Sentiment Analysis on IMDB Movie Reviews

Evaluation and comparison of multi machine learning approaches to predict a binary sentiment of a movie review

Roman Studer
BSc Data Science Student,
FHNW Brugg-Windisch, Switzerland
roman.studer1@students.fhnw.ch

Lukas Gehrig
BSc Data Science Student,
FHNW Brugg-Windisch, Switzerland
lukas.gehrig@students.fhnw.ch

Vincenzo Timmel
BSc Data Science Student,
FHNW Brugg-Windisch, Switzerland
vincenzo.timmel @students.fhnw.ch

***Abstract*— In this report we look at the binary sentiment analysis task of reviews (positive or negative sentiment) with different machine learning algorithms from simple linear models to deep neural networks. We show and discuss the whole process of how the data is prepared and how to interpret the results of the models.**

I. Introduction

A. Goal of the analysis

The sentiment analysis task is focused on the evaluation and comparison of several natural language processing methods for sentiment classification. For this purpose, we use a dataset with 50 thousand film reviews from the IMDB[1] platform. This dataset that is often used for benchmarks in the NLP environment . It is suitable for a binary sentiment classification as it is labeled with the binary feature "sentiment" which either takes on the value "positive", if the sentiment of the review is positive, or "negative" if the sentiment is negative [1]. An example record of the dataset would be:

Table 1, Example of a review in the IMDB dataset

Index	Review	Sentiment
124	“I like this movie”	positive

B. Process

At the start of the process, we defined several functions that allow a uniform pre-processing of the data. This ensures that all employed models receive the same data as input for training and test purpose. In chapter II we describe the pre-processing in more detail. With the pre-processed data, several models were trained and their accuracy on train and test set is recorded. In a further step, we compared the models. Assumptions about of how the results were obtained and whether they are robust are given at the end of the report in chapter 5.

II. Preprocessing

Language offers humans a way to communicate complex issues in an adaptable and national, even universal, way. This complexity is reflected in the almost endless possibilities of language use. Since machine learning models in the NLP domain require usually numerical values as input, a transformation of the text must take place. In most cases, the individual words of a sentence are divided into a list and each of them is transformed into its basic form. The process of transformation into the basic form is called stemming. Stemming (stem form reduction, normal form reduction) is the term used in information retrieval as well as in linguistic computer science to describe a procedure
by which different morphological variants of a word are reduced to their common root (e.g., the declension of “Wortes” to “Wort” and conjugation of "gesehen" or "sah" to "seh"). Models like BERT or T5 can also use hardly pre-processed texts as input [2]. We performed the following steps on the IMDB dataset:

1. Removing the HTML tags. These are not relevant for understanding the input.
2. removing brackets with text (it seems that bracket text was a hyperlink or quote or reference).
3. removing special characters, e. g. ?,!,/
4. removing stop words. Stop words are common or “filler” words, which contain little information. For example, connection words. They are removed.
5. put words into the basic form with the Porter-Stemmer-Algorithm, which applies multiple handwritten rules to reduce the word-length. Some examples: `likes`, `liked`, `likely` and `liking` will all be reduced to `like`.

For transformer models like BERT, step 4. and 5. were not applied. We performed an numerical encoding of the binary target variable “sentiment”. A positive review is now annotated with the integer 1 where a negative review is annotated with the integer 0. Table 2 shows an example of the prepared text:

Table 2, Example of a review after preprocessing

Index	Review	Sentiment
15594	[“saw”, “thi”, “movi”, “wa”,..]	0

A. Train-Test-Split

For reproducibility, we provide a function which returns the exactly same split of input data for all models. The 50 thousand reviews are split into a train-set of 40 thousand reviews (80%) and a test-set of 10 thousand reviews (20%).

III. Modelling

A. Baseline Model

To evaluate if the ml-models are better than a simple handcrafted model, we created a simple baseline model based on some simple rules. The approach of the baseline model is to count positive and negative words in a review and suggest a positive or negative sentiment based on this count. For this, we imported a list of positive words as well as a list of negative words. Since there is no learning algorithm behind

[1] IMDB: Internet Movie Database

this method, the score was calculated on the test-set only. For this method, both the list of negative and positive words and the IMDB-Dataset was pre-processed with all the steps mentioned in Chapter 2.

B. Evaluation metric

Because the train- and test-set are almost perfectly balanced, we use the accuracy and a confusion matrix as metric, to see whether our models handle one class much better than the other.

C. Matrix representation

1) Bag of Words

Bag of Words, a method often used for document classification based on the frequency of terms, converts a corpus into a vector with the token as keys and their frequency as elements. This can be converted into a matrix where the row represents the respective review frequency vector, and the column represents the corresponding word. Grammar and arrangement of input is not included in this model. In the train-set, a total of 156'200 unique words were counted after pre-processing. We used Bag of Words as input of a Logistic Regression, SVM as well as Multinomial Naive Bayes model.

2) TF-IDF

TF-IDF stands for Term Frequency - Inverse Document Frequency and is calculated for each word in the corpus. This value gives an insight into the importance of a term in the according document. The term frequency, i.e. the relative frequency of a term in a document combined with the relative, logarithmic frequency over all documents (see formula 1) gives words that occur frequently and, in many documents, less weight than words that only occur in a few documents.

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

$$\text{idf}(t,D) = \log \frac{N}{\left| \text{d} \in \text{D} : \text{t} \in \text{d} \right|}$$

Where TF-IDF is a combination of the two formulas:

$$tf - idf(t,d,D) = tf(t,d) \cdot idf(t,D)$$

The result of TF-IDF, using sklearn, is a matrix with the dimensions $(n_{samples}, 156'136)$, when only *unigrams* were applied. This is very close to the number of unique words. It is not exactly the same, because **sklearn.feature_extraction.text.TfidfVectorizer/CountVectorizer** uses a regex pattern, which was able to extract words attached or surrounded by either // or ^, while our method was not. Our method to remove special characters could be improved for future uses.

D. Classic models

In addition to the baseline model, we used some classical machine learning algorithms at the beginning. These included logistic regression with L2 regularization, support vector machines with hinge loss and a multinomial Naive Bayes model. Accuracy is measured for the three models using Bag of Words and TF-IDF as input. The best regularization-factor for each model was found via a 5-fold cross validation grid search, using **sklearn.model_selection.GridSearchCV**.

E. Deepl Learning Models

As an extension, we trained two deep learning models. The data we used for them were prepared with the same preprocessing steps as described in chapter II.

1) Bidirectional LSTM

Our LSTM model uses a pre-trained word embedding. Due to memory constraints, the fastText mini version was used for this case. GloVe (Global Vectors) as another word embedding, was also imported for comparison. Thus, two models with different embeddings were trained. The LSTM model was trained with a maximum review length of 128, a training batch size of 16, validation batch size of 6 and with 5 epochs. It takes the hidden state after the whole sequence (forward and backwards) and and scales reduces it's dimension using pooling or flattening. Afterwards a linear-layer is applied to get the output value.

2) BERT

Another language model, BERT, was trained on a corpus with 800 million words from BooksCorpus and 2,500 million words from Wikipedia. The input data was prepared as in Chapter II, except for steps 4 and 5. A maximum length of 512, a batch size of 16 was used for 4 epochs.

Accuracy was recorded for all models on train- and test-set. The results are discussed in the following chapter.

IV. Evaluation

The evaluation of the models was performed by the accuracy. It indicates the relative frequency of a correct prediction. A well-known metric for a binary classification purpose. For learning algorithms such as bidirectional LSTMs or BERT, in addition to the final score, the progression of the metric over the training epochs is interesting and important for detecting overfitting.

Model	**(min_df, max_df)**	**N_gram Range, Type**	**Accuracy**	
			Train	**Test**
Baseline	-	-	-	0.7131
LR (TF-IDF)	(0, 1)	(1,3), word	0.99628	0.6061
LR (BoW)	(0, 1)	(1,3), word	0.9963	0.715
SVM (TF-IDF)	(0, 1)	(1,3), word	0.99628	0.7485
SVM (BoW)	(0, 1)	(1,3), word	0.99626	0.5091
MNB (TF-IDF)	(0, 1)	(1,3), word	0.99626	0.7512
MNB (BoW)	(0, 1)	(1,3), word	0.99985	0.8857
LR (TF-IDF)	(0.001, 0.999)	(1,3), word	0.9294	0.8986
LR (BoW)	(0.001, 0.999)	(1,3), word	0.9645	0.8925
LR (TF-IDF)	(0.001, 0.999)	(3,3), word	0.6702	0.6569
LR (BoW)	(0.001, 0.999)	(3,3), word	0.66968	0.6552

SVM (TF-IDF)	(0.001, 0.999)	(3,3), word	0.9313	0.899
SVM (BoW)	(0.001, 0.999)	(3,3), word	0.9671	0.8938
SVM (TF-IDF)	(0.1, 0.9)	(3,3), word	0.768	0.7632
SVM (BoW)	(0.1, 0.9)	(3,3), word	0.7678	0.7625
SVM (TF-IDF)	(0.001, 0.999)	(1,4), word	0.93145	0.8991
SVM (BoW)	(0.001, 0.999)	(1,4), word	0.96698	0.8936
SVM (TF-IDF)	(0.001, 0.999)	(1,4), char	0.9112	0.8732
SVM (BoW)	(0.001, 0.999)	(1,4), char	0.92257	0.8647
SVM (TF-IDF)	(0.001, 0.999)	(1,3), char, word	0.93887	0.9013
MNB (TF-IDF)	(0.001, 0.999)	(1,3), word	0.89095	0.8727
MNB (BoW)	(0.001, 0.999)	(1,3), word	0.8757	0.8642
MNB (TF-IDF)	(0.001, 0.999)	(1,4), char	0.82485	0.8142
MNB (BoW)	(0.001, 0.999)	(1,4), char	0.78382	0.7775
LSTM (fastText)	-	1, word	0.9514	0.8731
LSTM (GloVe)	-	1, word	0.9355	0.8739
BERT	-	1, word	0.87663	0.7186

We see a high score for the training set and a significantly lower score for the test set in most models. This indicates overfitting of these models. This could have been addressed by either introducing regularization to the models which haven't already been regularized or by hyperparameter tuning. However, we focused more on the comparison of different models and their configurations rather than optimizing by any means. It is unexpected that the BERT model achieved similar accuracy compared to the baseline model. This is partly because the model is overfitted. Training with a higher batch size, in addition to a high dropout rate could mitigate this. However, due to hardware constraints, this option could not be explored. SVM with both word- and char- n_grams with a range of (1,3) provides by far the best result with an improvement of over 19% over the baseline.

SVM, Multinomial Naive Bayes, Logistic Regression and LSTM all provide a very good result. However, at the beginning, there was an error with the parameter *min_df* and *max_df* in **sklearn.feature_extraction.text**; from the notebook [3] we based part of this mini challenge on, the values for *min_df/max_df* were set on 0 and 1, respectively. Because we ignore all words which appear at most in 1 document, we were only able to overfit on the training-data. However using meaningful parameters for LR, SVM and MNB we were able to achieve good results. There also seem to be combinations of Models and Word-Vectorizing methods, which are very robust to *min_df = 0 and max_df = 1* which could lead to some interesting findings. We also think, by optimizing the LSTM-models with regularization and hyperparameter tuning, they could have reached or even exceeded the other models.

REFERENCES

[1] "Sentiment Analysis." http://ai.stanford.edu/~amaas/data/sentiment/ (accessed Oct. 31, 2021).

[2] "nlp - Using trained BERT Model and Data Preprocessing," *Stack Overflow*. https://stackoverflow.com/questions/63979544/using-trained-bert-model-and-data-preprocessing (accessed Nov. 03, 2021).

[3] "Sentiment Analysis of IMDB Movie Reviews" *kaggle* https://www.kaggle.com/lakshmi25npathi/sentiment-analysis-of-imdb-movie-reviews (accessed Nov. 09, 2021).

Tables

Table 1, Example of a review in the IMDB dataset1
Table 2, Example of a review after preprocessing1