Sentiment Analysis and Sequence Labelling using New York Times Comments
Machine Learning for Natural Language Processing 2020

Jeremy Marck
Finance
jeremy.marck@ensae.fr

Inès Deguy
Data Science
ines.deguy@ensae.fr

Abstract

Github link to our Notebook.

1 Problem Framing

This study is a prospective work aiming to perform sentiment analysis and sequence labelling using articles and comments of some articles from the New York Times. The following tasks are performed:

1. Predicting whether a comment will receive at least a positive vote or not, using `recommendations` as the target variable.
2. Building and exploring the sentiment scores distribution both for the comment database and the article one and comparing them.
3. Based on a comment, guessing the topic (using `sectionName` or `newDesk` as the target variable) of the article.
4. Predicting the nature of a comment (a comment or a user reply) using the `commentType` variable.

2 Experiments Protocol

Our data sets are "New-York Times comments" and "New-York Times articles" database available on Kaggle. We focused on articles and comments of April 2017, because of the RAM available in Google Colab when training our models. The articles database is composed of the description of more than 8,000 articles including their snippet, author information(s) and several other variables. Regarding the comments database, description of the comments are available. Articles and comments can be linked with the so-called `ArticleID` variable. The database also provides informations concerning comments contents, recommendations, informations from their authors etc... Once again, due to the oversize of the comments database (more than 240 000 comments) and computational constraints, we worked on a restricted part of the database. The choice of the restricted part is arbitrary. However, it should be noted that we tried to run our models locally, including the whole data set. As a consequence, models performances increase as the learning set is more important.

As far as this study is a NLP task, we first proceeded to a preprocessing work. We built functions cleaning the corpus and tokenizing the different sentences. We used different tokenization's library in order to identify which one is the most efficient such as NLTK or Spacy. We also built a function allowing us to acknowledge multi-words present on a sequence of tokens using the library gensim. Finally, we built functions used for performing the so-called word embedding task (Word2Vec, CountVectorizer) and functions that performed one-hot encoding and indexation.

Regarding predictive models, we used random forest, SVM, linear classifier and logistic regression objects from scikit-learn. Neural networks from Keras were also used. Some of these models have been tuned using Grid Search in order to optimize their prediction power .We also implemented our own logistic regression and neural network by using Python classes.

We trained our models splitting our data set into a train set and a test set. Our models have been trained over the train set and tested on the test set. Evaluation functions have been build to provide a quantitative/qualitative interpretation of our models.

3 Results

(1) Predicting comments likes. This part was dedicated to the prediction of the binary `recommendations` variable, stating whether a comment was received at least one like or not. Three main different classifiers were used: a Random Forest, a linear classifier and a Keras Neural Network. The linear classifier produced bad results and was not pertinent because a naive model would have produced better results, due to the fact that classes were unbalanced (77%-23%). Hence, considering accuracy as the evaluation metrics, a model is considered to be performant if it produces an accuracy greater than 0.77 on the test set. This is the case for the Random Forest and the Neural Network (around 0.85). Two main problems were faced: (1) Unbalanced classes and (2) RAM available through Google Colab. In order to increase our prediction power, next steps were dedicated to: (1) Use some technic to rebalance the classes. It did not increase accuracy a lot but increased precision! (2). Adding and building features. Prediction power increased a little bit. When pertinent models were used, we always overperformed naive predictors.

(2) Sentiment analysis. It was not a predictive task but a prospective one. Here are some conclusions from our study. We began by building a sentiment score function, taking text in input. (1) There is a huge polarization for comments sentiment scores (a lot of extreme sentiments) which is not the case for articles. (2) Articles headlines producing extreme sentiments are: Sports, music and media.(3) The sentiment is neutral for the most frequent top Keywords (meaning that most frequent keywords are objective informations, not very likely to be interpreted). (4) French political names are generally associated with neutral sentiments. Obama is associated with negative sentiments and Trump too.

(3) Guessing a topic. This problematic aims to predict the topic of a comment. We had the choice between two variables to do so (`sectionName` and `newDesk`). Using `newDesk` seemed to be more relevant to us because `sectionName` has 50% of the observations which are labelled as unknown. But due to the large number of classes (19) and the low numbers of observations we are able to use, our models were not robust. We decided to tune the target variable and remove some classes to improve our results. We deleted the labels with lowest frequency. We started to run our models with only four labels, observing that models produced satisfying results.Then, we determined how much classes were to be removed in order to get the best prediction. We found that we should use between 3 and 5 classes, associated with a 0.8 accuracy score.Logistic regression, SVM, Random Forest from scikit-learn and our own logistic regression and feedforward were used. We found that SVM, logistic regression and Random Forest give us approximately the same results with an accuracy close to 0.8.

(4) Nature of a comment. Goal: predict if a comment was a real comment or a user reply by using the `commentType` variable as our target. We used the same models as the ones used in the last problematic. Given our data set and our tuning, black-box models such as Random Forest and Neural Networks do not outperform the logistic regression in terms of prediction power. We obtain accuracy of 0.73 and a precision of 0.74.

4 Discussion/Conclusion

Global remarks: we performed a NLP project from scratch including exploration, preprocessing, prediction and comparison tasks. We manipulated famous NLP librairies and standard Machine/Deep Learning ones. One of the main problem we faced was due to the RAM available through Google Colab in order to improve our performances. However, we always built models 'learning something' (i.e. outperforming naive models). Unsurprisngly, given the size of our data set and our features, tuned random forests are a good alternative between computationally intensive models and simple ones but with a low predictive power.

Concerning possible future work, one could extend this study by performing other predictive tasks using variables included in the data set. One could also try to predict the sentiment score built in section 2. A feature engineering work could also be made for predictive tasks, including auto-encoders for instance. An alternative to improve prediction power could be to use other pre-trained word-embedders, or traine them directly.

References

- Natural Language Processing with Python, *Ewan Klein et Steven Bird*
- NLP: The Essential Guide to Neuro-Linguistic Programming, *Tom Hoobyar, Tom Dotz, Susan Sanders*