Sentiment Analysis:
Incremental learning to build domain models

Raimon Bosch

Master thesis, Universitat Pompeu Fabra (2013)

Intelligent Interactive Systems

Universitat Pompeu Fabra Barcelona

Prof. Dr. Leo Wanner,

Department of Information and Communication Technologies

Abstract

Nowadays, social contacts are vital to find relevant content. We need to connect with people with similar interests because they provide content that matters. Every day is more clear that in the future of document recommendations will be necessary to cross the traditional data with the data obtained from social networks. For instance, in order to provide the best content available we can use sentiment analysis techniques to prioritize content with good reviews. The aim of this project is to offer a better sentiment recognition strategy.

In this master thesis we have worked analyzing short messages about brands in Twitter trying to classify them between positive and negative using Sentiwordnet. After several experiments, we have seen that applying a semi-supervised approach we could increase the quality of the dictionary and adapt it to a specific domain. In the second part of the project we wanted to get one step further by analyzing relevant content inside those tweets to know also the reason why something is positive or negative. Due to the lack of strong grammatical structures inside tweets we had to go for an approach based on structured N-grams. For that, we have modelized a new idea called sentigram that consists in the aggregation of several N-grams. This approach allows to create models very precise to specific domains and at the same time capture the relation between aspects and sentiment words.

Keywords: sentiment analysis, natural language processing, opinion mining, twitter, n-grams, sentigrams, aspect identification, social networks, machine learning.

Contents

1. Motivation.....5
1.1 An opportunity to perform changes.....5
1.2 The information era is here.....5
1.3 Hidden data inside opinions.....5
1.4 Structure of this thesis.....6
1.5 Research goals.....6
2. Introduction to sentiment analysis.....7
2.1 Feature selection for sentiment analysis.....8
2.1.1 N-grams + Negations.....8
2.1.2 Pos-Tagging.....8
2.1.3 Syntax.....8
2.1.4 Term frequency.....9
2.1.5 Noun identification and other wildcards.....9
2.1.6 Punctuation signs + stopwords removal.....9
2.1.7 Authority and trust.....10
2.1.8 Twitter-specific features.....10
2.2 Classification techniques.....10
2.2.1 Naive Bayes.....10
2.2.2 Pointwise Mutual Information.....11
2.2.3 Vector Distance.....11
2.2.4 Support Vector Machines.....12
2.2.5 Maximum Entropy.....12
2.2.6 Voting among several classifiers.....13
2.3 Applications.....13
3. State of the art.....14
3.1 Use of sentiment dictionaries.....14
3.2 Subjectivity Summarization.....15
3.3 Using Twitter to make predictions of events.....17
3.4 Cross-domain models.....20
3.5 Edge cases and ambiguity.....21
3.6 Theoretical aspects.....22
3.6.1 The Art of feature selection.....22
3.6.2 Training and classifying data.....23
3.6.3 The power of subjectivity summarizations.....24
3.6.4 Is it possible to predict events with sentiment analysis?.....24
3.7 Topics not covered in state-of-the-art.....25
3.7.1 Knowledge dictionaries.....25
3.7.2 Structured N-grams.....25
3.7.3 Sentiment as a problem, desementilaization as a solution.....25
3.7.4 Buzz detection.....26

3.7.5 External factors on comments.......26
3.7.6 Big data can help?.......26
4. Proposal.......27
4.1 Sentigrams.......28
4.2 Incremental learning.......28
4.3 Automatization....... 29
4.4 Implementation.......30
4.4.1 Twitter Api....... 30
4.4.2 Sentiwordnet/Dictionary.......30
4.4.3 Hadoop Processor....... 31
4.4.4 Rails application (Web/Client side)....... 32
4.4.5 Annotations....... 32
4.4.6 Weka Module....... 32
5. Experiments....... 33
5.1 Dataset.......33
5.2 Experiments with Sentiwordnet....... 33
5.2.1 Co-occurrence scores.......33
5.2.2 Negative N-grams....... 34
5.2.3 Stemming....... 34
5.2.4 No Disambiguations.......34
5.3 Experiments with Sentiwordnet after random-walk.......35
5.3.1 Random-walk.......35
5.3.2 Random-walk with domain words....... 35
5.4 Experiments on sentigram prediction with Weka.......36
6. Conclusions.......37
6.1 Final remarks.......37
6.2 Research questions....... 37
6.3 Future work....... 38
7. References.......39

1. Motivation

Sentiment Analysis is a technology that will be very important in the next years. With opinion mining we can distinguish poor content from high quality content. And also, we have a tool able to calculate a social rank for any brand that offers online content. With the technologies available we can know if a brand has more good opinions than bad opinions and find the reasons why those opinions are positive or negative. This can allow brands to understand better its clients and anticipate change. On the other side it allows clients to do more informed decisions and stay alert about bad practices. The research field for this project will be natural language processing and specifically we will explore sentiment analysis technologies in order to extract useful information about brands, politics or products.

1.1 An opportunity to perform changes

What we can do to perform better decisions? How can we recover some decision power in our daily routine? It is very common to think that big changes must be done by leaders that are there to decide for us. But this shouldn't be true. Every time that we are in a conversation, every time we read a book or even every time we buy a product we make little decisions that can positively influence in our community. Becoming a world-changing person is a hard task. It is not possible to change the world without information. If we want to solve a problem we need to point out to the people who caused this problem. We have to define some good practices and make sure that most of us follow those rules. For that, is necessary information and we need the ability to share this information quickly and do it with enough openness to detect those informations that might look manipulated or not contrasted.

1.2 The information era is here

The rise of social networks has given us this opportunity of having an alternative media where we can connect directly with close people (in a topic or in a group of friends) to transmit useful information. Search engines are also an important part of this change. Through a search engine we can answer questions that some years ago took days or weeks to solve. We can easily access to knowledge and discuss about this knowledge which makes us more able to question thoughts and take informed decisions.

A change is happening right now. By having free access to information the quantity of knowledge that we can process in one day it has nothing to do with the quantity that we used to process before internet was here. Furthermore, we do not need to memorize pointless information anymore. Since we have gadgets able to answer questions in seconds we do not need to put effort in memorizing rules or ideas. We can focus on understanding concepts. The maxim that a good researcher is that one that asks the right questions is more true than ever.

1.3 Hidden data inside opinions

Sentiment analysis is a discipline that can be defined as a set of structured properties that we want to find inside a text. When we read an opinion we want to know what is talking about (object), and what are the characteristics of this object (features). For each one of this features we want to know the opinion direction (positive, negative). And finally, we want to add all this opinions directions in a summary. What we can do with this summary? This summary can be presented to the user to inform him about a topic of his interest.

One of the missing areas in sentiment analysis is how do we isolate opinions from big media from small consumer's opinions. This consumer that sells a phone because the battery life does not fit its expectations, this consumer that does not want to involve in a company anymore because he is disappointed, or this citizen that expresses opinions outside of the mainstream. Every time we buy, we vote, we search all those expectations that we have about the world will be taken into account to show you enough data to take the best decision. Reading opinions before buying something has become an habit, but there is not still an application able to centralize this opinions effectively like Google does with information. Will it be the next Google a sentiment search engine? We don't know yet. What is sure is that people need more elements to understand better the world and sentiment analysis is one of them.

1.4 Structure of this thesis

This master thesis work is structured in 5 chapters. Motivation (1), Introduction to sentiment analysis (2), State of The Art (3), Our Proposal (4), Experiments (5).

In chapters 2 and 3 we will present the field of sentiment analysis and we will explore the different state-of-the-art solutions that have been written before. We will explain what are the main development areas on sentiment analysis. Also, we will point out the main papers in the area specially those studies that had based its analysis on tweets.

Inchapters 4 and 5 we will present our findings and our experiments and we will explain in detail how our prototype works and what differences it has with other state-of-the-art solutions.

1.5 Research goals

The main goal of this work is to build a software prototype able to rank social opinions from Twitter. We have chosen Twitter because it is a public social network where users can complain easily about the things they don't like. If we are able to give a social rank to each twitter account for any possible topic we will be able to provide to the users a list with the best brands, products and services to choose. This information will be crucial if we want to provide quality content to users. Sentiment analysis can predict which brands are reliable and which don't. And in consequence which content has more general acceptation.

Our prototype should be able to adapt to several domains with minimum effort and deal with the lack of strong grammatical rules of short messages on Twitter. Also we should get accuracy results very competitive or better than state-of-the-art and design a set of experiments that prove that our system works perfectly with a public dataset.

2. Introduction to sentiment analysis

In (Liu, 2010) we have a very good definition of sentiment analysis. He proposes a model where unstructured texts are defined as structured data. The structure proposed is a quintuple (o_j, f_{jk}, oo_{ijkl}, h_i, t_j). So we have an object o_j with a set of features f_{jk}. This structure contains opinion orientations oo_{ijkl} that can be referred to the root object or to specific features and at the same time, those opinions can have a specific orientation (positive, negative, anger, joy, etc.) and strongness. If we want to complete this structure we can also add non-subjective opinions as facts (neutral opinion orientations). So the structure will have a collection of facts and opinions referred to an object and its features. Also we can consider the opinion holder h_i: the person or organization that claims this opinion at a specific time t_j. If we want to draw an example we can consider a review of a mobile phone where the object can be an IPhone and the features will be screen, battery and price. The different opinions could be "The resolution screen is awesome", "The battery life is not what I the expected", "For me is too expensive". And facts could be: "Retina screen", "Battery life: 10 hours", "The price is 600 euros". Furthermore, the set opinions and facts can have references to other objects. An opinion like "IPhone is better than Samsung Galaxy" can be categorized as a comparative opinion (instead of direct opinion).

But going from unstructured to a structured approach is complex. The most basic method to detect sentiment polarities inside texts is using the bag of words method. This method consists in having a dictionary with the polarity of each word and use it to decide. But how do we deal with ambiguity here? Sometimes a the same word can have different meanings depending on the context. This means that even when you have a dictionary entry for a word you will have to choose the exact meaning of this word. And what we can do if the author of the text is being satiric or ironic? Maybe he is using words out of context to refer to a specific word.

Most of the work done on the area of sentiment classification is based on machine learning techniques. (Pang et al., 2002) did a comparison of machine learning techniques applied to topic classification vs. sentiment classification. They used movie reviews for that which is very convenient because each review contains a punctuation from 1 to 5 (using stars). They tried to apply methods useful for topic-classification such as Naive Bayes, Maximum Entropy classification and Support Vector Machines using features like unigrams, bigrams so each document was represented as a word vector. The main conclusion of the study was that the differences between Naive Bayes and other methods is not significant. Also they saw that accuracy was lower than in topic classification (rarely higher than 80%) which implies that to improve accuracy ratios is necessary to choose other kind of features not considered for topic classification. In topic classification the presence of several words indicate that the topic is A or B. But an opinion is harder to classify since sometimes is an aggregation of several opinions with different degrees of strongness and polarities. We have to consider the contrast between them to decide if an opinion is A or B.

So we can consider that the art of sentiment analysis is the art of feature selection. Researchers have been trying out lots of possibilities here: N-grams, POS-Tagging, Syntax features, and others. The most used technique is N-grams which consists in creating a feature with the presence of certain words or concatenations of words. Also, Part of Speech Tagging is very useful since some verb tenses, adjectives and adverbs are indicators of subjectivity. The syntax analysis of a text can be also used to reinforce properties of words (negation, contrast, reinforcement). Playing with context can give you advantage, it is very usual to build domain-dependent models to detect polarity only in a specific domain although some researches have built cross-domain solutions.

2.1 Feature selection for sentiment analysis

This is the most important step when you are designing a sentiment analysis system. If you do not choose carefully your features you might end up with a noisy classifier or with low rates of accuracy. The most effective techniques are N-Grams, POS-tagging features and syntax features. Although in practice is better to combine several features to increase accuracies as much as possible.

2.1.1 N-grams + Negations

N-grams consists in using combinations of words as features. For instance the combination "beautiful" might be a unigram for positive opinions, "I like" would be a bigram for positive opinions, and "I don't like" could be considered a trigram that hides a negative opinion. There are several techniques to optimize N-grams. For instance, you can avoid several N-grams that are very common and they do not provide classification information

You can gain some precision on N-grams attaching negative words inside (not+like, doesn't+work, etc.), also choosing the right dimension is also important: in general bigrams provide a good balance between a coverage (unigrams) and an ability to capture the sentiment expression patterns (trigrams) as have been proved in (Pak and Paroubek, 2010).

2.1.2 Pos-Tagging

Tagging the grammatical features of each word is also a very good strategy to improve accuracy ratios and detect useful patterns for classifications. For example, authors of subjective texts usually describe themselves in first person while verbs in objective texts are usually in the third person. Subjective texts tend to use simple past tense instead of the past participle. As opposite to the positive set, the negative set contains more often verbs in the past tense, because many authors express their negative sentiments about their loss or disappointment. (Pak and Paroubek, 2010) did an extensive study to use twitter as a corpus for sentiment classifiers. They have used the strategy of selecting tweets with different emoticons: Happy emoticons: ":-)", ":)", "=)", ":D" etc. and sad emoticons: ":-(", ":(", "=(", ";(" etc. They tested TreeTager on this data to tag all words in each tweet and try to find grammatical patterns. The classifier presented was based on the multinomial Naïve Bayes classifier that uses N-gram and POS-tags as features.

2.1.3 Syntax

Syntax is another feature that can be useful in some cases. For instance, in the sentence "My flat is warm and cosy" we can say that the second adjective must be positive because first adjective is positive. This is indicated by the conjunction “and”. The identification of this connectors inside a text can be used as valence shifters such as negation, intensifiers, and diminutive (Kennedy and Inkpen, 2006).

2.1.4 Term frequency

Considering the number of times that a word appears in positive or negative contexts can be a factor to realize sentiment categorizations. Term frequency analysis can also be used to detect the domain of a text, so certain frequencies on a text can be used to choose a model or not. An example of a model based on term frequencies is (Bakliwal et al., 2012). The model defines the probability of being positive or negative by following this formula:

$$\begin{aligned} P_f &= Frequency\ in\ Positive\ Training\ Set \\ N_f &= Frequency\ in\ Negative\ Training\ Set \\ P_p &= Positive\ Probability\ of\ the\ token. \\ &= P_f/(P_f + N_f) \\ N_p &= Negative\ Probability\ of\ the\ token. \\ &= N_f/(P_f + N_f) \end{aligned} \tag{1}$$

Figure 2.1. Term frequency model.

This kind of models require an important training set to work correctly, but they tend to be very effective (around 87% of accuracy).

2.1.5 Noun identification and other wildcards

Detecting parts of speech that do not provide any subjectivity is a good path to increase accuracy. Usually nouns are an example of those words without specific sentiment weight. Opinum (Bonev et al., 2012) is an example of this kind of solution. They used named entity recognition algorithms in order to erase certain specific names from the training set. For instance, converting “Bank BBVA is very bad.” in “Bank BANK_NAME is very bad”. With this technique they can isolate domain-independent negative words from domain-dependent negative words and develop cross-domain models.

2.1.6 Punctuation signs + stopwords removal

This is a process that can give you a little boost in classifications. It is very usual to detect sentiments encoded in emoticons or punctuation signs like exclamations marks. Also, stopwords do not provide any real value. Stopwords are words used very often that we use to connect concepts and to be familiar with a language. But this high frequency inherent in stopwords means that is better not use them in classifications. The process of stopword removal usually can increment accuracy by 1 or 2 points as we have seen in (Bakliwal et al., 2012).

In the case of emoticons or specific punctuation signs that are encoding sentiments is good to include this values in a trainer to gain more accuracy. Also, punctuation signs more basic like commas or points that are indicating separation can help to separate several opinions and do not mix them.

2.1.7 Authority and trust

In some web sites certain opinions are more important than others. For instance, in eBay we trust more on those users that have already sold something. In reviews this is very similar, some lead writers are more influential than others and if they write a good opinion about a product this product should have an extra boosting. In (Ando and Ishizaki, 2012) we have an interesting paper about this fact on how we consider more useful some opinions than others. They created a dataset with 500 sentences including exclamation marks (!) and they asked to participants to select those sentences that might be more influential. With this analysis they detected the most important features in a hotel. So those opinions that discuss about “room”, “service”, “meal” and “scenery” are more influential than others.

2.1.8 Twitter-specific features

When the use-case is restricted to twitter things like retweets, hashtags, mentions can be indicators of sentiments. It is very usual that a twitter reply (mention) is done to criticize or refute an opinion so the apparition of ‘@’ will give to the tweet more probability of be subjective. Also the apparition of links gives a different dimension to the text inside a tweet. Sometimes a user can include a link to an image or a link to a text to complement the content on its opinion. Without analyzing this content sometimes is better to discard those messages from the dataset.

2.2 Classification techniques

In this chapter we will review briefly the main classification techniques used on sentiment analysis systems.

2.2.1 Naive Bayes

A Naive Bayes (Das and Chen, 2007), (Pang et al., 2002), (Bonev et al., 2012) classifier assumes that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature, given the class variable. For instance, a fruit may be considered to be an apple if it is red, round, and about 3" in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of the presence or absence of the other features.

So for a document that is already assigned to a class c, we now that:

$$P(c \mid d) = \frac{P(c)P(d \mid c)}{P(d)},$$

Figure 2.2. Naive Bayes formula (1)

This implies that we can rewrite P(c | d) as the probability of each feature to be in the class c because we assume that all features are independent.

$$P_{\mathrm{NB}}(c \mid d) := \frac{P(c)\left(\prod_{i=1}^{m} P(f_i \mid c)^{n_i(d)}\right)}{P(d)}.$$

Figure 2.3. Naive Bayes formula (2)

2.2.2 Pointwise Mutual Information

Pointwise Mutual Information (PMI) (Read, 2004), (Su et al., 2006) is a method for semantic orientation analysis which does not require a large body of training text. The PMI is based in the closeness of a phrase to a set of paradigm words that will define the different sentiments that we want to capture i.e. table R:

Positive	Negative
good, nice, excellent, positive, fortunate, correct, superior	bad, nasty, poor, negative, unfortunate, wrong, inferior

Table 2.4- Examples of postive and negative words

The statistical dependency between the phrase and each pole of semantic orientation is estimated using the World Wide Web (using a search engine, for instance). The search engine is consulted to determine the number of co-occurrences with the paradigm word sets and these hits are used in the following formula:

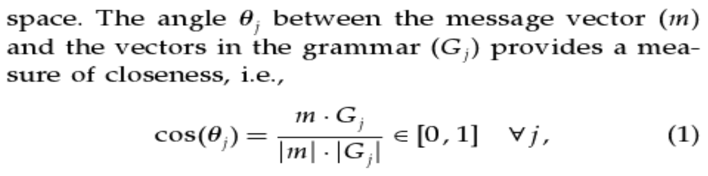
$$\mathrm{SO}(\textit{phrase}) = \log_2\left(\frac{\mathrm{hits}(\textit{phrase}\ \mathrm{NEAR}\ \textit{positive})\mathrm{hits}(\textit{negative})}{\mathrm{hits}(\textit{phrase}\ \mathrm{NEAR}\ \textit{negative})\mathrm{hits}(\textit{positive})}\right)$$

Figure 2.5. PMI formula

Finally, we can have the probability of certain words in a phrase of being positive or negative depending on their co-occurrences in the WWW.

2.2.3 Vector Distance

A vector distance classifier (Das and Chen, 2007) uses a group of pretrained vectors in order to decide the category of a new message. The grammar rules that identify each category are known as Gj.


space. The angle θ_j between the message vector (m) and the vectors in the grammar (G_j) provides a measure of closeness, i.e.,

$$\cos(\theta_j) = \frac{m \cdot G_j}{|m| \cdot |G_j|} \in [0, 1] \quad \forall j, \qquad (1)$$

Figure 2.6. Vector distance formula

2.2.4 Support Vector Machines

The approach in SVM (Bakliwal et al., 2012), (Kim et al., 2006), (Pang et al., 2002) is totally different. In this case we want to find an hyperplane able to separate correctly all the documents assigned to a class c. Furthermore, the algorithm finds an hyperplane with a margin as higher as possible separated from other classes and the class c. We talk about hyperplane and not plane because is generalized to N dimensions. One dimension for each feature that we want to consider.

$$L_P(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i [y_i(w^T x_i + b) - 1]$$

Figure 2.7. Lagrangian Dual formula

To solve the hyperplane we need to compute derivations from Lagrangian Dual to find the alpha for each dimension. Those alpha are known as support vectors that can be used to calculate the hyperplane in the formula:

$$\vec{w} := \sum_j \alpha_j c_j \vec{d}_j, \quad \alpha_j \geq 0,$$

Figure 2.8. Formula to calculate weights in SVM

2.2.5 Maximum Entropy

Maximum Entropy (Pang et al., 2002) does not make any assumption about the relationship between features. So this algorithm can work with features that are conditionally dependent or independent. This is why sometimes Maximum Entropy can perform better than Naive Bayes.

$$P_{\mathrm{ME}}(c \mid d) := \frac{1}{Z(d)} \exp\left(\sum_i \lambda_{i,c} F_{i,c}(d,c)\right),$$

where $Z(d)$ is a normalization function. $F_{i,c}$ is a *feature/class function* for feature f_i and class c, defined as follows:[6]

$$F_{i,c}(d,c') := \begin{cases} 1, & n_i(d) > 0 \text{ and } c' = c \\ 0 & \text{otherwise} \end{cases}.$$

Figure 2.9. Maximum Entropy formula.

Looking at figure 2.9, we can see how P(c | d) is defined as a exponential where each feature is associated to the pair $F_{i,c}(d,c')$ that can be 0 in some cases. Which means that a feature is only considered for a pair (d,c') if and only if exists another document with this feature associated to c'. For instance the feature trigram “does not work” would be only considered for negative documents because is never included in positive documents.

2.2.6 Voting among several classifiers

Another good approach is designing a sentiment analysis system that uses the combination of several classifiers. Sometimes the lack of quality of a classifier in some cases can be compensated by another. (Das and Chen, 2007) designed a classifier to extract sentiment from stock message boards. This classifier uses a combination of several techniques and uses a voting scheme to decide if a word is bullish (optimistic), bearish (pessimistic), and neutral (comprising either spam or messages that are neither bullish nor bearish). Final classification is based on simple majority vote amongst the five classifiers i.e. three of five classifiers should agree on the message type. If a majority is not obtained, the message is not classified. This approach reduces the number of messages classified, but improves accuracy.

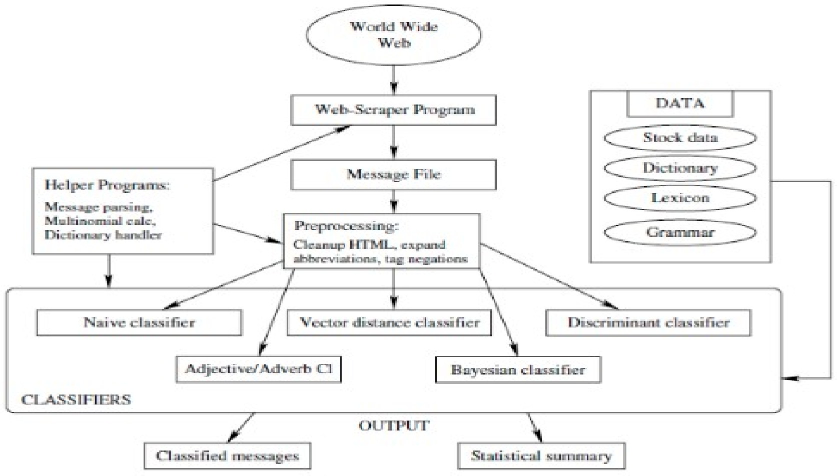


Figure 2.10. Das and Chen solution with several classifiers

2.3 Applications

Sentiment analysis is a discipline that can be used for many applications. The first group is review related websites for instance a review-oriented search engine that collects opinions from different sources. Another big group is recommendation systems, using sentiment analysis in this kind of applications we can outperform older systems by suggesting documents with good opinions instead of bad opinions. Also, business intelligence is one of the best candidates for this technology, there has been lots of interests in companies to track activity and detect low sales. By using this algorithms you can aggregate information on comments and detect patterns to have relevant information about a specific product with low sales. It has been very used in electoral campaigns and politics to measure the pulse of electors. By analyzing tweets you can have an idea on what is the general status of your party or what your voters think about an specific issue. This technology have been already used to predict elections with very accurate results.

3. State of the art

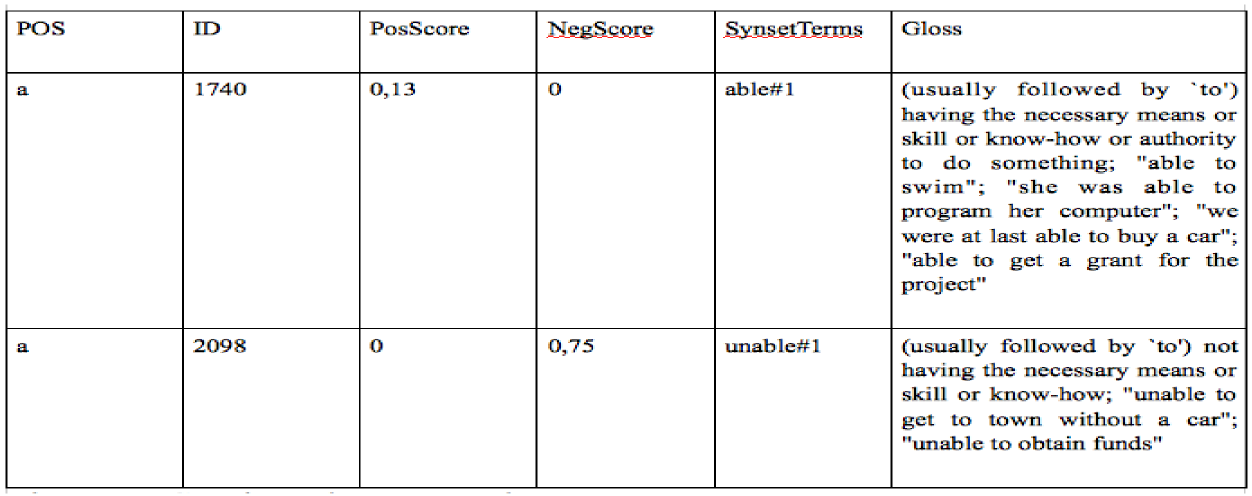
In the following sections 3.X we will review different fields of research considered as the state of the art of sentiment analysis. In Chapter 3.1 we will review dictionary techniques. After this we will talk about subjectivity summarization (chapter 3.2), twitter prediction (chapter 3.3), cross-domain models (chapter 3.4), and edge cases like irony or informal texts (chapter 3.5).

Finally we will discuss about the stengths of each field of research (chapter 3.6) and we will review the research topics that are not developed yet and considered as research opportunities (chapter 3.7).

3.1 Use of sentiment dictionaries

General Inquirer, NTU Sentiment Dictionary, Opinion's Finder Subjectivity Lexicon or SentiWordnet are examples of dictionaries with information about sentiments. This dictionaries can be used as a starting point to build a model to detect positive or negative opinions inside a text. By using dictionaries we can design "bag of words" methods and decide if a opinion is positive or negative depending on the information that we find on dictionaries for each word in a text.

Sentiwordnet (Baccianella et al., 2010) is an evolution from WordNet that provides a negative and a positive score for each synset. The dictionary is created through the use of some synset seeds known as pragmatically positive and pragmatically negative and the use synonyms and antonyms to pass on those scores to other synsets. This propagation is limited by a radius k. After this initial step is finished the random-walk technique is applied. So the glosses (definitions used to disambiguate a synset) inside each synset are used to train a classifier.



POS	ID	PosScore	NegScore	SynsetTerms	Gloss
a	1740	0,13	0	able#1	(usually followed by `to') having the necessary means or skill or know-how or authority to do something; "able to swim"; "she was able to program her computer"; "we were at last able to buy a car"; "able to get a grant for the project"
a	2098	0	0,75	unable#1	(usually followed by `to') not having the necessary means or skill or know-how; "unable to get to town without a car"; "unable to obtain funds"

Figure 3.1. Sentiwordnet examples.

The random-walk step consists of viewing WordNet as a graph, and running an iterative, “random-walk” process in which the positive and negative scores, starting from those determined in the previous step, possibly change at each iteration. The random-walk step terminates when the iterative process has converged.

(Godbole et al., 2007) based its classification in a lexicon obtained from WordNet. For more effectivity they designed different lexicons for each topic. So a lexicon for politics is totally different that a lexicon for health. From an initial lexicon they designed a graph model to expand polarities to other words. For instance, if the word “good” is marked as positive, all synonyms of “good” are marked as positive and all antonyms of “good” are marked as negative. Then a new iteration is performed for next level (with the synonyms of the synonyms and the antonyms of the antonyms) and so on. Depending on the distance the polarity score is different. Applying the formula 1/cd where $c > 1$ and d is the number of nodes away. With this kind of formulation the system ends up with polarities defined for all the words. Once we have a score for each word, we can calculate polarity scores of each text by dividing the sum of all polarity scores in a text between number of total words. The score was tested against names of celebrities: Maria Sharapova got the best score.

(Esuli and Sebastiani, 2007) presented a very interesting approach applying pageRank algorithm to determine term polarities. For that they used extended WordNet to build a graph where each synset has certain polarity depending on the polarity of its members. The main hypothesis is that there won’t be huge variations and each synset will have a similar degree of positivity and a similar degree of negativity. That, will produce a graph of relations between different synsets that will transfer its polarity properties to its neighbors. One interesting point of this experiment is that they computed the pageRank separately for positive synsets and negative synsets (starting all the graph from scratch for each case). Also they have seen that effectiveness is much better with positive terms, so classifying negative terms is a harder task. As a conclusion, they see that this kind of model can be applied to other use cases related with semantic properties of words.

(Thelwall et al., 2010) developed SentiStrength algorithm using comments from MySpace. The main complexity of this comments is the informality of text. Expressions and abbreviations used in SMS are very used on this social network. They classified a set of comments assigning a score from 1 to 5. Each comment could have a strengthness in the positive and negative axis at same time. In order to get consistent scores to each comment they annotated the dataset among different participants to find participants that gave consistent punctuations to all comments. Participants who gave abnormal punctuations were rejected.

The algorithm starts with an initial list of positive and negative words (assigned to a specific negative and positive punctuation from 1 to 5). By using this word scores we can assign a score to each comment. Ideally this comment score has to match with the punctuations given manually. The algorithm does iterations until convergence between calculated scores and manual scores is reached. So if the word “love” has initially 4 points and assigning 3 points the overall accuracy is better, score changes for this word. The algorithm not only includes words, also includes emoticons and punctuation signs such as “!”. This incremental model was compared against n-gram models and outperformed all of them.

3.2 Subjectivity Summarization

Another hot topic on this area is subjectivity summarization. This consists in summarizing only one type of opinions from a text or a set of texts. This approach is more interesting than detecting the polarity of a given text or phrase because users at the end might prefer a program able to give a general opinion about a brand than not a program able to do individual categorizations. Summarizing thousands or millions of texts in seconds is something that humans can't do. In the field of subjectivity summarization we can differentiate two types of summaries. The first one is a document-summary, where we provide a sentiment summary for a single document. This can be obtained by using graph representations of content and trying to find the most representative nodes and edges. The second type is multi-document summary where we provide a summary from aggregating multiple documents. The first step for this kind of summarization is detecting similar or identical opinions across the documents in order to show the most common ones. In this summaries you might also want to include when several users agree in a topic and record the different reasons for that. For instance, two users can agree that a movie is good but one could be because its special effects and another one because its plot. In contrast with traditional summarization where redundancy is erased, in multi-document summarization we use redundant opinions to build a general opinion.

One of the early papers in document subjectivity summarization was done by (Pang and Lee, 2004). They developed a graph algorithm based on minimum cuts to capture subjectivity inside a text. The basic idea behind minimum cuts is create groups behind each sentence inside a text. After having each sentence assigned to all classes with certain probability (for instance C1=positive and C2=negative), the algorithm creates relationships between the different sentences in pairs and calculates its group probability to be in a class.

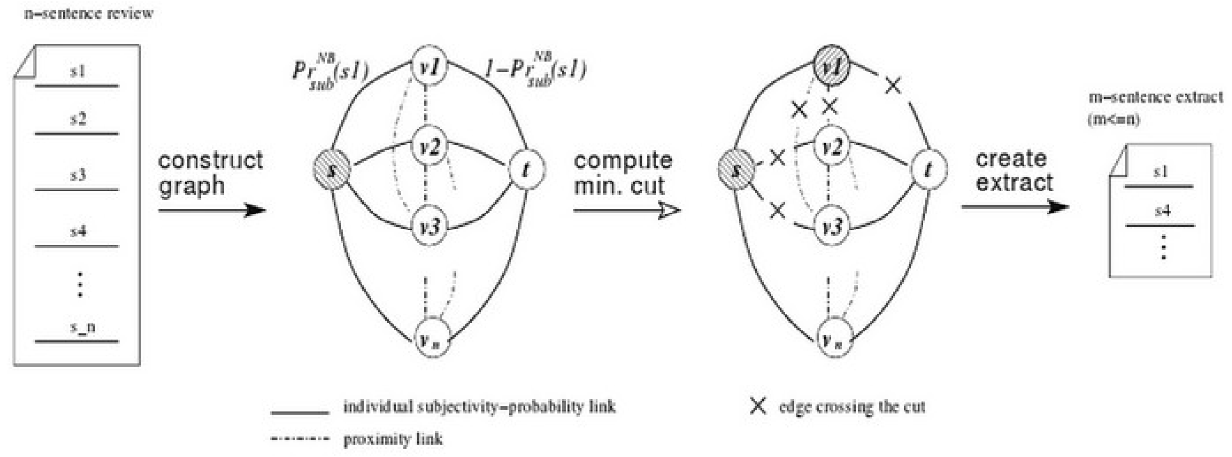


Figure 3.2. Graph-cut-based creation of subjective extracts

At the end the algorithm keeps all nodes that are likely to be in the same class and those sentences with higher probability to be in a class considering them together. To do that, you erase the weakest relationships (less likely to be in a class). With this system you can aggregate all the specific sentiments associated to one class inside a text and keep only the most important sentences.

But there are other techniques to present a sentiment summary to the user without textual representations. Thermometer graphics are represented by topic boxes with a determined size for each topic depending on the number of times that is mentioned. And one color assigned to each topic depending on its degree of positiveness. So the user can have a general idea of the topics discussed and the sentiment without reading text. You can also represent the information in histograms in the 5 stars review systems, in a way that you can analyze the different characteristics taken into account when the review is 1 star, 2 stars, 3 stars, etc... and also see clearly which star is more common. Rose plot is another technique to represent similarities between products. If for instance two products have long battery life they should be considered similar. In this kind of plot you have a circle where each end represents one characteristic (that can be represented in red or green depending of its positiveness) so you can see two different products and read two rose plots to have a quick idea which is better characteristic by characteristic. Time is another variable that can be included in this kind of graphics to analyze the sentiment evolution of a topic over time.

Another factor to consider when doing summarizations is reviewer quality. Some opinions could be more meaningful than others. For instance, in Amazon the user can mark an opinion as "helpful" or "not useful". By considering this flag we can detect opinion leaders and build our summary on top rated opinions. But even with those flags as input, we can still consider opinions that are not really useful or consider too much leader opinions (what in literature is called the richer-get-richer effect). This can be fixed by creating a usefulness classifier to analyze things like writing quality, term popularity or subjectiveness of the review. Other issue in reviewer quality is spam. In some sites like Amazon Mechanical Turk people is paid to write positive opinions. How do we process those opinions to not alter real opinions? Usually those spammers are using the same writing style in all opinions so we can detect patterns and erase them.

3.3 Using Twitter to make predictions of events

The most interesting topic inside state-of-the-art is prediction of events with Twitter. Some models have been designed in order to predict elections (Tumasjan et al., 2010) or market stock (Bollen et al., 2011). The focus can be positiveness and negativity although there are models based on other sentiments like calm, alert, or happiness. Those dimensions can give you models more well suited to predict. Twitter has emerged as the best source of information for this kind of tasks since most of its content is public and it is used by more than 500 million people.

In the area of prediction models we have many studies related with sentiment analysis on Twitter. (Tumasjan et al., 2010) worked with tweets about the last german election in 2010. The methodology taken is based in 104.003 tweets taken the month before the election. They choose tweets mentioning parties and political leaders of those parties. Using Linguistic Inquiry (LIWC2007) they got the degree of each tweet belonging to empirically defined psychological and structural categories. The categories are: future orientation, past orientation, positive emotions, negative emotions, sadness, anxiety, anger, tentativeness, certainty, work, achievement, and money.

This information was used to compute the similarities between political leaders (figure 3.3). As we can see in the results the leader of liberal party Westerwelle is the one that has more polarized results (positive and negative opinions):

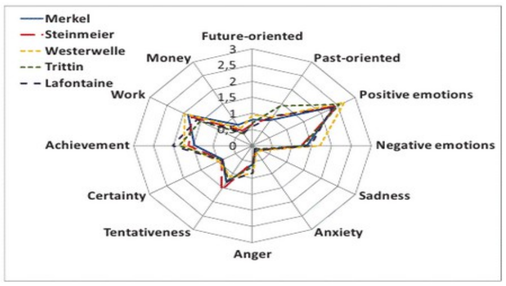


Figure 3.3. Profile of leading candidates

As a prediction of the election result they used the mention count method with a Mean Absolute Error (MAE) of 1.65% with respect the final result. The variation between the share on twitter and the final result can gives us an idea if a party has a positive or negative sentiments on twitter i.e. Grüne was able to get 11.4% with only a share of 8.0%, while SPD loses almost 2.2%. As a conclusion, number of tweets mentioning a political party can be considered a plausible reflection of the vote share. But the study does not go beyond this fact and does not analyze why some parties had a positive error and others a negative error.

(O’Connor et al., 2010) did a very interesting approach trying to correlate polls with tweets in two different topics: economic confidence and US political elections in 2008. For economic confidence they chose the following polls: Consumer Confidence Index from the Consumer Board, the Index of Consumer Sentiment (ICS) and the Gallup Organization’s “Economic Confidence” index. For the election they chose Gallup’s daily tracking poll for the presidential job approval and some data from Pollster.com.

Sentiment scores were calculated with a very basic technique: Xt = count(positive words ^ topic) / count(negative words ^ topic). The distinction between positive words from negative words is done through OpinionFinder dictionary. Some mistakes had to be considered to refine results, for instance the word ‘will’ was counted as weak positive and it messed up some scores. They had to revise manually some parts of the dictionary to have more realistic scores. Although at the end the system had some polarity errors, the aggregated text sentiment obtained was very similar to the real polls except that the information obtained in Twitter was more real-time that the information published in polls. Also, information obtained from Twitter needed some smoothing since in day-to-day sentiment is much more volatile than polls.

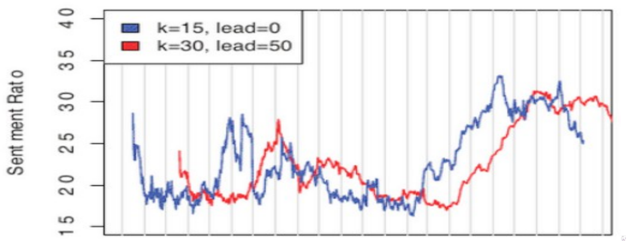


Figure 3.4. Correlation between Twitter and polls

A very interesting framework to predict changes in stock market is (Bollen et al., 2011). For that, they used around 10 millions of tweets for 6 months and they took tweets with phrases like “i feel”,“i am feeling”,“i’m feeling”,“i dont feel”, “I’m”, “Im”,“I am”, and “makes me”. The idea they wanted to test if there was some correlation between the mood of people and the variations in stock market. Once they had this data, they applied two processes: 1/ OpinionFinder to calculate the degree of positiveness and negativeness. 2/ GPOMS which measures variations in 6 different sentiments: Calm, Alert, Sure, Vital, Kind and Happy. To find those correlations they used Granger causality analysis. That makes the assumption that if a variable X causes Y then changes in X will systematically occur before changes in Y. The accuracy of the model was around 87%.

The first test realized was with the USA election of 2008:

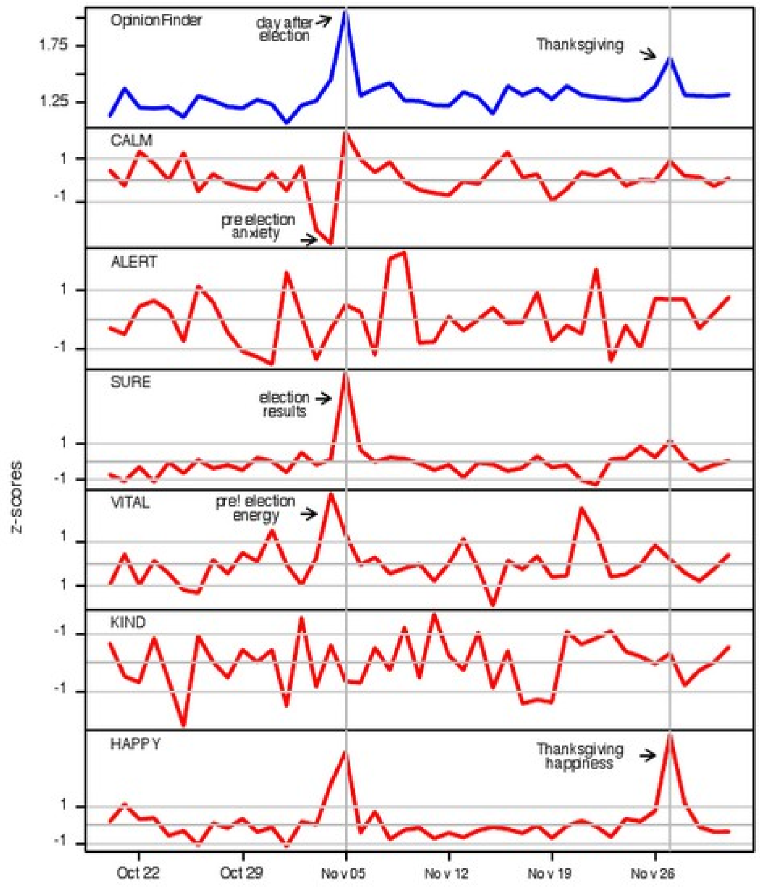


Figure 3.5. Comparison between GPOMS sentiments and OpinionFinder

As we can see in figure 3.5 each sentiment has some correlations with the positive/negative score obtained through OpinionFinder. We can see how calm has a deep fall in the day before the election. Or some happiness peaks for Thanksgiving day.

The second test was done comparing the data obtained through OpinionFinder and GPOMS to try to find correlations between market stock and the different emotions. They discovered that calm is the best indicator:

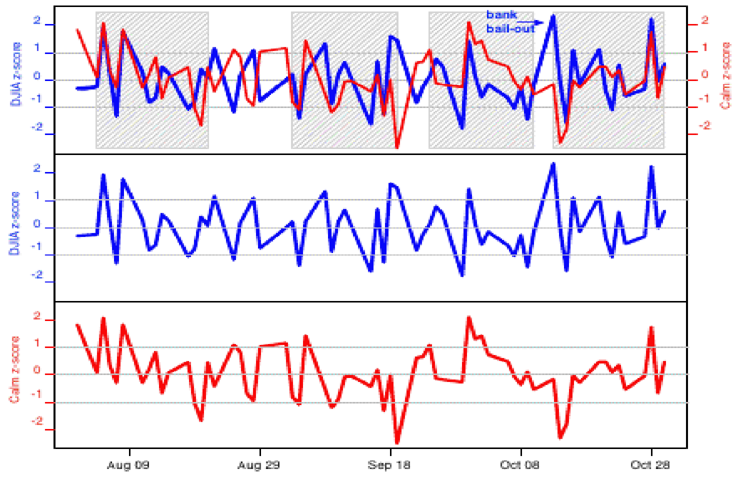


Figure 3.6. Calm scores vs. stock market

As we can see in figure 3.6 most of the peaks and the falls in the stock market are correlated with calm score except a couple of peaks that correspond to announcements of federal reservation which means that unexpected news has an effect in the economy that cannot be predicted on this model.

3.4 Cross-domain models

(Pan et al., 2010) developed a model in the area of cross-domain sentiment classification. For that, they used a bipartite graph, so domain-independent words are assigned to left part of a graph and domain-dependent words are assigned to right part of a graph as we can see in figure 3.7. This dependence or independence can be found by measuring the number of times that a word appears in different domains. If it appears in many domains it is independent, if only appears associated with texts of one domain is dependent.

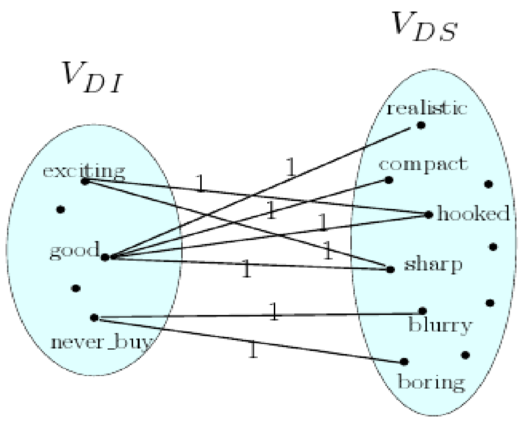


Figure 3.7. Bipartite Graph

As we can see in figure 3.7 this kind of graph already creates some clusters such as (blurry, boring) or (sharp, hooked). Each of this clusters represents a domain. Furthermore, by using the co-occurrence matrix we can guess the polarity of certain words so if “exciting” is used with “hooked” we can derive that “hooked” must be positive in the domain of movies, while using “never buy” with “blurry” should be a negative indicator in the domain of appliances.

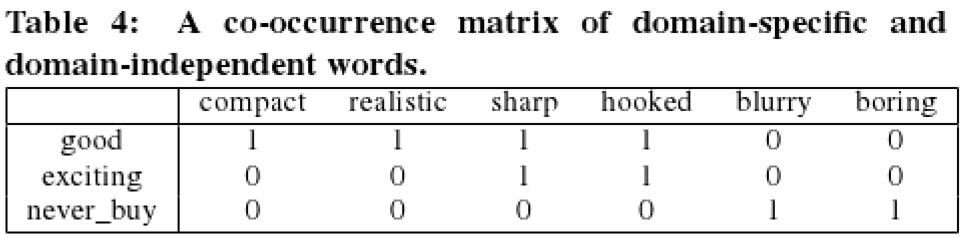


Table 4: A co-occurrence matrix of domain-specific and domain-independent words.

	compact	realistic	sharp	hooked	blurry	boring
good	1	1	1	1	0	0
exciting	0	0	1	1	0	0
never_buy	0	0	0	0	1	1

Table 3.8. Bipartite-graph co-occurrence matrix

By applying spectral clustering techniques (i.e. comparing eigenvectors) we can calculate the similarity between a new opinion and a domain opinion and estimate better its polarity.

In the last WASSA 2012, we saw Opinum (Bonev et al., 2012). An approach based in statistics. In this work, they captured n-grams to detect common text structures that can be counted as positive or negative. The theory behind this paper is that anyone that knows a language can detect if a opinion is positive or negative independently of the subject domain. This means that is possible to develop a domain-independent polarity detector by not introducing domain words in the training data. For that, they replaced entities such as names of banks, organizations and people by wildcards to focus on domain-independent words. They saw also, that unigrams and bigrams were not enough for sentiment analysis so they wanted to capture phrases like “an offer you can’t refuse” or “the best way to lose your money” (5-grams). The model is purely probabilistic so for a given text T with N sentences they calculate the probability of each sentence s to pertain to positive Mp, and the probability of each sentence s to pertain to the negative Mn. At the end, they subtract sums in T(P(s|Mp)) - sums in T(P(s|Mn)) and depending if this number is close to 1, 0 or -1 the system decides if a text T is positive, neutral or negative.

3.5 Edge cases and ambiguity

When discussing about sentiment analysis is very usual when someone asks what we can do with opinions that look positive but they are not. Sometimes is not enough by looking for presence of words because sometimes a negative opinion can be expressed without negative words and viceversa. For instance: “If you are reading this because it is your darling fragrance, please wear it at home exclusively, and tape the windows shut.” For understanding that this is a negative opinion you have to relate the concept of windows shut with fragance, but without this context it would be impossible to mark this opinion as negative. This context-sensivity and domain-dependence is one of the main challenges in sentiment analysis. You have to be also careful with dependency and order of sentences for instance an opinion like: “I loved this movie when I was little, now I hate it.” where the second sentence somehow erases the positivity of the first.

Irony is a specific case difficult to classify. (Reyes and Rosso, 2011) did a very interesting study on how to detect irony reviews on amazon shop comments section. All the idea about this study became from a viral campaign about a T-shirt called “The Mountain Three Wolf Moon” http://amzn.to/H4sbGc that basically was getting lots of 5 stars reviews with very ironic, sarcastic or even satiric descriptions. Some examples might be: “Unfortunately I already had this exact picture tattooed on my chest, but this shirt is very useful in colder weather.” or “Pros: Fits my girthy frame, has wolves on it, attracts women. Cons: Only 3 wolves (could probably use a few more on the 'guns'), cannot see wolves when sitting with arms crossed, wolves would have been better if they glowed in the dark.” For those cases they designed a classifier for irony detection. They based its classification in N-grams, POS-n-grams, funny profiling (with a dictionary taken from WordNet to detect funny words), positive/negative profiling, affective profiling (since affective words are very common in ironic and satiric texts), and pleasantness profiling (that calculates the probability of having a context favorable and unfavorable for irony).

Figure 3.9. Three wolf T-shirt

3.6 Theoretical aspects

3.6.1 The Art of feature selection

The use of dictionaries can give advantages to our sentiment analysis systems. It has no sense starting from 0 when you can access to very good dictionaries to fetch information from. Even starting from WordNet seems pointless since we can use Sentiwordnet. Furthermore, Sentiwordnet provides the different meanings for each word and a little gloss that can help you with disambiguations. The idea of graph sentiment propagation between synsets is very powerful. This provides methods to modify the scores assigned to each word to adapt better to each domain. The basic idea behind those algorithms is modifying word scores until convergence is reached. In our case this convergence can be considered reached when accuracy results do not improve anymore. Also is very important the apparition of dictionaries like Sentistrength. The degree of strongness is a very important factor in any opinion. How do you say things marks also the weight of those opinions, so if you detect than in a text there are 5 or 6 opinions about object features how do you choose the most important one? Probably sentiment strength is a good way to do this election. Sentiwordnet also classifies with a certain degree of positivity and negativity so choosing values near to 1 is a good policy. But sometimes, this enthusiasm or strongness is encoded in context and not in specific words. And this is a lack that dictionaries cannot capture.

Sentiment classification is not an ordinary text classification problem. Here we need to work with more complex features to get acceptable accuracy ratios. A system with accuracy lower or equal to 80% means that the system is incorrect 1 of each 5 times. But anyway, still you can get very acceptable accuracy ratios only with N-grams. We have to remember that most of the subjectivity is hidden in words combinations such as "I don't like", "I like", "beautiful", "sad", etc… Also, converting words to its grammatical representation is helpful. It is demonstrated that the presence of adjectives or certain past tenses are indicators of subjectivity. The data inside dictionaries like Sentiwordnet can also be converted into features. A dictionary can give you the negative/positive orientation of a word which means that you can distinguish between objectivity and subjectivity. In state-of-the-art solutions is usual the utilization of wildcards to not include in classifiers domain specific words such as names. We do not want to end up with a model where the name of a bank is inherently related with negativity because we are interested in capturing the words that are surrounding this bank. This is an important step through cross-domain sentiment classification models and the ability to detect sentiments in any text.

A social network or a website can provide other language independent features for sentiment classification. For instance, in a traveling site a positive opinion given by someone who travels a lot should count more than an opinion of someone who only travelled once. And probably there are other factors, that can be used for this boosting. A comment done by someone that has a strong affinity with the author will be more precise that an opinion expressed by someone with no affinity at all. When there is no affinity the prejudices appear and you can get some noisy associations that had nothing to do with reality.

3.6.2 Training and classifying data

Once we have decided the set features that we want to use we have to decide how to train our data and choose a classifier. There are several techniques to train a sentiment classifier. In text categorization problems it is important to choose between supervised or unsupervised approaches. Also, if you have a small set of labeled data you can create a bigger input dataset by doing some tricks. For instance, if a phrase has a greater tendency to co-occur within certain context windows with the word “poor” or with the word “excellent” this phrase could be added to a positive dataset. Bootstrapping is another unsupervised approach that is useful. The idea is to use the output of an available initial classifier to create labeled data, to which a supervised learning algorithm may be applied. In order to create a training set from Twitter it is very common the strategy of selecting tweets with different emoticons: Happy emoticons: “:-)”, “:)”, “=)”, “:D” versus sad emoticons: “:-(”, “:(”, “=(”, “;(” etc. From this starting point you have a very good separation about tweets potentially negative and potentially positive so you can tag manually from there to erase edge cases and create an initial dataset.

Once we have training data we can use some of the machine learning algorithms available to build a model able to perform successful classifications. Naive Bayes usually works good enough although other approaches more advanced such as Support Vector Machines have better accuracy results. We have to consider how much time the classifier will spend in building the model and realizing the individual categorizations. Sometimes it could be good to sacrifice 1 or 2 points of accuracy in exchange of some performance. If we do not want to rely on machine learning, other classification methods are possible. Pointwise Mutual Information or term frequency models can guess if a phrase or a text is positive or negative by using big quantities of data (such as the World Wide Web) to measure the co-occurrences ratios.

3.6.3 The power of subjectivity summarizations

As we discussed, subjectivity summarization is a task more valuable than direct classification. The ability to present a summary with the main opinions about a brand or a product is what user demands. User will not be happy just reading: “BRAND_A has 80% of positive opinions”. An user wants to know more, he wants to know what are the main reasons behind this 80%. Is it the price? Is it the quality? How many opinions were used to compute this percentage?

From the perspective of the brand all is different. The brand wants to analyze thousands of comments on different social networks to do some marketing buzz, this means detecting the main concerns of users to react fast. Where are my weaknesses? Why this new product is not working? Is my new product too expensive?

In this sense, subjectivity summarization depends on the use case. A use case focused on the brand is different from a use case focused in the user. We have several techniques to solve this problems. To start is very important to detect the sentiment strength inside a text to be able to detect the most strong opinion in a text. If we want to present a summary about an Iphone and we have to select among several opinions, it will be important to have a score assigned to each opinion to choose the opinion with highest score. When aggregating several opinions we should choose those opinions that are more repeated and have higher strongness. And also consider other facts such as authority of the opinion holder or freshness.

As we have seen in Minimum Cuts algorithm (Pang and Lee, 2004) the idea of joined probability of being part of a class is a concept very useful. If we want to capture the negative opinions inside a text (and not the positive) we can find the combination of sentences that have more probability to pertain to a class. But how do we transform those selected phrases considered negative or positive in features? This is a very important step in summarization that is not covered in literature. Because a phrase such as “This iphone is expensive” is referring to the feature price and there is not a direct relation unless a human marks this relation specifically.

3.6.4 Is it possible to predict events with sentiment analysis?

We have analyzed in chapter how some researches have been working in prediction through Twitter. This is a very interesting field of research for obvious reasons. The ability to predict the future is an opportunity to anticipate change and be more competitive. Stock markets or election processes are complex and sometimes a little interaction done by a big media can be influential enough to change a sign. As we have seen, to make predictions is important to focus sentiments a little bit more. It is not enough to say that an opinion is positive or negative, we have to connect each opinion with a concept such as calm, happiness or anger.

Trying to respond to the original question: Is it possible to predict events with Twitter? I would say no. Is possible to approximate a prediction, but there are external influences that are out of control. Opinions are a reflect of facts and is impossible to capture certain facts by following rumours and opinions. For instance, a fact like a earthquake will not be reflected in rumours and is very influential for sentiment. Another fact like a monetary intervention can be reflected in some rumours by specialists in the field. But the desire of that happening does not make that real (at least in the short term). So this is an example of a fact with a positive impact in economy that cannot be exactly predicted. What it is possible is measure the overall sentiment of people about a piece of news, a concept and analyze how this is influential in real indicators.

3.7 Topics not covered in state-of-the-art

In this section we include a list of topics that are not considered in most of the papers in the research area. Those topics could be considered as research opportunities for us or for future researchers since nobody has came out with a good solution for that. In the discussion section we will take some of this ideas to develop our proposal.

3.7.1 Knowledge dictionaries

Some projects like Sentiwordnet provide a negative and a positive score for each word in a dictionary. It would be interesting to calculate something like the degree of affinity between concepts. For instance, which is the degree of proximity between the word "happiness" and the word "calm"? By having this ability of keeping crossed scores we could find correlations between real indicators and those scores to find those words or concepts that define better those real indicators.

3.7.2 Structured N-grams

Semantic technologies are very interesting to perform classifications. In this sense, solving this problems with N-grams seems a very reductionistic approach. Usually consecutive words are considered as the base of the language. "I like", "I don't like" are indicators of positive or negative opinion, but sometimes there are hidden conceptual relations that are stronger than that. For instance a predicate-subject relation could be an indicator encoded in two words NAME-VERB. Even we could work with more complex graphs such as NAME-VERB-OBJECT. Those conceptual relations do not need to be consecutive, sometimes we can exclude some details that are not important for opinion like a nested phrase "This house, located in 17h street, has 120 square feet.". It is also possible to introduce directed graphs to make emphasis in the direction of the meaning. "Star wars is boring" could be converted in a graph where the root is "Star Wars" and we have one edge directed to the root "boring".

Those structured N-grams could be created with probabilistic models. Analyzing the content of a phrase we can try to create correlations between the different members in a phrase to guess which combinations are more meaningful. Do we have to choose a verb and a noun? Do we have to choose a noun and an adjective? Is it better to choose only an adjective? By working with a training dataset we could find heuristics to perfectionate accuracy results. The weight of dictionaries and semantics in this kind of heuristics can be crucial. So maybe a combination verb-noun between two specific topics can be more important than other.

3.7.3 Sentiment as a problem, desementilaization as a solution

Humans are passionate about what they do. Is impossible to not reflect some sentiment or attitude when you write about something. If you are writing an article about the israeli-palestinian problem you probably will pick one of both sides and reflect that in your article. But sometimes, we need to know the truth, we need to know the reasons and not the statements given by someone. In this information age it would as important as analyzing subjectivity having technologies able to take a text and do distinctions about what is subjectivity and what is objectivity. This technologies could be helpful to calculate the degree of impartiality of a big media towards one issue. We could know what is the position of “New York Times” towards “Gun control”. We could know if this issue is addressed with impartiality or not and if it is not addressed with impartiality which is both sides has been taken. This analysis is especially important with big medias supposed to be impartial such as public televisions.

3.7.4 Buzz detection

This is a topic that is not very covered in literature. In sentiment subjectivity we have seen techniques to detect the key phrases in a text towards one sentiment. But we haven’t seen techniques to convert this “key phrases” in “key topics”. The ability to convert phrases in topics is a key space in sentiment analysis. In this sense the techniques around buzz detection could be very helpful once we have selected those key phrases.

3.7.5 External factors on comments

When analyzing comments on a topic it is very important to analyze other factors that are not linguistic. For instance, the degree of affinity between the author and the person who did the comment. Usually the replies are done to open a discussion or because you do not agree with the author about some statement. Other kind of replies, could be more constructive between both members have certain degree of affinity which means that the discussion will be more about the details of a statement than not an amendment to all. In twitter this analysis is very obvious when people are discussing about ideological issues.

In this group of external factors on twitter we have other things such as external links or images. Sometimes a reply to a tweet could be an image. How do we take this information inside this image and we incorporate that in the language? Is it enough with the title of the image? Or do we have to analyze the pixels inside this image? This analysis is equally important when someone includes a link in a reply. Which is this link talking about? Is it objective or subjective? Which is the postulate that it defends and how it relates with the comment done. Twitter has reached a point where is not enough analyzing language. We have to analyze images, external links and video to provide a proper sentiment analysis.

3.7.6 Big data can help?

We have seen some algorithms that are using the biggest dataset you can use: The World Wide Web. This was the case of Pointwise Mutual information. But those algorithms are limited to the number of times that you can query Google without being banned. To work correctly you should have a copy of the WWW in your datacenter and be able to query without restrictions. Since this is difficult to achieve except for Google or Yahoo, another possibility is develop a crawler that saves the number of hits of each word and its correlations in a big sequential file. This big sequential file could be queried from outside with technologies such as Hadoop that instead of doing random accesses to disk, treats this information sequentially.

4. Proposal

After reviewing all the material we decided to test an approach with dictionaries. This seemed the approach more realistic to do. Better than using machine learning over N-grams, syntax or POS-tagging since the information in tweets is too short and usually does not follow strong grammatical rules. With dictionaries we can create a closed system with a finite number of N-grams. Furthermore, a dictionary seemed a necessary starting point for other advanced tasks such as summarization or data analysis.

We tested how well a predefined dictionary was able to classify documents but after first tests we saw that Sentiwordnet accuracy results weren't very good. For that, we designed a semi-supervised approach with a little interface where we can annotate tweets. This annotations can be reused to improve the dictionary in next iterations by adding new words to it. Once we had this supervised dictionary able to assign scores correctly to a given tweet, we developed the second part of the project: detect key aspects on a tweet. As important as knowing that an opinion is positive or negative is knowing the reason why. A model based only in sentiment words can be very poor because a negative word can be dependent on context. To avoid that, we captured the relationships between sentiment words and aspect words in each tweet. So if a user tweets "The staff of this airline is awful" we create a relationship between "staff" and "awful". So in the annotation we record that the user says that something is "awful" because its "staff".

All this relationships are included in our annotation system. After annotating a big amount of tweets capturing all those sentiment words and aspect words we can experiment with machine learning and try to use this data to make predictions on new data. So the main challenge on this part of the work is how do we represent this text information to create models as accurate as possible.

To sum up this ideas, we defined this 3 hypothesis:

HYPOTHESIS (H1): We can create groups of N-grams that influence specifically to one aspect in a negative or a positive orientation. This is what we call **sentigrams**.

The idea of sentigrams is one step ahead from the reductionistic problem of positive or negative. In sentigrams, we can see the reasons why a user thinks that something is good or bad and also the key sentiment words that make this difference.

HYPOTHESIS (H2): By using **incremental learning** the system improves in each iteration. Increasing precision.

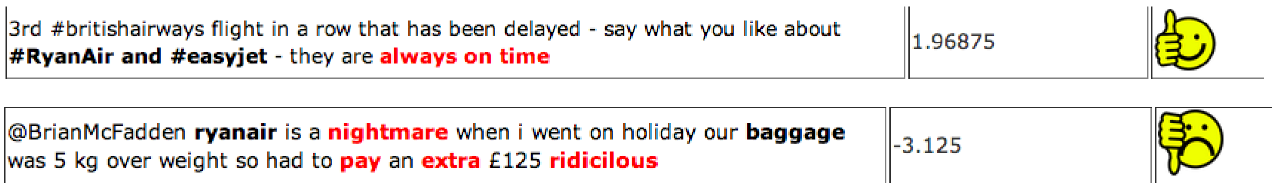
This means that the system is designed as a semi-supervised intelligent system. That improves in each iteration as we get more training data. The use of a dictionary that uses the information on annotations to process new words makes the system able to distinguish new cases and calculate better punctuations.

HYPOTHESIS (H3): After certain number of iterations is reached we can assign a sentigrams to a tweet **automatically** w/o manual intervention.

As we get more and more interactive annotations for one specific domain we are able to perform more complex automatic classifications and not only say that an opinion is "good or bad". If we guess sentigrams correctly, we can say "why an opinion is good or bad" and perform a more detailed analysis on groups of tweets.

4.1 Sentigrams

The annotations used in this project include if the tweet is positive or negative (independently of the score given by the dictionary), the sentiment words that make this tweet positive or negative (marked in red, figure 4.1), and aspects of this opinions (marked in black, figure 4.1). The sentiment words are those words that contain sentiment and that have a positive or a negative score. The aspects are "the reason why" a user is complaining or does agree about something. This combination of information is what we call sentigram. A sentigram is the agreagation of several words and it represents a structured N-gram able to capture a set of words (sentiment words and aspects) with enough weight to determine if a tweet is positive or negative



3rd #britishairways flight in a row that has been delayed - say what you like about **#RyanAir and #easyjet** - they are **always on time**	1.96875	👍
@BrianMcFadden **ryanair** is a **nightmare** when i went on holiday our **baggage** was 5 kg over weight so had to **pay** an **extra** £125 **ridicilous**	-3.125	👎

Figure 4.1. Manually annotated tweets

In the examples seen in figure 4.1 we can see how "always on time" is an N-gram detected a sentiment word that basically refers to "Ryanair and Easyjet". So in this case the user is not reviewing about a specific aspect about the airline and is giving a general opinion.

In the second example in figure 4.1 we see two negative sentigrams ("ryanair" => "nightmare") and ("baggage" => "pay", "extra", "ridiculous"). Very interesting to see how in this context paying for baggage is always bad when maybe in other context paying can be good. The idea on this example is that we have two sentigrams that express different opinions first in general about the airline "ryanair" and secondly about the aspect "baggage". The sum of those two opinions make this message clearly negative.

4.2 Incremental learning

When we introduce the idea of semi-supervised learning the user can adapt the system to a specific domain and specialize the system to a use-case. For instance, an airline could use this system to detect complaints from clients and react fast to negative opinions. It could also detect when good opinions are done and analyze those users that are happy with the airline and increase resources on that area that is working well. When we design a non-supervised system or a cross-domain solution we can lose this attention to detail that a domain-specific model can reach.

In our use-case what we do is using all the words marked as sentiment words (marked in red) to add new entries to the dictionary. This dictionary uses random-walk algorithm to adapt the scores of each word to maximize the number of matches. This means that depending on the domain the scores of the words could end differently. The idea of random-walk consists in doing a change in one random word of the dictionary. If this change increases the number of matches, we take it. If not we rollback and we try a different change. After several iterations we reach a point where any change can't increase our number of matches. This is the point where we reach convergence and the dictionary is ready to work.

4.3 Automatization

The process of supervision is interesting to create better dictionaries but we have other technologies that can take advantage of this data such as machine learning. With enough training data we can reach a point where the system can detect sentigrams on a specific domain automatically. When our application reaches an adequate number of iterations. We are able to create a model with training data able to guess sentigrams. For that task, we use Weka.

One of the main problems is how we introduce this annotation data in Weka because a tweet. This is a problem quite complex from a technological point of view. Because we potentially have an infinite number of states. A tweet is composed by 140 characters that usually contain from 1 to 32 words. From those 1 to 32 words we have to select which of them are aspects, which of them are sentiment words and which of them are not relevant. How many possible combinations we have? A simple problem with 3 words has already 3^3 possible combinations. Imagine this same problem with 32 words.

To solve this problem we use a divide-and-conquer strategy. This means that for each tweet we create N partial observations. One observation for each position in the tweet. This idea is inspired by Viterbi algorithm that uses information in neighboring states to predict next states. So if a tweet contains 4 words, we need to create 4 observations (first in position 0, second in position 1, third in position 2 and fourth in position 3). Each observation has information about neighboring words and the output that we obtain according with this information.

(i.e. "**easyjet** is a **joke**")

```
0, 801829636, -545403680, 1561023766, 2119008529, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 1
1, 801829636, -545403680, 1561023766, 2119008529, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0
2, 801829636, -545403680, 1561023766, 2119008529, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0
3, 801829636, -545403680, 1561023766, 2119008529, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 2
4, 801829636, -545403680, 1561023766, 2119008529, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0
5, 801829636, -545403680, 1561023766, 2119008529, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0
6, 801829636, -545403680, 1561023766, 2119008529, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0
7, 801829636, -545403680, 1561023766, 2119008529, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0
8, 801829636, -545403680, 1561023766, 2119008529, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0
9, 801829636, -545403680, 1561023766, 2119008529, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0

...
32, 801829636, -545403680, 1561023766, 2119008529, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0
```

Figure 4.2. Example of Weka @DATA input for a specific tweet

To represent words we use a unique integer. That can be calculated using hashCode() of a string. Is not an exact unique code because we do not cover all the possible combinations of strings, but it will work in most cases.

The possible class outcomes are {0,1,2} so we are not working strictly in a binary problem. But at least we have reduced the dimensionality of the problem to 3. {0} represents that the word is not relevant, {1} represents an aspect and {2} represents a sentiment word.

4.4 Implementation

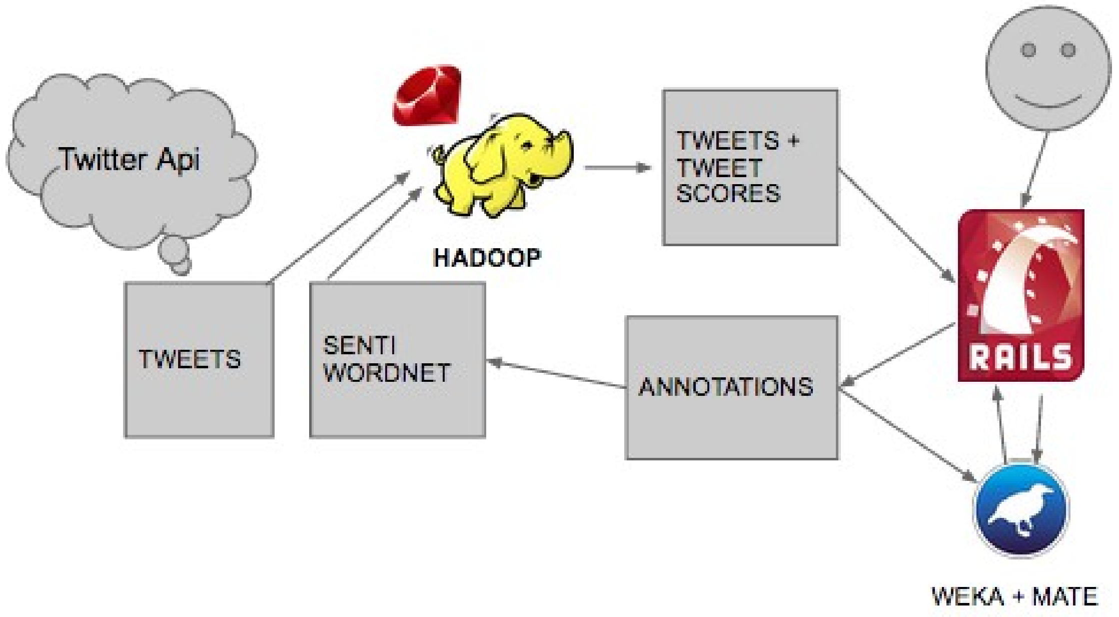


Figure 4.3. Design of our proposal

In the design shown in figure 4.3 we see the technical structure of this project. Basically we have a list of tweets obtained from the Twitter Api and a dictionary (initially taken from Sentiwordnet). From this point we want to know which tweets have words from this dictionary and which don't and calculate a tweet score that will be the sum of the scores of each of those words. After this aggregation we have one tweet score for each tweet and we can supervise the results in a user interface. In this interface we will decide which tweets were correctly classified and which don't aggregating new words to our dictionary if needed. This is what we call incremental learning. New information is integrated in the system to get more precise tweet scores. Now we will review briefly the different parts of the project.

4.4.1 Twitter Api

This part of the code realizes queries to the twitter search api for a list of topics specified by the user. Usually the user can compare 2 or more topics in order to see differences between them (for instance comparing two political adversaries or two competitor companies). The code performs searches by text to find all tweets in the last week mentioning a determined word. As it searches, the code creates a file with the list of tweets in JSON format (separated by end of lines). This line-by-line format can be parsed easily by Hadoop.

4.4.2 Sentiwordnet/Dictionary

Originally we take information from Sentiwordnet dictionary. This is a dictionary available at http://sentiwordnet.isti.cnr.it/ and is structured in the following way:

a	00009346	0	0.625	abstinent#1 abstentious#1	self-restraining; not indulging an appetite

```
especially for food or drink; "not totally abstinent but abstemious"
a	00009618	0.5	0.25	spartan#4 austere#3 ascetical#2 ascetic#2	practicing great self-denial; "Be systematically ascetic...do...something for no other reason than that you would rather not do it"- William James; "a desert nomad's austere life"; "a spartan diet"; "a spartan existence"
a	00009978	0	0	gluttonous#1	given to excess in consumption of especially food or drink; "over-fed women and their gluttonous husbands"; "a gluttonous debauch"; "a gluttonous appetite for food and praise and pleasure"
a	00010385	0	0	crapulous#2	given to gross intemperance in eating or drinking; "a crapulous old reprobate"
a	00010537	0	0.5	crapulous#1 crapulent#1	suffering from excessive eating or drinking; "crapulent sleep"; "a crapulous stomach"
```

Figure 4.4. Example of Sentiwordnet

The dictionary is also using a line-by-line text format where each line represents a meaning. This meaning can be represented by several words (i.e. crapulous#1 crapulent#1). The hashtag at the end is used for hyperonymic cases. Words written as the same but with different meanings depending of the context.

The Sentiword line also includes a positive score and a negative score that is basically what we have to use in our system to calculate if a tweet is positive or negative. Checking if a tweet has words on this dictionary and summing all those scores. Finally it includes a little gloss to understand the differences and a numeric id.

4.4.3 Hadoop Processor

Before starting to write code we had to decide which tool to use. We needed a something able to process big quantities of data without wasting too much memory. Saving a list with millions of tweets could not be scalable at certain point because would not fit in memory and this would force our system to perform random reads on disk. Hadoop seemed the best option to avoid that.

Hadoop uses MapReduce which is a programming model used by Google to support parallel computation in commodity hardware. The name of MapReduce is used because the programming model is divided in two main macros called Map and Reduce. In the Map phase we process a big file from several threads at same time. Each thread has a different starting point in order to cover all the file with all threads (some parts are processed twice or three times to have redundancy between machines). Each one of this threads will read the information provided in the subparts of this file to emit information to the Reduce step. The Map step decides how the information is grouped in the reduce step. And the reduce step receives this parts of information and performs group operations to create new data.

A typical example to understand MapReduce is the count-word example. So each one of this threads will read a part of a text. The Map step would emit pairs such as (word, 1) and the reduce step would collect all this (word,1) pairs and count them to emit (word, n) groups.

In this use case we have two sources of information. In one side, we have tweets and in the other side we have Sentiwordnet data. So we need to design a MapReduce process able to process two types of inputs: tweets and sentiword entries. The process of mapping will be emitting each N-gram inside a tweet and each word in the dictionary. In the reduce step we receive aggregations of N-grams with information associated to its tweets and its sentiwords entries. In this reduce step, we will be interested in keeping N-grams that are appearing in Sentiwordnet to create a file with all the N-grams with relevant information.

After we have this file, we perform a second MapReduce process which consists in reconstructing the information about tweets through the sentiword information. So we emit pairs such as (tweet, sentiword) and in the reduce step we collect groups such as (tweet [sentiword_1, sentiword_2, …, sentiword_N]). After this process we can have a list of all tweets with the information about all the sentiwords that were defined in the dictionary. For this sentiwords we will have also associated information like positive score and negative score.

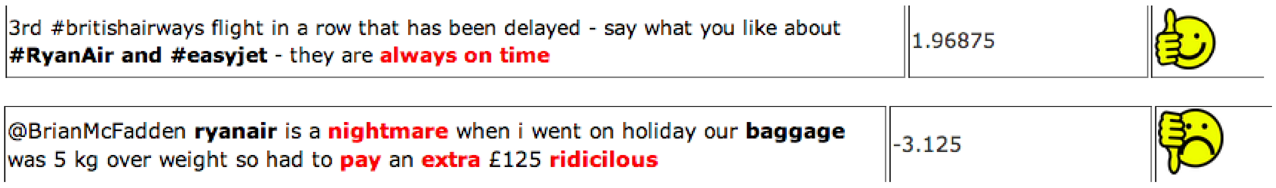
With all this data grouped in one point we can calculate now the tweet score associated to each tweet and we can determine if a tweet is positive or negative.

4.4.4 Rails application (Web/Client side)

Rails is the part of the code responsible of the operations with the user. Here we have a set of interfaces where we annotate tweets manually to improve the system (iteration by iteration). From here we can also design interfaces with the data of the different tweets and calculate statistics.

4.4.5 Annotations

All the annotations done in the interface are used in next of iterations of the system to improve the dictionary or to create better Weka Models.



3rd #britishairways flight in a row that has been delayed - say what you like about **#RyanAir and #easyjet** - they are **always on time**	1.96875	
@BrianMcFadden **ryanair** is a **nightmare** when i went on holiday our **baggage** was 5 kg over weight so had to **pay** an **extra** £125 **ridicilous**	-3.125	

Figure 4.5. Manually annotated tweets

The annotation consists in (+/-) indication (given by the emoticons). This is independent of the score given by the dictionary. We can have a tweet that is wrong in our dictionary and we can correct it from this interface. The second part of the annotation are the sentiment words (marked in red) that indicate those words that should be included in the dictionary on next iterations. And the third part include the aspects (marked in black) so those words which are the object of the user opinion.

In our code, the annotation can be converted to a series of {0,1,2} where 0 indicates that the word has no information, 1 indicates an aspect, and 2 indicates a sentiment word. We should give an output for each word position in a tweet.

4.4.6 Weka Module

The Weka Module is designed to perform sentigram predictions on new tweets. This module is trained with the information obtained from annotations. By adding sentigrams to new tweets we are be able to create interfaces analyzing one specific topic on Twitter and aggregate groups of sentigrams to show a summary of this topic.

5. Experiments

5.1 Dataset

For our experiments we took data through the Twitter Api from different airlines. We basically took data from the top 10 airlines in Europe http://en.wikipedia.org/wiki/List_of_largest_airlines_in_Europe. Since this api does not allow to query for messages older than 1 week, we have taken data from the first week of June 2013 aprox. 12.195 tweets. We used the filter of language to select only messages in english, but some of them are in other languages because this filter is not detecting all cases. The dataset is in JSON separated by end of lines. So any process can read from here line by line, parse the JSON and then it will get information about a specific tweet.

One characteristic of this domain is that most of messages are negative. So people is not usually happy with airlines because baggage policies, lack of space in seats or long reclamation processes. This means that detecting positivity in this domain is a task totally different that it could be in any other domain. Only a few selected words indicate positivity and they are correlated with very specific aspects.

In order to test results we have annotated manually 557 tweets of those 12.195. This is our gold standard to test accuracy in our experiments. In each one of those tweets we have indicated if the tweet is positive or negative, the different aspects, and the sentiment words. This information is used to test correctness in the tweet scores given by the dictionary and also to test our machine learning solutions that have to guess sentigrams.

5.2 Experiments with Sentiwordnet

The first version of our software was working exclusively with Sentiwordnet. So the concept of incremental learning was not yet defined. When we saw that the precision was not very far away from 50% we decided to perform some modifications to increase accuracy.

The first test done with the original dictionary has given us an accuracy of 46.50%. This basically means, that scores given by the original dictionary are not very helpful for the airlines domain. We know that this is a binary problem with two possible answers (positive or negative). If the accuracy is around 50% this implies that the accuracy is like flipping a coin or doing a random raffle. Anyway, in the original scores we start to see some tendencies in tweet scores for instance in words with strong positive or negative meaning.

In order to improve accuracy ratios, we tried some changes in code that we will review in this chapter.

5.2.1 Co-occurrence scores

This idea was based in Pointwise Mutual Information idea. Basically we wanted to assign a punctuation to words already marked as negative in the original dictionary. If a negative or a positive word was found inside a tweet (because one of those words was found in Sentiwordnet) we divide its score between all the neighboring words to emit a co-occurrence score. With this technique, words that co-occur more often with negative words will tend to have a co-occurrence score negative. With this second score we can complement the tweet score.

After some experiments we saw that adding more data the accuracy rate of the co-occurrence score was better. So if instead of processing 1 week of information we process 2 weeks of information the co-occurrence score was more helpful. But anyway, we got better results using exclusively sentiwordnet score and ignoring co-occurrence score. This means that this idea of co-occurrence was noisy at least with our original design.

5.2.2 Negative N-grams

The process of aggregating sentiment words in each tweet need some optimizations to perform better. For instance, we need to consider negations on N-grams to avoid assigning positive scores when something is negative. For that we simplify tweets that include negations such as “don’t”, “isn’t”, “wasn’t” replacing them for the word “not”. Now when a N-gram is preceded by the word “not” we emit this N-gram as “NOT N-gram”. In the side of Sentiwordnet we do the same we emit all words as “sentiword” and “NOT sentiword”. When a sentiword is emitted as “NOT sentiword” we emit as positive_score = 0, and negative_score = -positive_score.

5.2.3 Stemming

Stemming is a technique to reduce a word to its root. There are some stemming algorithms that help in information retrieval systems increasing recall. Using this technique if you search for the word “languages” you will find results for “language”. This means that the search is performed with the root word “languag”.

In our system we use stemming to group words by stems. This allows to consider together words that are plural with singular or present with past. This helps to join word sentiment scores in words of the same family that could be sharing positive and negative influence. In experiments stemming was a little bit noisy if we apply it alone, but in combination with negative n-grams it starts to be effective.

5.2.4 No Disambiguations

In Sentiwordnet the same word can be used for several meanings. Usually if a word has more than one meaning is indicated with the hashtag #n. For instance the word “be” can be represented as be#1, be#2, be#3 in the dictionary. Each word + hashtag represents one different meaning of a word. Consequently, this implies different sentiment scores for each meaning.

In our use-case we have not done any disambiguation between the different meanings of a word. Instead of that, we perform an average of all the positive and negative scores for all meanings. So all meanings, punctuate as a unique sentiment word. This would be very bad if we wanted to create a cross-domain classification model, but since we want to train models for specific domains it is more difficult to have problems with disambiguations. And in edge cases we hope that the scores will tend to be (0,0).

Method	**Accuracy**
Sentiwordnet	46.50%
Sentiwordnet + Co-occurrence scores	45.32%
Sentiwordnet + Stemming	45.96%
Sentiwordnet + Negative N-grams	47.94%
Sentiwordnet + Stemming + Negative N-grams + Co-occurrence scores	48.56%
Sentiwordnet + Stemming + Negative N-grams	50.98%

Table 5.1. Comparison of accuracy with all variations and the Sentiwordnet original dictionary

5.3 Experiments with Sentiwordnet after random-walk

5.3.1 Random-walk

An accuracy of 51% was not good enough, so at this point we decided to perform some changes using random-walk algorithm. This algorithm consists in modifying word scores randomly until number of matches increases. For that, we work over a set of annotated data that is our gold standard to perform accuracy tests.

The procedure consists in performing iterations over the dictionary performing random changes. Those random changes are only accepted if they increase number of matches. So if a change in the word scores decreases number of matches the change is reverted, and we try another change. The number iterations depends on the size of the dictionary so a good number of iterations is usually 100 times SIZE_DIC to ensure that every word of the dictionary is at least modified 100 times. This number of iterations ensures convergence in most cases.

Method	**Iterations**	**Accuracy**
Random-walk + Stemming + Negative N-grams	SIZE_DIC	64.09%
Random-walk + Stemming + Negative N-grams	10*SIZE_DIC	84.56%
Random-walk + Stemming + Negative N-grams	100*SIZE_DIC	95.33%

Table 5.2. Comparison of accuracy with all the variations and Sentiwordnet after random-walk

5.3.2 Random-walk with domain words

After seeing good results with original Sentiwordnet with random-walk we have tried to incorporate to the dictionary the words that we have included in our annotations (marked in red). The idea of this annotations was to incorporate domain sentiment words to the dictionary as (pos_score: 0, neg_score: 0) and make random-walk balance this scores in conjunction with original dictionary. Finally the domain words we have used do not make a big difference. This could be because the selection we used is not accurate from a linguistic point of view or maybe because some noisy situations not considered. If for instance we select the word “on time” as a specific for domain airlines and we already have in Sentiwordnet the word “time” we are duplicating information. So once the scores of word “time” are adjusted, the word “on time” remains equal.

Probably a better linguistic selection of this domain words and some fixing in this ambiguity cases could fix the problem. In any case, the performance between both solutions is nearly similar so including our own sentiment words is feasible and does not require much additional computing time.

Method	Iterations	Accuracy
Random-walk w/ domain words + Stemming + Negative N-grams	SIZE_DIC	63.02%
Random-walk w/ domain words + Stemming + Negative N-grams	10*SIZE_DIC	84.91%
Random-walk w/ domain words + Stemming + Negative N-grams	100*SIZE_DIC	94.08%

Table 5.3. Comparison of accuracy with all the variations and Sentiwordnet after random-walk with domain words.

5.4 Experiments on sentigram prediction with Weka

To test the automatic detection of sentigrams we have used the method described in chapter 4.3 that consists in creating one observation for each word inside a tweet and create one unique identificator for each word. Our objective is analyze all the words in a tweet and the position of the word we are in to guess if we are in a sentiment word {2}, in an apsect {1} or in a non-relevant word {0}. With the conjunction of all this guesses (one per word) we can classify all the important words on a tweet as a whole. And solve the problem as it were binary when in reality is a multi-class problem.

We have tested our solution with three different machine learning methods: Naive Bayes, K-Nearest Neighbors and Bagging. On the other hand, we have tested three approaches: guessing only sentiment words, guessing only aspects and guessing aspects and sentiwords in conjunction (which is equivalent to guessing the entire sentigram). The third approach is not as precise as approaches 1 and 2, but it will be able to capture correlations between sentiment words and aspects where in the other 2 solutions this distinction cannot be done. In table R we can see results:

Method	Accuracy
Sentiment words w/ Naive Bayes	84.64%
Sentiment words w/ KNN	92.01%
Sentiment words w/ Bagging	92.07%
Aspects w/ Naive Bayes	88.94%
Aspects w/ KNN	92.98%
Aspects w/ Bagging	93.63%
Sentigrams w/ Naive Bayes	78.96%
Sentigrams w/ KNN	85.21%
Sentigrams w/ Bagging	85.92%

Table 5.4. Comparison of accuracy with several machine learning methods to detect sentigrams.

We can see how classifying sentiment words is a problem where we can reach an accuracy of 92% while with aspects the problem reaches almost 94%. When we transform the problem in a problem with three outputs {0,1,2} the accuracy reaches almost 86%.

6. Conclusions

6.1 Final remarks

During this thesis work we have seen how sentiment analysis is a mature field of research with very good ideas already implemented. We have done an overview of some state-of-the-art solutions applied to sentiment classification, subjectivity summarization, or sentiment prediction.

We have demonstrated how solutions like Sentiwordnet are not as easy to extend as we may think. The selection of words provided can be adapted to multiple domains using random-walk algorithm and adapting internal weights to each use case. But is not trivial to add new words to this dictionary. We have to be careful from a linguistic point of view choosing new words for this dictionary to not add noise and decrease precision in our system. This basically proves that the original selection of verbs, adjectives and nouns in Sentiwordnet is very powerful and adaptable to multiple domains.

We have developed the idea of sentigram as extension of N-grams. Using sentigrams instead of N-grams gives us a the ability to detect relations between words that are not necessarily consecutive and not necessarily syntax-related. This kind of solution is very useful for the use case of Twitter where opinions are very short and they don't follow strong syntactic rules since the space is limited to 140 characters. Another good advantage of this approach is that encapsulates opinion direction and aspects in the same unit of information. This is useful for context-dependent words so our machine learning system trained with annotations will try to find correlations between aspects and opinion directions before assigning them to a specific group. This encapsulation is also helpful for data analytics so we can analyze groups of tweets and extract information such as most repeated sentigrams.

The results on sentigram prediction are very good considering that we do not provide any grammatical or syntactical information apart from the word order or the position we are in. Also we have to consider that we are working with a limited number of annotations. As the system is incremental, each iteration will add new annotations to the system and the accuracy ratio will grow more. Furthermore with the combination of sentigram annotations, individual annotations (aspects and sentiments), and the information provided in dictionaries we can minimize the problems given by false positives.

6.2 Research questions

HYPOTHESIS (H1): We can create groups of N-grams that influence specifically to one aspect in a negative or a positive orientation. This is what we call **sentigrams**.

We have demonstrated that this is possible. Analyzing tweets of airlines we have detected groups of N-grams non adjacent and non related by syntax that indicate opinion orientation of a tweet and its aspect. By using this new unit of information we can analyze an entire twitter topic and capture the general opinion of a group of users.

HYPOTHESIS (H2): By using **incremental learning** the system improves in each iteration.

Increasing precision.

We have applied incremental learning to improve Sentiwordnet dictionary and we have seen that this process of adding new words must be done very carefully to avoid noise. In our experiments we couldn't prove that adding new sentiment domain words gives always more precision. This could be because the words already selected in Sentiwordnet are covering most opinion orientations or because our selection of domain sentiment words was incorrect from a linguistic perspective. In any case, this noise added can be compensated by doing more random-walk iterations to rebalance sentiment scores.

Also, we have applied incremental learning to improve sentigram prediction. In this case, we have seen that Weka performs better when it has more annotations to work with. So we assume that new iterations in our system will retrieve better results than the results we have obtained in our experiments.

HYPOTHESIS (H3): After certain number of iterations is reached we can assign a sentigrams to a tweet **automatically** w/o manual intervention.

The part of the problem that can be considered as nearly solved automatically is detecting if a tweet is positive or negative. We have seen how using Sentiwordnet after random-walk we can get accuracy ratios near to 95%. Furthermore, we do not need to add new words to the dictionary to reach this ratios.

In the case of sentigram prediction, automatic detection is possible but with certain limitations. As we get more iterations in our system we process new combinations of sentiment words and aspects that can be reused to detect new opinions. But the precision obtained it is still below 90% so to get automatic detection of sentigrams we will need to design a system that combines information of individual classifiers and joined classifiers to compensate this 10-15% error ratio. With this combination the probability of false positives would be lower and we could detect the most clear cases in exchange of less information detected. With less amount of information (but always correct) we can analyze twitter topics and detect most common sentigrams to perform subjectivity summarizations.

6.3 Future work

For future work it will be interesting to focus more on subjectivity summarization and use this system to detect sentigrams in trending topics. We should explore then which sentigrams are more useful and how to combine them to produce automatic texts. We can exploit ideas such as affinity between users or social rank to determine which sentigrams are more essential to build this summary. Finally we should test this automated summaries with real users and find which of them are more close to reality.

Another line of work is improve the selection of domain words. So it would be interesting to focus our investigation in which kind of linguistic rules we should follow to get better Sentiwordnet accuracy ratios and extend this dictionary for specific domains. Other good idea is applying sentigrams to longer texts such as opinion reviews or articles. From more complex texts we could incorporate grammatical information to the algorithm and develop a model that could be used for shorter texts in the long term.

7. References

M. Ando and S. Ishizaki, “Analysis of travel review data from reader’s point of view,” in *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*, 2012, pp. 47–51.

S. Asur and B. A. Huberman, “Predicting the future with social media,” in *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, 2010, vol. 1, pp. 492–499.

S. Baccianella, A. Esuli, and F. Sebastiani, “Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining,” in *Proceedings of the 7th conference on International Language Resources and Evaluation (LREC’10), Valletta, Malta, May*, 2010.

A. Bakliwal, P. Arora, S. Madhappan, N. Kapre, M. Singh, and V. Varma, “Mining sentiments from Tweets,” *WASSA 2012*, p. 11, 2012.

J. Bollen, H. Mao, and X. Zeng, “Twitter mood predicts the stock market,” *Journal of Computational Science*, vol. 2, no. 1, pp. 1–8, 2011.

B. Bonev, G. Ramírez-Sánchez, and S. O. Rojas, “Opinum: statistical sentiment analysis for opinion classification,” in *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*, 2012, pp. 29–37.

S. R. Das and M. Y. Chen, “Yahoo! for Amazon: Sentiment extraction from small talk on the web,” *Management Science*, vol. 53, no. 9, pp. 1375–1388, 2007.

A. Esuli and F. Sebastiani, “Pageranking wordnet synsets: An application to opinion mining,” in *Annual meeting-association for computational linguistics*, 2007, vol. 45, p. 424.

P. A. Gloor, J. Krauss, S. Nann, K. Fischbach, and D. Schoder, “Web science 2.0: Identifying trends through semantic social network analysis,” in *Computational Science and Engineering, 2009. CSE’09. International Conference on*, 2009, vol. 4, pp. 215–222.

A. Go, R. Bhayani, and L. Huang, “Twitter sentiment classification using distant supervision,” *CS224N Project Report, Stanford*, pp. 1–12, 2009.

N. Godbole, M. Srinivasaiah, and S. Skiena, “Large-scale sentiment analysis for news and blogs,” in *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, 2007, vol. 2.

N. Jindal and B. Liu, “Identifying comparative sentences in text documents,” in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006, pp. 244–251.

A. Kennedy and D. Inkpen, “Sentiment classification of movie reviews using contextual valence shifters,” *Computational Intelligence*, vol. 22, no. 2, pp. 110–125, 2006.

S.-M. Kim, P. Pantel, T. Chklovski, and M. Pennacchiotti, “Automatically assessing review helpfulness,” in *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, 2006, pp. 423–430.

J. Leskovec, D. Huttenlocher, and J. Kleinberg, “Predicting positive and negative links in online social networks,” in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 641–650.

B. Liu, “Sentiment analysis and subjectivity,” *Handbook of natural language processing*, vol. 2, p. 568, 2010.

R. McDonald, K. Hannan, T. Neylon, M. Wells, and J. Reynar, “Structured models for fine-to-coarse sentiment analysis,” in *Annual Meeting-Association For Computational Linguistics*, 2007, vol. 45, p. 432.

A. M. Misbah and I. Imam, “Mining opinions in Arabic text using an improved ‘Semantic

Orientation using Pointwise Mutual Information' Algorithm," in *Informatics and Systems (INFOS), 2012 8th International Conference on*, 2012, p. SE–61.

B. O'Connor, R. Balasubramanyan, B. R. Routledge, and N. A. Smith, "From tweets to polls: Linking text sentiment to public opinion time series," in *Proceedings of the International AAAI Conference on Weblogs and Social Media*, 2010, pp. 122–129.

A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining," in *Proceedings of LREC*, 2010, vol. 2010.

S. J. Pan, X. Ni, J.-T. Sun, Q. Yang, and Z. Chen, "Cross-domain sentiment classification via spectral feature alignment," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 751–760.

B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, 2004, p. 271.

B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Foundations and trends in information retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.

B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, 2002, pp. 79–86.

J. Read, "Recognising affect in text using pointwise-mutual information," *Unpublished M. Sc. Dissertation, University of Sussex, UK*, 2004.

A. Reyes and P. Rosso, "Mining subjective knowledge from customer reviews: A specific case of irony detection," in *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, 2011, pp. 118–124.

Q. Su, K. Xiang, H. Wang, B. Sun, and S. Yu, "Using pointwise mutual information to identify implicit features in customer reviews," in *Computer Processing of Oriental*

Languages. Beyond the Orient: The Research Challenges Ahead, Springer, 2006, pp. 22–30.

M. Thelwall, K. Buckley, and G. Paltoglou, "Sentiment in Twitter events," *Journal of the American Society for Information Science and Technology*, vol. 62, no. 2, pp. 406–418, 2011.

M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas, "Sentiment strength detection in short informal text," *Journal of the American Society for Information Science and Technology*, vol. 61, no. 12, pp. 2544–2558, 2010.

A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe, "Predicting elections with twitter: What 140 characters reveal about political sentiment," in *Proceedings of the fourth international AAAI conference on weblogs and social media*, 2010, pp. 178–185.

J. Yin, P. Thomas, N. Narang, and C. Paris, "Unifying local and global agreement and disagreement classification in online debates," in *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*, 2012, pp. 61–69.