Università degli studi di Milano-Bicocca

Text Mining And Search

Final Project

Sentiment Analysis of Amazon Reviews

Authors:
Lorenzo Camaione - 850380 - l.camaione@campus.unimib.it
Davide Toniolo - 800458- d.toniolo2@campus.unimib.it

September 3, 2020

Abstract

In this paper a number of solution for a text classification task are presented. The objective of the project is to predict, from its text, whether the rating of a review is either positive or negative. The dataset consists of reviews of fashion products sold on Amazon. The different models evaluated perform well, but their results are skewed towards the most frequent class.

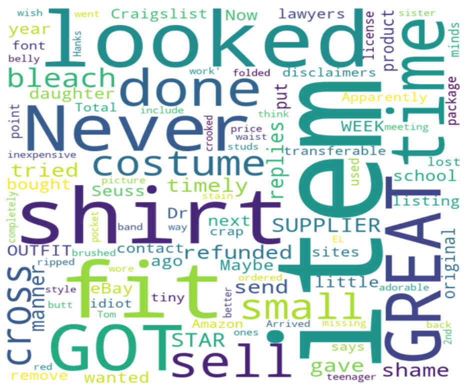
1 Introduction

Sentiment analysis consists of classifying a document in a set of predefined categories. In its simplest form, the emotions are positive or negative, while more complex analyses use a greater number of classes.

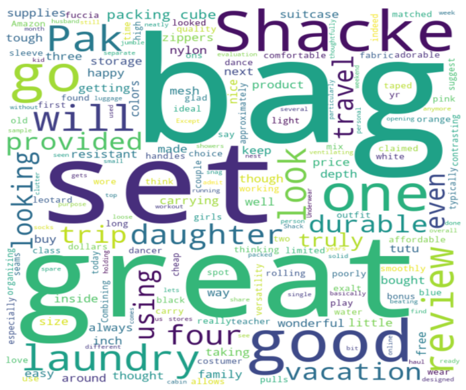
The dataset used in this project is a collection of Amazon reviews regarding some specific classes of products. In particular the products chosen are either clothes, shoes or jewellery. This dataset is made available directly by Amazon. The data span a period of 18 years, including a large number of reviews up to March 2013. The objective of the project is to develop a model capable of predicting the rating of a review from its textual content.

In the Amazon e-commerce people usually review products they buy. In this process they typically choose a rating between 1 and 5 and then add a text to try to explain why they picked that rating.

The reviews having either one star and five stars have been explored creating a word cloud for each group. These word clouds don't contain stop-words.



(a) One star reviews wordcloud



(b) Five star reviews wordcloud

Looking at the word clouds it's clear that there are some differences. Although some words are in both word clouds, others are not: the one star word cloud has words like *never* and *crap* while the five stars one has words like *great* and *good*. The two word clouds look different, but the majority of the words are shared between the two.

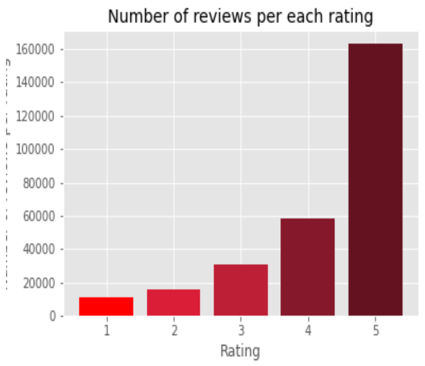
2 The Dataset

The dataset consists of 280 000 reviews collected from the *Clothing, Shoes and Jewellery* products of the Amazon online store in the May 1996 - July 2014 time frame [2], [1]. It comes as a sequence of JSON documents with the following attributes:

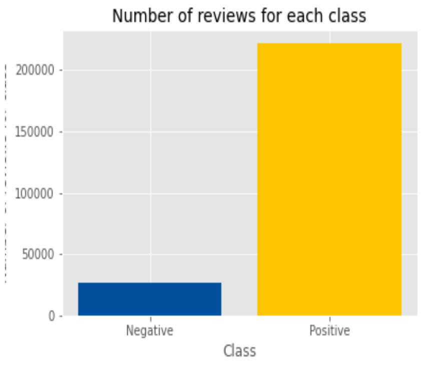
- `reviewerID` - ID of the reviewer
- `asin` - ID of the product
- `reviewerName` - name of the reviewer
- `helpful` - helpfulness rating of the review. It is a array of ints and of length 2. A value of `[1, 2]` means that among the two users that viewed that review, one of them found it useful.
- `reviewText` - the text of the review
- `overall` - rating of the product. Floating point number, 1 to 5. There are no fractional ratings.
- `summary` - the summary of the review made by the author
- `unixReviewTime` - the time when the review was posted in an human-readable (American) format.

  The overall file size is 150 MB.

The target variable class frequencies are showed in the following histograms: in the first one it is shown as is in the dataset while in the second one the histogram shows the variable after it has been discretized.



(a) Frequencies of the target variable overall



(b) Frequencies of the target variable after it has been discretized

3 The Methodological Approach

Different NLP pipelines and different classifiers are compared in terms of results and speed.

3.1 Preprocessing

As soon as the file is read, the relevant fields are extracted and everything else gets discarded. Given that the task consists of classifing the review into positive or negative given the textual content, only the fields `reviewText` (input variable) and `overall` (target variable) are kept. The latter is then transformed so that the 1 and 2 classes are encoded as 0 (negative) and the 4-5 starts as 1 (positive). The neutral 3 star reviews are discarded.

3.2 Tokenization

The raw textual data isn't suitable for manipulation by a machine, so it needs to be transformed into a more computer-friendly form. This transformation consists of a series of operations, the first of which is called *tokenization*. A *token* is a single, self contained "piece of text". Splitting each sentence into words isn't sufficient: a concept or idea may be represented by multiple words and if this is the case, separating them may lead to worse results.

As an example, the sentence "John wants to visit Italy this summer" can be naïvely tokenized by splitting into words, but in the sentence "Catherine has found a new state of the art model for NLP!" the words "state of the art" represent a single concept and shouldn't be separeted.

A great variety of algorihtms for tokenization are available. For this project the method `word_tokenize` by `nltk` was used.

3.3 Stop Words Removal

When building a vocabulary, it is often the case that the most frequent words (or tokens) are not the most informative. For example, in the English language "the" is by far the most common word, but it is hardly of any use to discern documents. A common practice is to remove words that occur too frequently, in order to reduce the size of the vocabulary and reduce the number of features.

`nltk`'s stop-words list for the english language was used.

3.4 Stemming and Lemmatization

These terms refer to two related practices used to reduce vocabulary size and improve computability: a given word is reduced to its root (e.g. "giver" becomes "give"). Stemming is a more crude and less sophisticated approach, but a more faster one.

3.5 Choosing the Text Representation

The preprocessed text then needs to be encoded in some form that is suitable to the classification algorithms that the researcher wishes to use.

The simplest of all use "unigrams" (that are, single tokens): they make use of each token in the document individually, without regard for their order or semantic/logical relationship with each other. The output of these encoding approaches is a matrix W in which the first axis refers to the vocabulary entries, while the second refers to the document in the corpus.

The entries of W can be obtained in different ways and each approach gets its own unique name:

- set of words - W_{ij} is a boolean that records the presence or absence of the i-th token in the j-th document.
- bag of words - W_{ij} is the number of times the i-th token appears in the j-th word.

- various weighting schemes: document length, max tf, tf-idf.

For this project the set of words and the tf-idf approaches where used. Bi-grams tecniques and higher have not been explored, given that they have much higher memory requirements (approx quadratic and cubic respectively).

3.6 Reducing Memory Usage

Even with a simple unigram text representation, if the document corpus is large enough, the W matrix can occupy a significant portion of memory, several GB in size.

To address this problem, tokens that are either too common or too rare can be excluded. Then, the matrix can be decomposed using PCA or some other dimentionality reduction technique.

To compute the matrix `sklearn`'s `Tf-idf` class was used. The advantages of Scikit-Learn's implementation are twofold: firstly, their code is fast, faster than a simple Python implementation by the end user would likely be. Second, they output a sparse matrix. Given that many of the entries of the W are usually 0, this hack allows for a huge memory saving. Also, when used for fitting compatible (both algorithm-wise and implementation-wise) classifiers, this can greatly reduce the number of computations required.

The dimensionality of the matrix is reduced using Trucated SVD (i.e. PCA where only the components corresponding to the higher variances are retained). By doing so, a dense matrix of much lower dimension is obtained: the number of output features is set to 100, the default.

3.7 Classification

Different Machine Learning classification algorithms were explored and evaluated. Given that the dataset is highly unbalanced, the performances of the different classifiers were compared using the precision and recall metrics.

In order to obtain a balanced fit out of the unbalanced dataset, each class was weighted with the moltiplicative inverse of its relative frequency: in principle a weighting scheme like this allows the researcher to use all the available training data and still get a classifier that doesn't favor the most frequent classes.

This technique was used with all the algorithms, with the sole exception of the K Nearest Neighbours classifier, given that it doesn't make sense in

	Metric	Gaussian NB	Random Forest	kNN	Linear SVM	ANN
Class 0	Precision	0.18	0.75	0.24	0.32	0.37
	Recall	0.74	0.10	0.34	0.84	0.82
	F1-measure	0.28	0.18	0.28	0.46	0.51
Class 1	Precision	0.95	0.90	0.84	0.98	0.97
	Recall	0.58	1.00	0.87	0.79	0.83
	F1-measure	0.72	0.95	0.86	0.87	0.90
Macro Avg	Precision	0.56	0.82	0.43	0.65	0.67
	Recall	0.66	0.55	0.43	0.81	0.83
	F1-measure	0.50	0.56	0.42	0.67	0.70
Accuracy		60%	90%	73%	79%	83%

Table 1: The performances of the various classificators. Results on the positive class are better across the board.

this case.

4 Results and Evaluation

Obtaining high levels of the target metrics was a challenging task. As showed in table 1, even with the weighting of the classes during the training stage, the classifiers still perform better on the positive reviews across the board.

Perhaps the problem could be solved by a more aggressive weighting scheme. However, various trials showed disappointing results.

This, coupled with the fact that the unbalanced results are common to all the classifiers, leads to the intuition that the problem may be lying in the data. The word clouds for the negative and positive reviews 2a and 2b, are very similar. Maybe, even with the tf-idf scheme (that should penalize words that are common across all documents).

Thus, the problem may lie in the use of a unigram representation.

Another issue may be the dimentionality reduction technique used: the truncated SVD selects the components associated with the highest variance. However, this variance could be common to all the classes and thus of limited use when classifying.

For example, the highest variance components could be associated with the different subcategories of the dataset (e.g. clothing vs shoes vs jewelley) and would be useful for topic detection, but not for sentiment analysis.

Therefore, a filter that is capable of retaining the features that have the highest variance *between* the classes might be able to improve the results.

5 Conclusions and Future Work

The NLP piepeline and classifiers used show sufficiently high metrics. The text preprocessing is able to extract the features from the text without using an excessive amount of memory.

However, the unbalanced results obtained by the classifiers suggests that a more sophisticated text representation method might improve the final scores.

Also, using a dimentionality reduction technique that is able to keep the most informative features will likely help.

References

[1] McAuley et al. "Image-based recommendations on styles and substitutes". In: (2015).

[2] McAuley He. "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering". In: (2016).