Multi-class Sentiment Analysis Using Amazon Alexa Reviews

Machine Learning for Natural Language Processing 2020

Jing TAN **Nicolas TILQUIN**
ENSAE Paristech
jing.tan@ensae.fr nicolas.tilquin@ensae.fr

In recent years, natural language processing (NLP) has become increasingly popular in commercial use. Thanks to the contribution of researchers and computer scientists, there now exist rather rich online resources of NLP models and tool-kits. Being involved in the master's program of data science, we had the chance to have an introductory course in NLP and to acquire basic knowledge of this domain. This course ends with a practical project. With the Amazon Alexa data-set[1], we are going to analyze customers' reviews on these products and conduct sentiment analysis to classify the reviews. This project focuses especially on review text and ratings. We want to build an NLP sentiment analysis model which learns from training set samples and establish a class prediction rule to predict ratings of new reviews.

1 Problem Framing

We will process as follows:

1. First of all, we will try to understand our data-set by statistical description of data, such as the distribution of classes and text length in each class of ratings. Next we will clean text data by removing unnecessary punctuation, stop words, etc. Then tokenize and vectorize the text by word frequency in the corpus and use machine learning methods for supervised classification to classify reviews into multiple rating classes.

2. In the second part of the project, we will extract features and processing word embedding which captures word frequency as well as context in a sentence. We will apply a skip n-gram model in a CNN which is short for convolutional neural network and a pre-trained BERT transformer with a gated recurrent unit i.e. GRU. The BERT model is more adaptive to analyze the word frequency, compared to the CNN. Another advantage of this model is that it consider both forward and backward context for each words.

[1] https://www.kaggle.com/sid321axn/amazon-alexa-reviews

2 Data-set Description

Descriptive statistics[2] of the data-set shows that our data is unbalanced. We would like to see the distribution of ratings with their respective weights and found out that 72.6% of the data-set have a rating of 5.

rating	count	mean	std	min	25%	50%	75%	max
1	161.0	37.472050	40.560150	0.0	6.0	24.0	53.0	237.0
2	96.0	48.718750	52.215082	0.0	15.0	33.0	60.0	319.0
3	152.0	40.848684	52.936143	0.0	10.0	27.0	52.5	374.0
4	455.0	35.101099	42.376001	0.0	7.0	19.0	49.0	267.0
5	2286.0	21.325459	30.041677	0.0	5.0	12.0	27.0	557.0

Fig. 1: Number of samples in each class

We will come back to it more carefully during the modeling process. We also point out the fact that we have a small amount of data (i.e. 3150 observations).

Before moving on to modeling, we started with the usual NLP process of transforming our raw text into a sequence of tokens. For this purpose, we used word tokenization which is the process of separating sentences into words. In addition, it is customary to erase special characters and punctuation marks, to lowercase the whole text, to remove stop words and then we applied lemmatization techniques to keep only the radical part of the word. All this, on the one hand, allows to keep only the useful information for the modeling and to recover a vector of word. Since we are not sure of the impact of removing stop words of text, we applied a second training with stops words and compare the performance of model. It turns out that by removing stop words of the text, we obtained a better performance.

3 Models for classification

Machine learning methods

To carry out our study, we used several machine learning classifiers such for *RandomForestClassification*, *SupportVectorClassification*, and *LogisticRegression*. We observed that with the imbalanced data the random forest classifier out performed other models because that their hierarchical structure allows them to learn signals from different classes and it manages to capture the behaviors of under-

[2] https://github.com/MariahJT/NLP_Project/blob/master/NLP_SA_Amazon_Alexa%20Part%20I_%20DatasetDescriptionAndModeling.ipynb

represented classes.

We have segmented our study in two parts, the first part, we studied the characteristics of words using basic embedding methods to provide the best classification; the second part[3] is based on deep learning and neural networks such as BERT to capture a more complicated structure between words relationships.

This chapter aims to improve the sentiment classification performance by using neural networks. Firstly, we use a CNN with pre-trained weights for word embedding using "glove.6B.100d". This model will serve as a baseline for the BERT pre-trained model with a transformer later.

N-gram embedding and CNN

With text classification, we are treating sequential 1-dimensional data since text are vectorized. A 1x2 filter can look over a 2 sequential words in a piece of text, i.e. a bi-gram. We will use multiple filters of the same size which will look at the bi-grams (a 1x2 filter) and tri-grams (a 1x3 filter) within the text. And we no longer explicitly need to create the n-grams and append them to the end of the sentence. This mehod is supposed to be adaptive for short text. We use *CrossEntropyLoss* as loss function in this multi-class classification problem. *CrossEntropyLoss* performs a softmax function over our model outputs and the loss is given by the cross entropy between that and the label.

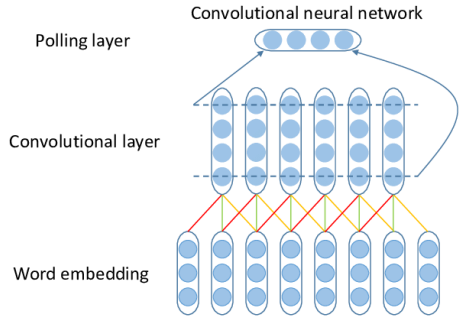

Fig. 2: Word embedding and CNN

We the parameters of the BERT model already pre trained.

BERT pretrained model

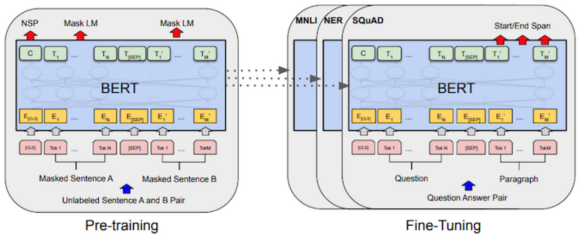

Fig. 3: Bert Transformers

For the Bert model, the transformer has already been trained with a specific vocabulary and it has appropriate tokenizers for each of the transformer models provided. In our project we are using the BERT model. We load BERT tokenizers by specifying the pre-trained tokenizer's name for example *bert-base-uncased*, and *bert-base-multilingual-uncased*. The *bert-base-multilingual-uncased* is a larger pre-trained base thus provide more noise to the training of our classification. When tested with these two models, the *bert-base-uncased* gives a much better performance.

Then we built two classification models, the *BertSentiment-Classification* with a sequence classification head on top (a linear layer on top of the pooled output), and *BertGRUSentimentClassification*. In the Bert GRU model bert transformer will be fed into a GRU to produce a prediction for the sentiment of the input sentence.

4 Results

You will find below a summary table of the implemented models. We can see from this table that the best score among machine learning models is given by logistic regression with TF-IDF as feature engineering.

Models	Accuracy	F1-score
RF with imbalanced	0.802	0.543
RF with balanced class weight	0.789	0.547
RF with SMOTE minority oversampling	0.784	0.544
RF with oversampling on all classes and balanced class weight	0.727	0.505
Logistic Regression with TF-IDF and unbalanced data	0.760	0.292
Logistic Regression with TF-IDF and RandomOverSampler	0.810	0.579
CNN with over sampling	0.881	0.377
BERT and unbalanced data	0.717	0.205
BERT GRU and unbalanced data	0.784	0.489
BERT GRU with over sampling	**0.910**	**0.884**

Table 1: Performance of different models

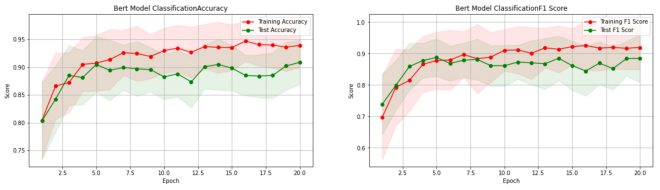

Fig. 4: Score of Bert GRU Model

5 Discussion/Conclusion

The BERT GRU model with over-sampled data obtains the best performance both in accuracy and F1 score. This model even runs faster than the naive BERT model since the gradients are not updated on forward path. We mentioned above, the main difficulties encountered : unbalanced data and small dataset. These two aspects have consequences on our algorithms. We have adopted the oversampling technique to balance our sample.

[3] https://github.com/MariahJT/NLP_Project/blob/master/NLP_SA_Amazon_Alexa%20Part%20II_%20Neural%20Networks%20.ipynb