Sentiment Analysis - Subjectivity & Polarity

Francesco Laiti

University of Trento

francesco.laiti@studenti.unitn.it

This report explains the development and evaluation of two models with respect to a baseline for a sentiment analysis task.

1. Introduction

This work performs sentiment analysis at document-level using the `movie_reviews` dataset, available in NLTK, to classify whether they are positive or negative reviews, known as polarity classification. As suggested by Bo Pang et al. [1], removing subjectivity sentences from movie reviews "can prevent the polarity classifier from considering irrelevant or even potentially misleading text". For this reason, the `subjectivity` dataset, available in NLTK too, is used to train a subjectivity detector and filter out the `movie_reviews` dataset from objective sentences. For this project, I started with a probabilistic classifier and then moved to deep learning models to hopefully achieve better performance. I implemented a Naive Bayes-based approach used as baseline, then I implemented a GRU-based architecture using the Attention mechanism and pre-trained word embeddings, and finally a Transformers-based approach.

2. Task Formalisation

Sentiment analysis, also known as opinion mining, is a natural language understanding task that aims to extract subjective information from text. It is a powerful tool to gain insights into customer opinions and attitudes. The task of sentimental analysis can be investigated at three different levels: document level, sentence level, and aspect level. The first two levels are interesting and highly challenging. However, the third level is more difficult because it performs a fine-grained investigation. At document-level, the process aims to classify the overall sentiment of a document. It works best when it is written by one person and it has an opinion about a single entity. At sentence-level, the focus is on the sentence. Not all sentences contain personal opinions; for this reason, they need to be classified as objective or subjective and usually remove the objective ones. Document and sentence levels do not provide the necessary detail on all aspects of the entity because they do not determine what people like and dislike. Aspect-level tries to address this problem by finding sentiments with respect to the specific aspects of entities [2]. Several factors make sentiment analysis challenging: language complexity, the ambiguity of a word or phrase that can have multiple meanings, the use of emoji, slang, or abbreviation that can have different meanings across different cultures and contexts, spam, irony, and sarcasm that involve saying the opposite of what is meant. Sentiment analysis is not a single problem, but rather a "suitcase" research problem that necessitates the completion of numerous NLP tasks. Several steps are required to extract sentiments from a given text, and many NLPs problems must be solved [2]. The aim of this project is to train and evaluate a polarity classifier on movie reviews at document-level. As mentioned in section 1, I trained and evaluated a subjectivity classifier too to filter out movie reviews' subjective sentences and see the benefit from it.

3. Data Description & Analysis

Two datasets are used for this project: `subjectivity` and `movie_reviews` dataset, both released in 2004 by Pang and Lee, already tokenized and lowercased, and available in the NLTK package `corpus`.

Subjectivity Dataset. The `subjectivity` dataset contains 5,000 subjective and 5,000 objective sentences. It is used to train and evaluate a supervised binary classifier for subjectivity detection. For labeling, I choose to map *subjective* as *0* and *objective* as *1*. Regarding statistics, the vocabulary contains 23906 words. The average sentence length is 24 words while the minimum and maximum are respectively 10 and 120 words.

Movie Reviews Dataset. The `movie_reviews` dataset is used for polarity classification and it contains 1,000 negative reviews and 1,000 positive reviews. It is used to train and evaluate a supervised binary classifier for polarity detection. For labeling, I choose to map *negative* as *0* and *positive* as *1*. The vocabulary size is 2× larger than the previous dataset, with 39768 words. The average number of sentences per document is 36 while the minimum and maximum are respectively 1 and 188 sentences. Regarding the number of words, on average a document has 792 words with a minimum of 19 and a maximum of 2879 words.

I have also tested a version of removing the stop words using the NLTK built-in function. In this case, the vocabulary size of subjectivity and polarity drops, respectively, to 23753 and 39617 words, with an average of ≈150 words removed.

For this project, I tried also to process the entire datasets as raw documents in order to tokenize them again with different tokenizers, like spaCy or NLTK. No improvements have been observed, so the experiments were conducted without changes to the original dataset.

The models have been trained and evaluated using a Stratified K-Folds cross-validator with a number of folds equal to 5. I choose to use a batch size that is a power of 2. For reproducibility reasons, I decided to use a fixed random state during the split of the dataset and fix the generator of the data loader in PyTorch.

4. Model

In this section, I explain the three models proposed to perform the sentiment analysis task as mentioned in section 1.

4.1. Multinomial Naive Bayes

This model, a multinomial Naive Bayes classifier, is the one proposed as baseline in the project assignment, and it has been implemented following the *sentiment analysis* notebook of the lab using the `scikit-learn` library.

4.2. GRU - Gated Recurrent Unit

The second model proposed is an RNN-based model, the GRU (Gated Recurrent Unit) architecture. It is built using the `PyTorch` library. This architecture's choice over the vanilla RNN and the LSTM is justified for these reasons: (1) vanilla RNN has a problem of vanishing gradient and cannot capture long-term sentences and, because we are dealing with long sequences, it is not suitable for this project, (2) LSTM has more parameters and requires more computational resources than GRU. Moreover, the data are limited and, as observed by Yang et al. [3], GRU performance surpasses LSTM in the scenario of long text and small datasets. This project uses small datasets and, in particular, the movie reviews dataset which contains long reviews. Thus a GRU is implemented.

The proposed GRU is a bidirectional version, which takes into account both the context before and after the word, whereas a traditional GRU can only consider the context before the word. It shows better performance than the non-bidirectional version. For this model, I have tried to use different tokenizers, as mentioned in section 3, but without improvements. It is worth mentioning that the datasets are already tokenized, so the original version has been used. I implemented a class `Tokenizer` to create a dictionary that maps the words in the training set to numbers and then encodes sentences or documents tokens to numbers. I have also written an optional function to build a matrix of weights for the embedding layer using pre-trained word embeddings. The input to the model is a sequence of words. Each word is first encoded into an integer representation and then embedded into a vector of 300 values, obtaining as output the corresponding word embedding. The number of features in the hidden state is set to 128. The number of layers is set to 2 because it can capture more complex patterns in data and, indeed, it showed better performance than just 1 layer. As the last layer of the network, I added a linear classifier that outputs two features. The model has been built with optional configurations described next:

1. *Attention mechanism.* I implemented a Bahdanau Attention mechanism [4] because not all words have the same importance and thus help the model to select the most informative words. In this way, the model can be guided to focus on meaningful words useful for the task;
2. *Pre-trained word embedding.* I implemented GloVe [5], a pre-trained word embedding, using the *6B tokens* version to improve the performances of the model, and I made it trainable with a different learning rate than the model, in order to try different learning rate parameter;
3. *Stop words.* I used the built-in NLTK function to remove the stop words to see how they affect performances.

I made these optional configurations easy to plug and unplug from the model. For regularization I applied, before the two fully connected layers, a dropout with a probability equal to 0.5 [6]. For training, I run it for 30 epochs, a sufficient number for overfitting the training set. The learning rate of the model is 10^{-3}. I decided to have the possibility to change the learning rate of the embedding layer to finetune it in different ways. I tried also to freeze the pre-trained embedding weights. The word embedding learning rate, when using GloVe, is set to 10^{-2} because it showed better performances. The batch size is set to 4096 for subjectivity (could be higher but it overfits very quickly and the behavior of the GPU is irregular) and 512 for polarity. The optimizer used is Adam and regarding the loss, I choose the Cross-Entropy loss. Following the *sequence nn* lab, I used the `pack_padded_sequence` to avoid computation over pad tokens by reducing the computational cost.

4.3. BERT

The last model presented is a Transformers-based architecture. I used the `transformers` library to fine-tune large pre-trained models on the downstream task. This helps me to implement complex architectures, avoid training them from scratch and achieve better performances. For the choice of pre-trained models, I explored and searched by myself in the models' zoo available at huggingface.co/models. The focus has been on BERT [7]. As mentioned in the last part of the *sequence nn* lab, Transformers tokenizers split the document differently from classic rule-based approaches. For this reason, I process all the datasets using a BERT-based tokenizer `bert-base-uncased` before the training and evaluating steps. The tokenizer was not fine-tuned. As written in the model's card, the maximum length of the sequence has to be $\leq$512 subword tokens. As pointed out on the discussion forum of Hugging Face [8], BERT has its own word representation, and thus that might be words tokenized in a particular way. For this reason, many movie reviews, which are length, on average, 792 words, will be truncated to 512 tokens during the tokenization process.

For subjectivity classification I choose a BERT-based model named `cffl/bert-base-styleclassification-subjective-neutral`. As the model card report, the model has been fine-tuned on the Wiki Neutrality Corpus, a parallel corpus of 180,000 articles labeled as "neutral" or "biased". For polarity classification, I choose again a BERT-based model named `fabriceyhc/bert-base-uncased-amazon_polarity`, which is a fine-tuned version of BERT on the Amazon reviews dataset. For both models, I use a Cross-Entropy loss, which is the default loss output from the model. Following the guide [9] provided by Hugging Face, I fine-tune the 🤗 Transformers models for 10 epochs (a sufficient number to overfit the training set), use a low learning rate 5^{-5}, and the AdamW as optimizer with the default weight decay equals to 0.01. Batch sizes for subjectivity, polarity without filtered sents and polarity with filtered sents are respectively 128, 16, and 32, based on the availability of the GPU memory.

5. Evaluation

The three models have been trained and evaluated both for subjectivity and polarity tasks. I also tried to train and evaluate the polarity classifier with and without objective sentences. To save the best model weights, I adopted the early stopping of the training to interrupt the training when the performance is getting worse by observing the accuracy evaluation score. I used the parameter `patience`, set by default to 3. For the attention visualization of the GRU model, I adapted the code provided by [10] to generate a `.tex` file and use the library `seaborn` to generate attention heat-map images. Regarding the metrics, I used the accuracy and the F1-score computed as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1 score} = \frac{2 * Precision * Recall}{Precision + Recall}$$

where TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative.

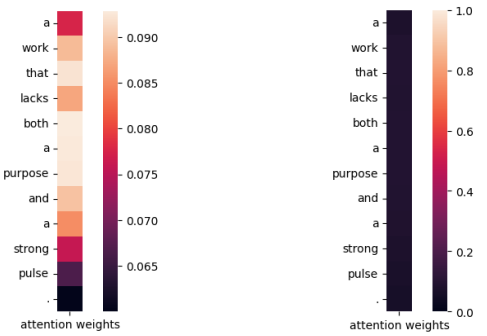
Note that the tests were conducted on Google Colab Free and an NVIDIA GeForce RTX 3090.

5.1. Subjectivity classification

Table 1: *Accuracy and F1-score of subjectivity classification. Scores reported, in percentage, the average of Stratified 5-fold runs and the standard deviation.*

Model	Accuracy	F1-score
Multinomial NB	92.0 ±0.7	91.9 ±0.6
GRU	91.1 ±0.3	91.0 ±0.4
GRU + Attention	91.0 ±0.4	90.9 ±0.3
GRU + GloVe	91.4 ±0.7	91.4 ±0.6
GRU + Att. + GloVe	92.9 ±0.5	92.9 ±0.4
GRU + Att. + GV + SW	90.9 ±0.4	90.9 ±0.4
BERT	**96.8 ±0.4**	**96.8 ±0.4**

In Table 1 are reported the accuracy and the F1-score for the subjectivity task. The baseline, a Multinomial Naive Bayes model, obtains a mean score higher than GRU in all the cases except the model configured with the Attention mechanism (Att.) plus the pre-trained word embedding GloVe (GV). Removing stop words (SW) does not improve the performance of the model, even with the best configuration of GRU that obtains the highest accuracy (Attention + GloVe). It has the lowest scores compared to the other models, and after experimenting with other configurations, the accuracy dropped in every case. This can be related to stop words that are important to express subjective opinions, such as *very*, *because*, *about*, and other types of words that are removed. For this reason, the stop word configuration has not to be tested in the next section. Regarding the word embedding layer, when using GloVe, I tried different setups.



(a) *Color scale in range* $[min(\mathbf{x}), max(\mathbf{x})]$.

(b) *Color scale in range* $[0, 1]$.

Figure 1: *Attention weights heatmap for subjectivity classification, where* $\mathbf{x}$ *is the attention weights vector. Displayed sentence labeled as subjective from the original source.*

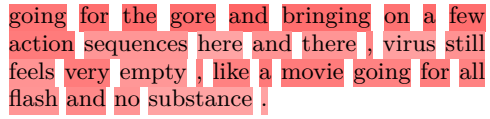
Adopting a learning rate for the embedding equal to 10^{-2} performs better than freezing the parameters or using the same learning rate of the network, 10^{-3}. In Table 1 are reported the results with embedding GloVe learning rate equal to 10^{-2}. The Transformer outperformed with respect to the other models. Contrary to what has been supposed, GRU performances are quite disappointing. This behavior can be related to the scarcity of data. Neural networks require lots of data to generalize well and to be properly trained. Training the GRU on a large dataset may lead to better performances, but in that case, an LSTM model would be considered. The attention mechanism does not improve the performance compared to the vanilla GRU. Observing Figure 1b, the weights are very small and thus not helping the classifier to focus on particular and meaningful words for the task. Figure 1a reveals the information of the weights, but the differences between weights are very small. This can be related, again, to the lack of data, but also to the length of sentences. Sentences in the `subjectivity` dataset are short and the attention could be useless. Transformers are not affected by the low quantity of data, thanks to the pre-training on large amounts of data. Fine-tuning, combined with a low learning rate, obtains very good results. The early stopping killed the training process before reaching the maximum value of the epochs in all the cases. The best model weights for subjectivity classifications, saved for later to filter out the movie reviews, are reported in Table 2.

Table 2: *Best model weights for subjectivity classification. Note that folds start from 0.*

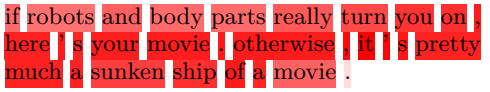
Model	Fold	Accuracy	F1-score
GRU	4	91.7	91.5
GRU + Attention	3	91.5	91.4
GRU + GloVe	1	92.5	92.6
GRU + Att. + GV	1	93.7	93.7
BERT	2	97.3	97.3

5.2. Polarity classification

As explained in section 1, removing objective sentences from movie reviews could improve the performances of the polarity classifier. For this reason, the best classifier of each model (see Table 2) trained on the `subjectivity` dataset has been used to filter out the objective sentences from the `movie_reviews` dataset. In particular: (1) in the Multinomial Naive Bayes model, after the computation of scores, the model has been trained again on the whole `subjectivity` dataset; (2) in the GRU model, after analyzing the performances of each configuration, I used the GRU with Attention and word embedding GloVe to filter out the objective sentences; (3) in the Transformer model, I chose to use the best model with the highest accuracy and F1-score. On average, the number of sentences removed from the multinomial Naive Bayes, GRU, and BERT subjective detectors is, respectively, 11, 15, and 15. Analyzing the distribution, the mode values of BERT and GRU are, respectively, 9 and 12. Table 3 reports the accuracy and F1-score for each model of the polarity task obtained with and without filtering the sentences. Results for the GRU are disappointing also in this task. The baseline obtained a higher score compared to the GRU, except for the GRU with only GloVe and with Attention mechanism plus GloVe. At the beginning of the experiments, I experienced an issue with the `patience` parameter, which killed the training process before the network started to learn and improve. In the beginning, the GRU seems to perform very poorly, without improvements, and getting worse every epoch. Setting `patience` to 7 or removing this constraint, make the network able to learn and achieve comparable results. As observed in section 5.1, the word embedding GloVe performs better with a learning rate equal to 10^{-2}. BERT outperforms also in this task, with a big advantage over the other model. Filtering out the objective sentences help the baseline and BERT models to obtain higher scores, and so it can be confirmed what Bo Pang et al. [1] proposed in 2004. This is not true for the GRU, where performance drops, or increases by a small number with the vanilla GRU. This can be related to the quality of filtering, where the subjectivity detector works with an error that could potentially remove useful sentences or leave useless ones. In Figure 2, we can appreciate how



(a) *Objective sentence from a movie review.*



(b) *Subjective sentence. Classified as subjective sent from the subjectivity detector.*

Figure 2: *Text attention heatmap visualization of the attention weights for polarity classification.*

the Attention mechanism works in polarity classification. Colors have been obtained by normalizing the attention vector before applying the soft-max, and then removing the outliers of the weights vector to visualize better the contribution of each word. The examples reported are sentences extracted from the same movie review and the intensity of the color highlights the weight values. We can observe that the subjective sentence in Figure 2b contains words that have higher values compared to Figure 2a. This means that Attention in the polarity classification learns to pay more attention to subjective words that are important for classifying the review rather than objective ones. Despite the visualization, the weight values are very low and do not help the network, as already seen in the previous section 5.1, and observable by the poor results in Table 3. The reason why BERT works better in the case of filtering the objective sentences might be related also to the input limitation mentioned in section 4.3, where sequence tokens ≥ 512 are truncated. Observing movie reviews, most of them start with objective sentences related to the plot of the movie, and personal opinions start some sentences after that introduction. Given the fact that a document on average has 792 words, means the reviews have a high probability to be truncated and thus remove important personal information, useful to determine the polarity of the document. On the other hand, by filtering out the objective sentences, a document presents an average of 498 words, quite half of the unfiltered case, which means that most of the reviews can be fit in the model without truncation, and also has a high probability to include only subjective sentences. This process ensures more quality input data to the model.

Table 3: *Accuracy and F1-score of polarity classification. Scores reported, in percentage, the average of Stratified 5-fold runs and the standard deviation.*

Model	**Accuracy**	
	Polarity	Filtered polarity
Multinomial NB	81.4 ±1.4	84.1 ±2.1
GRU	77.8 ±3.0	78.0 ±0.2
GRU + Attention	81.2 ±2.2	80.8 ±0.1
GRU + GloVe	85.4 ±1.5	82.8 ±0.1
GRU + Att. + GV	84.7 ±0.7	81.0 ±0.1
BERT	90.3 ±0.6	**93.8 ±1.0**
Model	**F1-score**	
	Polarity	Filtered polarity
Multinomial NB	81.1 ±1.6	83.9 ±2.2
GRU	77.4 ±3.2	77.9 ±0.4
GRU + Attention	81.2 ±2.2	81.2 ±0.1
GRU + GloVe	85.2 ±1.9	83.4 ±0.1
GRU + Att. + GV	85.1 ±0.8	80.7 ±0.1
BERT	89.7 ±0.8	**93.9 ±1.0**

6. Conclusion

In this report, I proposed, trained, and analyzed the performances of three methods to perform sentiment analysis using a polarity classifier, supported by a subjectivity detector to pre-process data by removing misleading objective sentences from movie reviews. Transformers BERT outperforms with respect to the baseline and GRU models in all the tasks.

7. References

[1] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," *Computing Research Repository - CORR*, vol. 271-278, pp. 271–278, 07 2004.

[2] M. Birjali, M. Kasri, and A. Beni-Hssane, "A comprehensive survey on sentiment analysis: Approaches, challenges and trends," *Knowledge-Based Systems*, vol. 226, p. 107134, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095070512100397X

[3] S. Yang, X. Yu, and Y. Zhou, "Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example," in *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, 2020, pp. 98–101.

[4] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014. [Online]. Available: https://arxiv.org/abs/1409.0473

[5] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[6] "stackexchange: Where should i place dropout layers in a neural network," https://stats.stackexchange.com/questions/240305/where-should-i-place-dropout-layers-in-a-neural-network, accessed: 2023-01-13.

[7] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[8] "Hugging Face discussion forum: Is 512 token in bert, token or character level?" https://discuss.huggingface.co/t/is-512-token-in-bert-token-or-character-level/16406, accessed: 2023-01-14.

[9] "Hugging Face: Fine-tuning with custom datasets," https://huggingface.co/transformers/v3.2.0/custom_datasets.html, accessed: 2023-01-13.

[10] J. Yang and Y. Zhang, "Ncrf++: An open-source neural sequence labeling toolkit," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018. [Online]. Available: http://aclweb.org/anthology/P18-4013