation of the paragraph output by the LSTM goes through a linear layer followed by the softmax function. Thus, the output contains a probabilistic score for each label in the output vocabulary. The label with the highest score is predicted for that paragraph.

Having a limited output vocabulary is a serious limitation to this approach. Indeed, our train set is quite small, so not all French political figures will be present in it. Those who are not cannot be identified as speakers. Out of 154 speakers in our data, only 137 are present in the train set. So even in our small test set, there are already 17 new speakers who cannot be accounted for.

It was therefore not surprising that the results achieved with this first model were unsatisfactory. After training, the accuracy on the test set is of only 44.80%. This is less than the accuracy for a majority class baseline. Indeed, a system that always predicted the majority class `None` for every input would achieve an accuracy of 51.45% on the same test set. The test accuracy varies a lot between iterations This model was barely learning at all, given that after the first epoch its validation accuracy was of 48.60% and after the last epoch it was not much higher, at only 49.24%. Thus, we decided not to pursue this approach any longer and to look for a different system, that was not a classifier.

3.2 Vanilla Bi-LSTM Sequential Classifier

Instead of classifying whole paragraphs, we decided to take inspiration from the Named Entity Recognition (NER) task. This consists in identifying the named entities (persons, organizations, places, etc.) in a given piece of text. It is usually done using the inside–outside–beginning (IOB) tagging system. Each token is labeled with either `inside` (`I`) if it belongs to a named entity, `outside` (`O`) if it does not, or `beginning` (`B`) if it is the first token of named entity and the previous label was `I`. [3]

The attribution task is similar to NER, only we restrict ourselves to tagging named entities of a specific type — persons — and, among those, only the ones that are expressing an opinion. Hence, we decided to create an IOB tagger for speakers based on a bi-LSTM. This implied a slight simplification of the problem, as we could no longer hope to identify speakers who were not explicitly mentioned in the text. But, as explained in section 2.2, this is a reasoned choice and has a relatively low cost in terms of coverage of our annotated data.

In order to train such a model, we had to modify our set of examples. Instead of pairs of the form (tokenized paragraph sequence, speaker label), we needed (tokenized paragraph sequence, IOB tags sequence) pairs, with both sequences having the same length. To choose the appropriate tag for each token in the paragraph text, we looked at whether that token was present in speaker annotation. If it was, the tag was `I` or `B` and if not, `O`.

[2] The code for training the monolabel model is available here.

[3] The code for the vanilla sequence classifier is available here.

The model architecture was quite similar to the previous one, only instead of using the hidden state of the LSTM after processing the whole paragraph, we used the hidden state after processing each token in the paragraph sequence. Each hidden state goes through a linear layer and the softmax function is applied to the result, as previously. However, for this model the output layer is only of size 3 and the output vector contains the score for each label `I`, `B` and `O`.

The results achieved by this model were significantly better than the previous one. The accuracy on the test set is of 82.70%. This is not directly comparable to the accuracy of the previous model, since their outputs are not of the same form. This accuracy is still lower than the majority class baseline. For this test set, a system that consistently predicted `O` for every token would achieve an accuracy of 98.51%.

On a positive note, we can at least be certain that this model is learning, since its accuracy on the validation set goes from 59.39% on the first epoch to as high as 98.76%. So we decided that the IOB tagger approach was worth pursuing.

3.3 Bi-LSTM Sequential Classifier pre-trained for Named Entity Recognition

The scarcity of training data is a major hindrance to good performance. So, we thought pre-training our model on a related task for which more data is available could be beneficial. As mentioned before, we took inspiration from NER tagging to create our attribution tagger because of the similarity between the two tasks. Attribution tagging can even be seen as a sub-type of NER tagging. Pre-training our model with NER tagging seemed reasonable in these conditions. [4]

We used [Dup19], a publicly available French corpus for NER, for pre-training. The fact that the annotated texts in this corpus are also news extracts made it particularly appropriate for our purposes. However, the format of the examples had to be changed to match the downstream task of attribution. The articles had to be split into example paragraphs. Since this was only a pre-training task and not our end-goal, we decided to use a simpler approach and just break every article into segments of a standard length. We chose a sequence length of 100 tokens, which is close to the average length of paragraphs in our speaker attribution corpus. 100 tokens is much longer than the average sentence length in French, ensuring that our model will see similar text structures in pretraining and fine-tuning, while still being short enough that an LSTM can properly handle the information.

In preprocessing, we tokenized the articles from the NER corpus the same as for our speaker attribution data, then applied the IOB tagging schema. Since the goal for our final system was only to apply speaker attribution to persons (ignoring organizations, countries, etc.), we made the decision to discard all named entity labels except `person`. In the end we were left with 2792 sequences for pretraining, and an additional 349 examples of NER data for both validation and testing.

Since the task was functionally the same as for the vanilla sequence tagger, we were able to use the same model architecture. The model was trained first on the NER data until it had reached a reasonable level of performance, and then fine tuned on our corpus of speaker attribution data. This resulted in significant gains in performance compared to the vanilla implementation: the accuracy on the test set was 99.0%, higher than the majority class baseline, which was 98.80%.

4 Evaluation

In order to have a more fine-grained evaluation of our models, we computed precision, recall and the f1-measure for each of them. The results are reported in Table 1. As mentioned before, we compared

[4]The code for the pretraining and fine-tuning is available here.

Classifier type	Accuracy	Accuracy baseline	Precision	Precision baseline	Recall	F1
Mono-label	44.80	51.45	25.00	48.55	12.24	16.43
Vanilla sequential	82.70	98.51	99.25	1.50	65.09	78.62
Pre-trained sequential	99.00	98.80	98.76	1.20	98.60	98.68

Table 1: Performance metrics for our three different models for statement attribution.

each model's accuracy to that of a system that would simply guess the majority class in every case (`None` for mono-label classifiers and `O`) for sequential classifiers. The only model to perform better than the baseline in terms of accuracy was the sequential classifier pre trained on NER tagging.

Given that the baseline accuracy is already very high for the taggers, it is difficult to measure the extent of the improvement achieved by the last model. So, we computed a baseline precision too to measure our results against. The baseline precision is the one theoretically achieved by a model that would randomly guess for each example between the negative class (`None` in one case and `O` in the other) and the positive class (all other classes, which means a speaker is detected). In this case, the retrieved elements are a random subset of the test set. Since the subset is random, the proportion of positive elements in the subset should be similar to the proportion of positives in the entire test set. So precision would theoretically be the proportion of positive examples in the test set. Using this definition, both sequential classifiers perform significantly better than the precision baseline. However, the pretrained version is the only one that achieves a good recall too and therefore a good f1-measure (98.68% for the pretrained model against 78.62% for the vanilla model).

5 Conclusion

As discussed above, being able to accurately identify when a speaker is making a claim is an important part of news analysis. While this task is straightforward for the average person, for practical application it is made difficult by the sheer volume of news data which is constantly being produced. Given this large amount of data, an automatic system for labelling speaker attribution could be very useful for any news analysis task. Using pretraining on the named entity recognition task in combination with fine-tuning on manually labelled data, we have successfully trained a system which performs with both high precision and recall on the task of identifying speaker attribution. Specifically, with precision of 98.6, we believe this system is good enough for industrial application.

For possible improvements and areas to expand this work, the first and most obvious is increasing the size and quality of the labelled speaker attribution data. Although manually annotating data is a time-consuming process, this would certainly improve the quality and robustness of the model. Additionally, it would be interesting to measure inter-rater agreement on this task, since correctly identifying speakers can be a difficult task at times even for humans. This could help put our models performance in a more clear perspective. Concerning the model itself, there are numerous techniques in deep learning to improve the performance of NLP systems: dropout, parameter regularization, and subword tokenization methods to name a few. Any of these could have a positive impact on performance and may be worth pursing, but it may be more interesting to compare the LSTM model to a different architecture, such as a transformer. Fine-tuning transformer models like BERT has been shown to reach very good performance on a variety of NLP tasks, and few-shot learning with large language models like BERT or GPT3 has also shown promising results. Exploring the performance of one of these strategies could be an interesting area for future work.

Acknowledgement

We would like to thank Professor Benoît Crabbé for providing the framework for the code of the original Bi-LSTM NER tagger and for his valuable advice.

References

[Dup19] Yoann Dupont. Un corpus libre, évolutif et versionné en entités nommées du français. In *TALN 2019-Traitement Automatique des Langues Naturelles*, 2019.