NLP and the Web – Milestone 2

Finn Andersen
Michael Kennecke
Fabian Schick

Abstract

This report deals with the foundations of the submission made for the course Natural Language Processing (NLP) and the Web at TU Darmstadt. The Milestone 2 Project which is subject of this paper deals with different classifiers that categorize product reviews based on their domain (kitchen or electronics) and their polarity (positive or negative). The given task is editing the test data in such a way that the precision score of the domain-classifier stays the same but the precision of the polarity classifier is decreased as much as possible. This paper shows that you need to implement diverse algorithms and functions to achieve a low precision score of less than 12%.

1 Introduction

In this project which is the second milestone of the course Natural Language Processing and the Web at TU Darmstadt we deal with product reviews, two given classifiers and a given vectorizer. The provided data is a csv-file that contains 400 different product reviews and their correct labels for both classifiers. The first one indicates whether the review belongs to the domain electronics or kitchen and the second classifier predicts if the review has a positive or negative intention in general which is called polarity. The projects goal is to keep the precision of the domain classifier and decrease the polarity score as much as possible. In the following chapters general approaches and thoughts to the explained task will be introduced and explained. Furthermore there will follow a discussion of the results and a conclusion. The first two sections in chapter two deal with data loading, preprocessing and exploration. Afterwards there follows a description of the developed fooling approaches and their integration in a Jupyter notebook. Finally the results are discussed and concluded.

2 Methodology

In general, the process of fooling the classifier is divided into several steps. First the data needs to be loaded and saved in a specified format in a Pandas dataframe. Next the different reviews will be preprocessed and their domain and polarity are determined. For the prediction of the polarity a preprocessed sentence is used to achieve a higher accuracy. This will be explained in more detail in chapter 2.2. After that the data will be prepared for running different fooling algorithms. In the end all functions will be executed until the polarity is reversed and the domain is the predicted one. Finally, every comment is stored in a separate dataframe. If the fooling approaches do not work, the original comments are stored in this dataframe.

2.1 Loading data

Before starting to work with the reviews the Python program loads the file "evaluation_examples.csv", saves it in a dataframe and tokenizes every comment using Spacy's nlp() function. Next all product reviews are split into sentences and appended to a list of all sentences. By now the data is in a format that can be used for all processing and analysis steps. For having a more precise overview of which tokens or sentence structures have a high influence on polarity and domain all sentences are classified by the given classifiers. As a result, the dataframe of sentences contains the returned values for both classifiers.

As a next step all items of the dataframe, the reviews, are processed with the nlp() function of Spacy. This procedure returns all reviews in tokenized forms and from now on it is possible using part of speech (POS) tags, lemma etc. of all tokens.

2.2 Basis analysis of reviews

Before developing and running the approaches that shall fool the polarity classifier a basic analysis of the given data is made. The for-loop runs over all sentences and identifies all verbs and adjectives together with their domain and polarity of the sentence. Our strategy is that mostly these two types of tokens have a strong influence both on the domain and the polarity classifier. Therefore, the next step is aggregating all items found and calculating their polarity score. Furthermore, a ranking is created by calculating the differences of occurrences in domain electronic and kitchen. This ranking shows which tokens mostly occur in one domain and their assigned polarity.

Additionally, it can be observed that the accuracy of the polarity classifier can be increased, if only some parts of speech are used for classification. For example, adjectives, verbs, adverbs and numbers have a high impact on the result while some parts of speech decrease the accuracy. As a result, a function is implemented which returns only some parts of speech per comment.

2.3 Development of fooling approaches

For decreasing the polarity score there are several approaches developed that shall fool the classifier. All these functions are based on the preprocessed data and use the previous results. The next sections describe those approaches.

2.3.1 Assign “not” to verbs & adjectives

One basic approach is assigning a “not” to all verbs and adjectives in case there isn’t such a word existing before. If there is an existing “not” before a word the function will delete it. By this, hopefully many polarities of reviews are fooled just by turning around all the meanings of words. This function is executed for all reviews.

2.3.2 Exchange verbs & adjectives (1)

This function which is called turn_sentences() checks all reviews and each token based on their polarity. If the polarity of a token is not equal to the desired polarity of the whole review it is exchanged by a word that has the exact opposite polarity according to our analysis which was mentioned before. For all verbs and adjectives this function will be executed. Furthermore, all words which have the lemma “not” will be deleted. Thereby there are only strong adjectives and verbs for the desired polarity and domain.

Having a review that is negative in general and one verb is “hate”, this procedure would replace the word with “love” for instance. For adjectives this would work in a similar way.

2.3.3 Exchange verbs & adjectives (2)

This function which is called turn_sentences2() checks all reviews and each token based on their polarity. It changes all verbs and adjectives to the most common one for the desired polarity and domain. Furthermore, all words which have the lemma “not” will be deleted. Thereby only strong adjectives and verbs for the desired polarity and domain are existing.

In comparison to the previous function, this one changes the original reviews even more because every verb or adjective is exchanged. In the previous case only words that don’t match the desired polarity are exchanged for the correct ones.

2.3.4 Adding strong adjectives before nouns

This procedure extends the functionality of the approaches shown before. In addition to exchanging verbs and adjectives this function will add strong adjectives before nouns. Such proceeding leads to better fooling results regarding the provided classifiers. Normally, nouns just indicate the main content of a sentence. If you add strong adjectives you will strengthen the desired polarity of the phrase.

To do so, the word “awesome” is added before every noun if the review shall be turned to a positive polarity. In the other case, if a review shall have a negative polarity, “horrible” will be added.

2.4 Realization in loop procedure

After defining several functions with different fooling approaches there is a loop that executes them based on some prerequisites.

For every product review the program calculates the polarity and domain to avoid the use of original labels from the file. To get a more confident result the classifier for polarity only receives some parts of speech of the comment, so it is more likely that it returns the correct (original) polarity. Next, the

program tries a function and checks whether the polarity is changed to the desired one and the domain stays the same. If so, it continues with the next review and if not, it runs the next fooling approach and checks again. All functions are executed in case the previous ones haven't led to the desired, wrong polarity. After executing this loop, the program is able to end its fooling attempts for a specified review. If the program was successful and the polarity is turned around, the new version of this comment is saved in a different dataframe. In the other case, if the described functions were not able to change the polarity without changing the domain, the comment will not be manipulated. In addition to the comments the original label and polarity will be stored in the dataframe with the "fakes". Thereby we are still able to predict the polarity and domain with the given classifiers.

In the end, all manipulated reviews are saved in a file called "processed.csv" and the classifiers are executed for the whole document. Lastly the analysis shows the achieved score of domain and polarity.

3 Discussion

With this procedure, we achieved that the precision for domains remains the same with a score of 93.5% but the score for polarity is decreased to 11.25%.

All in all, the results show that it is possible to change the polarity of all the given comments, with the described functions, but without the usage of the original labels we are not. In reality we are not allowed to use the original labels. If we use the "prepared comments" the polarity classifier has an accuracy of 88,75%. The remaining 11.25% of failures are the failures in our fooling methods, too.

This report shows that a pattern based language manipulation is possible for computer programs. Nevertheless, it's important to note that the described approaches mostly return comments which are not related to the old ones. The only target of those approaches is changing the polarity and keeping the domain but not to return meaningful sentences which could be written by humans.

Furthermore, this report shows that different POS-tags more or less have an influence on different "meanings" of the sentences. By removing some POS it gets easier to predict the polarity (and/or) domain. This can be reasoned with the features of the vectorizer.

4 Conclusion

All in all, the Python program and its function are all running and in combination they produce good results. In the end the polarity score is 11.25% without any changes to the score of the domain classifier. Without the preprocessing of the comments for the initial polarity prediction in the loop the lowest polarity to achieve would have been 15.5%.

In relation to the task this result is a major step in the desired direction. The goal was decreasing the polarity score towards 0% starting from 84.5% and the shown approaches manage to nearly reach this target. But it also shows that there are still some other classifications than the desired one. One noticeable reason is the classifier and its precision or failure rate. The way of classification which we weren't allowed to change or adapt is not perfect which consequently leads to failures in prediction. These failures occur with the manipulated reviews too. So, it might be the case that the manipulation works correct but the classifier returns the "wrong" prediction.

To sum it up, in the beginning the idea was to generate one holistic fooling approach but first tests showed that the reviews are all individual and different in their characteristics which needed different means and measures. Therefore, several approaches are developed and the best working one is chosen for each review. As a result, the program achieves a good polarity score of less than 12%.