Team 42

Text-based sentiment analysis

an UmBERTo approach

AVALLE Dario
EURECOM
avalled@eurecom.fr

FONTANA Umberto
EURECOM
fontana@eurecom.fr

SORBI Marco
EURECOM
sorbi@eurecom.fr

SPINA Gabriele
EURECOM
spina@eurecom.fr

***Abstract*—Social media in recent years became the vector for spreading opinions and ideas. By monitoring the publications of social media and extracting the sentiment it is possible to get useful insights into public political opinion, customer service, and perform market research. But which part of the text lead to the sentiment description? In this report, we focused on detecting the meaningful span of the text that contains the strongest sentiment using the combination of ELMo architecture to perform word embedding and a Bi-LSTM to find the span. We then trained a RoBERTa model to deal with the sentiment analysis and compared the results obtained by using the original tweet, the true span, and the predicted span.**

I. Introduction

One of the most popular tasks in natural language processing is sentiment analysis, i.e. a classification task that aims to predict a user's sentiment. This task has many practical use cases, such as market research and improving customer service. Ambiguities in language, sarcasm, and context-dependent sentiments, represent still a significant problem in achieving high precision in sentiment classification.

Determining which are the most informative words about sentiment is crucial for the classification. This is known as the span detection task, which has enormous potential in practical applications and can help the model focus on the most influential aspects while filtering out irrelevant information.

In our work, we mainly focused on span detection. We extracted a context-dependant embedding for each word and we trained a bidirectional LSTM (Bi-LSTM) to predict the span, given as ground truth in the dataset. We then compared the performance of a pre-trained RoBERTa model [1] on the sentimental classification task using the original text, the original span, and our predicted span.

II. Data Exploration

The dataset is composed of tweets, each associated with a label representing the sentiment of the tweet (positive, neutral, and negative) and the span, i.e. a substring representing the portion of the tweet that is most representative for the purposes of sentiment analysis.

The provided dataset is divided into train and test sets, which consist of almost 25000 and 3000 tweets respectively. Approximately 8000 tweets of the training set are positives, 10000 are neutrals and 7000 are negatives. From this statistic, we can see that the labels are roughly balanced, with a more relevant presence of neutral tweets.

The average length of a tweet (both training and test) is approximately 69 characters while the average length of the span is approximately 37 characters. The distributions of lengths are shown in Fig. 1.

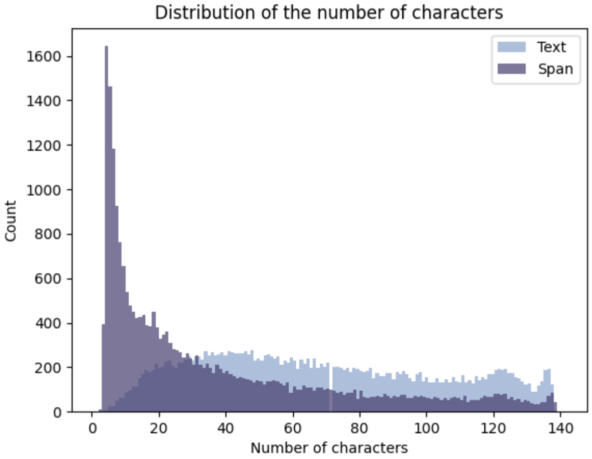


Fig. 1: Length distribution of tweets and spans

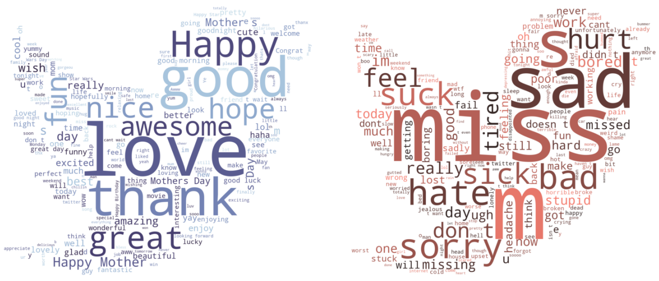
The training dataset was further split into train and validation sets with an 80-20 ratio, ensuring to maintain the original balance of labels.

An important factor is that sometimes the span does not consist only of whole words, but can also contain single characters, truncated words, and punctuation symbols. Fig. 2 shows the most common words in the spans of both the positive and negative tweets. Indeed, we can notice that some of those are just single characters.

III. Preprocessing

A. Data Cleaning

We applied some preprocessing techniques to clean the text and to make its format more standard. At first, we replaced HTML entities with their original character representation, then we removed all the '#' symbols, the URLs, and the user tags (@user_id), as they should not contain any information about sentiment and their presence could induce noise to the model. We also compressed the words having the same character repeated more than two times: as an example, we compressed the word "happppppy" to "happy". Finally, we trimmed the whitespaces and removed repeated punctuation.



(a) Positive spans

(b) Negative spans

Fig. 2: Word clouds of the most common words in the spans.

Using spell checker tools was taken into consideration to adjust some spelling mistakes that are frequent in the nature of Twitter; however using such automated tools introduced some errors, as some slang words were corrected into different words of the English dictionary, changing the context of the tweet and potentially its sentiment. We, therefore, decided not to use these tools.

B. *Word Embeddings - ELMo*

To detect the relevant span in the tweet, we embedded the sentences in a vector space and labelled each token of the sentence according to the available span in the dataset. We tokenized the tweet using the SpaCy NLP pipeline [2] and then we associated each of them with one of the binary labels 'S' (the token is part of the true span) or 'O' (the token is not part of the span). A context-dependant embedding for each token was extracted by using the ELMo model [3]. Its advantages are multiple, in particular:

1) ELMo word representations are functions of the entire input sentence with the possibility to model both the complex characteristics of word use and how these uses vary across linguistic contexts;
2) is pre-trained on a large corpus (approximately 30 million sentences) and is freely available in Python with the AllenNLP library [4].

At the end of this passage, we obtained, for each sentence composed of W words, a sentence representation $W \times D$, where $D = 1024$.

IV. Models

As a baseline model for the sentiment classification task, we relied on the RoBERTa transformer architecture [5] trained on $\sim$ 58M tweets and fine-tuned for sentiment analysis with the TweetEval benchmark [6]. Later, the same architecture has been fine-tuned on our dataset to be more task-specific.

For the span detection task, the model used was a simple Bi-LSTM module followed by a Dropout layer, a ReLU activation function, and a final linear layer which performs the label prediction of the input word.

The final workflow is the combination of the two tasks: the span detector is used to extract relevant spans from the tweets, which are then used to perform the sentiment classification with the RoBERTa model.

V. Experiments and Results

A. *Span Detection*

For extracting the span, after having created the embedding of the tweets with the ELMo model, we trained the span detector model with 10 epochs and a learning rate of 10^{-4}.

We observed that the prediction made by the span detector were not all contiguous, and the results consisted of sparsed words inside the tweets. To solve this problem, we included in the span all the words in between the two non-contiguous positive predictions, extending the length of the span. Other predictions consisted only of non-meaningful entities (such as *'it', '! !', 'I'*), or resulted in an empty span. We dealt with these situations by considering as span of the overall sentence.

We tested the model on the test set by measuring its performance using the Jaccard index (or Jaccard similarity coefficient) and the F1 binary score. The scores are, respectively, **0.587** and **0.652**. The examples in Table I are representative of the pitfalls of our model. In particular, the aggregation of separated spans leads to some very long sentences that include the true subtext.

TABLE I: Some examples of the extracted span from the tweets

Original Text	**Selected Text**	**Predicted Span**
We can't even call you from belgium suck	*m suck*	*suck*
Need a camera blower.. my camera censor is dirteeh..	*dirteeh..*	*.. my camera censor is dirteeh..*
Hello, yourself. Enjoy London. Watch out for the Hackeys. They're mental.	*They're mental.*	*, yourself. Enjoy London. Watch out for the Hackeys. They're mental.*

B. *Sentiment Analysis*

As mentioned above, we used a RoBERTa model for the sentiment analysis task. It was tested using both the entire text of the tweet and the span, either the one available as ground truth or the one predicted by using our Bi-LSTM. The baseline of this task is a pre-trained RoBERTa model without fine-tuning. Subsequently, for each experiment, we fine-tuned its parameters by training the model for 5 epochs, using the Adam optimizer with a learning rate of 10^{-5}. The cross-entropy loss has been used as optimization criterion.

The results are shown in Tab II, which reports the validation and test scores for each experiment. For the model fine-tuned on the predicted span, we display also its confusion matrix in Fig. 3, which shows that the model is roughly balanced in predicting the different classes.

TABLE II: F1-macro scores of the model trained in different ways

	VAL	TEST
Pre-trained (baseline)	0.723	0.714
Fine-tuned on text	0.809	0.791
Fine-tuned on true span	0.894	0.899
Fine-tuned on predicted span	0.808	0.788

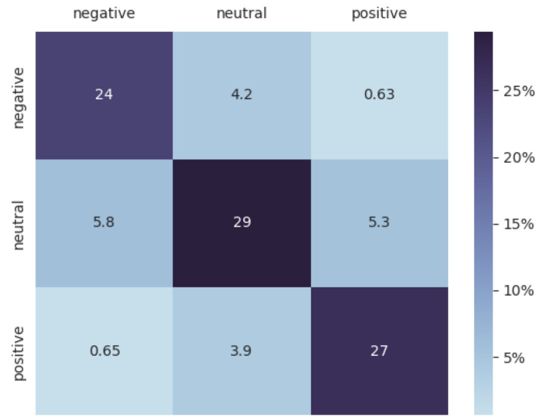


Fig. 3: Confusion matrix of RoBERTa fine-tuned on the predicted span, in percentage

VI. Conclusions

To understand the significance of individual tokens in the classification of sentences as positive or negative, we extracted the positive and negative scores of each token in a tweet. Fig. 4 shows the scores computed by the transformer-interpret tool [7], a Python library that relies on the PyTorch explainability package Captum [8] adapted for transformers. A positive value for the positive label indicates that the token likely contributes to classifying a sentence as positive, otherwise, if the positive score is negative, the token would negatively impact the overall positivity of the sentence. The same principle is applied to the negative label. It can be seen in the example that a word like '*excited*' is considered highly positive (high positive score, low negative score), while words like '*week*' can be considered more or less neutral.

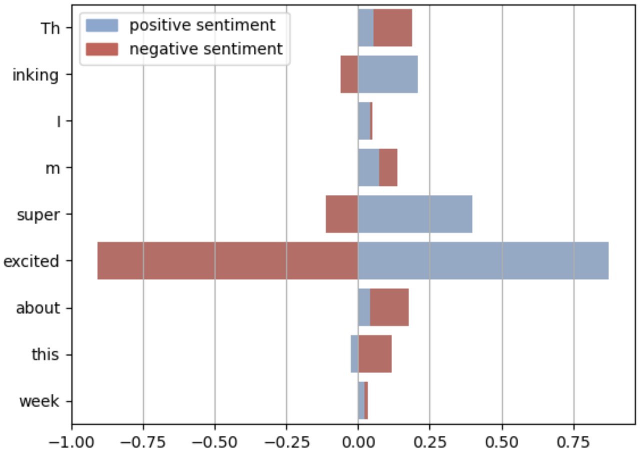


Fig. 4: Weight of each token in a tweet in favor of the positive and negative sentiments

In this example, the original selected text is "super excited", in fact these two words are highly positively oriented, and the span predicted by our model is "super excited about this week". This was an expected behavior, and we noticed this same pattern in all the tweets we analyzed using the tool.

It is clear that the span of the text plays a crucial role in the sentiment classification task, where using the true span to fine-tune the model outperforms the same model trained on the whole text. Our predicted span negatively contributed to the task, surely due to the poor performance of the span extractor model.

It can be seen that RoBERTa trained only on the true span outperforms the model trained on the whole text by about 0.1. This justifies our idea that span detection can actually bring improvements to the original task of sentiment analysis and motivates further research in this direction.

VII. Future Works

The span detector work with Bi-LSTM and ELMo embedding didn't improve the performance of the model and didn't manage to outperform the model trained on only text. N. Kim Thi-Thanh et al. implemented an architecture similar to ours with the addition of a Conditional Random Field (CRF) in [9]. CRFs are particularly effective in detecting sentence span because they can capture sequential dependencies thanks to the connection between inputs and outputs, unlike LSTM and Bi-LSTM networks where memory cells/recurrent components are employed. Moreover, looking at the solutions of both the 11th task of the SemEval2020 challenge and the Twitter Sentiment Extraction challenge [10] [11], it is clear that a better solution to detect subsentences is the combination of a BERT/RoBERTa model with a CRF. In future work, it would be interesting to compare the results obtained with these two architectures.

References

[1] L. Yinhan, O. Myle, G. Naman, D. Jingfei, J. Mandar, C. Danqi, L. Omer, L. Mike, Z. Luke, and S. Veselin, "Roberta: A robustly optimized bert pretraining approach," 2019.

[2] "SpaCy Python API." https://spacy.io/api.

[3] E. P. Matthew, N. Mark, I. Mohit, G. Matt, C. Christopher, L. Kenton, and Z. Luke, "Deep contextualized word representations," 2018.

[4] "Allennlp Python API." https://allenai.org/allennlp.

[5] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019.

[6] B. Francesco, C.-C. Jose, N. Leonardo, and E.-A. Luis, "Tweeteval: Unified benchmark and comparative evaluation for tweet classification," 2020.

[7] "Transformers Interpret Python Tool." https://github.com/cdpierse/transformers-interpret.

[8] "Captum, Model Interpretability for Pytorch." https://captum.ai/.

[9] N. Kim Thi-Thanh, H. Sieu Khai, P. Phuc Huynh, P. Luong Luc, N. Duc-Vu, and N. Kiet Van, "Span detection for aspect-based sentiment analysis in vietnamese," 2021.

[10] G. Da San Martino, A. Barrón-Cedeño, H. Wachsmuth, R. Petrov, and P. Nakov, "SemEval-2020 task 11: Detection of propaganda techniques in news articles," in *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, (Barcelona (online)), pp. 1377–1414, International Committee for Computational Linguistics, Dec. 2020.

[11] "Twitter sentiment extraction." https://www.kaggle.com/competitions/tweet-sentiment-extraction.