Intelligent Systems: Yelp Reviews Classifier

Miguel Ángel García-Gutiérrez Espina

Universidad Politécnica de Madrid
`miguel.gespina@alumnos.upm.es`

1 Introduction

This document corresponds to the NLP assignment of the course of Intelligent Systems in the winter semester 2021/2022. Text data with a categorical label are processed and a machine learning model is built. Then the results are laid out and discussed.
R [6] and its package *dplyr* [8] for a brief preprocessing of the dataframe, and *Python 3* [7] and mainly its packages *pandas*[4], *nltk*[2] and *pytorch*[5] for processing the text and building the machine learning model. The Python code was run on *Google Colab* taking advantage of their free NVidia GPUs.

Dataset The dataset chosen was the *Yelp Reviews Dataset* [1]. It is a collection of reviews of businesses (mostly restaurants). These reviews contain a number of stars, which corresponds to the rating that the user gives the business (integers from 1 to 5), and a text review commenting it.
The goal of this work is to train a model that is able to predict the number of stars based on the comment the user gives. I hope and expect that the classifier reaches an accuracy higher than 20%, which would correspond to the one obtained if the classifier decided the number of stars randomly.

2 Methodology & Results

First, I used *R* to do a brief analysis of the data to get an overview of it. I found out that the dataset was unbalanced in the number of stars as can be seen in Figure 1 and Table 1. Therefore I sampled from the dataset to obtain 20,200 perfectly balanced unique *(review, stars)* tuples.

2.1 Text Processing

I processed the text data using *Python 3 [7]* and its *nltk* package. This consisted in tokenization, removal of stopwords, lemmatization and stemming.

1. **Tokenization**: I used *nltk*'s *RegexpTokenizer* and also removed punctuation and transformed all the text to low caps.
2. **Stopwords**: I removed *nltk*'s English stopwords. In this step, I also only took the first 20 words of each review because of memory issues.
3. **Lemmatization**: I used *nltk*'s and *WordNet*'s lemmatizer.
4. **Stemming**: I used *nltk*'s *PorterStemmer*.

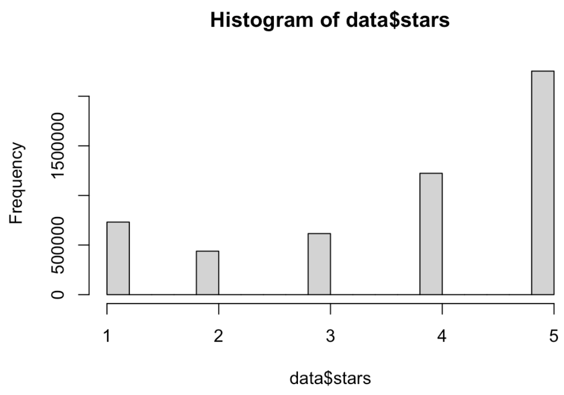

Fig. 1: Histogram of the frequency of the number of stars. It shows that the dataset is unbalanced.

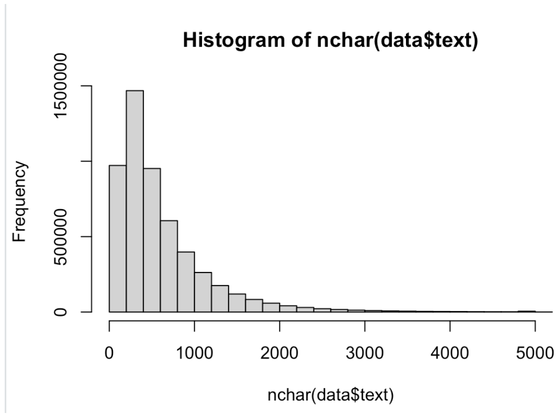

Fig. 2: Histogram of the frequency of the number of characters in the yelp reviews.

Table 1: Metrics about the number of stars of the yelp reviews

	Number of Stars
Min.	1
1st Qu.	3
Median	4
Mean	3.728
3rd Qu.	5
Max.	5

2.2 Classifier

First I tried a character-level convolutional neural network,[3] but it yielded a low accuracy so I moved on to a convolutonal neural network that encoded every word into a one-hot vector. Note that I did not use the identity matrix (numpy.eye or numpy.identity). To save memory, I wrote a helper function that produces a one-hot vector without creating a big matrix.

I trained a CNN with learning rate of 0.0005 on the first 20,000 reviews and only the first 20 words (without stopwords) of every review. I tested it on 200 unseen reviews and got an accuracy of 38%, which is pretty good for an NLP classification task with as many as 5 classes.

I tried making adjustments to the kernel size of the CNN and arrived at one that I found optimal. I also tried tweaking the number of convolutional and fully connected layers, but since our word vectors were very big, I did not dispose of enough RAM to store the weights that a CNN with many layers would have to store. At the end, I wound up using only two convolutional layers and one fully connected layer.

3 Conclusion

The prediction results of the classifier fit my expectations. The score of 38% is well above the one that would be obtained randomly (20%) and acceptable for an NLP task like this one. I made the realization that machine learning on text data requires more computation power and RAM than other types of data. This is due to how text needs to be encoded compared to numerical data. Finally, I would like to state that this task has been done with a small fraction of the total data that *Yelp* makes available. I am sure that with greater computation power, one could achieve much better results.

References

1. Nabiha Asghar. Yelp dataset challenge: Review rating prediction, 2016.
2. Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.
3. Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999.
4. Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.
5. Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.
6. R Core Team et al. R: A language and environment for statistical computing. 2013.
7. Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.
8. Hadley Wickham, Romain François, Lionel Henry, and Kirill Müller. *dplyr: A Grammar of Data Manipulation*, 2018. R package version 0.7.6.