Sentiment Analysis Challenge

Fandio Njylla Esdras E.
4th year in Computer Science at National Advanced School of Engineering Yaounde, CM,
fandioemma@gmail.com

1 Introduction

The analysis of customer/user opinions and comments is a real problem that most companies face. Indeed, when a product is released, customers usually provide opinions in the form of a paragraph. The analysis of these comments can quickly become a rather heavy task and to do so may require the creation of a team, which is a significant cost for the company. To overcome this problem, researchers have designed and implemented various models to highlight the feelings (positive, negative, neutral...) of a given text, models that have proven to be very reliable in the sense that the state-of-the-art in this field presents models exceeding 95 % (https://paperswithcode.com/sota/sentiment-analysis-on-imdb). Our work consisted in establishing and implementing methods of sentiment analysis based on the IMDB Movie Reviews dataset. The github repository https://github.com/Esdras123/Sentiment_analysis_of_IMDB_reviews.git contains the different notebooks we created and this report details the progress of this work.

2 Dataset

2.1 Description

The dataset on which our study was based is the IMDB Movies Review Dataset. It contains a list of 50,000 film reviews. 25,000 are labelled as positive (positive reviews) and 25,000 are labelled as negative (negative reviews). After observing the data, it appears that no value in the dataset is null and that the length of the reviews varies a lot (there are reviews written in one sentence and others written in several paragraphs). The methods to be used must therefore be able to keep the links between the different words even if they are far apart. The analysis of the notices in the dataset shows that there is a lot of useless data in these texts: usual words (in, of, the, at, ...), html tags (¡br¿...), special characters,... One step of the different methods will therefore be to clean up this data.

2.2 Data preprocessing

Several steps are necessary when pre-processing data :

- Removal of HTML tags, noisy text and special characters
- Removal of common words from paragraphs (stopwords): It is important to note that here we used a small set of english stopwords ('in', 'of', 'at', 'a', 'the') rather than using the big set of english stophwords because sometimes removing all the english stopwords reduce the performance of the model.
- Stemming: Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes.

After this pre-processing, the paragraphs and sentences describing the notices were transformed into token sequences. Tokenization is essentially splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms. Each of these smaller units are called tokens.

2.2.1 Data split in training/validation and test sets

To divide the dataset in training/validation and test sets, I used the model_selection method from sklearn with the commands:
X_train, X_test, y_train, y_test = train_test_split(df['review'], df['sentiment'], test_size=0.2, random_state=1)

X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.25, random_state=1) (0.25 x 0.8 = 0.2)
We split the data in 60% for training data, 20% for validation data and 20% for test data.

3 Performance metrics used

The problem we're facing here is a classification problem. We have therefore chosen precision as a metric because generally for classification problems, what matters is the probability that the input is well classified.

4 Approaches

After studying different reports and research, we decided to study 2 NLP methods: the BERT method and the combination of Feature extraction, ngrams model and Linear svm method.

4.1 BERT method

4.1.1 What is BERT ?

BERT is an open source machine learning framework for natural language processing (NLP). BERT is designed to help computers understand the meaning of ambiguous language in text by using surrounding text to establish context. The BERT framework was pre-trained using text from Wikipedia and can be fine-tuned with question and answer datasets.
BERT, which stands for Bidirectional Encoder Representations from Transformers, is based on Transformers, a deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based upon their connection. (In NLP, this process is called attention.)
Historically, language models could only read text input sequentially – either left-to-right or right-to-left – but couldn't do both at the same time. BERT is different because it is designed to read in both directions at once. This capability, enabled by the introduction of Transformers, is known as bidirectionality.
Using this bidirectional capability, BERT is pre-trained on two different, but related, NLP tasks: Masked Language Modeling and Next Sentence Prediction.

4.1.2 How it works ?

BERT was pre-trained using only an unlabeled, plain text corpus (the entirety of the English Wikipedia, and the Brown Corps) and it continues to learn unsupervised from the unlabeled text and improve even as its being used in practical applications (ie Google search). Its pre-training serves as a base layer of "knowledge" to build from. From there, BERT can adapt to the ever-growing body of searchable content and queries and be fine-tuned to a user's specifications (transfer learning).
BERT is also the first NLP technique to rely solely on self-attention mechanism, which is made possible by the bidirectional Transformers at the center of BERT's design. This is significant because often, a word may change meaning as a sentence develops. Each word added augments the overall meaning of the word being focused on by the NLP algorithm. The more words that are present in total in each sentence or phrase, the more ambiguous the word in focus becomes. BERT accounts for the augmented meaning by reading bidirectionally, accounting for the effect of all other words in a sentence on the focus word and eliminating the left-to-right momentum that biases words towards a certain meaning as a sentence progresses.

4.1.3 What is BERT used for ?

BERT is currently being used at Google to optimize the interpretation of user search queries. BERT excels at several functions that make this possible, including:

- Sequence-to-sequence based language generation tasks such as:
  - Question answering
  - Abstract summarization
  - Sentence prediction
  - Conversational response generation
- Natural language understanding tasks such as:
  - Polysemy and Coreference (words that sound or look the same but have different meanings) resolution
  - Word sense disambiguation
  - Natural language inference
  - Sentiment classification

4.1.4 Strength of BERT

- Truly Bidirectional: BERT is deeply bidirectional due to its novel masked language modeling technique.

- Model Input: BERT tokenizes words into sub-words (using WordPiece) and those are then given as input to the model.

- Uses Transformers: using transformers enables the parallelization of training which is an important factor when working with large amounts of data.

- Uses self-attention

4.2 The combination of Feature Extraction, ngram model and Linear svm Model

4.2.1 Feature extraction

Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing. A characteristic of these large data sets is a large number of variables that require a lot of computing resources to process. Feature extraction is the name for methods that select and /or combine variables into features, effectively reducing the amount of data that must be processed, while still accurately and completely describing the original data set.
The process of feature extraction is useful when you need to reduce the number of resources needed for processing without losing important or relevant information. Feature extraction can also reduce the amount of redundant data for a given analysis. Also, the reduction of the data and the machine's efforts in building variable combinations (features) facilitate the speed of learning and generalization steps in the machine learning process.

This process is usually used for autoencoders, bag of words and image processing.

4.2.2 N-grams model

In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sample of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. The n-grams typically are collected from a text or speech corpus.
An n-gram model is a type of probabilistic language model for predicting the next item in such a sequence in the form of a (n 1)–order Markov model. n-gram models are now widely used in probability, communication theory, computational linguistics (for instance, statistical natural language processing), computational biology (for instance, biological sequence analysis), and data compression. Two benefits of n-gram models (and algorithms that use them) are simplicity and scalability – with larger n, a model can store more context with a well-understood space–time tradeoff, enabling small experiments to scale up efficiently.

4.2.3 Linear SVM

SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the data into classes.
At first approximation what SVMs do is to find a separating line(or hyperplane) between data of two classes. SVM is an algorithm that takes the data as an input and outputs a line that separates those classes if possible.

4.2.4 Combination of those methods

The second method we have implemented is a combination of the different concepts presented above. The combination is as follows:
Firstly, we did a feature extraction which already reduced the size of each of the texts making up the data by extracting the important features.Then, we transformed this input data into bags of 3 words maximum (we did not create bigger bags in order to reduce the processing time, and to reduce the size of the data). Then using sklearn's CountVectorizer, we transformed each of the input data into different vectors. At the end, we simply applied the classification model on the data with classes 0 (for negative) and 1 for positive using LinearSVC.

4.2.5 Strength and Weaknesses

- **Strength :** This method permits to obtain a good accuracy quickly (the training does not take too much time

- **Weaknesses :** This method does'nt work well when the data in input are long texts because bags of word does'nt not capture greatly relations between words in sentences.

5 Observed Results

For each of the two methods used, we trained the model first with non-pre-processed data and then with the pre-processed data according to the principle given in I. The observed results were

5.1 BERT Method

5.1.1 With non-pre-processed data

Figure 1: Classification report

	precision	recall	f1-score	support
0	0.96	0.92	0.94	5044
1	0.92	0.96	0.94	4956
accuracy			0.94	10000
macro avg	0.94	0.94	0.94	10000
weighted avg	0.94	0.94	0.94	10000

We observed that the precision was 0.96 for negative reviews and 0.92 for positives reviews. The average precision was 0.94.

5.1.2 With pre-processed data

Figure 2: Classification report

	precision	recall	f1-score	support
0	0.88	0.96	0.92	4938
1	0.96	0.87	0.91	5062
accuracy			0.91	10000
macro avg	0.92	0.91	0.91	10000
weighted avg	0.92	0.91	0.91	10000

We observed that the precision was 0.96 for positives reviews and 0.88 for negative reviews. his difference is surely due to the fact that usually people writes longer paragraphs when they are not satisfied with a product or film than when they are satisfied and the text pre-processing affects more longer texts.

The average precision was 0.92.

5.2 Combination of Feature extraction, N-gram methods and Linear SVM method

5.2.1 With non-pre-processed data

Figure 3: Final Accuracy

```
Final Accuracy: 0.9074
```

We observed that the final accuracy was 0.9088.

5.2.2 With pre-processed data

Figure 4: Final Accuracy

```
Final Accuracy: 0.9088
```

We observed that the final accuracy was 0.9074.

6 Conclusions

After having tested the two methods below, we observed that in each case the model performs better when data is not pre-processed. It is a little bit strange because usually it is not the case.
We also observe that the BERT method performs better to solve this problem.

References

[1] https://paperswithcode.com/sota/sentiment-analysis-on-imdb

[2] https://www.analyticsvidhya.com/blog/2019/06/understanding-transformers-nlp-state-of-the-art-models/

[3] https://www.quora.com/How-can-I-measure-natural-language-processing-accuracy-Is-there-any-standard-or-industry-acceptable-criteria

[4] https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989

[5] https://towardsdatascience.com/hacking-scikit-learns-vectorizers-9ef26a7170af

[6] https://towardsdatascience.com/sentiment-analysis-with-python-part-1-5ce197074184

[7] https://towardsdatascience.com/sentiment-analysis-with-python-part-2-4f71e7bde59a

[8] https://machinelearningmastery.com/predict-sentiment-movie-reviews-using-deep-learning/

[9] https://towardsdatascience.com/understanding-bert-is-it-a-game-changer-in-nlp-7cca943cf3ad