# Final NLU Project Report

*Elena Basei (229706)*

University of Trento

elena.basei@studenti.unitn.it

Abstract

This document is the report of the final project developed for the Natural Language Understanding course at the University of Trento. The primary objective was to develop a Sentiment Analysis pipeline consisting of two tasks: Subjectivity Detection and Polarity Classification.

## 1. Introduction

Sentiment Analysis is a natural language processing (NLP) technique which aimed at determine the emotional orientation of a body of text. In recent years, with the rise of social media and e-commerce, it has been extensively used and developed for applications ranging from marketing to customer service.

In this report will be presented my solution for Sentiment Analysis, which also uses, besides Polarity Classification, Subjectivity Detection to improve its performance as explained in the work released by Pang et al. [1]. The neural network I propose follows almost completely the structure described in the paper of Satapathy et al. [2], which also underlines the importance of Subjectivity Detection when it comes to Polarity Classification.

## 2. Task Formalisation

As mentioned in the introduction, the pipeline to resolve the task consists of two sub-tasks, each of which will be explained in the following subsections.

### 2.1. Subjectivity Detection

The Subjectivity Detection task objective is to identify and remove *factual* or *neutral* content from a body of text, retaining only subjective sentences. This procedure is especially important when we seek to determine the polarity of a set of sentences as we only need subjective sentences to analyze the author's opinions. The reason behind this is that analyzing an unfiltered text may lead the network to learn wrong word patterns that do not help determine the orientation of a text, reducing instead its accuracy.

### 2.2. Polarity Classification

Polarity classification constitutes the core of the Sentiment Analysis pipeline, it is the sub-task responsible for determining the orientation of a text. In general, it is addressed with approaches like dictionary-based ones, which mainly focus on the construction and use of word-level sentiment lexicons or concept-level dictionaries, and those based on machine learning, which depend on the availability of an annotated corpus with polarity labels to determine the polarity of new samples.

In this paper, the proposed solution uses a machine learning approach that leverages the labels of the dataset chosen to train the network.

## 3. Data Description & Analysis

To train and evaluate my model I used two datasets, the NLTK MovieReviews and the NLTK Subjectivity.

The first dataset [1] is composed of 2000 movie reviews written before 2002 by 312 different authors, 1000 of them are positive while the remaining 1000 are negative. The total number of words is 1583820 while the vocabulary length is 39768. To determine the polarity of the reviews, the authors searched for explicit rating score inside them. This was not an easy process as a review may not have the author's rating with it, and when it does, the rating can appear at different places in the file in different forms. Only few of them were recognizable, which were then extracted via a set of ad-hoc rules. In essence, a review's classification was determined based on the first rating they were able to identify. In my model, these data are used to train the polarity classifier and they are randomly spit between training and testing, for the training we have 80% of the

reviews, while for the testing we have the remaining 20%.

The NLTK Subjectivity dataset [1] contains 5000 subjective sentences and 5000 objective ones. The data has been acquired from snippets of movie reviews from Rotten Tomatoes (http://www.rottentomatoes.com/) and plot summaries for movies from the Internet Movie Database (http://www.imdb.com). The lables were assigned assuming that all snippets from the Rotten Tomatoes pages are subjective, and all sentences from IMDb plot summaries are objective. Also in this case, the dataset is split between training and testing as 80%-20% and it is used to train the subjectivity classifier.

# 4. Model

The pipeline I developed is modeled on the approach proposed by Satapathy et al. [2]. They presented a Multitask Learning framework with BERT-base embedding composed by a Bidirectional LSTM layer, a Self Attention layer, a Neural Tensor Network layer and a final Softmax layer. The results they obtained using this network structure show how this framework exceeds baseline levels and achieves a new state-of-the-art status in multitask learning. They also demonstrate that the information across datasets for related tasks can be helpful in understanding task-specific features. In this particular case of use, they have proven that using Subjectivity Detection along with Polarity Classification improves the final network accuracy.

## 4.1. Pipeline Structure

To build my network I followed the paper, changing however some of the layers. For this project I wanted to focus my results on the impact Subjectivity Detection has on the polarity classifier accuracy, to do so I removed the Neural Tensor Network layer and built a network that can be used for both subtasks. Moreover, I changed the simple Self Attention layer in a Multihead Attention one [3].

The final pipeline is hence structured as follow:

- BERT-tokenizer [4] to encode a body of text in a tensor, it is set with max_length for truncation equal to 512 words.
- NN structured as in Figure 1 for subjectivity classification. This network is used to filter the sentences in a body of text and retaining only subjective sentences.

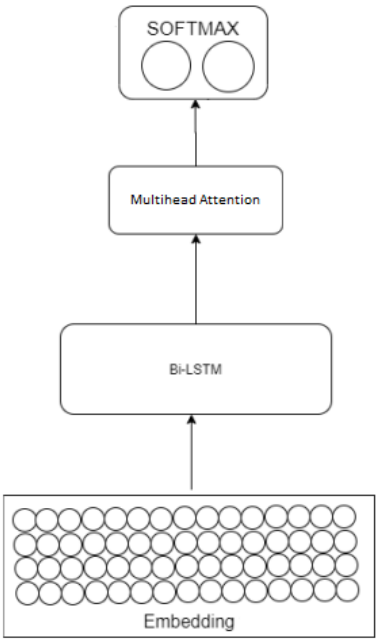


Figure 1: *Structure of the model*

- Text filtering function that uses the results of the subjectivity classifier.
- NN structured as in Figure 1 for polarity classification. The input to this network is the filtered text resulting from the previous step.

## 4.2. Model Training

Regarding the training, I used firstly the NLTK Subjectivity dataset for the subjectivity classifier and then the filtered NLTK MovieReviews for the polarity classifier. For the sake of completeness, I also trained another instance of the polarity classifier without filtering the dataset to compare the results with the complete pipeline ones.

For both the training I used ADAM optimization algorithm with learning rate equal to 0.001 and the Cross-Entropy loss as the loss function. The number of epochs I used is 20 with batch_size equal to 64.

## 4.3. Baseline

The baseline I used for this project is the Naive Bayes classifier, as suggested in the assignment details. The pipeline I developed for the baseline is similar to the one I just exposed. To encode the datasets I used the CountVectorizer, testing also how

different combination of stop_words and number of ngrams considered affect the final result. For the subjectivity classifier and the polarity one I used a Multinomial Naive Bayes model.

## 5. Evaluation

In order to evaluate the model, the loss and the accuracy are taken into account. The loss used is the Cross-Entropy one while the accuracy is computed as:

$$ACC = \frac{\sum_{n=1}^{N} y_n^*}{N}$$

with $N$ total number of samples and $y_n^*$ equal to

$$y_n^* = \begin{cases} 1, & \text{if sample n is correctly classified} \\ 0, & \text{otherwise} \end{cases}$$

In the complete pipeline, the subjectivity classifier is treated as a pre-trained module that does not need to compute the gradient on its parameters. Therefore, the overall pipeline accuracy is computed as the accuracy of the polarity classifier.

### 5.1. Tests results

In the following graphs (Figure 2) are reported the results of the two main tests along with the results of the subjectivity classifier. The overall pipeline final train accuracy is 96% while the test is slightly smaller, as reported in the first graph. This is probably due to the small dimension of the dataset used for the train and test procedures. Another factor that I think has influenced the final outcome is the truncation process carried out by the BERT-tokenizer when encoding a body of text. Although truncating is performed after the 512th word (maximum value possible), some of the reviews are longer than this and we can have a loss of information.

For the other test I removed the second and third step of the pipeline, giving to the polarity classifier unfiltered data. As shown in the graphs, the model performance is worst that the complete one, demonstrating how Subjectivity Detection impacts the final outcome.

All the results I obtained are in line with the ones exposed in [2].

### 5.2. Baseline of comparison

During the testing of the baseline I tried different combination of the CountVectorizer parameters to

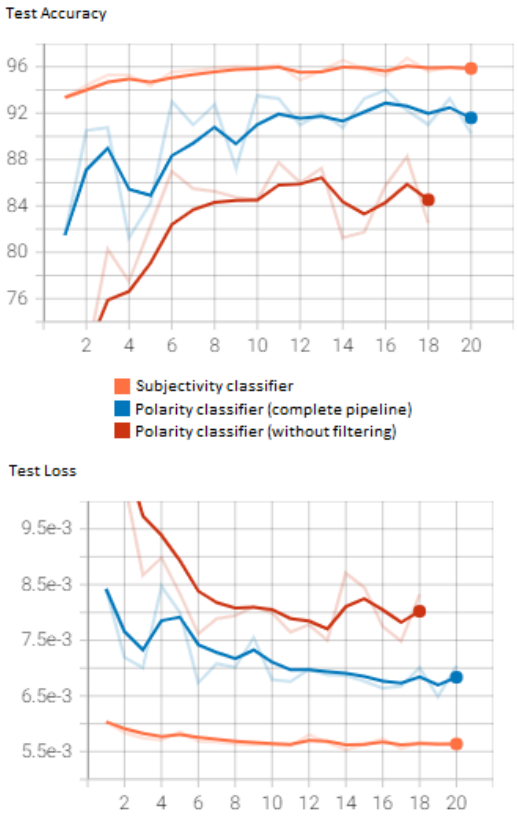


Figure 2: *Tests results*

better understand how words relations influence the result when working with Sentiment Analysis. In Table 1 are reported the final test accuracies if the different combination of parameters. The first row labels refer to the polarity classifier while the first column ones refer to the subjectivity classifier.

| | Standard | SW | NGRAMS | SW+NGRAMS |
|---|---|---|---|---|
| Standard | 0.841% | 0.832% | 0.877% | 0.822% |
| SW | 0.842% | 0.836% | 0.876% | 0.830% |
| NGRAMS | 0.854% | 0.850% | 0.875% | 0.841% |
| SW+NGRAMS | 0.842% | 0.834% | 0.875% | 0.832% |

Table 1: *Naive Bayes tests accuracy*

For every classifier, I used a different CountVectorizer, each of which is tested with four possible combinations of parameters:

- *Standard* - The stop_words are not removed and are considered only ngrams of size 1.
- *SW* - The stop_words are removed but we still consider only ngrams of size 1.
- *NGRAMS* - We do not remove the stop_words but we consider ngrams from size 1 to size 5.

- *SW+NGRAMS* - We both remove the stop_words and consider ngrams from size 1 to size 5.

From the results I obtained emerges that the best option is to consider different size of ngrams and keep the stop_words for the polarity classifier CountVectorizer. Regarding the subjectivity classifier one, we have instead that choosing one combination rather than another does not influence the performance significantly.

# 6. Conclusion

In conclusion, the overall results are good and in line with the ones of Satapathy et al. [2], although obtained with different implementations. Moreover, they are better than those obtained using the baseline. It would be interesting to try to implement the entire solution of Satapathy et al. and confront the features found with my model ones. Their model seeks to share information between tasks to find better features, hence studying the differences could contribute to improve both the models.

# 7. References


[1] Bo Pang and Lillian Lee, "*A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts*"

[2] Ranjan Satapathy, Shweta Pardeshi, Erik Cambria, "*Polarity and Subjectivity Detection with Multitask Learning and BERT Embedding*"

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, "*Attention is all you need*"

[4] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, Alexander M. Rush, "*HuggingFace's Transformers: State-of-the-art Natural Language Processing*"