Final NLU project report

Federico Diprima 224400

University of Trento

federico.diprima@studenti.unitn.it

Abstract

This document represents the report of the final project for the Natural Language Understanding course of the University of Trento. The main goal was to implement a Sentiment Analysis [10] system that consists of Subjectivity Detection and Polarity Classification.

1. Introduction

Sentiment analysis or opinion mining, is the use of natural language processing and text analysis to identify and study people's opinions, sentiments, emotions and attitudes toward entities such as products, services, organizations and their attributes.

With this project I present my approaches to face two different sentiment analysis subtasks. I decided to start analyzing the performances of a simple Naive Bayes classifier (used as baseline) in the same way that we have seen during the classes, then I have tried different methods in order to improve the results.

As we will see later, some of these approaches will give me comforting results compared to the baseline.

2. Task Formalisation

In this work i focused mainly on two subtasks of sentiment analysis, Subjectivity Detection and Polarity Classification.

Subjectivity Detection is the task of sentiment analysis of differentiating between opinionated (Subjective) and non-opinionated (Objective) sentences. The term "objective" refers to the sentences which carry factual information, the term "subjective" instead, describes the incident contains non-factual information, such as personal opinions, judgment, and predictions.

Polarity Classification is the process which aims to automatically identify the general sentiment expressed by a text. Some systems looks, for instance, at emotional states contained in a sentences, such as enjoyment, anger, disgust, sadness, fear, and surprise. Some others focus on rating an opinion expressed by a document on a scale from 1 to 5 stars for instance. In this case the goal of the project is to evaluate whether a sentence is overall positive or negative, a simple binary classification.

These two subtasks are not necessarily independent, in fact Pang et al. [7] showed that removing objective sentences from a document before classifying its polarity helps in improving the performances. That's because "..this can prevent the polarity classifier from considering irrelevant or even potentially misleading text.", in this way the polarity classifier take into consideration only subjective sentences, those that can actually discriminate whether an opinion is good or bad. For instance if we consider a movie review, it could be made up of objective and subjective sentences. Objective sentences for describe the plot and the roles of the actors and subjective ones that express effectively the feelings and the opinions of the users. The objective sentence "The protagonist tries to protect her good name" describe a part of the plot of the movie and it tells us nothing about the user's opinion and in fact could well be embedded in a negative movie review, even if contains the word "good". Objective sentences are useless in polarity classification task, since they do not carry opinions.

In all the models that I will propose during this project, I will start from the subjectivity task by training a classifier that can predict the subjectivity of each sentence. Then I'll move on to the polarity classification task at document level by also analyzing the pros and cons of filtering out objective sentences from the whole dataset before training it.

3. Data Description & Analysis

3.1. Subjectivity Dataset

The dataset used for subjectivity task is called *subjectivity*, it is introduced in by Pang and Lee on June 2004 and consists of 10,000 sentences, the fist half (5,000) are objective and the second 5,000 are subjective. The total number of words is around 240,000 and the vocabulary size is 23,906. Minimum sentence length is 10 words, the maximum is 120 and the average is 24 words. The most common word, excluding punctuation is the word "the".

This dataset is simply composed by a list of sentences in which each sentence is a list of strings, the target labels are not included, so I added them manually. In all the proposed models the dataset is split into Train and Test set and it is used to train and evaluate our subjectivity binary classifier.

3.2. Movie_reviews Dataset

The dataset used for Polarity Classification is called instead *movie_reviews* and it was also released by Pang and Lee in 2004. It is made by 2,000 reviews, of which 1,000 are positive and 1,000 are negative. The number of words is obviously greater than the previous dataset, 1,583,820 words and the vocabulary size is 39,768. Also in this case the most common word is "the".

This dataset is a list of reviews in which every review is a list of sentences (which is a list of strings) so it is a bit more structured than the one before. The maximum length of a review is 115 sentences and the minimum one is 1 sentence, with an average of 24 sentences per review. In Table 1 we can see some other statistics of this dataset.

Also in this case the ground truth targets are not present, each review is paired with a binary label, "Positive" or "Negative". The dataset is splitted in Train and Test set and it is used to train the polarity classifier, which should predict whether a review on a movie, is positive or negative.

Statistics	Max	Min	Avg
Sentences per review	115	1	33
Words per review	2,879	19	792
Word per sentence	187	1	24

Table 1: *movie_review statistics*

4. Models

As I briefly said before I used different methods to solve both tasks, starting from the simplest to something a little more complex. In this section I will describe the main features of the models I tried.

4.1. Naive Bayes Classifier

The first model I implemented is a Naive Bayes Classifier, the same proposed in the project description and the same seen during classes. With the following architecture I will try to improve the results obtained with this baseline.

4.2. LSTM

The second approach involves the introduction of a LSTM Recurrent Neural Network. Long Short Term Memory networks, usually just called LSTMs are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter and Schmidhuber in 1997 [3] and it became very popular in the following years. The key idea of an LSTM cell is to have some gates that regulates the flow of information inside it and this allows to remember information for long periods of time.

Since the dataset consists of a text document, as mentioned before, the first step was to convert these words into an id representation and then embedded in a vector of 300 values, before passing it to the net. Regarding the LSTM i decided to use a hidden representation composed of 100 values and a bidirectional architecture with 2 layer, because gave me better results. For each word given as input, the output is composed by the hidden representation of the last layer, since a bidirectional cell is used, each hidden vector will be composed by the forward and backward parts concatenated together. The obtained output is then summed up to create the final context vector, sent to a dropout layer with probability 0.5 for regularization and finally to the classifier, a linear layer with only one output neuron. In the image below (Figure 1) we can see the structure of the model. For practical reasons I have inserted only one LSTM layer in the figure.

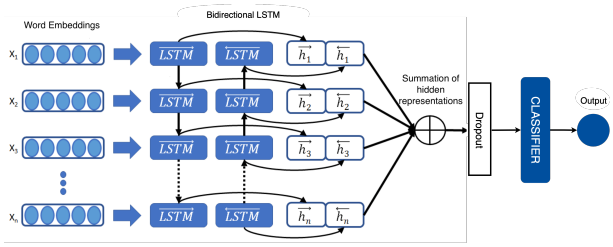

Figure 1: *LSTM based classifier*

Another small problem concerning the structure of the text sequences in the datasets, was that, obviously, the sentences were not all the same length. A function called *collate_fn* reused from laboratories was used to solve this problem, adding 0 pads to "fill" the shorter sentences. Now the input is ready to be fed into the net with a batch size of 256 examples each, for both subjectivity and polarity classification.

This architecture has been trained for 30 epochs, with a patience of 5 epochs if no improvements occurs. The used optimizer is ADAM [5] with a learning rate of 0.001 for subjectivity task and 0.005 for the polarity one. The loss function used is the binary Cross Entropy with Logits Loss [1], which combines a Sigmoid layer and the BCELoss in one single class. This version is more numerically stable than using a plain Sigmoid followed by a BCELoss as, by combining the operations into one layer, we take advantage of the log-sum-exp trick for numerical stability.

4.3. CNN

The third method I tried to solve the two problems is a CNN-based approach [4]. Traditionally, CNNs are used to analyse images and are made up of one or more convolutional layers, which use filters to scan an image and produce a processed version. Each filter has a shape, each element of the filter has a weight associated with it and this weights are learned via back-propagation. The intuitive idea behind learning these weights is that every convolutional layer act like feature extractor, extracting parts of the image that are most important for the final goal. In the same way that a 2D filter can scan and extract information from an image, a one dimensional filter of size N (1xN) can look over a N sequential words in a piece of text. In this CNN model we will use multiple filters of different sizes which will look at n-grams (bigrams, trigrams, ecc..) within the text. The intuition here is that the appearance of certain n-grams within the text will be a good indication of the final classification.

Also in this case the first step was to convert each word into an id representation and then embedded it in a vector of 150 values. Here a sentence is visualized in 2 dimensions, each word along one axis and the elements of embedding vectors across the other dimension. Consider the image below (Figure 2), with our word vectors represented in blue. Here we have a sentence composed by 4 words with 5 dimensional embeddings, creating a 4x5 tensor (this is only a simplification, in reality every word is embedded in a vector of size 150, not 5).

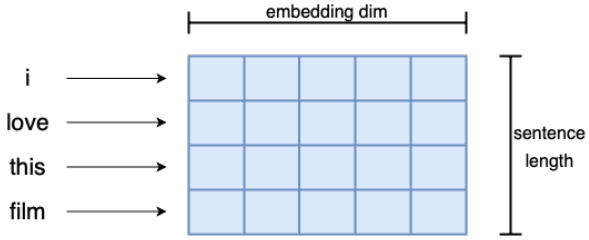

Figure 2: *Example of a CNN input*

Now the data is ready to be fed to the convolutional layers. The filters i used are of size n x 150 in order to cover entirely n subsequent words, for instance, a filter that covers two words at a time (i.e. bi-grams) will be 2x150 filter. The output of each position of the filter will be a single real number that is the weighted sum of all elements covered by the filter, just like a normal convolution. The next step is to apply max pooling on the output of the convolutional layers in order to take only the maximum value over a dimension. The idea here is that the

maximum value is the "most important" feature for determining the class of the text.

In this CNN model i decided to use filter of size 2, 3, 4 and 5, each dimension corresponds to a convolutional layer and for each convolutional layer 50 different filter have been used. To be clearer, the first covolutional layer is composed by 50 different filter of size 2, the second one is made by 50 filters of size 3, and so on, with a total of 4 convolutional layer followed by 4 pooling layer. All the pooled output were concatenated together, sent to a dropout layer (p=0.5) for regularization and finally given to the classifier. Figure 3 shows the final model.

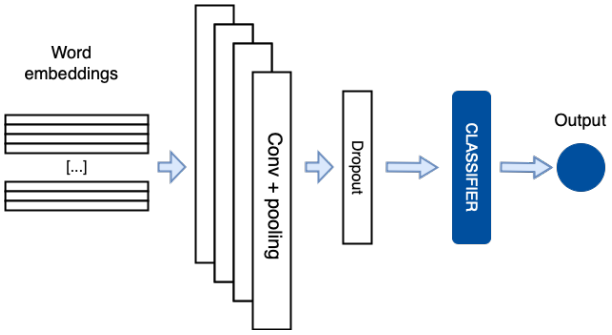

Figure 3: *CNN based classifier*

Also in this case I used a batch size of 256 with the *collate_fn* function to train the two classifiers. The loss function, the optimizer and the learning rates are the same as the previous model.

This model has been trained for 30 epochs, with a patience of 5 epochs if no improvements occur.

4.4. Transformers

The last model i propose is based on the use of transformers. A Transformer, introduced in the paper*"Attention is all you need"* [9], is simply an encoder-decoder architecture which introduce the concept of the self attention in order to improve the results on NLP tasks. This attention mechanism looks at input sequences and decides at each step which parts of the that sentence are more important than the others, giving it an higher weight. To explain better, when you read a text, you always focus on the word you read but at the same time your mind still holds the important keywords of the text in memory in order to provide context. Attention works exactly in this way.

For computation reasons and to have better results, for this last step of the project I decided to fine-tune two different models (one for each task) using the weights available in the Transformers library provided by HugghingFace. For subjectivity classification, I use the weights of the pre-trained BERT [2]. Originally BERT model was trained with two objectives: Masked language modeling (MLM) and Next sentence prediction (NSP). In this project this architecture is modified adding a linear layer with 2 output neurons in order to deal with a binary classification problem. Bert has a much deeper structure than the models previously seen, with around 110 millions parameters, it is fine-tune for 10 epochs with a smaller batch size of 64 examples and with an ADAM optimizer with a learning rate of $5 * 10^{-5}$.

For the polarity classification instead, for time reasons, i decided to use a pre-trained DistilBERT model [8]. DistilBERT is distilled version of BERT which preserves 97% of its language understanding capabilities being 60% faster, with far fewer parameters (around 66 millions). Also this model is trained on *movie_reviews* dataset for 10 epochs but with a much smaller batch size of 8 examples, to comply with the limits of the Colaboratory free GPU. The optimizer used is ADAMW [6], an improvement version of ADAM, with a learning rate of $5*10^{-5}$ and a weight decay of 0.01. Also in this case the architecture is modified with a linear layer at the end in order to face with the classification task.

Talking about the tokenizers, I used the pre-trained ones proposed by the two articles (BERT tokenizer and DistilBERT tokenizer). The tokenization process is obviously the first operation applied to the text examples before feeding them to the network.

As you can see, I decided to use models that was pre-trained on tasks that are different from the ones of this project. For example i used the weights of a BERT trained to solve Masked language modeling tasks, applying them to a classification problem. Despite this, I have achieved very good performances. I can therefore affirm that a complex model like BERT has very deep language understanding capabilities and it can solve problems that are even very different from those with which he was trained on. During my experiments I tried also to fine-tune a BERT model who had been trained more specifically on Wiki Neutrality Corpus(WNC) dataset, with the task of discriminating subjective and neutral sentences (*cffl/bert-base-styleclassification-subjective-neutral*), but the results was almost the same, so i decided to use the most general version of BERT.

5. Evaluation

As i briefly said before, for training each model I used the free GPU provided by Google Colaboratory. So, the first step was to write all the code on my personal machine, then i move to Colab notebook to actually start the training procedure. In all the proposed models, I started from the subjectivity task by training a classifier which predict the subjectivity of sentences and i saved the weights of the best achieved model in a .pt file. Then I moved on to the polarity classification task with the option to keep the original dataset or to use the subjectivity classifier trained previously to filter out the objective sentences.

The dataset is splitted in 2 parts, *train_set* and *test_set*, with a percentage of 75% / 25%, the first one is used to train each model and the second one to evaluate the achieved performances.

To evaluate the performances the Accuracy and the F1_score have been used. Here we can see the formal mathematical formulations of the two metrics.

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

$$F_1 = 2 \cdot \frac{\text{P} \cdot \text{R}}{\text{P} + \text{R}}$$

where P stands for precision and R stands for recall:

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad R = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

In this section i will compare the baseline results (Naive Bayes Classifier) with the ones proposed by me. For deciding if one model is better than another the best average accuracy

on the *test_set* will be used, that is the number of the correct predictions divided for the number of all the examples in the set.

5.1. Subjectivity classification results

In Table 2 we can observe the accuracy and F1-score in the subjectivity classification of, Naive Bayes Classifier, LSTM, CNN and BERT models.

Model	Accuracy	F1-score
Naive Bayes Classifier	92.09	92.20
LSTM	91.07	90.78
CNN	90.11	89.86
BERT	**96.92**	**96.89**

Table 2: *Accuracy and F1-score of subjectivity classification*

As we can see the architecture which has been able to perform better than all the other models is the BERT. On the other hand if we compare the other baselines we realize immediately that LSTM and CNN based models are not capable to reach the Naive Bayes accuracy and F1-score. I think that a possible reason of that problem is the limited size the dataset, neural networks in general need a large amount of examples in order to be successfully trained from scratch, a dataset of only 10,000 sentences could be too small in my opinion.

BERT model can achieve a very high accuracy (almost 97%) even with a small dataset, the reason could be that it had previously been trained on a much larger dataset. Another thing that it is interesting to observe is that a CNN-based model, an architecture that is usually used mainly on images, can have quite acceptable results even if applied on text sequences.

5.2. Polarity classification results

Regarding the polarity classification task we can see in Table 3 the results achieved with the models i proposed with and without filtering out objective sentences. In each model proposed to solve the polarity task, the filter is represented by the same model but trained for the subjectivity task. To be clearer, if we talk about LSTM-based model, firstly i trained the LSTM-based model for subjectivity classification, then i used this trained weights to detect and filter out the objective sentences from the polarity dataset (*movie_reviews*), lastly i trained the LSTM-based model on the filter dataset for polarity classification.

Model	Accuracy	F1-score
Naive Bayes Classifier	81.40	80.98
Naive Bayes Classifier + Filtering	84.40	84.10
LSTM	85.42	84.86
LSTM + Filtering	84.63	84.57
CNN	82.17	81.90
CNN + Filtering	84.78	84.50
DIstilBERT	86.92	86.19
DistilBERT + Filtering	**90.58**	**89.92**

Table 3: *Accuracy and F1-score of polarity classification with and without filtering*

Observing the accuracy and F1_score values we can see that, as suggested by Peng et al. [7], removing the objective sentences from the reviews leads us to obtain better results in most cases. Furthermore, with this filtering the dataset becomes smaller and consequently the training and evaluation time is greatly accelerated. The Naive Bayes Classifier, CNN-based and also the DistilBERT model benefits from this technique and in all this case the best average accuracy reaches higher rates if compared with the same structure without filtering.

Looking at the table the only exception is represented by the LSTM-based model, in which the accuracy is better if we use the original dataset without filtering any sentences. it is difficult for me to find the reason for this "failure", the only thing that comes to my mind is that the filter is not very accurate.

It is easy to see that the best possible model among those proposed is the pre-trained DistilBERT fine-tuned on the *movie_reviews* dataset with filtering on objective sentences. The filter in this case is very precise, (BERT in the subjectivity task) with an accuracy close to 97%, certainly this is one of the reasons why this model improves by nearly 4 percentage points if fine-tuned with this filtered dataset.

6. Conclusion

In this project report, I presented four models to deal with Sentiment Analysis, in particular with the tasks of subjectivity detection and polarity classification. I also studied the effects of filtering out objective sentences from from the polarity dataset in the polarity task. The first model is a Naive Bayes Classifier. The following is a Bidirectional LSTM, with two layers. The third one, is a CNN-based model, applied to text sequences instead of images. And the last one is a fine-tuned BERT model.

The experimental results showed that the last model was able to obtain better results over all the other approaches. Moreover, it was easy to find that in most cases, filtering out objective sentences with a subjective classifier in the polarity classification, can improve even more the performances.

References

[1] *BCE with logistic loss*. URL: https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html.

[2] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. DOI: 10.48550/ARXIV.1810.04805. URL: https://arxiv.org/abs/1810.04805.

[3] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[4] Yoon Kim. *Convolutional Neural Networks for Sentence Classification*. 2014. DOI: 10.48550/ARXIV.1408.5882. URL: https://arxiv.org/abs/1408.5882.

[5] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

[6] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2017. DOI: 10.48550/ARXIV.1711.05101. URL: https://arxiv.org/abs/1711.05101.

[7] Bo Pang and Lillian Lee. “A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts”. In: (2004). DOI: 10.48550/ARXIV.CS/0409058. URL: https://arxiv.org/abs/cs/0409058.

[8] Victor Sanh et al. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2019. DOI: 10.48550/ARXIV.1910.01108. URL: https://arxiv.org/abs/1910.01108.

[9] Ashish Vaswani et al. *Attention Is All You Need*. 2017. DOI: 10.48550/ARXIV.1706.03762. URL: https://arxiv.org/abs/1706.03762.

[10] Lei Zhang, Shuai Wang, and Bing Liu. *Deep Learning for Sentiment Analysis : A Survey*. 2018. DOI: 10.48550/ARXIV.1801.07883. URL: https://arxiv.org/abs/1801.07883.