Final NLU project report

Matteo Guglielmi (232088)

University of Trento

matteo.guglielmi@studenti.unitn.it

Abstract

This work focuses on explaining the salient steps taken to implement a Sentiment Analysis algorithm that consists in *Subjectivity* and *Polarity Classification*.
More in detail, a custom model is compared to a baseline in order to highlight the improvements achieved. To carry out this project, three different datasets were used : the "*MovieRewievs*" and "*Subjectivity*" datasets both downloaded from the NLTK python module and the public "*IMDB*" dataset dowloaded from Kaggle[1].

1. Introduction

Sentiment analysis consists in performing sentence classification with the final goal to predict whether a sentence/review/comment expresses a positive or negative sentiment.
Sentiment analysis is a very important task in the field of Natural Language Processing (NLP) and has been studied for many years. It is a very challenging task because of the presence of negation, sarcasm, irony, and other linguistic phenomena that make the task very difficult.
From a commercial point of view, sentiment analysis is a very important task because it can be used to automatically analyze the opinion of the customers about a product or a service. For the latter reason, many industries are interested in this specific applicatio to understand the degree of appreciation a specific product has.

As briefly mentioned before, in this work I'll be comparing two different models to perform Sentiment and polarity classification.

1.1. Baseline

The baseline model consists in :

- a *CountVectorizer*[2], which acts as an encoder, to extract the features from the text;
- a *Naïve Bayes*[3] classifier to perform the final classification.

It is worth mentioning that both datasets were pre-processed applying double negative marking to consider double negations (double negations corresponds to a neutral effect).
More in detail, the step-by-step procedure followed to perform the classification is the following:

1. Pre-processing: the text is pre-processed by collapsing the severale sentences in documents and the double negations are marked
2. Feature extraction: the text belonging to the Subjectivity dataset is encoded using a *CountVectorizer* which counts the number of occurrences of each word in the text;
3. Classification: the encoded text from the previous point is classified using a *Naïve Bayes* classifier;
4. Evaluation: the classification is evaluated using a 10-fold cross-validation precedure exloiting the *accuracy* metric to address the overall performances;
5. Filtering : the trained model is used to filter the objective sentences from the Movie Review dataset;
6. Feature extraction: the text belonging to the Movie Review dataset is encoded using a *CountVectorizer* as point (2);
7. Classification: the encoded text from the previous point is classified using a *Naïve Bayes* classifier as point (3);

1.2. Proposed model

The proposed model follows the same logical concept as the baseline but with a completely different approach.
Very briefly, the proposed model consists in :

- the *Bert Tokenizer*[4] to extract the features from the text and the *BertModel*[5];
- a shallow *BiLSTM* network used to filter objective movie reviews from the Movie Review dataset, accepting the output of the previous point as input;
- a *(*BertForSequenceClassification) to perform the final classification on the MovieReviews dataset as a whole.

2. References

[1] "IMDB Dataset of 50K Movie Reviews," https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews.

[2] "sklearn.feature_extraction.text.CountVectorizer," https://scikit-learn.org/stable/modules/generated/sklearn.featurei_extraction.text.CountVectorizer.

[3] "sklearn.naive_bayes.MultinomialNB," https://scikit-learn.org/stable/modules/generated/sklearn.naivei_bayes.MultinomialNB.htmlsklear

[4] "BertTokenizer," https://huggingface.co/docs/transformers/modeli_doc/berttransform

[5] "BertModel," https://huggingface.co/docs/transformers/v4.21.1/en/model_doc/berttra