Sentiment Analysis on Movie Reviews

Matteo Ambrosini

University of Trento

`matteo.ambrosini@studenti.unitn.it`

Abstract

The project focuses on the natural language processing facet of Sentiment Analysis. In particular, we will use a Support Vector Machine Classifier to carry out Polarity Classification task, and a Naive Bayes Classifier to carry out Subjectivity Detection. Both classifiers are wrapped into a Two Stage 'Block', and work jointly to best carry out Sentiment Analysis tasks.
In following sections we will delve deeper into the approach we favoured for our project. In Section 5 (Results) we present the scores as a result of an experiment involving several possible combinations that might be work with the Two Stage Classifier.

1. Introduction

Sentiment Analysis (sometimes known also as opinion mining) is a branch of Natural Language Processing. Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine.
In this project we will focus on carrying out two major tasks of sentiment analysis, namely **Subjectivity Detection** and **Polarity Classification**. Our go-for approach – which is the one showing best results on polarity classification also over rather shallow deep learning architectures – is a Two Stage Classifier structure. Simplistically we use a classifier trained on `nltk`'s `movie_reviews` dataset to detect whether subjectivity is present within a sentence (in this case a movie review).

The repository structure to which this report file is attached is structured as follows:

```
📂 repo_root
    📂 data
        📁 movie_reviews
        📁 output
        📁 rotten_imdb
        📁 subjectivity
    📂 src
        📜 utils.py
    📜 01-preprocessing.ipynb
    📜 01-tfidf.ipynb
    📜 02-main.py
```

For the complete code please refer to the GitHub repository.

2. Problem Formalization

With this project we want to be able to effectively classify documents as *subjective* or *objective*. Such a task is intuitively referred to as subjectivity detection. In addition, we want to grasp the humor a writer wants to convey – namely *positive* or *negative*. The second task goes by polarity detection.
Despite Subjectivity and Polarity detection might be addressed in two different ways, we provide proof of evidence that carrying out both tasks in parallel yields better results.

3. Data Description & Preprocessing

This section illustrates the resources used for carrying out sentiment analysis tasks.

3.1. Subjectivity Detection

The dataset used to carry out subjectivity detection at sentence-level is conveniently available within `nltk` library as `'sentiment'`, for a complete overview of the dataset please refer to cornell.edu. It is composed of 5000 subjective and 5000 objective sentences.

3.2. Polarity Detection

The dataset used to carry out polarity detection at document-level is the one from Pang and Lee [1], and is as well available within `nltk` library as `'movie_reviews'`. For a complete overview of the dataset please refer to cornell.edu. It is made of 1000 positive reviews and 1000 negative reviews picked from the larger IMDB movie reviews dataset.
In addition, for the implementation of the *DiffPosNeg* filtering block – as proposed in our mentoring paper from Nguyen and Pham [2] – we use Esuli and Sebastiani's [3] *SentiWordNet* resources available within `nltk`. SentiWordNet is a lexical resource for opinion mining. SentiWordNet assigns to words – and synonyms that take on common meanings under some context – a *positivity* score, a *negativity* score, and an *objectivity* score.

4. Approach

Initially, to work out the Subjectivity Detection problem, we test a diverse number of deep learning structures on the Movie Reviews dataset. In particular, we start off with a Multi Layer Perceptron which we consider as the lowest bound (*accuracy:* 92.5%, *recall:* 61.0%). The second architecture we test is an LSTM (Long Short Term Memory) recurrent neural network. This has been implemented with and without an *Attention* module. Results with and without the Attention Module are not far from each other (*accuracy:* $\sim 88\%$, *recall:* $\sim 59\%$). The

last architecture amongst the ones with deep learning capabilities is an interesting CNN featuring an unconventional three input-convolution implementation as shown in Satapathy and Pardeshi [4] paper. Results for this last neural network are as follows : *accuracy:* 94.9*%, recall:* 61.1*%*. Table 1 sums up results for deep learning techniques.
Because we are trying to classify documents (or more broadly some text) to be either positive or negative, subjectivity detection can be considered to be a **binary classification** task. The analysis of above results – high training accuracy and low recall – provide sufficient proof of evidence to state that such architectures have a strong overfitting flare. Options to attack overfitting issues with neural networks are to either feed them with more data (and subsequently increase the number of features to be detected) or not use neural networks at all. Because using `nltk`'s Movie Reviews and Subjectivity datasets is a requirement, we decided to work our way back and drop deep learning completely.

Model	Accuracy	Recall
MLP	92.5%	61.0%
LSTM	88.3%	59.5%
LSTM + Attention	88.4%	59.1%
BERT Embedding	94.9%	61.1%

Table 1: *Table showing accuracy vs recall results for Deep Learning architectures trained on* `nltk`*'s Movie Reviews dataset.*

With Nguyen and Pham [2] paper mentoring us, and aiming for state-of-the-art comparable results, we build a Two Stage Classifier composed of a *filter* called DiffPosNeg, a Naïve Bayes classifier, and a Support Vector Machine Classifier.

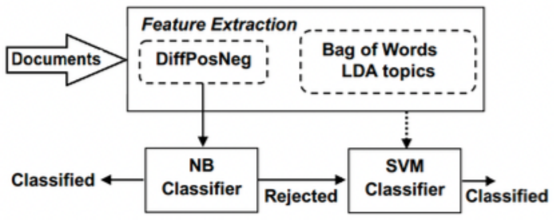

Figure 1: *Original structure of the Two-Stage Classifier as proposed in Nguyen and Pham paper.*

As depicted in Figure 1, documents are initially fed to a Naive Bayes (NB) Classifier which is in charge of categorizing documents. NB goes about classifying a document by extracting **high level** (or document-level) features from documents through a filtering criteria based on a set of two thresholds, namely:

$$\begin{cases} \tau_{pos} > P\left(pos|D\right) \wedge P\left(pos|D\right) \geq P\left(neg|D\right) \\ \tau_{pos} \in [0,1] \end{cases} \tag{1}$$

OR

$$\begin{cases} \tau_{neg} > P\left(neg|D\right) \wedge P\left(neg|D\right) \geq P\left(pos|D\right) \\ \tau_{neg} \in [0,1] \end{cases} \tag{2}$$

Simply put, τ_{pos} and τ_{neg} are thresholds to set the minimum confidence level NB should have to correctly classify a document D as *positive* or *negative*, respectively. In case such criteria are not met, the document in analysis is classified as *rejected* and passes on to the second and last classification stage: SVC (Support Vector Classifier). At this step, features are extracted at a **low level**, in particular at a sentence level – if feeding the SVC with a BoW feature – or at token-level – if feeding the SVC with token-level features such as TF*IDF.

In the following paragraphs we explain in greater detail the how features are extracted from text and what do they mean:

4.1. DiffPosNeg

DiffPosNeg representation – as explained in Nguyen and Pham [2] paper – is defined as the absolute numerical distance between those sentences with a positive 'orientation' and those with a negative 'orientation'. Our version of the DiffPosNeg implementation leverages Esuli and Sebastiani's [3] *SentiWordNet* as learned in one of the NLU class laboratories held by Evgeny A. Stepanov. We opted for SentiWordNet because it proved to work just as good as the original paper's construction choice, but more than that it conveniently allows for a token-level implementation. Polarities can then then combined, producing 'orientations' at sentence level and even further at document level.

4.2. BoW

Whenever any of the requirements set by equations (1) and (2) are not met, then NB classifier rejects the document and branches it off to the SVM Classifier (SVC). At this point – as mentioned at the beginning of this section – SVC is used to extract features at sentence level. To properly carry out such a task we feed it with an additional feature that is BoW (Bag of Words).
BoW representation describes some text as a set of words (*bag*) disregarding grammar – and even word order – but keeping track of the number of times each word appears within a bag (*multiplicity*).
With BoW we represent each sentence as a binary vector being 1 if the word appears within the sentence in analysis, 0 otherwise. We tested such an approach by performing a 10-fold Cross-Validation on both a Multinomial Naïve Bayes classifier and a Bernoulli Naïve Bayes Classifier[1] and with two different representation types. Test results are provided in Table 2.

	Multinomial NB		**Bernoulli NB**	
	count	**tfid**	**count**	**tfid**
score	0.96	0.97	0.95	0.95

Table 2: *10-Fold Cross-Validation results for both Multinomial and Bernoulli Naïve Bayes*

[1]Tests have are performed on Cornell *Subjectivity* dataset.

4.3. TF*IDF

TF*IDF is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. In practice, for each token we extract this value using **`nltk`**'s word tokenizer, along with its *negation* feature, its *part-of-speech* – as proposed in the Universal Tagset [5] research paper – as well as its *positional encoding*.
We were guided through tokel-level feature extraction by Kamal's research paper [6] which tries to produce [*subjective*, *objective*] labels for a sentence based on the set of tokens (words) it is composed of. Such an experiment showed to be the opposite of effective on our dataset, probably due to the meager size of it. However – after a not quite straightforward fine-tuning – we were able to tweak and turn the right knobs and obtain results that yield an average F1 Score of 56% on a 10-Fold Cross-Validation.
Please refer to repository structure [1] to find the code for such experiment.

4.4. DimRed

As for our fourth and last parameter, we implemented a dimensionality reduction module to assess whether it might positively or negatively affect the SVM Classifier. To perform dimensionality reduction we use Linear Discriminant Analysis (LDA).

5. Results

In this section we will show and describe results yield by different experiments. In particular, we tested our implementation of the Two-Stage Classifier with the list of parameters described below. The code implementation can be found in the original repository.

- `first-stage vectorizer`: method used with the Naïve Bayes Classifier (first classification stage). First stage vectorizer can be either *bow* or *diffposneg*, hence using *BoW* feature or *DiffPosNeg* feature respectively.
- `second-stage vectorizer`: method used with the SVC Classifier (second classification stage). Second stage vectorizer can be ⟨*bow*, *tfidf*⟩ when first stage vectorizer is set to *diffposneg*, but only *tfidf* when first stage vectorizer is set to *bow*. The reason behind such a choice is
- `subjectivity detector`: method intuitively used to perform subjectivity detection. The parameter can be set to either *filter* or *aggregate*. Whenever it is the case it assumes the value **filter**, then sentences within documents are rejected before being passed over to polarity classification (the second stage). Otherwise, subjectivity for a document is extracted as the result of a vote based on the subjectivity labels of the document's inner sentences. Note that the latter setting treats subjectivity detection as a proper **feature extraction step** as each additional feature component gets added to each document's feature vector.
- `reduce dimensionality`: boolean parameter allowing dimensionality reduction or not. As a *remark*, whenever we find ourselves with the combo

  $$\langle Reduce\ Dim,\ Subj\ Det\rangle == \langle True,\ aggregate\rangle$$

  dimensionality reduction is applied before extracting the subjectivity features at document level.
  Mathematically, the extracted feature results in a mono-dimensional feature vector described by the value of the projection direction yield by LDA classifier and the subjectivity label.

Experimental results are provided in Table 3 as a result of running each and every possible combination of the parameters listed above.

Driven by pure inquisitive nature, we ran the same code on two different machines with two different CPU architectures and obtained the results in terms of time shown in Table 4.

6. Discussion

With this project, we aim at addressing Subjectivity Detection and Polarity Classification, two tasks belonging to the branch of Sentiment Analysis. The way we carried out such tasks is through the implementation of a Two-Stage Classifier featuring a Naïve Bayes Classifier and a Support Vector Machine Classifier. At the first stage NB classifies documents which are categorized as 'easy-to-classify'. 'Hard-to-classify' documents – hence those ones that are most likely to be miss-classified during stage one – are instead handed over to the more capable SVM classifier once rejected.

Examination of results in Table 3 providing several combinations of parameters for the Two-Stage Classifier, thoughts and considerations that arise are the following:

- although being *DiffPosNeg* a noteworthy feature due to its simple implementation, it proves to be less effective than a considerably more structure *BoW* representation. Geometrically, points of interest for the classifier are further spatially spread whenever represented as BoW rather than DiffPosNeg, making it easier for the classifier to produce an accurate result. However let us remind that the higher the representation dimensionality, the greater the computational effort.
- SVM classifier yields best results whenever $\langle Reduce\ Dim,\ Subj\ Det\rangle == \langle True,\ aggregate\rangle$. Note that with this setting results are persistent no matter what parameters for *vect 1* and *vect 2* are set to. With reference to what has been said for such a combo in Section 4.4, we can clearly state that the quality of feature has a meaningful impact on performance.
- computation times on both CPUs show to be significant unburdened by LDA dimensionality reduction getting also higher F1 scores. However – as highlighted at the previous observation –
- the Two-Stage Classifier often yields similar results to either NB or SVC in terms of F1 Score. What this means is that documents are always completely handed over to second stage or directly classified at first stage, there is no in between. Below are fragments of the log file (available in

	Vect 1	Vect 2	Subj Det	Red Dim	Two-Stage F1	NB F1	SVC F1
1	diffposneg	bow	aggregate	True	0.52	0.52	0.91
2	diffposneg	bow	aggregate	False	0.73	0.52	0.73
3	diffposneg	bow	filter	True	0.52	0.53	0.89
4	diffposneg	bow	filter	False	0.75	0.52	0.75
5	diffposneg	tfidf	aggregate	True	0.49	0.52	0.90
6	diffposneg	tfidf	aggregate	False	0.83	0.52	0.83
7	diffposneg	tfidf	filter	True	0.50	0.51	0.90
8	diffposneg	tfidf	filter	False	0.87	0.52	0.87
9	bow	tfidf	aggregate	True	0.81	0.81	0.90
10	bow	tfidf	aggregate	False	0.81	0.81	0.83
11	bow	tfidf	filter	True	0.85	0.85	0.90
12	bow	tfidf	filter	False	0.85	0.85	0.87

Table 3: *Table showing experimental results yield by running each and every possible combination of the parameters listed in the Results* 5 *section.*

	Intel i9			Apple Silicon		
	Two-Stage Elapsed	NB Elapsed	SVC Elapsed	Two-Stage Elapsed	NB Elapsed	SVC Elapsed
1	01*m*0*s*	00*m*37*s*	00*m*37*s*	00*m*16*s*	00*m*06*s*	00*m*09*s*
2	04*m*12*s*	00*m*33*s*	22*m*22*s*	05*m*01*s*	00*m*06*s*	05*m*16*s*
3	00*m*21*s*	00*m*27*s*	00*m*05*s*	00*m*07*s*	00*m*09*s*	00*m*02*s*
4	04*m*15*s*	00*m*50*s*	21*m*52*s*	05*m*03*s*	00*m*09*s*	08*m*34*s*
5	00*m*43*s*	00*m*24*s*	00*m*19*s*	00*m*16*s*	00*m*06*s*	00*m*09*s*
6	02*m*47*s*	00*m*25*s*	02*m*22*s*	05*m*50*s*	00*m*06*s*	06*m*24*s*
7	00*m*21*s*	00*m*27*s*	00*m*05*s*	00*m*07*s*	00*m*09*s*	00*m*02*s*
8	02*m*34*s*	00*m*31*s*	02*m*14*s*	05*m*32*s*	00*m*09*s*	05*m*25*s*
9	00*m*09*s*	00*m*11*s*	00*m*25*s*	00*m*04*s*	00*m*04*s*	00*m*09*s*
10	00*m*10*s*	00*m*09*s*	06*m*32*s*	00*m*04*s*	00*m*04*s*	05*m*58*s*
11	00*m*06*s*	00*m*37*s*	00*m*07*s*	00*m*02*s*	00*m*07*s*	00*m*02*s*
12	00*m*05*s*	00*m*20*s*	01*m*47*s*	00*m*05*s*	00*m*07*s*	05*m*31*s*

Table 4: *Inquisitiveness – time results for two different CPU structures. Note that we use scikit-learn (or scikit-learn dependent) libraries to carry out both training and evaluation tasks. As scikit-learn does not yet support GPU implementation both codes are ensured to have run fully on CPU resources.*

the complete repository at the path '*data/output*') showing that second stage is usually called either 100% of the time or 0.8% of the time.

Listing 1: *log fragment*

```
DiffPosNeg—Transformed 400 documents in 3s
Two Stage Clf—SVC was called 100% of the time
...            ...            ...
Two Stage Clf—Done fitting in 3m38s
Two Stage Clf—SVC was called 2% of the time
```

7. References

[1] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity," 2004, pp. 271–278.

[2] S. B. P. D. Q., D. Q. Nguyen, "A two-stage classifier for sentiment analysis," Ph.D. dissertation, Vietnam National University, Hanoi, jun 2013.

[3] *SENTI-WORDNET: A publicly available lexical resource for opinion mining.*, Genoa, Italy, Aug./Sep. 2006.

[4] E. C. R. Satapathy, S. R. Pardeshi, "Polarity and subjectivity detection with multitask learning and bert embedding," *Special Issue Machine Learning Perspective in the Convolutional Neural Network Eras*, Feb 2022.

[5] S. Petrov, D. Das, and R. McDonald, "A universal part-of-speech tagset," 2011. [Online]. Available: https://arxiv.org/abs/1104.2086

[6] A. Kamal, "Subjectivity classification using machine learning techniques for mining feature-opinion pairs from web opinion sources," 2013. [Online]. Available: https://arxiv.org/abs/1312.6962