Sentiment Analysis with GloVe

Álvaro Parafita

June 2017

1 Introduction

TripAdvisor is a travel website focused on storing user-generated reviews of travel-related content. Among several categories of such content, restaurant information and reviews are included.

This project tries to predict the sentiment from the reviews of such restaurants, specifically, by predicting a score between 1 and 5 based on the text of the reviews. The scope will be restricted to Barcelona restaurants found in the website and to its reviews that are in English. For that reason, as a preprocessing step, a language identifier model will be needed before proceeding to the Sentiment Analysis model.

In order to predict the sentiment of the reviews, word and text vectors (embeddings) will be used, a well-stablished technique in Natural Language Processing (NLP) that embeds words and texts in dense vector spaces encoding their meanings. These vectors, then, will be used as input of three classifier models, which will be used to assess the potential of these embeddings in the prediction of text sentiment.

1.1 Dataset

The dataset consists of a JSON file retrieved with scraping techniques from the web of TripAdvisor. In terms of numbers, it consists of 7,343 different restaurants, with 861,591 reviews in total. This file contains a unique identifier for each restaurant and all its reviews, with the title, text and numerical score of each review, among several other metadata fields. This project focuses on the text review and its rating, so all the other fields of the restaurant will be discarded.

1.2 Data exploration

As an initial step, it is necessary to perform a previous exploration of the dataset to detect possible errors that should be avoided before training the actual model.

Firstly, since we want to work with English reviews (due to the fact that most Natural Language Processing resources are for the English language) we should be sure that there are enough English reviews in the dataset. If we look, for example, at the user locations (which are also included in the dataset), we find that in the top 10 locations, apart from Barcelona city, some of them are from abroad. As a result, one can expect to find some English reviews.

If we now take a look at a random sample of review texts, we find reviews written in Spanish, but also in English and German, among other languages. The most common languages found, when looking at increasing samples of reviews, were Spanish, English, French and Italian, in that order. There were also reviews in Dutch, Chinese or Russian, for example.

Apart from that, 1368 reviews ended with "\nMore", which suggests that they were cropped by the website and do not include the whole contents of the original review. As a result, these entries should be discarded during the preprocessing step.

The other variable in which we are interested is the actual rating of a review. In figure 1 we see that higher scores are the most common ones. This unbalance should be taken into account across the whole project to avoid biasing the models to certain scores.

This exploratory analysis ignores the rest of the variables in the dataset, since the main objective is to work exclusively with the text contents of the reviews. The only other point to mention is the fact that there were some duplicate entries among the reviews, so those had to be discarded in order to avoid redundancies in the final model.

1.3 Related Work

Sentiment Analysis, restricted to text, tries to identify the attitude of the author of a text when talking about a specific concept. Traditional approaches work mainly with Lexicons ([1]): manually labelled dictionaries of words annotated with their semantic orientation. More recent approaches try to leverage the success of Deep Learning with good results ([2]), using Recurrent Neural Networks (for example, LSTM networks) or Convolutional Neural Networks ([3]).

This paper focuses instead on Word and Document Embeddings, as stated before. Mikolov et. al proposed in [4] the use of shallow Artificial Neural Networks to create these representations, based on previously calculated Word Em-

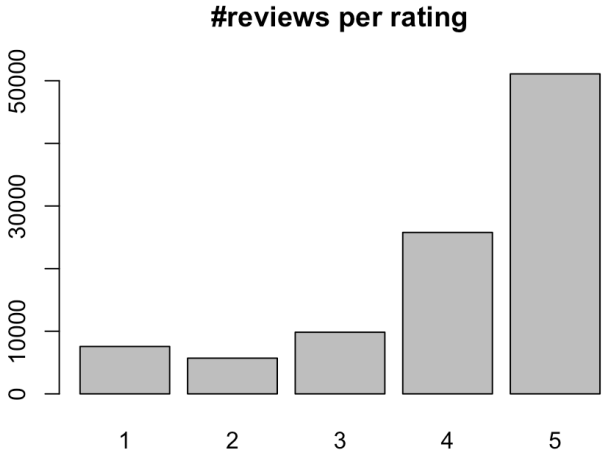

Figure 1: Reviews per rating

beddings. This project avoids the use of Neural Networks, due to computation limitations, and instead uses a simpler alternative to compute these Document Embeddings, based again on precomputed Word Embeddings from the GloVe dataset [5], a publicly available dataset consisting of word-vector pairs computed with the GloVe algorithm over huge text samples.

2 Preprocessing and language identification

The entire JSON file is first preprocessed to fit it in a Tibble structure (a modern R DataFrame) that is more comfortable to work with. Every review that doesn't contain text, or that ends with "\nMore" (which means that it is an incomplete review) is discarded.

This project will focus on English words, so we need a language identifier. We first require a labeled set of reviews, for which a specific tagging module was defined, the *language_tagger.py* file. This Python module shows random reviews one by one and lets the user tag them with a single keystroke as English, Spanish, French, Italian or Other reviews. Even though this project focuses on English reviews, the tagger includes other languages in case the tagging set should be reused in other projects.

Now, with that tagged dataset, consisting of 604 entries, we need to define a language classifier that predicts whether a review is written in English or in another language. The input of such a classifier will be the set of character trigrams contained in every review. A character trigram is a sequence of three consecutive characters found in the text. According to [6, Chapter 2], "Lan-

guage identification based on classifiers that use short character subsequences as features is highly effective; most languages have distinctive signature patterns".

We first split the tagged dataset in two sets, train and test, of sizes 500 and 104, respectively. We then compute the vocabulary of trigrams, obtaining 3518 possibilities. In order to avoid using infrequent trigrams, we select the 25% most common ones, getting then 886 different values. Finally, we compute the Bag-Of-Words (BOW) vector of each review, a vector consisting of zeros and ones with whether the trigram belonging to each column appears in the document.

This new list of BOW vectors will be used as input of a Random Forest Classifier trying to predict if a review is in English or not. We perform hyperparameter optimization of the model by comparing the OOB precision and sensitivity for each combination of the two hyperparameters considered: number of trees in the forest and maximum depth of each of those trees.

The main scoring function considered for the tuning and evaluation of the model is precision, since our main objective is to reduce the probability of a non-English review appearing in the resulting English dataset. However, we want to be sure that sensitivity doesn't decrease as a result, since that would mean a significant reduction of the resulting dataset. The results of these executions are plotted in figure 2.

If we focus on increasing precision, then a maximum depth of 4 or 5 seems the better option, because it leads to the highest values for precision while not compromising sensitivity. As for the differences in the number of trees, we see that around 100 trees it starts to stay constant, so we should focus on values of at most 100 to get the simplest model possible. If we look at the plot in figure 3, we can see that the constant region starts at 90 trees, so we decide to keep only 100 trees with a maximum depth of 4, to use a simpler enough model that still has good performance.

The Random Forest model with that choice of parameters is finally evaluated through the precision score and a confusion matrix on the test set. The result is a 100% precision score on the test, which does not necessarily mean that the model is perfect. A bigger tagged dataset should be used to assess the real average precision and its variance. However, the results are promising and ensure a decent performance to filter our original dataset only to those reviews written in English.

Finally, we train a final model with that choice of parameters and using both the train and test set to finally predict which of the remaining reviews (not appearing in the tagged dataset) are in English. This new subset will form the dataset with which we will perform Sentiment Analysis. It consists of 151,014 English reviews.

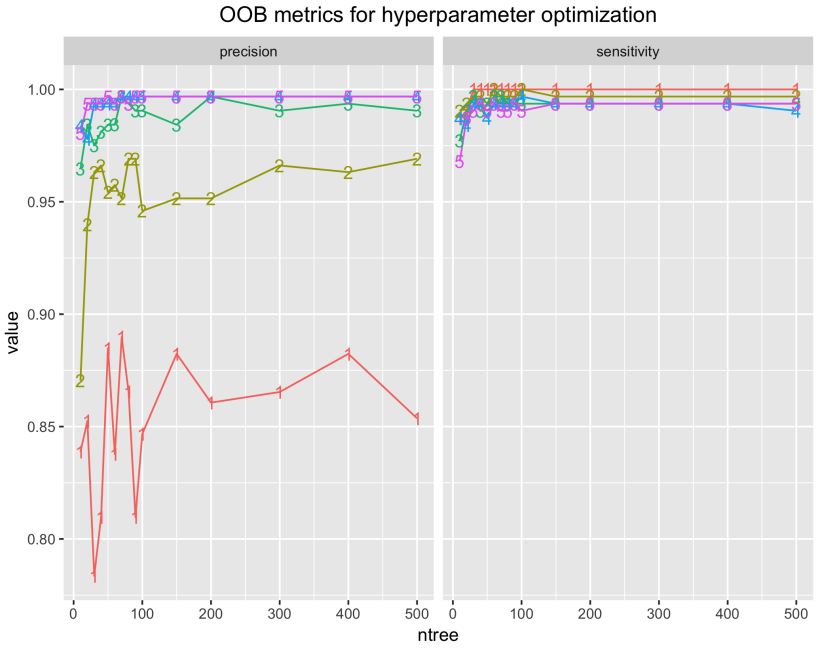

Figure 2: OOB precision and sensitivity for hyperpameter optimization

3 Sentiment Analysis with GloVe-based document vectors

The English Reviews dataset is divided again in a train-test setting. The train set will consist of 100,000 reviews and the test set of the remaining 51,014 reviews that have not been selected previously for the training set. In this way, we can evaluate the performance of our model with a dataset that has not been used to train or optimize our model, thus making for a valid estimation of the model performance.

As a preprocessing step, we perform word tokenization and number, whitespace, punctuation and stopword removal on each of the reviews, using the *tm* R library.

3.1 GloVe and document vectors

In order to use the words in a review as input for a model, an encoding of the text must be provided. The standard approach is using a BOW representation of the text based on a fixed-length vocabulary. The method assessed in this paper, however, uses dense word vectors and computes a document vector from the words it contains.

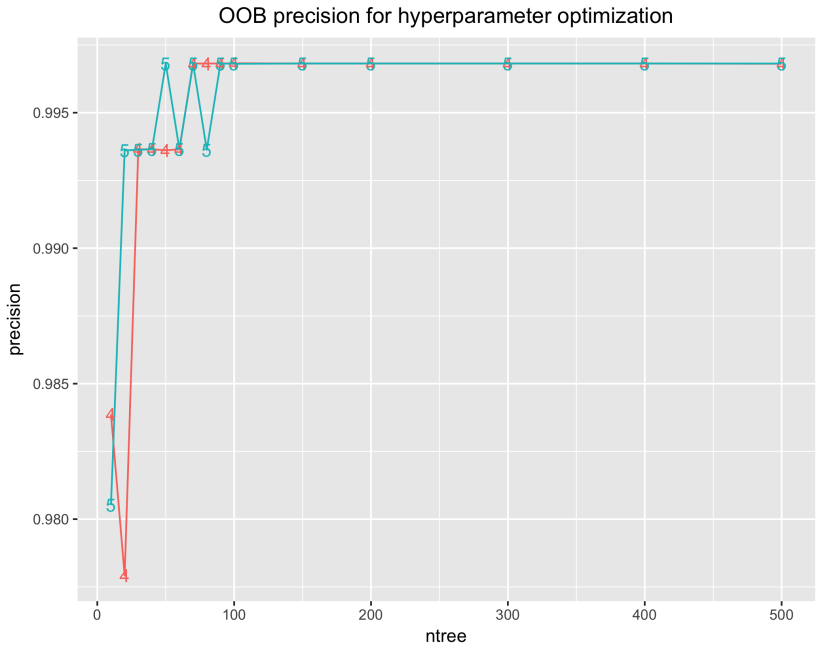

Figure 3: OOB precision for hyperpameter optimization, with maximum depth of 4 and 5

In this project, word vectors from the GloVe dataset [5] are used to create our document vectors. The GloVe dataset consists of 300-dimensional embeddings of words derived from public text data. The GloVe algorithm leverages techniques from global matrix factorization models and local context window methods, such as word2vec, described by Mikolov et al. in [7]. The result of their execution is a dataset consisting of pairs word-embedding that are publicly available.

Our code loads the GloVe dataset and assigns each word to its corresponding embedding. Some words do not appear in the dataset, so those were discarded. These vectors represent each word in a review and are used to compute the document vector, the actual input of our Sentiment Analysis model.

In order to compute that document vector, a weighted average of the word vectors of each document is performed. The weights of this average are the TF-IDF scores of each word, considering for the IDF score only the documents found in the train set. The resulting document vectors are used as input for our Sentiment Classifier.

As a final note, the size of the vocabulary in the IDF scores is of 39,950 different words, considerably bigger than the 300 dimensions from the GloVe vectors.

3.2 PCA

Since the embedding dimension is 300, a good approach would be to try dimensionality reduction techniques, such as PCA, before training the predictive model. In working with PCA, we should take into account that the unbalance in the number of entries for each possible rating could affect the overall inertia in our dataset, thus affecting the results of the PCA. As a consequence, in order to avoid that bias, we add weights to each individual in the dataset passed as input to the PCA so as to decrease the importance of those individuals from the most common ratings.

The screeplot of the PCA is displayed in figure 4. By using the Kaiser rule (taking the Principal Components that have associated eigenvalues bigger than the average eigenvalue) we keep 75 components, that account for 70% of the variability in the data.

Finally, figure 5 shows the first factorial plane of the training set with the ratings of each review. Figure 6 shows again the first factorial plane, this time filtering those reviews with rating 3 (that could be interpreted as a neutral sentiment) and displays as positive sentiment (TRUE) those reviews with rating 4 or 5 and negative sentiment (FALSE) for ratings 1 and 2. As we can see in these plots, the PCA, just with the first factorial plane, returns a structure where the sentiment could be interpreted. Finally, figure 7 shows the latter plot

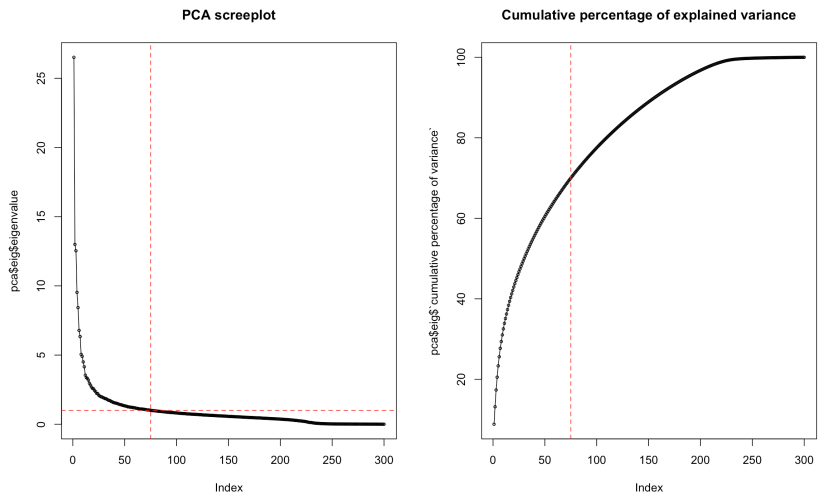

Figure 4: Screeplot and cumulative explained variance in PCA

with three dimensions, to prove that this separation in sentiment is not due to superposition in the first two components.

3.3 LDA, QDA and SVM

The methods assessed for our prediction are Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA) and Support Vector Machines (SVMs). All three are chosen due to their good performance on numerical, continuous variables, which is actually how the entire dataset is expressed through the document embeddings and the PCA.

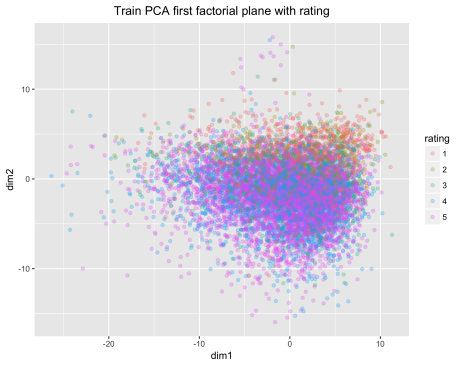

Figure 5: FFP with rating

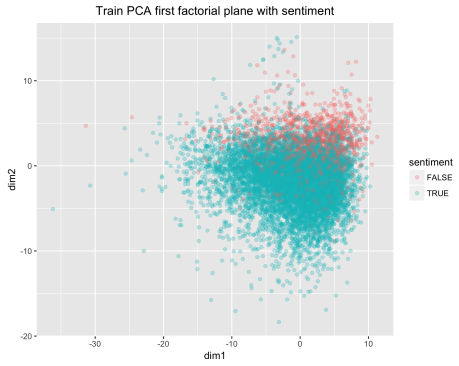

Figure 6: FFP with sentiment

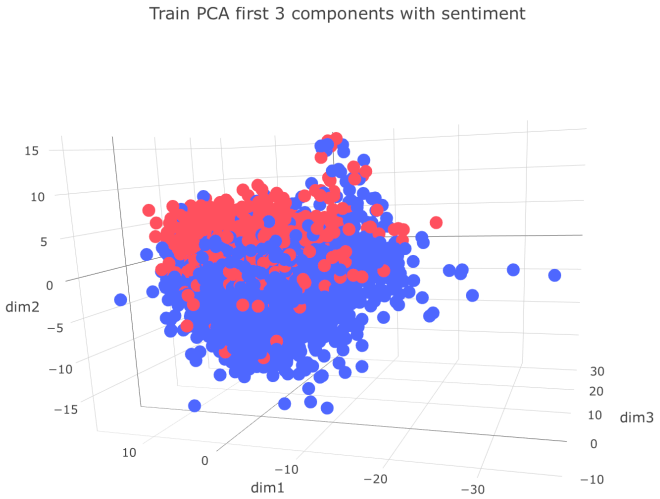

Figure 7: First three components with sentiment

As said before, the input of both models will be the PCA-transformed vector representations of the reviews. The target variable will be a categorical vector containing the five possible scores assigned with a rating, an integer score between 1 and 5. Additionally, the model will also be trained trying to predict just the sentiment of a review (positive for 4 or 5, negative for 1 or 2, and not trained with any 3-review, due to its ambiguity in terms of sentiment).

No hyperparameter optimization is done except for SVM, where we focus on optimizing the C hyperparameter, by evaluating several values of C with a validation set and choosing the one with the highest accuracy. In both problems, $C = 1$ is the best option. For SVM, however, we had to train with a sample of 10000 reviews instead of the whole training set, because with the whole dataset it took unreasonable amounts of time. The usage of other libraries to train the SVM model could be attempted to overcome this setback.

When the three models are trained, the rating and sentiment is computed over the test set and compared with the real values. With the resulting confusion matrix, we can evaluate the performance of each model. The metrics used to assess the quality of the models for the rating prediction problem will be both accuracy (ratio of correct guesses) and the mean squared error. Even though we're running the problem as a multi-class classification model, the result has an inherent ordering that the mean squared error can assess well.

For sentiment classification, since we have a binary classification problem, we can look at other metrics too. Those are depicted in the following tables.

Table 1 shows the results for the rating prediction problem and table 2 shows the results for the sentiment classification problem. Both were computed over the same test set.

model	accuracy	MSE
LDA	**0.598**	1.018
QDA	0.531	1.471
SVM	0.553	**0.977**

Table 1: Performance on the rating prediction problem

model	acc.	+prec.	-prec.	sens.	spec.	F+	F-
LDA	**0.932**	0.944	**0.845**	**0.978**	0.671	**0.961**	0.748
QDA	0.872	**0.965**	0.548	0.881	**0.819**	0.921	0.656
SVM	0.931	0.962	0.761	0.957	0.786	0.959	**0.773**

Table 2: Performance on the sentiment classification problem

In all cases, QDA is the worst model in terms of accuracy and most of the metrics. It is worth mentioning that SVM achieves very good results even when working with just a tenth of the training samples that LDA and QDA received. This might suggest that with an increased train size, SVM could outperform the results from LDA, but nothing conclusive can be derived just from this experiment.

4 Conclusions

This project tries to show the potential of GloVe word vectors as a mean of describing documents, which are even able of encoding sentiment in their embeddings. This can be seen by the fact that a generic classifier such as LDA can achieve an accuracy of over 90% in a sentiment classification problem just using those vectors.

Nevertheless, other approaches are more adequate for the problem of Sentiment Analysis. In particular, Neural Networks working with word embeddings are considered the state of the art in this field. However, this project tried to use this other approach, explicitly the power of word embeddings, in a setting without the computational requirements needed for training Recurrent Neural Networks.

Other approaches for creating the document vectors should be considered too. In particular, Mikolov et al. suggest in [4] two Neural Network architectures to create document embeddings from word embeddings, both of them similar to the ones in [7] for word vectors. This alternative could be tested with a bigger dataset to improve our approach. Its main advantage is that it is capable of considering word ordering in the creation of the word embedding, something that our method, that just computes a weighted average of the word vectors, cannot represent.

References

[1] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307, 2011.

[2] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, pages 1422–1432, 2015.

[3] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[4] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196. JMLR Workshop and Conference Proceedings, 2014.

[5] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[6] Prabhakar Raghavan Manning, Christopher D. and Hinrich Schütze. *Introduction to information retrieval*. Cambridge: Cambridge university press, 2008.

[7] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.