UNIVERSITAT DE BARCELONA

FUNDAMENTALS OF DATA SCIENCE MASTER'S THESIS

Sentiment Analysis of student evaluation of teaching

Author:
Indra IKAUNIECE

Supervisor:
Venelin KOVACHEV
Eloi PUERTAS
Antonia MARTÍ

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamentals of Data Science

in the

Facultat de Matemàtiques i Informàtica

July 1, 2018

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

MSc

Sentiment Analysis of student evaluation of teaching

by Indra IKAUNIECE

After every semester in University of Barcelona all students are asked to fill a survey about professors and subjects from the previous semester. Students provide evaluation by answering two different kinds of questions - quantitative (numeric), and qualitative (open text). It would be useful for the professors, the program coordinators, and for the directors of the departments to have an automatic quantitative overview of the textual answers.

The goals of this project are twofold: 1) to create a supervised dataset for sentiment analysis and polarity detection of student opinions in two languages (Catalan and Spanish); and 2) to validate the dataset empirically and propose competitive baselines by investigating, implementing and comparing sentiment analysis algorithms and methods to automatically classify student comments as positive, negative or neutral.

Chapter 1

Introduction

Sentiment analysis is a fast-growing research area that focuses on extracting sentiment from textual documents. Common applications are sentiment extraction from Tweets, product reviews and comments on the Internet, with a goal to make better marketing decisions, recommendations, improve different services, as well as to advance human-computer interaction. Sentiment analysis and the sub-task of polarity detection can be approached as a binary classification task ("positive" or "negative"), a multi-classification task by adding a "neutral" class, or a task of estimating positivity and negativity on a scale with discrete or continuous numbers.

The approaches are supervised and unsupervised, with supervised approaches generally performing better but also requiring a large manually annotated corpus in the relevant language. Unsupervised approaches, on the other hand, use linguistic knowledge and predefined linguistic rules that are applied to a text corpora. Results on sentiment analysis are good for English, as it has many annotated datasets for supervised learning. However in last years the focus has also been placed on other languages, where annotated datasets have been created.

This project focuses on creating a corpus and applying supervised methods on answers in student questionnaires from Computer Science bachelor programs in the University of Barcelona. After every semester students fill in the surveys about subjects and lectures, and comment their opinions in open questions. These answers are important part of the teacher evaluation, and it would be useful to have a quantitative evaluation of the open answers. For example, it would eventually be useful to have a system that given all the surveys reports on how many positive and negative comments each professor has. This project is a first step towards a more complex system of collecting and evaluating these student answers. We focus on creating a dataset from student surveys, developing code for processing this dataset, and applying and comparing some of the most common methods for sentiment polarity detection.

The main goals of this project are:

1) Exploring related work that has been done for English, and other languages for dataset creation and supervised sentiment analysis. Learning about common approaches and state-of-the-art methods.

2) Creating a dataset.

   - Data extraction. Student surveys contain all the data in *pdf* format, so the first step of this project is to extract the data.
   - Data pre-processing, including, cleaning the data, anonymizing any mention of professor names. This step also includes detection of language, because, while the questions of survey are in Catalan, student answers are often Spanish and English as well.

- Data annotation, that is, marking, each sentence as "positive", "negative" or "neutral". After completing this step, we will have created a supervised dataset on which models can be applied.

3) Performing experiments with different features and models. Develop code for repeating these experiments. Comparing these experiments on original comment language (Catalan and Spanish) and their translations to English. Answer the following questions:

   - which is the best feature setup for multi-class sentiment polarity detection for this particular dataset;
   - which model performs the best;
   - does automatically translating Catalan and Spanish comments to English improve the classification results.

Chapter 2

Background information and theory

2.1 Related work

Sentiment analysis has been an active research topic in the past years, although most experiments and algorithms have been developed for English, as it is the most used language on the Internet, therefore it has the most texts for experiments (for example, annotated datasets with movie reviews (McAuley and Leskovec, 2013) (Pang and Lee, 2004)). More recently, interest has shifted to other languages, such as, Catalan and Basque, for example, there has been created an annotated corpus with Basque and Catalan hotel reviews (Barnes, Lambert, and Badia, 2018).

Spanish is also quite common on the Internet, so there have been several publications about sentiment polarity detection in Spanish which I will describe in this chapter.

The article "Computational approaches to subjectivity and sentiment analysis" (Balahur Dobrescu, Mihalcea, and Montoyo, 2014) presents an overview of the latest trends and current challenges in the field of sentiment analysis, and as the first challenge they name the multilingual sentiment analysis. To tackle this problem many researchers use translations to English, and then applying existing English lexicons.

Another paper that describes a survey on the main approaches on sentiment extraction on English, other European as well as several Indian languages is the paper called "A Survey on Sentiment Analysis and Opinion Mining Techniques" (Kaur and Gupta, 2013). In this paper as the most popular approaches are listed the subjective lexicon approach (each word has a score that describes it as positive, negative or neutral), n-gram modeling (uni-gram, bi-gram, tri-gram models) and machine learning (supervised learning with extracting different features from the text).

What most related papers have in common is that they all use a manually annotated corpus for supervised methods in the required language. Before extracting features they perform data cleaning, text tokenization and segmentation. All the papers have the pipeline of firstly selecting and extracting features from text (that is, converting raw text to numerical vectors), then they apply models and algorithms, and then report obtained results.

In this chapter I will investigate features, models and evaluation that were used in some of the publications that performed related experiments.

Features

In an experiment about language-independent movie review polarity detection, paper "Language-Independent Sentiment Polarity Detection in Movie Reviews: A Case

Study of English and Spanish" (Graovac and Pavlovic-Lažetic, 2014), they used byte-level n-grams as features. The n-gram is usually defined on a word level, but can also be defined on a character or byte level, where the difference is that character level n-grams doesn't include digits, punctuation, whitespace, etc., while the byte-level n-grams include all characters. In this publication they claim that using byte-level n-grams leads to larger relative insensitivity to spelling errors, as well as this approach doesn't require any linguistic knowledge and is language and topic independent, with the only disadvantage that this leads to a large number of n-grams. For each training category (all positive reviews are combined in one category and all negative reviews in another) they extract n-grams (in their case for Spanish the best results were achieved with $n = 11$), then for each n-gram they calculate a normalized frequency, order n-grams an a descending order, and select first L n-grams (they found that for Spanish the best results were with $L = 40000$).

In the article "Sentiment polarity detection in Spanish reviews combining supervised and unsupervised approaches" (Martín-Valdivia et al., 2013) they train classifier for film reviews in Spanish, and as training data they use Spanish corpora as well as corpora of Spanish reviews automatically translated to English. They combine two techniques: machine learning (supervised) and an unsupervised semantic orientation approach, which doesn't need training but just positive or negative orientation of the words. They combine three models: the first one uses machine learning on Spanish corpus, the second one uses the same model on the English corpus, and the third one uses the SentiWordNet (Baccianella, Esuli, and Sebastiani, 2010) resource on the English corpus to generate an unsupervised polarity. After this, they test different ways to combine these three models. For the supervised models they compared different features: TF (term frequency) which is the relative frequency of each word, TO (term occurrences) which is the absolute number of occurrences of each word, TF-IDF (term frequency–inverse document frequency) which combines TF and IDF which decreases the weights of words that appear very often (for example, the word "the") and increases the weights for words that occur rarely, as they might be more meaningful, and lastly they use BTO (binary term occurrences) where a word is 1 if it is present and 0 if it is not. For each set of features they test results without processing, as well as with stemming (using only word stems) and filtering out stop words (common meaningless words).

The paper "Assessing State-of-the-Art Sentiment Models on State-of-the-Art Sentiment Datasets" (Barnes, Klinger, and Schulte im Walde, 2017) compares several models on several different datasets using several different features. For supervised methods they use word embeddings with different dimensions (50-, 100-, 200-, 300-, and 600-dimensional sentiment embeddings) which were trained on a 2016 Wikipedia dump. Also they use models that are trained on a bag-of-words representation (each training example is represented as a vector in a size of the vocabulary, representing the number of times each word appears).

Another related experiment is described in a publication "Sentiment Polarity Detection From Amazon Reviews: An Experimental Study" (Sygkounas, Rizzo, and Troncy, 2016), where they use ensemble learning to implement five state-of-the-art classifiers, for which the features are n-grams and a dictionary with semantic values of emojis.

Models and methods

The experiment about language-independent movie review polarity detection (Graovac and Pavlovic-Lažetic, 2014), after having extracted the n-grams and their relative

frequencies for all positive and negative documents, for each test review again extract the n-grams and their relative frequencies and calculate a dissimilarity measure between the test review and the two categories. They use three different dissimilarity measures. The first is used by (Kešelj et al., 2003):

$$dK(P_1, P_2) = \sum_{n \in category profile} \left(\frac{2 \cdot (f_1(n) - f_2(n))}{f_1(n) + f_2(n)} \right)^2$$

where $f_1(n)$ are the relative frequencies of n-grams in the either positive or negative category profile P_1 and $f_2(n)$ the frequencies in the test document profile.

The second dissimilarity measure is from (Cavnar and Trenkle, 1994), and it is referred to as dOP (Out-of-Place). To calculate this measure for each n-gram in test document the same n-gram is located in the category profile and and then calculated how far it's location in category profile is from the location in the test profile. If the n-gram is not in the category profile, then the distance is equal to the number of n-grams in the profile. The sum of all these values for all n-grams is the dissimilarity measure dOP.

The third measure they use is a measure introduced by the first author of the same paper (Graovac, 2014), it is referred to as dSD (symmetric difference), and it represents the number of n-grams that appear in the union of the profiles but not in their intersection:

$$dSD(P_1, P_2) = |P_1 \triangle P_2|$$

where again P_1 is a category profile and P_2 is a test document profile.

In the article that combines the supervised and unsupervised approaches to detect polarity in Spanish reviews (Martín-Valdivia et al., 2013), authors used two learning algorithms, Support Vector Machines (SVM) and Naive Bayes (NB), but they only reported on SVM as it achieved better results. They report results on SVM trained on different features with and without stemming and stop words (they found the best result when they didn't use neither stemming nor stop words). They trained these models on both the original Spanish corpora as well as on the translated English corpora. For the unsupervised part, they calculated the SentiWordNet (SWN) score for each document of the English corpora according to method by (Denecke, 2008). To combine the result from these three classifiers they tested voting and stacking algorithms.

In the paper were authors compare several models of several datasets (Barnes, Klinger, and Schulte im Walde, 2017), first of all they train an L2-regularized logistic regression classifier on a bag-of-words classifier, then the same classifier on average of the word vectors (trained on Wikipedia dump, different dimensions). Then they also implement LSTM and PLSMT, and one layer CNN with one convolutional layer. For all the neural network models they initialize word representations with skip-gram algorithm with negative sampling (Mikolov et al., 2013).

Evaluation

All of the above mentioned papers use one or more of the following four evaluation measures: Precision (P), Recall (R), Accuracy (Acc) and $F1$, which are defined as:

$$P = \frac{TP}{TP + FP}, R = \frac{TP}{TP + FN}, Acc = \frac{TP + TN}{TP + TN + FP + FN}, F1 = \frac{2PR}{P + R}$$

where TP or True Positive is the number of correctly assigned test documents to the considered (for example positive) category, TN or True Negative is the number of correctly assigned negative label test documents, FN or False Negative is the number of negative assigned labels that should be positive, and FP or False Positive is the number of positive assigned labels that should really be negative.

Chapter 3

Data processing

3.1 Data collection

In our dataset we have two type of student surveys - a survey about a professor (related to a subject), and a survey about a subject (that might have more than one professor). Both surveys have questions in Catalan language, but answers are often also in Spanish or English. Surveys are formatted in *pdf* files.

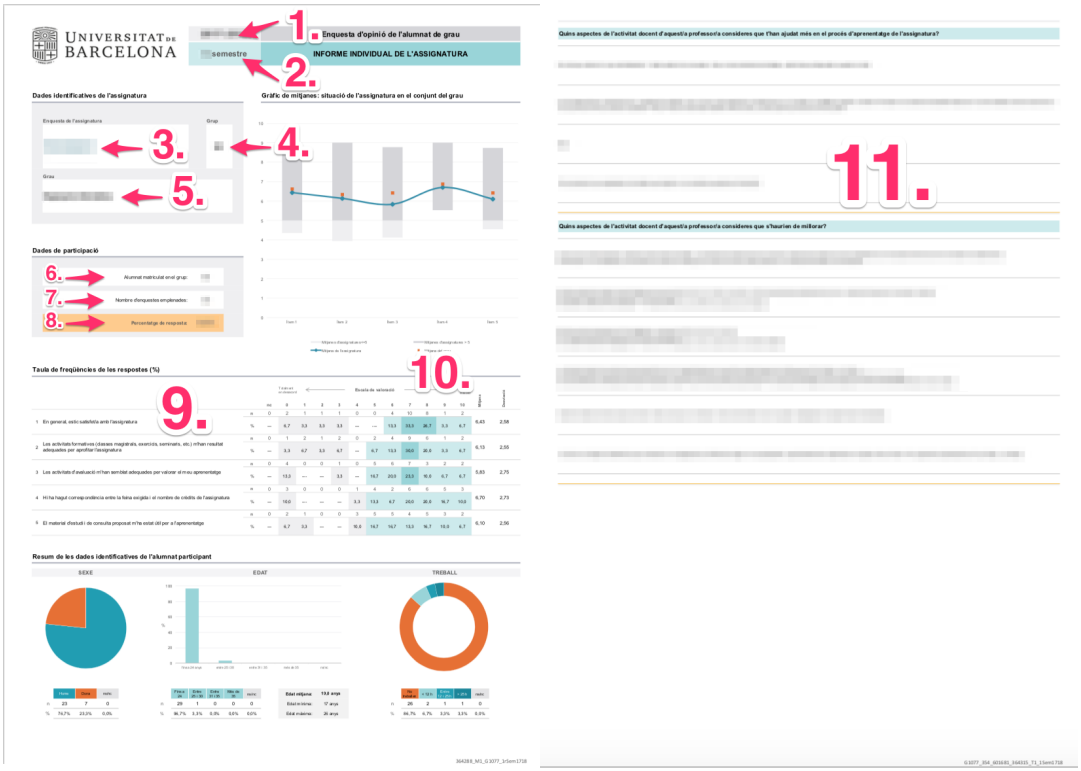

FIGURE 3.1: Example of a student survey

Figure 3.1 displays an example of a student survey, which contains the following information:

1. academic year,
2. semester,
3. professor name (for professor surveys),
4. group number,

5. subject name,
6. number of students enrolled in the course,
7. number of filled out surveys,
8. percentage of students who participated in the survey.

Furthermore each survey has some questions that are answered with grades from 0 to 10 (number 9. in Figure 3.1).

For professors the questions are:

1. whether the student is satisfied with the teaching activity of the professor,
2. whether the teacher maintains a good communication with students,
3. whether the professor clearly explains the contents of the subject,
4. whether the professor has adequately performed his tasks like teaching plan, program, delivery of evaluation etc.

And for subjects the questions are

1. whether the student is satisfied with the subject in general,
2. whether the learning activities were adequate,
3. whether the assessment activities were adequate,
4. whether the amount of work required was corresponding to the number of credits,
5. whether the study materials were useful.

For these questions there is information about frequency of the marks, the average for each question and the deviation of each question.

There are also open questions, for which our goal is to extract which sentiment is expressed. The order for all questions is random and currently it is not possible to match written answers to open questions with the points given to the questions from the first group. However it should be useful to explore this mapping in the future in order to obtain semi-supervised dataset.

The open questions for professors are:

1. which aspects of the teaching activity of this teacher has helped the students the most in the process of learning the subject,
2. which aspects of the teaching activity by this professor should be improved.

And the one open question for subjects is simply to write any comments, suggestions or observations.

For this thesis there are 113 surveys available for professors and 105 surveys for subjects. The surveys are from the academic years of 15/16, 16/17 and 17/18, and they are all from Computer Science bachelor programs in University of Barcelona.

To extract the information from the *pdf* I used a *Python* library *pdfquery* [1]. *Pdfquery* is a wrapper around several *Python* libraries for extracting data from *pdf* files. All the information from these files was extracted to a *MongoDB* [2] database so that later it is easy to query and process this information. This database has two collections - one for professors and one for subjects. To extract information from the first page, each box with information is found based on their exact coordinates on the *pdf* file. Then for the rest of the pages that have the open questions, all the pages are loaded, and then their *xml* content is downloaded. Then items of *xml* are sorted by their location on *pdf* (y axis) so that all answers and questions are in the correct order. To detect which items are questions we check for their coordinates, and read questions and answers one by one.

3.2 Data pre-processing

After the data was extracted from the *pdf* files to the MongoDB database, we preprocessed the data before feature extraction.

First of all, professor names were replaced by a placeholder within the comments to make all the data anonymous. Second, for each comment its language was detected, so that later comments with the same language can be grouped, and because feature extraction is language dependent. Third, all Spanish and Catalan comments were translated to English, in order to obtain new English corpora, that will be used in further experiments. Fourth, each sentence was processed with a part-of-speech tagger, tokenized and lemmatized. Last, each sentence was annotated, that is, marked as positive, negative, or neutral, in order to obtain a supervised dataset.

Data anonymization

Because the student surveys are not publicly available, it was important to replace all of the mentioned professor names with a placeholder. To find the names, a list of all the professor names from all the surveys was created, and these names and parts of these names were searched for in the comments. The meaningless parts of names were removed from the search list such as "de" and "mas". Then each name in comment was replaced with a placeholder "PROFESSOR_NAME".

Language detection

For language detection task *Python* libraries such as *langdetect* [3] and *polyglot* [4] were tested , but I found the *googletrans* [5] to be the most accurate (and also the slowest), as other libraries often failed to recognize languages, especially in shorter comments. The library *googletrans* implements Google Translate API in *Python*.

[1] https://github.com/jcushman/pdfquery

[2] Document oriented database, easy for storing and searching documents. https://www.mongodb.com/

[3] https://github.com/Mimino666/langdetect

[4] http://polyglot.readthedocs.io/en/latest/Transliteration.html

[5] https://pypi.org/project/googletrans/

Because this API has some restrictions regarding how many queries can be made at once, for looping through the comments I had to add some waiting time (0.4 seconds) between the processing of each comment.

With this library there were a few cases that wrongly detected Catalan language and French or Dutch, and some other mistakes when dealing with Catalan and English abbreviations. These cases were fixed manually.

After language detection, the corpus can be split in three parts with

- 1598 Catalan comments,
- 632 Spanish comments,
- 14 English comments.

The average word count per comment was 27.9 words for Spanish comments and 25.5 for Catalan comments, and average word count per sentence was 16.7 for Spanish sentences and 15.4 for Catalan sentences. In figures below it is visible that while the average word count per comment and sentence is quite high, most of the comments have less than 10 words.

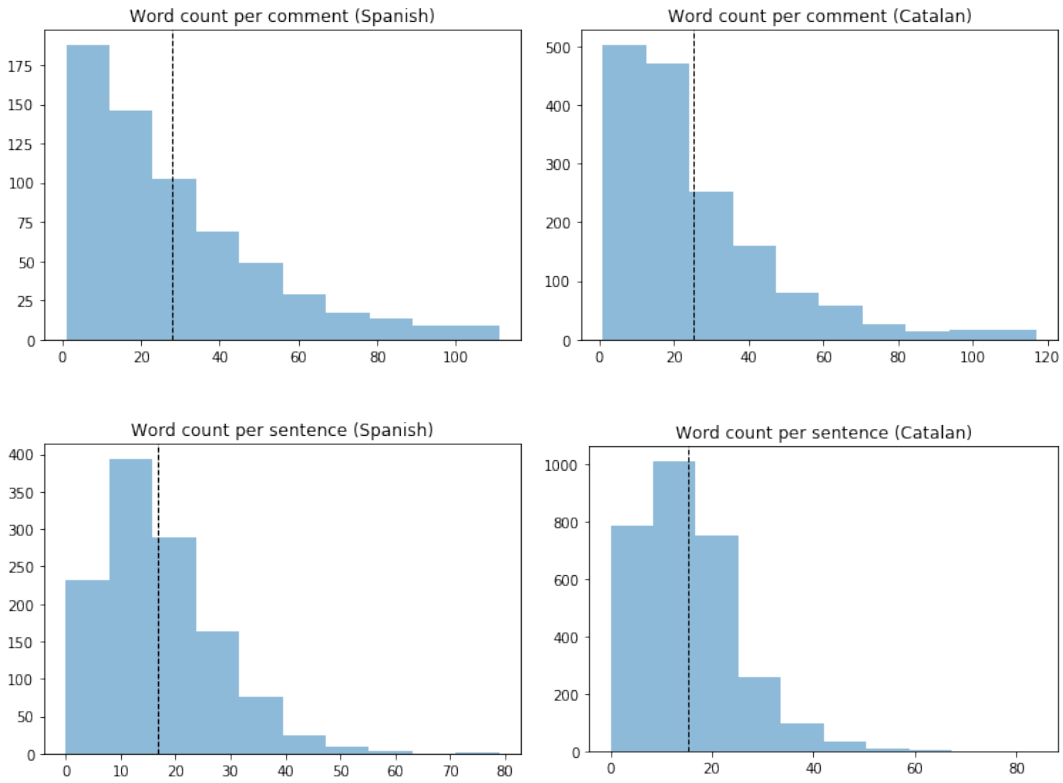

FIGURE 3.2: Average word count per comment and sentence for Catalan and Spanish

English corpora

To obtain the parallel English corpora, each non-English sentence was translated to English. For the task the same *Python* library *googletrans* as for the language detection was used. Each translated sentence was inserted in the MongoDB as a new field "sentences english".

Part-of-speech tagging

Part-of-speech tagger assigns to each word or a group of words its lemma (word in its dictionary form) and a part-of-speech tag.

Because *Polyglot* doesn't yet cover part-of-speech tagging for Catalan language, for this task *Freeling* [6] was used, which is an an open source language analysis tool that covers most feature extraction for both Spanish and Catalan.

For this tool (as well as for annotating) each sentence was saved in a *.txt* file, with file names like "language_PdfID_NumQuestion_NumSentenceInsideCommentary", and create a dictionary for *pdf* names and ID's. These files are anonymous, but to keep track of them and a database a dictionary of keys was created.

Annotation

To obtain a supervised dataset, each Spanish and Catalan sentence was annotated, that is, marked as either positive, negative or neutral. The annotation was done by STeL (El Servei de Tecnologia Lingüística de la Universitat de Barcelona) [7].

Creating files for the feature extraction

Next a new MongoDB collection was created with only the relevant fields, where each comment is a separate document. The fields are

- survey type (professor or subject),
- *pdf* file name as an ID,
- question number to which the comment belongs,
- language of the comment,
- sentences of a comment,
- sentences translated to English,
- sentiment polarity of each comment.

For these experiments the work was done both on sentence and comment level, therefore to split comments into sentences *Python* library *polyglot* was used, which is a natural language processing pipeline that supports different feature extraction for several languages. This library will also be used later for feature extraction.

For the purpose of the classification experiments, all database entries were converted to text files and grouped by language. The text files are available on GitHub [8].

[6] http://nlp.lsi.upc.edu/freeling/index.php/
[7] http://stel.ub.edu/
[8] https://github.com/IndraI/student-survey-analysis/blob/master/README.ipynb

Chapter 4

Feature extraction and Experimental setup

After data pre-processing, the resulting dataset has Spanish and Catalan sentences and their English translations marked as "positive", "neutral" or "negative". For each sentence there is also a corresponding *Freeling* output where all words in each sentence were lemmatized (each inflected word or group of words are converted to their dictionary form (lemma)), and each lemma has its part-of-speech tag.

For Spanish language comments, after pre-processing and annotation there are:

- 883 negative sentence;
- 169 positive sentences;
- 137 neutral sentences.

For Catalan language comments, after pre-processing and annotation there are:

- 1596 negative sentence;
- 885 positive sentences;
- 444 neutral sentences.

In this chapter I will describe how different features (bag-of-words for original and lemmatized texts, *n*-grams, *word2vec* features) were extracted from each sentence, as well as the whole comment and their English translation. Then I describe algorithms that are compared in the next chapter, and how they are evaluated.

4.1 Features

The following features are the most common features used for text representation for natural language processing. All these features were extracted for both sentences and comments in their original language (Spanish or Catalan) and their translation to English. These features and different combinations of them were used as input for different machine learning models (described in next section), and the results are analyzed in the next chapter. The code for extracting these features is in *GitHub* [1].

[1] https://github.com/IndraI/student-survey-analysis/blob/master/README.ipynb

4.1.1 Bag-of-words

The most simple way of representing texts is using a bag-of-words (BOW) model. These features will be used as a baseline for machine learning models. To create this model we need to create a vocabulary of all the words that appear in the training corpus. To obtain these words I removed punctuation, and lowercased all the words. The feature vector for each sentence (or comment) is in the length of the vocabulary. For Spanish the average feature vector length is 2801 (depending on train-test partition) and for Catalan it is 4952.

The most simple way to create feature vectors and the first feature type I test is for every sentence to put 0 for words that do not appear in that particular sentence and 1 for words that appear in that sentence. In some publications this method is also called BTO (binary term occurrences).

The second feature vector I used was to count frequencies of how many times each word appears in each sentence.

To reduce feature vector dimension and remove impact of meaningless words (stop-words) I deleted stop-words from the vocabulary. The list of Spanish and Catalan stop-words was downloaded from *GitHub*.

For both of these vectors I did experiments with and without TF-IDF weights that are described later in this chapter.

This type of feature vector on reports whether or how many times a word appears in text, but doesn't address questions like these:

- if a word is misspelled it will be interpreted as another word,
- each declension and conjugation of a word is interpreted as a separate word,
- word order and context is ignored.

4.1.2 *n*-grams

Creating n-gram features is similar to creating the bag-of-words features except that instead of taking one word, we take a sequence of n words. To create the vocabulary for n-grams we will create a list of all co-occurring words with a window of n. The length of each feature vector then will be the length of the vocabulary of n-grams. For Spanish and Catalan this length is varies from 10556 to 16864 depending on choice of n and train-test split.

For the experiments I tested how n-grams of different lengths perform. Because most of the sentences and comments consist of less than 20 words, it is probable that shorter n-grams might perform better. I will start with bi-grams ($n = 2$), tri-grams ($n = 3$), and see if further increasing the size of n improves the classification accuracy.

4.1.3 Lemmas and part-of-speech tags

Lemmatization is converting each word or group of words to its dictionary form, taking into account their context within the sentence. Part-of-speech tagging is labeling words with their part-of-speech tags (these tags contain information about the word class, declensions, conjugations, gender, count ans so on).

To convert words to lemmas and assign part-of-speech tags a language analysis tool *Freeling* was used. Tagsets for part-of-speech tagging for *Freeling* are based on

the guidelines from (EAGLES, 1996). With these guidelines it should be possible to "encode all existing morphological features for most European languages".

From lemmatized senteces I experimented with bag-of-words and n-gram feature vectors created from lemmatized words, POS tags as well as lemmatized words combined with their part-of-speech tags. Because the obtained part-of-speech tags are quite complex and include a lot of information (type, gender, conjugations and so on), I also experimented with keeping only the first letter of the part-of-speech tag, which indicates the type of word (noun, verb, pronoun and so on).

4.1.4 *Word2vec* embeddings

Another way to convert text to feature vector is to use pre-trained *word2vec* models. These models are neural networks that assign to each word a vector in n-dimensional vector space, such that words that appear in similar context will be be located closely in the vector space. To train such models a very large corpus is needed. For this task I downloaded all texts from Spanish and Catalan Wikipedia, and with this corpora I pre-trained word embeddings using *Gensim word2vec* [2] with 100 dimensions.

4.1.5 TF-IDF

TF-IDF or term frequency-inverse document frequency is a weight that can be calculated for each word, n-gram, lemma and so on. For the above mentioned features I calculated how many times a term (a unique word) appears in each sentence. This approach can be normalized taking into account that each sentence and comment is of different length, so for longer sentences terms can appear more times. Term frequency (TF) weight normalizes this by dividing the number each term appears in a sentence with the term count of that sentence.

$$\mathrm{TF}(t) = \frac{\text{number of times term appears}}{\text{total number of terms}}$$

Term frequency needs to be normalized further because some terms (for example, stop-words like "and" or "the") will have higher scores even though they are not meaningful. To decrease the weights for these terms inverse document frequency (IDF) can be calculated, which assigns larger positive weights if the term does not appear in too many other sentences, and smaller or negative weights if the term appears in most of the other sentences.

$$\mathrm{IDF}(t) = \log_e \left(\frac{\text{total number of sentences}}{\text{number of sentences that contain this term}} \right)$$

Term frequency-inverse document frequency (TF-IDF) combines both of these weights as

$$\text{TF-IDF}(t) = \mathrm{TF}(t) \times \mathrm{IDF}(t)$$

I apply this normalization to all of the above described feature vectors to see if this improves the classification. To perform this normalization I use *sklearn Tfidf-Transformer*.

[2] https://github.com/RaRe-Technologies/gensim

4.2 Machine learning models

4.2.1 Baseline score and sentiment polarity lexicon

Before applying any machine learning models calculated a baseline score by counting the number of positive and negative words in each sentence, and assigning a score "positive" or "negative" depending on which kind of words appear more, or a score "neutral" if a sentence has the same amount of positive and negative words. Lists of positive and negative words for Spanish and Catalan were downloaded from *Kaggle* dataset (Tatman, 2017) which contains sentiment polarity lexicons (positive and negative) for 81 languages.

Resulting accuracy for all the annotated data using this method was 0.481 for Spanish language and 0.399 for Catalan language. These are not good results, but in further experiments I will use these positive and negative vocabulary lists in combination with other features to see whether this improves the classification.

Another baseline score is the expected accuracy, that is the accuracy we can expect to get by simply guessing the most common class. This baseline is known as "most common class" baseline. . The Spanish dataset is very unbalanced with 74 percent of negative sentences, so the expected accuracy for Spanish is 74%. The Catalan dataset is more balanced, with 54 percent of negative sentences, so here the expected accuracy is 54%.

4.2.2 Model training and evaluation

To implement all algorithms I use *sklearn* [3]. Because with *sklearn* results can vary depending on the chosen random state, I will train and evaluate the model on 10 different random states, and average the results.

Because our dataset is unbalanced (there are many more data samples for negative data than for positive and neutral data) I will also experiment with adjusting the weights for each class.

For each setup (features + model) the train and test split was 3/4 to 1/4 of data. To evaluate the results of each setup a 4-fold cross-validation was performed by rotating the three-fourths of training data and testing on the remaining quarter of the data.

4.2.3 SVM

In the above described related work, support vector machines with linear kernel outperformed the other methods. This algorithm finds the maximum margin hyperplane that splits the training space into two classes. It is a computationally expensive algorithm, but because we have a small dataset, this will be the first method to test in this work. I trained a standard linear support vector machine with hinge loss and *l2* penalty. For implementation I will use *sklearn* with different penalty parameters and different number of iterations. The *sklearn* implementation supports multi class classification by applying the one-vs-all scheme.

4.2.4 Multinomial NB

Another method often mentioned in related work is Naive Bayes algorithm. Because we have discrete labels and our features contain frequencies and counts (of terms,

[3] http://scikit-learn.org/stable/index.html

n-grams) I tested Multinomial Naive Bayes algorithm. Even though in theory the Multinomial NB algorithm requires integer features, in practice normalized features (like TF-IDF) works as well. This algorithm models the probability of a term belonging to a certain class with a multinomial distribution (Manning, 2008).

4.2.5 Logistic regression

The next algorithm we tested was logistic regression. Logistic regression is a discriminative model that models the probabilities using sigmoid function. I implemented this algorithm with *sklearn* with *l2* penalty. With this implementation the multi class classification is supported by using one-vs-all approach. Like with support vector machines I tried different number of iterations, and adjusting weights for different classes.

4.2.6 Feed forward neural network

Last, for comparison, a simple feed forward neural network with 2 hidden layers was implemented with *Python* deep learning library *Keras* [4]. This model is trained with *Adam* optimizer with cross-entropy loss, 20 training epochs and batch size 128.

[4] https://keras.io/

Chapter 5

Experimental results

This chapter introduces the results after performing all the experiments. The main research questions of this chapter are:

- which feature setup gives the best accuracies for each class;
- which model performs the best;
- does automatically translating comments to English improve the classification results.

The experiments were performed at sentence level which is the original annotated dataset and at comment level. To obtain the labels for comments, I counted the number of positive, negative and neutral marked sentences in the comments, and chose the most frequent label. These labels on comment level I also added to *MongoDB* database. For these experiments I trained and tested on full comments.

In this report I only report results for Catalan, as for this language there is a larger and well balanced dataset, as well as because this language is less researched. Experiments for Spanish are in *Github*.

It is possible to get higher accuracy, by not balancing class weights. This way we can have high results for the negative and positive class, but also very low results for the neutral class (as it only has a few training samples). All of the reported results have balanced class weights. The other experiments are in *Jupyter* notebooks.

For the sentence level (Catalan dataset) the train-test split is

- 2191 instances for training;
- 731 instances for testing.

And for comment level this split is

- 1198 instances for training;
- 400 instances for testing.

This split varies a bit for different folds in cross-validation.

For each model family I only report results of the best performing setting. Similarly for each feature family (BOW, *n*-grams, lemmas and POS, *word2vec*) I tested different combinations with count, frequencies, TF-IDF, removing stop-words, but here I only report the best performing feature combinations.

5.1 Evaluation metrics

For selecting the best models and features I report only the accuracy metrics, which provides the proportion of correctly classified sentences or comments against all of the sentences or comments. In the notebooks however I also compare the confusion matrices to see if neutral class is not excluded.

$$\text{accuracy} = \frac{\text{count of correctly classified examples}}{\text{count of all the examples}}$$

For the best performing feature and model combinations I also report confusion matrix.

		Predicted class		
		positive	negative	neutral
True class	positive	pos-pos	pos-neg	pos-neu
	negative	neg-pos	neg-neg	neg-neu
	neutral	neu-pos	neu-neg	neu-neu

FIGURE 5.1: Confusion matrix

In addition for the best models I also report precision, recall and F_1 score for each class, where

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}; \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FP}}; F_1 = \frac{2\text{PR}}{\text{P} + \text{R}}$$

where TP - true positives, TN - true negatives, FP - false positives and FN - false negatives.

5.2 Results

For training support vector machines and logistic regression 5 iterations were performed. For these two models the best accuracy is often achieved when all the classes ("positive","negative" and "neural") have the same weights (that is, weight for each class is one). However in this case neutral class is usually very badly classified with good results on positive and negative classes. Because of this, all the reported examples have balanced weights where the weight of each class is inversely proportional to frequency of this class in the training data.

Below are reported model performances on different features created from original words, lemmatized words or POS tags, with results at sentence level, results with sentences automatically translated to English, and results at comment level.

In this report I do not report the results of Multinomial Naive Bayes algorithm as it performs badly and doesn't deal well with unbalanced classes.

5.2.1 Results with bag-of-words feature vectors created from original vocabulary

In the table 5.2. are reported results with different type of features using bag-of-words approach, with vectors created from original words (not lemmatized).

The first result - bag-of-words vectors with binary term occurrences as features trained with support vector machine algorithm is 0.647. This is the most simple machine learning model, therefore this result can be interpreted as a baseline result for the machine learning methods.

	SVM	Logistic regression
unsupervised baseline	0.399	
most common class baseline	0.54	
BOW BTO	0.647	0.68
BOW TF-IDF	0.677	0.706
BOW frequencies	0.633	0.683
BOW frequencies TF-IDF	0.673	**0.707**

FIGURE 5.2: Results with bag-of-words features (original texts) on sentence level

Further in table 5.2. we present the results of bag-of-words feature vectors with term frequencies. For support vector machines these vectors give lower results, while for logistic regression algorithm the results are a little better.

Using TF-IDF on both binary term occurrences and term frequencies significantly improves the result for both methods and both feature vectors. With logistic regression reaching accuracy of 0.707.

	SVM	Logistic regression
unsupervised baseline	0.399	
most common class baseline	0.54	
BOW BTO	0.623	0.661
BOW TF-IDF	0.654	**0.682**
BOW frequencies	0.607	0.644
BOW frequencies TF-IDF	0.651	0.671

FIGURE 5.3: Results with bag-of-words features (original texts) on comment level

In Figure 5.3. we display the results for the same model and feature setup as above but trained and evaluated at comment level. Resulting accuracies are a bit lower than at the sentence level. This can be explained with the fact that the comment level has fewer training data. Also annotations may not always be completely

accurate, because they were calculated automatically by counting the number of positive and negative sentences in each comment. This might not always be accurate, as it is possible that a comment has an equal number of positive and negative sentences, but is actually an overall negative or positive comment (rather than an overall neutral comment).

	SVM	Logistic regression
unsupervised baseline	0.399	
most common class baseline	0.54	
BOW BTO	0.644	0.679
BOW TF-IDF	0.692	0.705
BOW frequencies	0.642	0.681
BOW frequencies TF-IDF	0.693	**0.705**

FIGURE 5.4: Results with bag-of-words features on English translations

Figure 5.4. shows the same features and models trained and tested on original sentences translated to English. The results are improved for support vector machine and are about the same for logistic regression.

5.2.2 Results with bag-of-words feature vectors created from lemmatized texts and POS tags

As mentioned above, a disadvantage of using bag of words features can be that the same word in different forms or conjugations, or the same word with a spelling error will be interpreted as a completely separate word. This has bigger impact on Catalan language than on English as it has more word forms, for example, many inflected forms for verbs. This is avoided by using lemmas instead of tokens.

For Catalan language using lemmatized words reduces the length of vocabulary from 4941 to 3330 words.

	SVM	Logistic regression
unsupervised baseline	0.399	
most common class baseline	0.54	
BOW BTO	0.65	0.69
BOW TF-IDF	0.7	0.711
BOW frequencies	0.642	0.685
BOW frequencies TF-IDF	0.698	**0.712**

FIGURE 5.5: Results with bag-of-words features on lemmatized words

From table 5.5. it is clear that using lemmas rather than the original word forms has improved the results for all bag-of-words feature vectors and for both algorithms. Now the best accuracy - 0.712 - is obtained by using bag-of-words feature vectors with term frequencies and TF-IDF with lemmatized vocabulary, trained with logistic regression algorithm.

	SVM	Logistic regression
unsupervised baseline	0.399	
most common class baseline	0.54	
BOW BTO	0.499	0.548
BOW TF-IDF	0.518	0.564
BOW frequencies	0.466	0.546
BOW frequencies TF-IDF	0.528	**0.574**

FIGURE 5.6: Results with bag-of-words features on POS tags

Another type of feature to test is part-of-speech tags. Here for each word I took the first letter of part of speech tag (this denotes the type of word) and calculated all the bag-of-words vectors with these part-of-speech tags. Results are in the table above. It is clear that the results are not good with only using part-of-speech tags, but combining these features with others might improve results (this is tested in later sections). Applying TF-IDF improved results in all the cases, with best result again being the most complex bag-of-words features trained with logistic regression.

5.2.3 Results with *n*-gram feature vectors created from the original vocabulary

In figure 5.7. I displayed the results for logistic regression model with n-gram features for $n = 2, 3, 4, 5$ with and without applying TF-IDF normalization. These results are calculated on sentence level on lemmatized texts. It is clear that n-grams with n larger than 2 perform worse, and it is only worth to further look at bi-grams with and without TF-IDF.

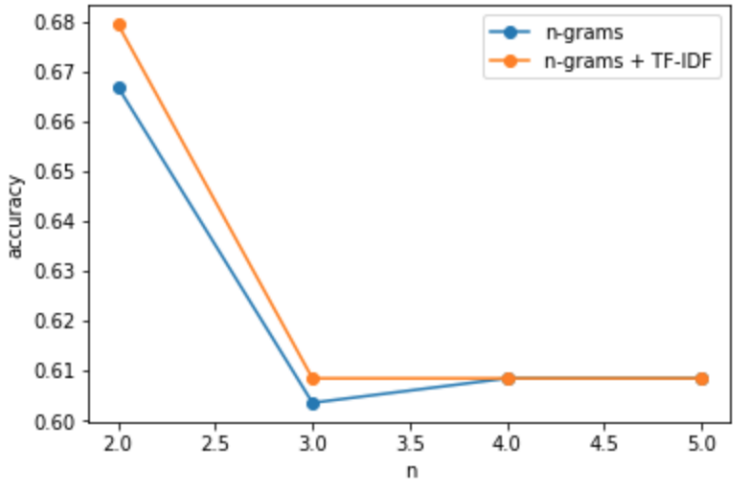

FIGURE 5.7: Logistic regression accuracy with different order *n*-grams

In the tables for sentence level, comment level and translated sentences, are reported model performances on bi-grams with without applying TF-IDF normalization.

Results for all three tables are worse than using bag-of-words features. English translations have almost the same accuracy as the original sentences. Accuracies are better on comment level, which is likely because many sentences are too short to contain several meaningful bi-grams.

	SVM	Logistic regression
unsupervised baseline	0.399	
most common class baseline	0.54	
bigrams	0.609	0.613
bigrams tf-idf	0.609	0.642

FIGURE 5.8: Results with 2-gram features (original texts) on sentence level

	SVM	Logistic regression
unsupervised baseline	0.399	
most common class baseline	0.54	
bigrams	0.625	0.645
bigrams tf-idf	0.6	0.666

FIGURE 5.9: Results with 2-gram features (original texts) on comment level

	SVM	Logistic regression
unsupervised baseline	0.399	
most common class baseline	0.54	
bigrams	0.61	0.613
bigrams tf-idf	0.609	0.642

FIGURE 5.10: Results with 2-gram features on English translations

5.2.4 Results with *n*-gram feature vectors created from lemmatized texts and POS tags

In the tables for lemmatized words and POS tags, are reported model performances on bi-grams and tri-grams with and without applying TF-IDF normalization.

As expected, using lemmatized words perform better than using the original words, however these features do not perform as well as the bag-of-words features. Still, these results are better than the machine learning baseline, that is, 0.647% accuracy, so concatenating these feature vectors with bag-of-words feature vectors might improve the accuracy of classification.

	SVM	Logistic regression
unsupervised baseline	0.399	
most common class baseline	0.54	
bigrams	0.647	0.667
bigrams tf-idf	0.654	**0.679**
trigrams	0.573	0.604
trigrams tf-idf	0.558	0.608

FIGURE 5.11: Results with *n*-gram features on lemmatized words

In the table 5.12. the results of using part-of-speech *n*-grams as features are presented. These results are actually better than part-of-speech bag-of-words features where the best accuracy was 0.574%, while here the best accuracy is 0.588%.

	SVM	Logistic regression
unsupervised baseline	0.399	
most common class baseline	0.54	
bigrams	0.492	0.565
bigrams tf-idf	0.577	0.588
trigrams	0.52	0.547
trigrams tf-idf	0.587	0.588

FIGURE 5.12: Results with *n*-gram features on POS tags

5.2.5 Results with *word2vec* feature vectors

In the table 5.13. I report results for original sentences, full original comments, and lemmatized sentences using *word2vec* with TF-IDF features, as they always performed better that *word2vec* features without TF-IDF.

These vectors are trained on Catalan *Wikipedia* corpus. Texts in this corpora belong to very different domains from our survey domain texts, so for future to get better results it would be useful to train *word2vec* embeddings on a domain specific corpus. Still comparatively to the bag-of-words vectors, they perform quite as good.

	SVM	Logistic regression
unsupervised baseline	0.399	
most common class baseline	0.54	
original sentences	0.651	0.655
comments	0.633	0.643
lemmas	0.665	**0.668**

FIGURE 5.13: Results with *word2vec* with *TF-IDF* features on sentence level

5.2.6 Combining feature vectors

For this part I combined different feature vectors and applied the support vector machines and logistic regression on them. On the table 5.14 are reported some of the best combinations, with the best result achieved by using logistic regression with combined bag-of-words term frequency with TF-IDF, bi-grams with TF-IDF and tr-grams with TF-IDF, which gives the best accuracy 0.726.

	SVM	Logistic regression
unsupervised baseline	0.399	
most common class baseline	0.54	
bow freq tfidf + 2gram tfidf	0.71	0.722
bow + bow POS	0.69	0.712
bow + 2gram + 3gram	0.715	**0.726**
bow + 2gram + 3gram + 2gram POS	0.706	0.725

FIGURE 5.14: Results with combined feature vectors

5.2.7 Feed forward neural network

For comparison I trained a simple feed forward neural network with 2 hidden layers. This network was trained with only 20 epochs, and it generally performed similarly as the other two machine learning models. The accuracies were:

- 0.665 for bag-of-words with term frequencies and TF-IDF;
- 0.697 for lemmatized bag-of-words with term frequencies and TF-IDF;
- 0.71 for the best performing feature combination in the other machine learning models, that is, lemmatized bag-of-words with term frequencies and TF-IDF combined with bi-grams and tri-grams, both with TF-IDF.

5.3 Discussion of the results for Catalan language dataset

Catalan dataset was balanced in that 54% of examples were of negative class. However only 15% of the examples were from neutral class, which led to lower results for accuracy for neutral examples. In the practical part I compared models with and without balancing class weights, however in the report all the results are with balanced class weights.

The expected accuracy for Catalan dataset is 54%, as this is the percentage of the majority class, meaning that this number we can expect to get by always guessing the majority class. We used this number as a baseline accuracy for all the models.

In this report two machine learning models were compared - support vector machine and logistic regression. Logistic regression models generally performed better than the support vectors machine models. In the practical part also Multinomial Naive Bayes model was tested, and while it had a good accuracy it did not deal well with the unbalanced neutral class, and did not predict it.

For most of the first setups I also trained the same models on comment level and on the sentences that were automatically translated to English, to answer question whether translating to English will improve the accuracy. On comment level almost all features (except n-grams) performed worse than on the sentence level. This is because, first of all, on comment level there are much less training examples, and second, the annotations for comment level were calculated automatically from the sentence level annotations so they might contain some mistakes.

As for the English translations, they perform almost as good as the original sentences, and for support vector machines, in some cases they perform even a bit better. When translating texts automatically, translation mistakes can accumulate and

lead to worse results. However English language is the most researched for the sentiment analysis, and all the features I am comparing have been tested a lot and are often good for English language corpora. Also translating Catalan to English might result in more simple sentences as English does not have that many word inflections.

First the most simple features were used, that is, bag-of-words with binary term occurrences. These features set the baseline for machine learning models at 0.647 for support vector machines models and 0.707 for logistic regression models.

Then more complex features were discussed and compared, such as different bag-of-words features, *n*-gram features and *word2vec* features. The more complex bag-of-words features improved the classification accuracy, while the other more complex feature types alone did not result in higher accuracy.

However *n*-grams did lead to better results when using full comments rather than sentences as a training and testing data. This is because, as shown in a previous chapter, sentences are mostly very short, and combining them into comments give some longer sentences. However many comments consist of only one sentence.

Big improvement for all the feature setups was achieved by using word lemmas rather than words in their original inflections. All the feature setups and models gave somewhat better results with using only lemmas.

Further, after using all the feature vectors as input to machine learning models separately, I compare combining different feature vectors, which leads to the best accuracy when combining bag-of-words term frequency, bi-grams and tri-grams (all with TF-IDF and created from lemmatized sentences), with the best accuracy of 0.726.

Last, a simple feed forward neural network was tested. With the best feature setup, it resulted in 0.71 accuracy. This result could be improved by adding more training examples and performing the training for more epochs.

This final table reports the steps that helped improve the accuracy from the baseline to the final best results. The baseline for counting positive and negative words was 0.399 and the baseline of expected accuracy was 0.54. For support vector machine models the first baseline was 0.65 and for logistic regression models 0.726.

We can see that logistic regression models generally performs better that support vector machine models with all of the features reported in this table. However, we can see that, support vector machine models are very impacted by using different feature vectors, while for logistic regression models variations in results are not that wide.

	Unsupervised	SVM	Logistic regression
Sentiment polarity lexicon	0.399		
Expected accuracy	0.54		
BOW BTO		0.65	0.69
BOW freq TF-IDF		0.698	0.712
bow + 2gram + 2gram POS		0.705	0.719
bow freq tfidf + 2gram tfidf		0.71	0.722
bow + 2gram + 3gram		0.715	**0.726**

FIGURE 5.15: Accuracy improvements by using different feature vectors and methods.

In the two figures below (figure 5.16. and figure 5.17) are presented confusion matrices for the best setups (that is, bag-of-words with term frequencies, bi-grams and tri-grams) for both models, support vector machine and logistic regression. There is a big improvement from the expected accuracy, which is:

- 30% for the positive class;
- 54% for the negative class;
- 15% for the neutral class.

All the classes have improved accuracies. Results with support vector machine are a bit more balanced, but have lower overall accuracy than results with logistic regression.

		Predicted class		
		positive	negative	neutral
True class	positive	**0.687**	0.258	0.553
	negative	0.663	**0.878**	0.056
	neutral	0.13	0.62	**0.25**

FIGURE 5.16: Confusion matrix for the best accuracy with support vector machine.

		Predicted class		
		positive	negative	neutral
True class	positive	**0.673**	0.281	0.046
	negative	0.074	**0.895**	0.031
	neutral	0.157	0.63	**0.213**

FIGURE 5.17: Confusion matrix for the best accuracy with logistic regression.

5.4 Discussion of the results for Spanish language dataset

In the practical part also experiments with Spanish dataset was done. However, this datasets is very small (three times smaller than the Catalan dataset), as well as very unbalanced with 0.74% of the expected accuracy.

To deal with this class imbalance I tried to balance the number of training examples based on the less represented class, but this led to an even smaller number of training examples.

The same experiments as for Catalan were also done for Spanish. The best result achieved was 0.75%, that is only one percent above the expected accuracy. Most of the setups performed best when simply guessing the majority negative class, and when forced to predict other classes with adjusting class weights, led to worse overall accuracy.

This dataset needs to be improved by obtaining more data examples for training.

Chapter 6

Conclusion and future directions

The goals of this project were

1) to create a dataset for supervised learning from quantitative answers from student surveys from Computer Science bachelor programs in the University of Barcelona; and

2) to perform experiments with the obtained dataset for sentiment polarity detection with different feature setups and models.

First of all, some of the related work on dataset creation and sentiment polarity detection for similar datasets in English and Spanish was introduced and analyzed. The related work, however, was done on larger datasets, that were focused on different domains (movie reviews, *Amazon* product reviews).

For creating the dataset, first of all, I extracted all the answers from the survey *pdf* files and saved them to the database. Then I performed data cleaning, splitting comments in sentences, anonymization and language detection. After that, the anoymized data was sent to STel institute were they were annotated by a native Spanish and Catalan speakers. In addition, for all the words were lemmatized and assigned a part-of-speech tag, All these steps resulted in two supervised datasets - one in Spanish and other one in Catalan. Both datasets are novel and are addressing limitations of the currently available resources for NLP and Sentiment Analysis in terms of source language and domain. There are very few datasets available for languages other than English, and even fewer for lower resource languages like Catalan. The educational domain is also not very well represented in the areas of Sentiment Analysis and Polarity Detection, with most of the datasets built around product reviews.

Further, with the obtained supervised dataset I performed several experiments. For for all the experiments I tokenized the text by splitting it into words and removing punctuation. First of all, I calculated the unsupervised polarity score, by using a predefined word list with positive and negative Catalan words. Then I compared two supervised machine learning models that are often used for sentiment polarity detection with English datasets - support vector machine and logistic regression. I also tested the Multinomial Naive Bayes algorithm. For support vector machine and logistic regression algorithms, I compared results with and without balancing weights for each class. In this report all of the mentioned result are achieved by adding weights to each class that are inversely proportional to frequencies of each class in the training data. In addition I also trained a simple feed forward neural network to compare the results.

For the machine learning models I compare some of the most common features and their combinations from the related work. I start with the most simple feature vectors, that is, bag-of-words with binary term occurrences. Then I add term frequencies, TF-IDF, n-gram feature vectors and pre-trained *word2vec* feature vectors. I

created these feature vectors with the original inflected words, as well as with lemmatized words and with part-of-speech tags. All vectors created with lemmatized words performed significantly better than the vectors created with inflected words.

Logistic regression algorithm generally gave better accuracy results than support vector machine, however when looking at confusion matrices, results with support vector machine were a bit more balanced for all the classes than results with logistic regression.

The best result, 0.726 ,was achieved by using logistic regressions and concatenating feature vectors of bag-of-words with term frequencies, bi-grams and tri-grams (all created from lemmatized sentences, with TF-IDF).

Experiments were also performed on sentences translated from Catalan to English. Results were similar (but a bit worse) than results from the original sentences.

Experiments on comment level performed worse than experiments on sentence level, mainly because using full comments resulted in much less training examples.

These results also validate the quality of the dataset. They show that

1) the dataset can be used to learn to classify opinions in a supervised manner. This is evident by the systems outperforming the unsupervised baseline or the most common class baseline;

2) the dataset is not trivial. That is, while the different systems can obtain result higher than the baseline, their predictions are still far from human-level performance.

This project was a first step towards evaluating open question answers from student surveys. This project provides a framework for creating a supervised dataset of student opinions and a variety of polarity detection systems that can be applied to that dataset. To further improve the results, the dataset should be improved by adding more examples for training. If more neutral examples were obtained, it would lead to better accuracy for the neutral examples as well as to a better overall accuracy. If more training examples are obtain, it would also lead to better results with deep learning algorithms. To obtain better *word2vec* features, it would be useful to train these vectors on a more specific domain that the Wikipedia.

Another possibility to improve the annotations could be to connect the open textual answers in the surveys to the corresponding answers to the numerical questions. That would lead to a large annotated dataset without having to do the annotations manually.

A further application for this dataset would be to create system that given all the survey *pdf* files, would create a report of how many positive and negative comments each professor and subject has and to compare them with the numerical scores.

Bibliography

Baccianella, Stefano, Andrea Esuli, and Fabrizio Sebastiani (2010). *SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining*.

Balahur Dobrescu, Alexandra, Rada Mihalcea, and Andres Montoyo (2014). *Computational approaches to subjectivity and sentiment analysis: Present and envisaged methods and applications*.

Barnes, Jeremy, Roman Klinger, and Sabine Schulte im Walde (2017). "Assessing State-of-the-Art Sentiment Models on State-of-the-Art Sentiment Datasets". In: *CoRR* abs/1709.04219. arXiv: 1709.04219. URL: http://arxiv.org/abs/1709.04219.

Barnes, Jeremy, Patrik Lambert, and Toni Badia (2018). "MultiBooked: A Corpus of Basque and Catalan Hotel Reviews Annotated for Aspect-level Sentiment Classification". In:

Cavnar, William B. and John M. Trenkle (1994). "N-Gram-Based Text Categorization". In: *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pp. 161–175.

Denecke, K. (2008). "Using SentiWordNet for multilingual sentiment analysis". In: *2008 IEEE 24th International Conference on Data Engineering Workshop*, pp. 507–512. DOI: 10.1109/ICDEW.2008.4498370.

EAGLES (1996). "Recommendations for the Morphosyntactic Annotation of Corpora". In: *Expert Advisory Group on Language Engineering Standards*. URL: http://home.uni-leipzig.de/burr/Verb/htm/LinkedDocuments/annotate.pdf.

Graovac, Jelena (2014). "A variant of n-gram based language-independent text categorization". In: 18, pp. 677–695.

Graovac, Jelena and Gordana Pavlovic-Lažetic (2014). *Language-Independent Sentiment Polarity Detection in Movie Reviews: A Case Study of English and Spanish*.

Kaur, Amandeep and Vishal Gupta (2013). "A Survey on Sentiment Analysis and Opinion Mining Techniques". In: 5.

Kešelj, Vlado et al. (2003). "N-Gram-Based Author Profiles For Authorship Attribution". In:

Manning (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press. ISBN: 0521865719, 9780521865715.

Martín-Valdivia, MaríA-Teresa et al. (2013). "Sentiment Polarity Detection in Spanish Reviews Combining Supervised and Unsupervised Approaches". In: *Expert Syst. Appl.* 40.10, pp. 3934–3942. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2012.12.084. URL: http://dx.doi.org/10.1016/j.eswa.2012.12.084.

McAuley, J. and J. Leskovec (2013). "From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews." In: URL: http://www.cs.cornell.edu/people/pabo/movie-review-data.

Mikolov, Tomas et al. (2013). "Efficient Estimation of Word Representations in Vector Space". In: *CoRR* abs/1301.3781. arXiv: 1301.3781. URL: http://arxiv.org/abs/1301.3781.

Pang, Bo and Lillian Lee (2004). "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts". In: *Proceedings of the ACL*. URL: http://www.cs.cornell.edu/people/pabo/movie-review-data.

Sygkounas, Efstratios, Giuseppe Rizzo, and Raphaël Troncy (2016). "Sentiment Polarity Detection from Amazon Reviews: An Experimental Study". In: *Semantic Web Challenges*. Ed. by Harald Sack et al. Cham: Springer International Publishing, pp. 108–120. ISBN: 978-3-319-46565-4.

Tatman, Rachael (2017). "Sentiment Lexicons for 81 Languages". In: *kaggle*. URL: https://www.kaggle.com/rtatman/sentiment-lexicons-for-81-languages/data.