ADRIANO WEIHMAYER ALMEIDA

SENTIMENT ANALYSIS IN SHORT MESSAGES USING AFFECTIVE LEXICONS

Dissertation submitted to the Applied Computer Science Program (PPGIA) at the Pontifical Catholic University of Paraná as a partial requirement to obtain a Master's title.

CURITIBA

2017

ADRIANO WEIHMAYER ALMEIDA

SENTIMENT ANALYSIS IN SHORT MESSAGES USING AFFECTIVE LEXICONS

Dissertation submitted to the Applied Computer Science Program (PPGIA) at the Pontifical Catholic University of Paraná as a partial requirement to obtain a Master's title.

Major: *Metodologias e Técnicas de Computação*

Orientador: Prof. Dr. Fabrício Enembreck

CURITIBA

2017

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central

A447a
2017

Almeida, Adriano Weihmayer
Sentiment analysis in short messages using affective lexicons / Adriano Weihmayer Almeida ; orientador: Fabricio Enembreck. – 2017.
139 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2017
Bibliografia: f. 131-139

1. Semântica – Processamento de dados. 2. Linguística. 3. Léxico. 4. Informática. I. Enembreck, Fabrício. II. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. III. Título

CDD 20. ed. – 401.430285

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

PUCPR
GRUPO MARISTA

ATA DE SESSÃO PÙBLICA

DEFESA DE DISSERTAÇÃO DE MESTRADO Nº 05/2017

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA – PPGIa
PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ - PUCPR

Em sessão pública realizada às 14h00 de 28 de Setembro de 2017, no Auditório Guglielmo Marconi – Bloco 8, ocorreu a defesa da dissertação de mestrado intitulada **"Sentiment Analysis of Short Messages Using Affective Lexicons"** apresentada pelo aluno **Adriano W. Almeida**, como requisito parcial para a obtenção do título de **Mestre em Informática**, na área de concentração **Ciência da Computação,** perante a banca examinadora composta pelos seguintes membros:

Prof. Dr. Fabrício Enembreck (Orientador)- PUCPR

Prof. Dr. Emerson Cabrera Paraiso – PUCPR

Prof. Dr. Jones Granatyr – UNC

Após a apresentação da dissertação pelo aluno e correspondente arguição, a banca examinadora emitiu o seguinte parecer sobre a tese:

Membro	Parecer	
Prof. Dr. Fabrício Enembreck	(X) Aprovado	( ) Reprovado
Prof. Dr. Emerson Cabrera Paraiso	(X) Aprovado	( ) Reprovado
Prof. Dr. Jones Granatyr	(X) Aprovado	( ) Reprovado

Portanto, conforme as normas regimentais do PPGIa e da PUCPR, a tese foi considerada:

(X) **APROVADO**
(aprovação condicionada ao atendimento integral das correções e melhorias recomendadas pela banca examinadora, conforme anexo, dentro do prazo regimental)

( ) **REPROVADO**

E, para constar, lavrou-se a presente ata que vai assinada por todos os membros da banca examinadora. Curitiba, 28 de Setembro de 2017.

Prof. Dr. Fabrício Enembreck

Prof. Dr. Emerson Cabrera Paraiso

Prof. Dr. Jones Granatyr

Programa de Pós-Graduação em Informática
PPGIa
PUCPR

Dedicated to my wife and kids, my parents
and all that made this possible.

ACKNOWLEDGEMENTS

I would like to thank the staff and faculty of the PPGIA-PUCPR department and my fellow graduates, who have helped me in this journey to become a Master. In special my supervisor Dr. Fabrício Enembreck for his guidance and ideas that were crucial to improve the quality of this work.

Not less important, a big thanks to my wife that had to cope with my lack of time and attention to her and the kids. Gratitude is also due to my parents for the support given in my life so far, especially in this period.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

SA	*Sentiment Analysis*
OM	*Opinion Mining*
PRT	*Personality Recognition from Text*
EA	*Emotional Analysis*
BOW	*Bag of Words*
ML	*Machine Learning*
POS	*Part of Speech*
ISEAR	*International Survey of Emotion Antecedents and Reactions*
GA	*Genetic Algorithm*

Abstract

Sentiment analysis is an area of Natural Language Processing that aims to find the sentiment associated to texts. It has gained attention of researches due to the so-called web 2.0 that enabled users to generate their own content. The latest state-of-the-art works has demonstrated that affective lexicons improve the results of sentiment analysis in text, especially in microblogs like Twitter. Using this evidence, this work makes the hypothesis that adding more lexicons can help the sentiment analysis pipeline. It shows an empirical study of 17 affective lexicons and how they can help with the task of sentiment analysis in short messages. The main differential of this work is in the huge number of lexicons studied and applied, mixing polarity lexicons (positive, negative and neutral), commonly used in sentiment analysis, with emotional lexicons (anger, sadness, joy, etc.), used mainly in personality computing. It also presents a method that uses Genetic Algorithms to combine the lexicons with minimal effort generating a model with good performance when compared with the best models of the SemEval's Sentiment Analysis competition.

Keywords: Sentiment analysis; Affective; lexicons; Machine Learning

Resumo

A Análise de Sentimentos é uma área do Processamento de Linguagens Natural que busca encontrar sentimentos em textos. Ela ganhou mais atenção com o surgimento da web 2.0, já que os usuários podem se transformar em criadores de conteúdo. Os últimos trabalhos considerados estado da arte têm demonstrado que léxicos afetivos contribuem para melhorar o desempenho da análise de sentimento em textos, especificamente em microblogs como o Twitter. Baseado nesta evidência, este trabalho sugere a hipótese de que a adição de mais léxicos pode ajudar cada vez mais na tarefa de análise de sentimentos, demonstrando de maneira empírica a contribuição de 17 léxicos afetivos. O grande diferencial deste trabalho se encontra no estudo e aplicação de uma enorme quantidade de léxicos com anotações de polaridade, geralmente usados em análise de sentimentos, assim como léxicos com anotações de emoções (raiva, tristeza, angústia), comumente usado na área de extração de personalidade em texto. Também é mostrado um método que usa Algoritmos Genéticos para fazer a combinação destes léxicos com pequeno esforço gerando modelos com bom desempenho quando comparados com os encontrados na competição do SemEval.

Keywords: Análise de sentimentos; afetiva; léxicos; Aprendizagem de máquinas.

Chapter 1

Introduction

The Internet in its current state, the so-called Web 2.0, enabled individuals to become content creators. Anyone, sometimes even with little resources, can express their idea and reach a very high audience. For example, someone from Africa can share his everyday life with people from Europe. A music or video clip from an unknown artist can become an instantaneous hit all over the world creating a worldwide celebrity overnight.

The rapid communication changed the society's pace. Changes are fast and people react to them very quickly. This influence business and lifestyle, constantly changing the society mindset. Big websites emerged in this communication revolution becoming the place to share ideas in a collective manner. According to the website Statista[1] there were in January of 2016 1.55 Billion users on Facebook and 320 Million active users on Twitter while the world population is 7.3 Billion, according to www.census.gov[2]. It means that 1 in 6 people on Earth are somehow connected through Facebook. According to Eric Schmidt[3], CEO of Google, in 2010 every two days it was generated as much information as the whole history of humanity until 2003.

In the information era, knowledge is power and the ones that are able to read big amount of information and translate that into actions may have an advantage. With the massive amount of information created every day, it is impossible for humans to read everything. This is when automatic methods and most of all Natural Language processing

[1] http://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/ - visited on March 2016
[2] http://www.census.gov/popclock/ - visited on March 2016
[3] http://techcrunch.com/2010/08/04/schmidt-data/ - visited on March 2016

come into the game, translating a lot of content into simple and comprehensive information enabling companies and individuals to understand what people talk and feel collectively.

1.1. The problem

Affective computing is the name given for the part of Artificial Intelligence that tries to find the subjectivity expressed by human beings. Whether it is based on text, gestures, expressions, speech or even key strokes, the challenge is to make the computer quantify the current state of its user or its emotional characteristics, enabling an improvement in the interaction between human and machine. Specifically on texts, affective computing relies on techniques of text analysis called Natural Language Processing (NLP) to try to infer the personality, emotions and opinions of the text's author.

The terms *affect*, *feelings*, *emotion*, *sentiment* and *opinion* are used interchangeably but in the work of [MMSP14] they set some differences between them. Affect is something more abstract and it is in the top of their classification. Feelings are related to the memory we have about a situation so it can be labeled and detected from text. Emotions are the reactions from the body associated with the feelings and what is detected in texts is the consciousness experience of factors that characterize emotions. Sentiments should be related to past experiences with something creating a memory and finally, opinion are the punctual and personal interpretation of information and it may contain emotion or sentiment attached to it.

In practice, these differences in the definition are not so subtle. It is very hard for humans to properly distinguish and set apart what is being felt and put it on text. This mix of senses and situations created many areas of study in the so called affective computing.

Although Sentiment analysis (SA) and opinion mining (OM) are not exactly the same, they generally indicate the same task. According to [PaLe08], SA first appeared in the works of [DaCh01, Tong01] and it was defined as "automatic analysis of evaluative text and tracking of the predictive judgments therein". In the other hand [DDLL03] defined OM as "process a set of search results for a given item, generating a list of product attributes (quality, features, etc.) and aggregating opinions". In short words, SA would define the positive and negative sentiment towards a text while OM defines positive or negative sentiments towards a specific subject in the text.

Emotional analysis (EA) is the task to infer the emotion (anger, sadness, happiness, etc) contained in text. One of the biggest difficulties in this task is the lack of a consensus amongst researchers on how to properly represent emotions. [RuMe77] used three dimensions to define many emotions using the pair *pleasure-displeasure*, *arousal-nonarousal* and *dominance-submissiveness*. [COCC89] describe emotions as *likes* and *dislikes* of objects, *pleasure* and *displeasure* of agents and *approval* and *disapproval* of events. [Ekma92] classify "*happiness, surprise, fear, sadness, anger and disgust combined with contempt*" as basic emotions.

Differently from sentiment and emotional analysis, Personality Recognition from Text (PRT) tries to infer the personality of the author. The main difference between analyses is that while OM and SA are punctual analysis determining the sentiment of one piece of information PRT tries to infer something that is continuous, that is part of someone. While opinion analysis output relies on a general sense of the felling of an input where most of people can detect it, PRT rely on psychological features and is much more related to the psychological sciences [PCEP16].

Amongst the models that try to infer the personality, the most used is the Big Five proposed by [JoDL91]. It describes ones personality in a measure of 5 scales [ViMo00]:

- Extraversion: Active, Assertive, Energetic, Outgoing, Talkative, etc.
- Agreeableness: Appreciative, Kind, Generous, Forgiving, Sympathetic, Trusting, etc.
- Conscientiousness: Efficient, Organized, Planful, Reliable, Responsible, Thorough, etc.
- Neuroticism: Anxious, Self-pitying, Tense, Touchy, Unstable, Worrying, etc.
- Openness: Artistic, Curious, Imaginative, Insightful, Original, Wide interests, etc.

Self-assessment questionnaires like NEO-Personality-Inventory Revised (NEO-PI-R, 240 items) [CoMc95] are widely accepted and used as the ground truth to compare the automatic detection results in detecting personality traits.

In this scenario of affective computing many techniques were created to try to infer sentiment, opinion, emotions and personalities. In [Moha15], the author generates a long survey on methods and applications of affective computing in public health, politics, brand management, education, etc. This work will focus specifically on the use of affective lexicons,

dictionaries that carry information about the words and the affective weight associated to them, one of the most used techniques across-domains.

Amongst the disadvantages of this approach is that lexicons may have limited coverage of the words present in the analyzed text as their construction are domain specific [LiCC15]. Another problem is that the word's lexical and sentimental characteristics are not a consensus [WiWC05] and that is reflected in its sentiment values. A third problem is that the sentiment values may be associated to words directly thus not taking into account its syntactic or semantic meaning [AlPG12] creating wrong interpretation of the words polarities and sentiments. Adding to this list, lexicons were created in different times, using different techniques, they carry different measures and scales and were evaluated in different datasets.

Even with the afore mentioned list of issues, the use of lexicons is still a good technique to help infer sentiments on short texts like the ones found in Twitter. In the later section 3.6 it will be shown that the best SA models rely on affective lexicons to create features that are usually amongst the most relevant to the model's performance.

As far as we know, there is no work that study specifically the affective lexicons contribution and which ones are more indicated in the Twitter sentiment analysis task. This is the problem that we address in this work. How to use the many existent lexicons, created to be used in the general and similar but not equal tasks of SO, SA, EA and PRT and maximize the results of the specific task of Sentiment Analysis of short texts.

1.2. Motivations

Twitter[1] sentiment analysis has become so popular that in 2013, in the annual Conference on Semantic Evaluation Exercises (SemEval), an internal competition was created (Nakov et al., 2013). The main objective of the organizers was to help develop the studies on the subject and standardize the datasets and metrics used so that the methods could be compared properly and fairly. The competition has been so successful that it later became a track on its own. Since the initial edition, it has grown yearly in number of participants and the systems created every year outperforms the ones created before.

A deeper analysis of the winner systems will be presented in chapter 3 but in summary most of the winners rely on affective lexicons, specifically in some automatically created that were used to win the first-year edition.

[1] www.twitter.com

The motivation of this work is to create a system to compete against the state-of-the-art systems presented in the SemEval competition. It will be based on many already created lexicons used in different NLP tasks.

1.3. Hypothesis

Analyzing the progression of the created systems submitted to the SemEval competition it can be seen that, apart from the Neural Networks and Deep Learning models, what sets apart the winning solutions is the use of bigger and more task specific tailored lexicons [RNKM15].

The hypothesis that this work is based on is that the combination of lexicons, whether it contains sentimental or emotional content, whether it was manually or automatically created, produce bigger coverage in the corpus as well as more certainty to the words polarities. The combination of more lexicons can achieve better results on sentiment analysis task than the use of just a few.

1.4. Objectives

This work has the objective to test the affective lexicons individually and create a method that use many of these lexicons used on Personality Recognition from Texts, Emotional Analysis, Sentiment Analysis and Opinion Mining and check if they can help to correctly identify the sentiment (polarity) on micro-blogging posts and how to use and mix them to achieve the best performance in the task of Sentiment Analysis.

The following additional questions will enable a deeper understanding of the problem:

Q1: Are there lexicons with better performance? Is there a way to identify them?

Q2: Can different features created from the lexicons produce better performance than the simple count of the words and or sum of word polarities?

Q3: What lexicons combinations can yield the best performance? Can they all be put together to create a model with the best performance?

Q4: Is there a pattern that can identify the best lexicons?

1.5.Structure

In Chapter 1 a brief introduction to the Natural Language Processing area is done as well as some of the motivations and challenges on this area, the Sentiment and Emotional Analysis, Opinion Mining and the Personality Recognition in Text problems and the main objective of this work.

Chapter 2 will describe affective lexicons created for OM, SA and PRT.

Chapter 3 will lay the foundations of Sentiment Analysis, common features and machine learning algorithms used.

Chapter 4 describes the creation of the base model and the study of each of the many lexicons used in this work

Chapter 5 explores the use of Genetic Algorithm to search for the best lexicon combination and the results it achieves in the SemEval's datasets

Chapter 6 concludes this work with the conclusions and future works possibilities.

Chapter 2

Lexicons

According to the Oxford Dictionary, lexicon is “the vocabulary of a person, language or branch of knowledge”. In the last context, affective lexicons serve as a vocabulary of the sentiment associated to words. It is a lookup of the terms previous polarity and sentiment that can be used to indicate the general sentence sentiment. After they are created, their use is usually simple and computationally they don’t demand much processing. The words can be stored in memory and retrieved with linear time O(n) making them a good addition to the automatic sentiment analysis pipeline. Sentiment analysis systems based on lexicons have proven to be very effective [HaBB15, HPBS15, MoKZ13, MSHO14].

In the early 2000s lexicons were created mainly with hard labor of a team of researchers who needed to manually check each of the word in a context and then determine what the word sentiment was. A final annotator had to merge the inter-annotators opinions compiling the final list of words and the sentiment associated with them. As one could expect, people have different views and understanding of the text and that translates to different values assigned to the words. Consequently, affective lexicons are created using a major consensus rather than a total agreement of word’s sentiments. Manually created lexicons are expected to have a better representation of the real world sentiment (higher precision) but the mount of words that they recognize is smaller (lower recall) [ZGDH11].

With the recent evolution of computing power and the access to higher amount of human made texts, automatically made dictionary became possible. Specifically in short text messages or phrase analysis, Twitter’s API enables open access to practically unlimited corpora that can be used for text analysis. For instance in [TWQZ14] 10 million tweets were

collected and in [GoBH09], 16 million. The creation of dictionaries has become easier, cheaper and faster enabling the creation of context specific dictionaries that leads to higher coverage of words and consequently better precision and recall on the affective recognition task.

This chapter presents some of the affective lexicons that can be found in the literature. As the purpose of this work is to use lexicons employed in Sentiment Analysis, Emotional Analysis and Personality Recognition, they will be presented using this classification and a general category will present other lexicons that exists and need to be studied to better understand the creation of some affective lexicons.

2.1. General Lexicons

This section introduces the most used and referenced general lexicons. Although not being constructed directly for sentiment analysis, General Lexicons form the base of many of the affective lexicons presented in this work.

2.1.1. WordNet

WordNet's [Fell05] construction started in 1986. It is not an affective lexicon, but as it forms the base of many of the affective lexicons, it needs to be explained. It is a large lexical database of English nouns, adjectives, adverbs and verbs that are grouped forming groups of words with related concepts. These groups are linked by means of conceptual-semantic and lexical relations. The most common relation is the super-subordinate relation (hyperonymy or hyponymy) forming *synsets* like {furniture - piece of furniture}. Other noun relations are meronymy, part-whole relation like {car - wheel}, verbs relations include hierarchy as in {communicate}-{talk}-{whisper}, adjectives form antonymic synsets as in {wet}-{dry} or conceptually similar (semantically) as {dry}-{arid}. Figure 2-1 graphically show a {similar to} relation between the word *nascent* and other words meaning for example that the word *emergent* is similar to *nascent.*

As any manually created lexicon, WordNet has limitation of words, concepts and the relation of words but still is one of the biggest concept nets created. In the 3.0 version, the most recent, it is composed of 155,287 words and 206,941 word-sense pairs.

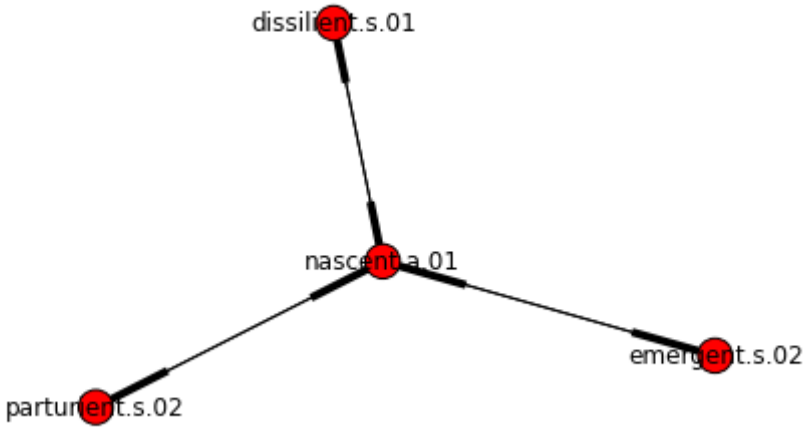

Figure 2-1 Graph generated from WordNet adjective nascent and {similar to} relations.

2.1.2. General Inquirer

General Inquirer [SHDS66] is one of the oldest manually created lexicons and contains 11,789 words gathered from 3 different sources that are classified amongst *182 categories* developed for social-sciences. The biggest category is "negative" with 2291 entries. There are 2 big valence categories (positive and negative), categories based on Harvard IV-4 that define semantic, pleasure, pain, virtue and vice, overstatement, language of a particular institution (academic, military, etc.) and others, Lasswell [LaKa50] value dictionaries that encompass deference domains (power, rectitude, respect, affiliation) and welfare domains (wealth, well-being, enlightenment and skill) and finally categories based on social cognition. Table 2-1 exemplifies some words that have positive valence and have their origin in the Harvard and Lasswell dictionaries. A better understanding of the values and columns can be found in the use instructions[1].

Entry	Source	Positiv
ABLE	H4Lvd	Positiv
ABUNDANCE	H4Lvd	Positiv
ABUNDANT	H4Lvd	Positiv
ACCEPT	H4Lvd	Positiv

Table 2-1 Example of words, their sources and one category of valence.

[1] http://www.wjh.harvard.edu/~inquirer/spreadsheet_guide.htm

2.1.3. MRC Psycholinguistic Database Machine Usable Dictionary

The dictionary created in [Wils88] is a collection of other works in the linguistic and psycholinguistic area. It is composed of 150,837 words, 115,331 unique, classified in up to 26 categories that represent classification of words according to frequency occurrence, word pronunciation, syntactic category, the status of the words (e.g. dialect, poetical, rare), the use of capital letters (indication of proper nouns), the indicative of being a derived word (e.g. Baptist from baptism), the plural form existence as the same words or irregular word and phonetic description. Table 2-2 show the distribution of the words across the many categories.

Name	Property	Occurrences
NLET	Number of letters in the word	150.837
NPHON	Number of phonemes in the word	38.438
NSYL	Number of syllables in the word	89.402
K-F-FREQ	Kucera and Francis written frequency	29.778
K-F-NCATS	Kucera and Francis number of categories	29.778
K-F-NSAMP	Kucera and Francis number of samples	29.778
T-L-FREQ	Throndike-Lorge frequency	25.308
BROWN-FREQ	Brown verbal frequency	14.529
FAM	Familiarity	9.392
CONC	Concreteness	8.228
IMAG	Imagery	9.240
MEANC	Mean Colorado meaningfulness	5.450
MEANP	Mean Paivio meaningfulness	1.504
AOA	Age of acquisition	3.503
TQ2	Type	44.976
WTYPE	Part of speech	150.769
PDWTYPE	PD part of speech	38.390
ALPHSYL	Alphasyllable	15.938
STATUS	Status	89.550
VAR	Variant phoneme	1.445
CAP	Written capitalised	4.585
IRREG	Irregular plural	23.111
WORD	The actual word	150.837
PHON	Phonetic transcription	38.420
DPHON	Edited phonetic transcription	136.982
STRESS	Stress pattern	38.390

Table 2-2 Original table of the properties described in the dictionary in [Wils88]

2.1.4. Brown Clustering

In [Brow92] the author described an algorithm that assign words to clusters. The basic idea of the algorithm is to assign the words to individual classes and then start merging the classes one by one maximizing the average mutual information of the classes. In this scenario, the mutual information will be a measure indicating how often 2 words are seen together in

the text. The final result, trained over a corpus of 365M words resulted in 260k individual words assigned automatically to 1000 clusters and can be seen in Figure 2-2.

Friday Monday Thursday Wednesday Tuesday Saturday Sunday weekends
June March July April January December October November September August
people guys folks fellows CEOs chaps doubters commies unfortunates blokes
water gas coal liquid acid sand carbon shale iron

Figure 2-2 Words in their formed cluster, as shown in the original paper [Brow92]

2.2. Opinion Lexicons

This section presents the most used opinion lexicons. It is not meant to be an exhaustive list but the most used in SemEval's and sentiment analysis works.

2.2.1. Bing Liu's Lexicon

The orientation dictionary constructed by Bing Liu is composed of 4783 negative words and 2006 positive words captured along many years of research. The construction started in [HuLi04] where they manually selected 30 adjectives that had positive and negative orientation. Using WordNet and the relation between words, they found the synonyms and antonyms of the seeds and used the new-found words as another set of seed to iteratively find the polarity of adjectives. Adjectives that were not found are discarded. Figure 2-3 shows an example of two adjectives, their antonyms and synonyms.

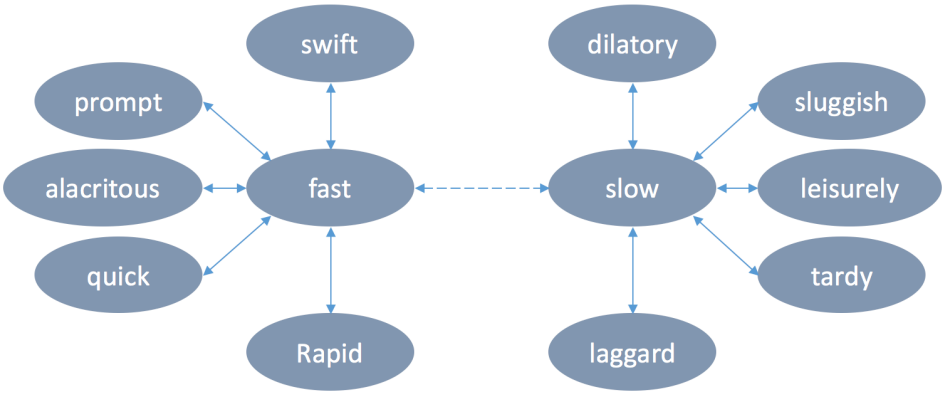

Figure 2-3 Bipolar adjectives, synonyms and antonyms [HuLi04]

The use of the dictionary is straight forward where the word polarity is indicated by its presence in either the negative or positive word list.

2.2.2. SentiWordNet

SentiWordNet [BaES08] is derived from the original WordNet and the words relations. Using a two-step approach, the authors first used a semi-supervised learning where a set of positive and negative seeds were chosen and iteratively new synsets were added to the list receiving its polarity based on the antonym or synonym relation. The word's glosses (word definition) of the first steps are then used as input for a committee of 8 classifiers trained to determine the positive, negative and objective score of each word. The output of the classifiers is averaged resulting in the words' final score.

The second step in the process is the creation of a graph where words form links in the graph if one is contained in the definition of the other. The polarity score received is the same calculated in the step one. A random walk process is then used to converge the values of all the synsets in a process defined as "inverse flow-model".

The final result is a list of words (SynsetTerms in Table 2-3) with their POS, the positive and negative score and a gloss definition. The objective score is calculated as the residual probability left from the sum of positive and negative score.

	# POS	ID	PosScore	NegScore	SynsetTerms	Gloss
0	a	1740	0.125	0.000	able#1	(usually followed by `to') having the necessar...
1	a	2098	0.000	0.750	unable#1	(usually followed by `to') not having the nece...
2	a	2312	0.000	0.000	dorsal#2 abaxial#1	facing away from the axis of an organ or organ...
3	a	2527	0.000	0.000	ventral#2 adaxial#1	nearest to or facing toward the axis of an org...
4	a	2730	0.000	0.000	acroscopic#1	facing or on the side toward the apex
5	a	2843	0.000	0.000	basiscopic#1	facing or on the side toward the base
6	a	2956	0.000	0.000	abducting#1 abducent#1	especially of muscles; drawing away from the m...
7	a	3131	0.000	0.000	adductive#1 adducting#1 adducent#1	especially of muscles; bringing together or dr...
8	a	3356	0.000	0.000	nascent#1	being born or beginning; "the nascent chicks";...

Table 2-3 Table of the first 9 words on SentiWordNet.

2.2.3. MSOL

MSOL [MoDD09] uses a different approach to construct the lexicon. As is own name states, Macquarie Sentiment Orientation Lexicon, the Macquarie Thesaurus [Bern86] was used as the base to find similar words. The thesaurus has about 100k words that are grouped in categories and subcategories creating semantic groups called paragraphs.

The construction of the lexicon was done in two steps. In the first step a set of seed words were collected using two different approaches. The first used 11 antonym-generating

affix patterns to find opposite words. For example, the pattern *X-disX* would match the pair *honest-dishonest* being the first positive and the second negative. The second approach used General Inquirer to identify another set of seeds and their polarity. As the GI is manually created, it carries a good confidence on the polarity assigned to words on it.

Both set of seeds were used in the second step to assign a polarity to the thesaurus paragraphs. For each paragraph, if the amount of positive words is bigger than the negative, positive polarity is assigned to the paragraph. Otherwise, a negative polarity is set.

Finally, to assign polarities to the words in the dictionary the same simple majority rule is used but this time the polarity is given to a word based on the number of positive and negative paragraphs the word is part of.

The final result is a set of 76,400 words, 30,458 (39.9%) positive and 45,942 (60.1%) negative words.

2.2.4. SentiStrength

[TBPC10] created an algorithm for emotion detection based on a set of Myspace posts. The main idea was to use short comments that had many abbreviations and informal texts. The corpus of 3,641 posts where manually annotated by a set of evaluators in scale of 1 to 5 for both positive and negative opinion, being 5 the indicative of stronger sentiment. This collection led to a word sentiment strength list of 298 positive terms and 465 negative terms with values from 2 to 5. Some of the terms include wildcards indicating they are a root of other words. Table 2-4 shows some examples of the words, the sentiments strength (polarity) associated with the term and the its origin.

Word	Sentiment	Origin
abandon*	-2	LIWC unless specified otherwise
abate	-2	General Inquirer Feb 2010
abdicate*	-2	General Inquirer Feb 2010
abhor*	-4	General Inquirer Feb 2010
abject	-2	General Inquirer Feb 2010
abnormal*	-2	General Inquirer Feb 2010
abolish*	-2	General Inquirer Feb 2010
abomina*	-3	General Inquirer Feb 2010
abrasive*	-2	General Inquirer Feb 2010

Table 2-4 Example of words in SentiStrength

In the manual processing of the text, other lists were created including booster word list (boost or reduce the emotion of subsequent terms), a negating word list that inverted polarity, an emoticons list with positive or negative sentiment and a slang lookup table.

2.2.5. NRC Hashtag Sentiment Lexicon

In [Moha12] the author showed that hashtagged emotion words like #angry, #joy may indicate alone the full sentiment of the tweet. Based on this idea, in [MoKZ13] the authors manually selected 30 positive and 47 negative hashtagged words entries from Roget's Thesaurus and used them as a seed for the collection of 775,000 tweets. The final lexicon created contained 39,143 unigrams, 178,851 bigrams and 308,808 non-contiguous pairs generated from the combination of unigram-unigram, unigram-bigram, and bigram-bigram pairs.

Using the pointwise mutual information as the base, they generated a simplified formula (Equation 2-1) to calculate the sentiment score for each word in the corpus. Notice how the sentiment score is a proportion of the frequency that a word is seen as negative or positive ($freq\ (w, \dots)$) times a weight determined by the amount of positive and negative words ($freq\ (\dots)$) in the vocabulary.

$$Sentiment\ Score\ (w) = log_2 \frac{freq\ (w, positive) * freq\ (negative)}{freq\ (w, negative) * freq\ (positive)} \qquad \text{Equation 2-1}$$

2.2.6. Sentiment 140 Lexicon

In the same work [MoKZ13] that the NRC Hashtag was presented the authors also presented the Sentiment 140 Lexicon. It was created the same way as the previous lexicon but the dataset used for the calculation of the words sentiment score came from a collection of 1.6 million tweets that contained positive and negative emoticons collected by [GoBH09]. The final set created is composed of 65,361 unigrams, 266,510 bigrams and 480,010 non-contiguous pairs with sentiment scores from -5 to 5 and the amount of times the words co-occurred with positive and negative tags (Table 2-5).

The authors compared the Sentiment 140 with the manually annotated sentiment lexicons MPQA, Bing Liu's and NRC Emotion Lexicon and they found that the agreement in

polarity between the automatic and the manual lexicons were between 71% and 78%. Terms with higher absolute values had better agreement from 89% to 99%.

To further check the creation of this automatic lexicon, the authors conducted a MaxDiff experiment and they concluded that the polarity values assigned automatically correspond to the human intuition.

Maxx diff (Louviere, 1991) is also called best-worst scale due to how the words scale is done. Evaluators are presented with many sets of four words where for each set one word is chosen as the most negative and the other as most positive. Almost like a tournament, the words are pit against each other and the ones that are chosen most of the time as positive are assigned higher ranks of positivity or negativity.

term	sentiment score	numPositive	numNegative
@jeffrey_donovan	5	6	0
familiar	5	6	0
@vppatel2011	5	6	0
emilio	5	7	0
@livetolovemcfly	5	11	0
@j2ad	5	6	0
http://twitpic.com/65p32	5	5	0
@missashleyn	5	5	0
http://twitpic.com/6fauf	5	6	0

Table 2-5 Unigrams, sentiment score and the amount of times the words was found with positive and negative tags

2.2.7. TS-Lex

[TWQZ14] described a neural network approach to construct a sentiment lexicon. Their model is an extension of Skip-gram model [MCCD13], a neural network that tries to infer words based on the nearby words creating a vector representation for each of the words in an n-dimensional space.

The process for assigning sentiment to words is called Sentiment-Specific Phrase Embedding (SSE) and is built in many steps. The architecture of the neural network created initially find individual word embedding (n-dimensional representation) using the Skip-gram model and then it averages the vector values of all words to create a representation of the sentence in the same n-dimensional space. This vector is used as an input to another step in the network that will learn an association with the phrase polarity. The output of the network

is a word embedding that represents a mix of the syntactical, semantic and sentimental connection between the words.

To finally classify the sentiment of the representations, the authors chosen 374 most frequent words to be used as seeds and used Urban Dictionary[1] to find similar words, totaling 1,512 positive, 1,345 negative and 962 neutral words. This created a gold labeled set of words with embedding representations that enabled to train a last classifier that outputs the probability of a word to be positive or negative.

Negative Words		Positive Words	
:(	-1.000000	)	1.000000
:-(	-1.000000	:d	1.000000
sorry	-1.000000	love	1.000000
sad	-1.000000	:-)	1.000000
bad	-1.000000	like	1.000000
hate	-1.000000	good	1.000000
ill	-1.000000	lol	1.000000
shit	-1.000000	happy	1.000000
sick	-1.000000	thanks	1.000000

Table 2-6 Extract of the negative and positive words in TS-Lex

2.2.8. MPQA

The MPQA Subjectivity Lexicon [WiWC05] is a list of subjective words allowing a phrase-level sentiment analysis system to identify neutral and polar phrases. It is based on a list of subjective clues from [RiWi03] and expanded with words from the General Inquirer, a dictionary and a thesaurus. Besides the polarity (positive, negative, neutral or both) the words received a reliability tag *strongsubj* and *weaksubj* indicating that in most of the contexts the word can be found with the indicated polarity or only in certain usages. Additionally, the lexicon has a stem indicator to represent if the words are radicals to other words. Table 2-7 depicts one word and the previous explained characteristics.

The lexicon is composed of over 8,000 words, 33.1% positive, 59.7% negative, 6.9% neutral and 0.3% are marked as having both polarities.

[1] www.urbandictionary.com

type=weaksubj len=1 word1=abandoned pos1=adj stemmed1=n priorpolarity=negative

Table 2-7 Example of word annotation in MPQA

2.3. Emotional Lexicons

Emotional lexicons are used in the studies of personality recognition and emotional identification in texts, as presented in Section 1.1. In this part it will be shown some of the emotional lexicons created and most used in this area.

2.3.1. ANEW

Affective norms for English words (ANEW) [BrLa99] is composed of 1034 words with scores on 3 different aspects: dominance (strength/weakness), arousal (excitement/calm) and valence (pleasure/displeasure) defined on the Semantic Differential Scale [RuMe77].

The authors presented a list of words to Introductory Psychology students and asked for them to evaluate each word in the 3 dimensions using a series of drawings called SAM figures [BrLa94] that allows to represent the intensity of feelings in a 9 point scale. The final score values list is an average of all the grades given for each of the words. The figure 2-4 shows the figures for dominance, arousal and valence, respectively. The 5 person drawings and the in between dots represent the values in the 9-point scale.

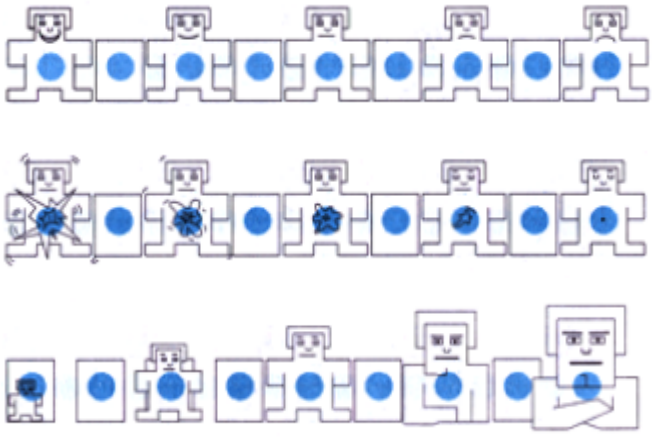

Figure 2-4 SAM figures. Extracted from [BrLa94]

2.3.2. WordNet-affect

WordNet-affect (WNA) is an extension of the WordNet lexicon. The authors [StVa04] manually mapped 1903 terms that referred to emotional states to categories like emotion, cognitive state, trait, behavior, attitude and feeling. Then all *synsets* of WordNet where these words appeared and were recognized as affective received and attribute called a-labels with the values from the previous mapping. Below is an example of the final lexicon created. The expansion of the words occurred in a semi-supervised manner where similar words received similar labels.

A-Labels	Examples
EMOTION	noun anger#1, verb fear #1
MOOD	noun animosisy#1, adjective amiable#1
TRAIT	noun aggressiveness#1, adjective competitive#1
COGNITIVE STATE	noun confusion#2, adjective dazed#2
PHYSICAL STATE	noun illness#1, adjective all_in#1
EDONIC SIGNAL	noun hurt#3, noun suffering#4
EMOTION-ELICITING SITUATION	noun awkwardness#3, adjective out_of_danger#1
EMOTIONAL RESPONSE	noun cold_sweat#1, verb tremble#2
BEHAVIOUR	noun offense#1, adjective inhibited#1
ATTITUDE	noun intolerance#1, noun defensive#1
SENSATION	noun coldness#1, verb feel#3

Table 2-8 A-labels and examples of corresponding synsets. Extracted from [StVa04]

2.3.3. Dictionary of Affect in Language

The DAL [Whis09] is a dictionary of 8742 words rated for pleasantness (how pleasant the word feels), activation (how active the word feels) and imagery (how easily the word calls an image to mind). Words with more than 10 occurrences were selected in a corpus composed by the Brown Corpus [KuFr79], essays, interviews and stories gathered by researchers at Laurentian University and received a rating for the 3 different aspects from around 200 volunteers using a 3 point scale for each.

The dictionary is part of a free program[1] that was created to identify how the word *feels* through text analyses.

word	pleasantness	activation	imagery
abandon	1	2.375	2.4
abandoned	1.1429	2.1	3
abandonment	1	2	1.4

Table 2-9 Example of words and the three categories

[1] https://www.shaktitechnology.com/whissel-dictionary-of-affect/index.htm

2.3.4. SenticNet

SenticNet [CSHH10] is based on Sentic Computing [CHHE10a] that uses Artificial Inteligence and Semantic Web techniques like sense reasoning and domain-specific ontologies to better predict sentiments and opinions in natural language. For the emotional part it uses a sentiment model called Hourglass [Plut01] to define the polarity of its terms. The model is based on the connection of the emotions as they occur in the same brain's part. For example, the state of anger also brings state of awareness that helps us to react quicker and with more strength while suppressing other resources that make us act with prudence. The classification of the sentiments is not made by sentiment categories but uses the four independent dimensions Pleasantness, Attention, Sensitivity and Aptitude that can be understood as the happiness in a context, the attention given, how comfortable the situation is and how disposed a person is to be in the same context again. Each of this dimension has six levels of activation adding to 24 possible basic emotions. The combination of the basic emotions creates more complex ones. For example, the emotion 'love' is given by the sum of Pleasantness and Aptitude.

In the building of the first version, SenticNet used 2 parts to define the values associated to the 4 dimensions. In the first part the knowledge in ConceptNet, a common-sense concept net in the model of WordNet, is blended with the affective knowledge of WordNet-Affect. In the second part, key terms are chosen and their activation is spread to the connected terms resulting after many steps in a transference of the activation to the concepts that have short paths or many different paths between them.

For the current version, SenticNet 3, 'energy flows' were used to connect many common and common-sense knowledge databases forming a database of 30,000 concepts.

Figure 2-6 shows the term *32 teeth*, how it is correlated with other concepts like *bad_taste* and with the values pleasantness, attention, etc. and the information structure based on semantic web standards using XML with specific semantic tags.

```
<rdf:Description rdf:about="http://sentic.net/api/en/concept/32_teeth">
    <rdf:type rdf:resource="http://sentic.net/api/concept"/>
    <text xmlns="http://sentic.net/api">32 teeth</text>
    <semantics xmlns="http://sentic.net/api" rdf:resource="http://sentic.net/api/en/concept/snot"/>
    <semantics xmlns="http://sentic.net/api" rdf:resource="http://sentic.net/api/en/concept/bad_taste"/>
    <semantics xmlns="http://sentic.net/api" rdf:resource="http://sentic.net/api/en/concept/mouth_part"/>
    <semantics xmlns="http://sentic.net/api" rdf:resource="http://sentic.net/api/en/concept/gum"/>
    <semantics xmlns="http://sentic.net/api" rdf:resource="http://sentic.net/api/en/concept/near_mouth"/>
    <pleasantness xmlns="http://sentic.net/api" rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.965</pleasantness>
    <attention xmlns="http://sentic.net/api" rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.775</attention>
    <sensitivity xmlns="http://sentic.net/api" rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0</sensitivity>
    <aptitude xmlns="http://sentic.net/api" rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.968</aptitude>
    <polarity xmlns="http://sentic.net/api" rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.903</polarity>
</rdf:Description>
```

Figure 2-5 An example of SenticNet 3 record.

2.3.5. NRC Word-Emotion Association Lexicon (EmoLex)

In [MoTu13] the lexicon was created using the so called "Wisdom of the Crowds" that simply implies that the collective opinion is better than the opinion of a single expert. The authors used Amazon Mechanical Turk[1] (AMT) to crowdsource the construction of the lexicon that contains Plutchik's 8 basic emotions [Plut01] for 14,182 words.

Unigrams and bigrams were chosen in Macquarie Thesaurus as well as terms in General Inquirer and WordNet Affect and were put into AMT to be annotated by five different human annotators in the 8 emotional categories. They later made sure that only valid annotations were considered to construct the final lexicon that uses the majority vote to decide if a word is related or not with a particular emotion.

The end result is a lexicon with the 14,182 annotated with an indicator of presence in the 8 emotions.

2.3.6. SentiSense

SentiSense [AlPG12] is another lexicon constructed using WordNet as the base. It is conceptually similar to what was done in WNA, but emotions were attached to the WordNet's concept level instead of the word level. The emotions categories are a mix of the emotions defined by Arnold (1960), Plutchik (1980) and Parrot (2001) and their antonyms, making a total of 14 distinct categories.

A set of 500 texts (250 news headlines and 250 hotel reviews) were presented to two judges that annotated for the concept related to each word, what emotional category was present. A total of 1200 synsets were collected and used as seeds to a second stage where similar and antonyms concepts in WordNet were collected and tagged accordingly. According

[1] https://www.mturk.com/mturk/welcome - visited April 2016

to the authors, not all relations could be used to mark new words. “Derived-from-adjective”, “pertains-to-noun” and “participle-of-verb” were the only relations, out of 8, that maintain the emotional meaning.

In the end, this semi-supervised approach was able to tag 5496 words and 2190 synsets in 14 categories. Figure 2-6 displays some examples of the synsets ID, part-of-speech (POS), gloss and emotion associated with it.

```
<Concept synset="SID-14526182-n" pos="noun" gloss="the general atmosphere of a place or situation and the effect that it has on people" emotion="ambiguous"/>
<Concept synset="SID-07947958-n" pos="noun" gloss="people who are free" emotion="like"/>
<Concept synset="SID-01041634-a" pos="adjective" gloss="disposed to seek revenge or intended for revenge" emotion="disgust"/>
<Concept synset="SID-06671484-n" pos="noun" gloss="a proposal for an appropriate course of action  " emotion="anticipation"/>
<Concept synset="SID-02367477-a" pos="adjective" gloss="made necessary by an unexpected situation or emergency" emotion="sadness"/>
<Concept synset="SID-00313633-r" pos="adverb" gloss="in a dishonorable manner or to a dishonorable degree" emotion="disgust"/>
<Concept synset="SID-02734952-v" pos="verb" gloss="be menacing, burdensome, or oppressive" emotion="fear"/>
<Concept synset="SID-01278818-a" pos="adjective" gloss="of major significance or importance" emotion="like"/>
<Concept synset="SID-00193480-a" pos="adjective" gloss="provoking horror" emotion="fear"/>
<Concept synset="SID-10002031-n" pos="noun" gloss="an advocate of democratic principles  " emotion="hope"/>
<Concept synset="SID-00732394-v" pos="verb" gloss="accept as inevitable" emotion="sadness"/>
```

Figure 2-6 Word example extracted from SentiSense 3.0

2.3.7. List of Emotional Words (LEW)

[FrGe13] constructed an emotional lexicon called List of Emotional Words. It is composed of 3027 emotional words divided into 3 main emotional dimensions that try to capture numerically basic aspects of the emotion and 92 emotional tags like grief, sad and happy.

The construction of the dictionary was based on 8 tales annotated for the presence of the emotional tags and values for 3 main emotional dimensions named affect, judgement and appreciation. The dimensions’ values were collected using the SAM 9-point scale shown in figure 2-4. To better deal with the difference between the annotations of the emotions done by 15 different subjects, they have created an emotional ontology with different levels of abstraction where deeper levels had more words that could describe the previous level.

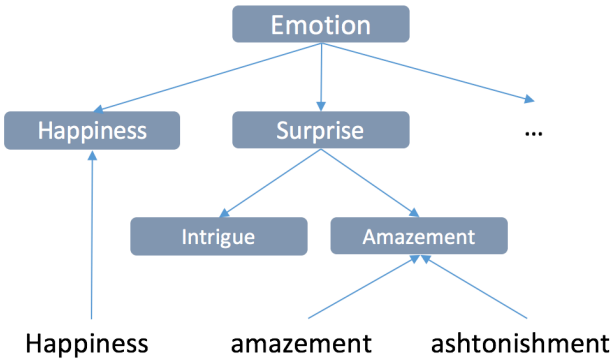

Figure 2-7 Fragment of the ontology created. Extracted from [FrGe13]

If no consensus was found in the word chosen by the annotators for the phrase, the deepest words were exchanged by the immediate parent level of the ontology until a majority consensus of the sentiment is acquired between the annotations. If no consensus was found, the neutral status was given.

To find words with emotional meaning, every word was checked in the General Inquirer for an affirmative mark in the categories “Positiv”, “Negative”, “Pstv”, “Ngtv”, “Active”, “Passive”, “Power”, “Strong”, “Submit” or “Weak”. These words were explicitly set for evaluation by the annotators and the values given in the SAM scale were used to create the value of the dimensions.

As the emotional categories present in the GI are different from the 3 dimensions used, activation was mapped to “active” or “passive”, the evaluation dimension was mapped to the “Positive”, “Negative”, “Pstv” or “Ngtv” features and the power dimension was mapped to “Power”, “Strong”, “Submit” or “Weak” features. The emotional categories were assigned using the emotional words given by the annotators.

To deal with negations, they used a dependency analyzer and a list of opposite sentiments, like happiness is opposite to sadness. Negated words would receive the opposite value for dimensions in reference to the mean of the scale, for example power=2 would become power=8, and the opposite sentiment category, happy would become sad.

To enrich the LEW lexicons first order synonyms and antonyms of the words were found in WordNet. For synonyms, similar values and sentiment categories were used. For antonyms the same logic of negated words was applied.

Word's stem	Label	Grief	Sad	Happy	Neutral	...
death	Noun	22.234	6.590	0	2.317	...
dark	Noun	0	6.844	6.704	2.945	...
grateful	Adj.	0	0	1.376	0	...
happy	Adj.	0	0	19.011	2.772	...

Table 2-10 Fragment of the LEW resource for emotional categories. Extracted from [FrGe13]

2.3.8. EmoSenticNet and EmoSenticSpace

In the work of [PGCH14] they propose an extension of SenticNet’s [CSHH10] initial four categories with additional six emotional categories of WNA [StVa04]: fear, disgust, anger, sadness, surprise and joy. To further enrich the emotional information 16 features from the ISEAR [Sche05] dataset were also used.

The International Survey of Emotion Antecedents and Reactions (ISEAR) dataset is a collection of texts collected in the 1990s across 37 countries with approximately 3000 respondents. The texts describe situations in which the respondents felt a particular emotion. The 7666 statements received 40 numerical or categorical features like age, gender and religion of the authors as well as emotions and the type of physical reactions described in the text like change in breath, muscles trembling and others. Table 2-11 presents the grouping and characteristics that were considered for the EmoSenticNet.

There were 3312 concepts present in the SenticNet plus WNA vocabulary that were common in the ISEAR dataset. Diverse similarities between these concepts were calculated like SenticNet score-based, WordNet distance-based, text distance-based, point-wise mutual information and emotional affinity. This created 13 new features that were used with the 16 categorical features to create a vector representation of the 3312 concepts.

Short Name	Description
Background	Background data related to the respondent: age, gender, religion, father's ocupation, mother's ocupation, country
General	General data related to the emotion felt in the situation described in the statement: Intensity, timing, lngevity
Physiological	Physiological data: ergotropic aroulas; trophotropic arousals, felt change in temperature
Behavioral	Expressive behavior data: movement, non-verbal activity, paralinguistic activity
Emotion	Emotion felt inthe situation described in the statement

Table 2-11 ISEAR features used in EmoSenticNet. Extracted from [PGCH14]

The vectors served as input to a set of 6 fuzzy c-means clustering models. The output was the probability of each concept to belong the emotional categories joy, anger, sadness, fear, surprise and disgust. These probabilities were later reduced to a single label using a second stage classifier, a SVM algorithm that used as characteristics the 2 strongest labels and as the gold label the WNA emotional labels. This concept-emotional label relation received the name of EmoSenticNet (ESN).

With the created lexicon, the authors used the idea of SenticSpace [CHHE10b] to create their own version of a vector representation of the words, called EmoSenticSpace. In short, two graphs are created representing ConceptNet's relations like *spoon-UsedFor-eating* and EmoSenticNet's emotional assignment like *birthday party-joy*. These graphs are represented as matrixes and using a technique called blending, both matrixes are blended creating a new matrix with the information shared by both matrixes. Finally, the number of dimensions is reduced to 100 using Truncated Singular Value Decomposition. The result is a

dictionary that represents each concept from ESN or ConceptNet's vocabulary as a 100-feature vector.

2.3.9. LIWC

LIWC [TaPe10] started in 1986 when the authors were studying if writing texts could improve health. A group of volunteers wrote hundreds of "deeply moving stories" that were analyzed for the study. As it was a time and psychological consuming task the authors imagined on creating a dictionary with positive and negative words that could help the task of automatic classification. Starting from a set of 2,800 text randomly selected, they automatically created a wordlist with categories like impersonal pronouns, auxiliary words, etc. that quickly elevated the number of categories from the original two to around 80.

Subsequently the categories were grouped in standard linguistic dimensions, psychological processes, personal concerns and spoken categories. A set of important words (highest frequency) were manually selected from the initial words and then three judges independently voted for the continuity of each of the words in the categories and in the dictionary.

In [PBJB15] the original word list was reviewed using a similar process and some categories were removed and some were added to form almost 90 different categories and 6400 words. This time different texts were used like newspaper, tweets, novels, expressive writings, blogs and expressive writings to identify important and commonly used sentiment words.

LIWC is a payed solution for text analysis that is highly used probably due to its big number of categories and the amount of effort spent during the years on revising it. A long list of papers that uses this work as a base can be found in [TaPe10]. It is highly used in personality recognition from texts and a revision of this use can be found in [PCEP16]. The Table 2-12 shows some of the categories and the number of words on it as a reference on how LIWC works.

Category	Abbrev	Examples	Words in Category
Word Count	WC	-	-
Summary Language Variables			
Analytical thinking	Analytic	-	-
Clout	Clout	-	-
Emotional tone	Tone	-	-
...			
Linguistic Dimensions			
Total Function words	funct	it, to, no, very	491
Total Pronouns	pronoun	I, them, itself	153
...			
Other Grammar			
Common verbs	verb	eat, come, carry	1000
Common adjectives	adj	free happy, long	764
...			

Table 2-12 LIWC sample categories and the number of words in each of them. Extracted from [TaPe10]

2.4. Lexicons Summary

Table 2-13 summarizes the lexicons studied so far and their characteristics. The affective type column present what kind of affect is associated with the words, the method column indicates if the lexicon was created Manually (MA), Automatically (AT) or Semi-automatically (SA), the type of construction shows if the lexicon was created based on a Dictionary or in a Corpus, the next column Corpus/Base names the base of the dictionary and finally, statistics shows the size and how the words are distributed in the corpus.

Notice how most of the lexicons were created after 2008 and semi-automatically. This may be explained by the fact that in 2006 the social media started to growth wildly with the creation of Facebook and Twitter. This lead to more interest from the academia in the area of finding emotions and opinions of users.

Another pattern that can be seen in data is that many of the lexicons created were based on WordNet or General Inquirer. Although they are manually created lexicons and carry less words, researches still see them as a trustable source for the words present in the list and create new ways of using them to infer new words polarities.

Finally, in section 3.6 it is presented the use of these lexicons in the context of sentiment analysis in Twitter. Still, there is a long list of applications that use affective lexicons like reputation systems [GSOD17], stock market prediction [OlCA16], multimodal content analysis [PCHH16], online dispute [WaCa16], author profiling [RaRo16], just to site some.

Name	Affective Type	Method	Type of construction	Corpus/Base	Statistics
General Inquirer 1966	Sentiment, emotions	MA	Dictionary Based	Harvard IV-4 Lasswell	11,789 in 182 categories
MRC 1988	Sentiment, emotions	MA	Corpus Based	Psycholinguistic Compilation	115k in 26 categories
Brown Clustering 1992	Cluster Number	AT	Corpus Based	Own corpust	260k in 1000 clusters
ANEW 1999	Emotional	MA	Dictionary	Word List	1034 in 3 different categories
Bing Liu 2004	Sentiment	SA	Dictionary based/Manually	Wordnet	4783 negative words 2006 positive words
WordNet Affect 2004	Emotional	SA	Dictionary	WordNet	1903 in 11 emotional categories
SentiWordNet 2008	Sentiment	SA	Dictionary based	Wordnet	117658 in 2 categories
MSOL 2009	Sentiment	AT	Dictionary Based	Macquarie Thesaurus General Inquirer	76k words, 30k positive and 45k negative words
DAL 2009	Emotional	MA	Corpus Based	Brown Corpus essays, interviews, stories	8472 in 3 dimensions
SenticNet 2010	Emotional	SA	Dictionary	Wordnet	30k in 4 cetegories
SentiStrength 2010	Sentiment	SA	Corpus	Myspace posts	298 positive words 465 negative words Emoticon List
NRC Emotional 2011	Sentiment, emotions	MA	Corpus	Macquarie Thesaurus General Inquirer WordNet Affect	14182 unigrams in 8 emotional dimensions
SentiSense 2012	Emotional	SA	Corpus/Dictionary based	WordNet 500 texts	59463 words and 2190 synsets in 14 categories
LEW 2013	Emotional	SA	Corpus/Dictionary based	8 tales General Inquirer WordNet	3027 in 3 dimentional dimensions and 92 emotional categories
NRC Hashtag 2013	Sentiment	SA	Corpus	Twitter posts	39k unigrams, 178k bigrams and 308k non-contiguous in a positive and negative scale
Sentiment 140 lexicon 2013	Sentiment	SA	Corpus	Twitter posts	65k unigrams, 266k bigrams, 480k non contiguous in positive and negative scale
EmoSenticNet 2014	Emotional	SA	Dictionary	WordNet Affect	13189 in 6 different emotions
EmoSenticSpace 2014	Emotional	SA	Dictionary	WordNet Affect	Unk amount represented as 100 features vectors
TS-LEX 2014	Sentiment	SA	Corpus/Dictionary based	Urban Dictionary	168845 negative terms with values 178781 positive terms with values
MPQA 2015	Sentiment	MA	Dictionary	General Inquirer Subjective clues	8k, 33.1% positive, 59.7% negative, 6.9% neutral and 0.3% both
LIWC 2015	Personality	SA	Corpus Based	Blogs (Schler et al, 2006) Expressive Writing Novels (Project Gutenberg) Natural Speech NY Times articles (1) Twitter (2)	Almost 6400 words, stems and emoticons in almost 90 categories

Table 2-13 Summary of the lexicons and their characteristics

Chapter 3

Foundations of Sentiment Analysis

In [LiZh12] the authors define opinion mining as a quintuple $(e_i, a_{ij}, oo_{ijkl}, h_k, t_l)$ meaning that in a specific time (t_l) someone (h_k) will have an opinion (oo_{ijkl}) over a specific aspect (a_{ij}) of and entity (e_i). The opinion may be positive, negative or neutral, or expressed as values of intensity/strength. Document sentiment classification is defined using an equivalent quintuple *(e, GENERAL, oo, h, t)* where *oo* signifies the opinion towards the GENERAL aspect and *e*, *h* and *t* are considered known or irrelevant. The same document classification can be used to define a phrase or, in the context of this work, a short text message like a tweet.

What seems obvious and trivial to humans, is not a simple task for an algorithm. Texts are written by their authors in varied ways and the synonyms that can be used to express the same idea give a very wide range on how to express someone's opinion.

Another challenge is that words can have different meanings in different contexts. Even in the same contexts, they may be differently interpreted by people. For example, the word "weasel" has many different meanings. The most obvious would be to name an animal or describing it but in some languages a sneaky or untrustworthy person is called a "weasel". This context is more related to one's attitude. Words can be combined and change the meaning of what is being said. Still considering the previous animal example, being said to be "fast as a weasel" may be considered a good thing. The adjective changed completely the meaning of the word representing now something related to speed.

A prior understanding of a subject may also be necessary to make a proper judgment. The example "The battery lasts for 5 hours" if applied to a laptop signify a good indicative but in the context of mobile phones, that would be a very bad mark.

The information can also be classified as objective and subjective information. The first is a fact, like the one described before where the battery hour is measured, and the second can be considered as an opinion where the author express his feeling towards the subject. [HHWW00] were among the first to study how objectivity and subjectivity can affect the opinion mining task.

The sentiment of a whole text may be changed with just a small detail. In [PaLV02], they describe their work on sentiment analysis of films reviews. The example *"This film should be brilliant. It sounds like a great plot, the actors are first grade, and the supporting cast is good as well, and Stallone is attempting to deliver a good performance. However, it can't hold up"* clearly shows that although the sentiment is good and keep growing during the text, the general sentiment for the review is bad.

In [BoOs69] the authors demonstrated the human tendency to use more positive words than negative words referenced as the *Pollyanna principle*. In the SA field, that means a skewed distribution where negative words are more difficult to identify and to train.

Lastly, sarcasm is also considered a difficult task [NiHu06, PaLe08] that depends on the knowledge of the context to properly identify that the author mean the opposite of what is written.

In this chapter, it is presented some of the most commonly used techniques for SA in Twitter posts, problem formulation of Sentiment Analysis, common features used, machine learning techniques and it finishes with the analysis of the winners of SemEval's works. For a more complete review on general SA, the reader is referred to [InDa10, LiZh12, MeHK14].

3.1. Sentiment Analysis problem formulation

Sentiment analysis can be treated simply as a *single-label multi-class* classification problem, where the algorithm must indicate whether the sentence belongs to one of three classes *C={positive, negative, neutral}*. Another way to deal with the classification is using a 5-point scale, *C={very positive, positive, neutral, negative, very negative}* or *C={-2, -1, 0, +1, +2}* and is called *ordinal classification* or a *regression problem* as the algorithm needs to give a *value* for the sentiment associated with the text. Different scales can be found in the industry and in academic work. Usually big online stores use a 5-point scale to summarize the review done by their clients.

In this work we will focus on the *single-label* classification. The performance of the systems will be analyzed using the *macro F1-score* for the positive and negative labels to enable the comparison between the proposed system and the ones from the SemEval competition [NaKR13].

The macro F1-score is defined as the average of the F1-scores of the individual classes. In SemEval's measure, it is considered only the positive and negative classes.

$$F_1 = \frac{F_1^{pos} + F_1^{neg}}{2} * 100 \tag{3-1}$$

In turn, the F1-score for each class is defined over the precision and recall calculated according to the confusion matrix of Table 3-1:

- Precision: total correctly predicted of a class over the total predicted for the class:

$$Prec^{pos} = \frac{PP}{PP + UP + NP} \tag{3-2}$$

- Recall: total predicted for the class over the total amount for the class:

$$Rec^{pos} = \frac{PP}{PP + PU + PN} \tag{3-3}$$

- F1-score: total predicted for the class over the total amount for the class:

$$F_1^{pos} = 2.\frac{Prec^{pos}.Rec^{pos}}{Prec^{pos} + Rec^{pos}} \tag{3-4}$$

		actual		
		positive	negative	neutral
predicted	positive	PP	PN	PU
	negative	NP	NN	NU
	neutral	UP	UN	UU

Table 3-1 A confusion matrix of the predicted and real classes

3.2. Automatic Classification

It is clear for the reader at this point that although is possible to manually classify sets of texts, for big corpus like Twitter messages it is nearly impossible. Researches have a wide range of algorithms called Machine Learning (ML) algorithms that use techniques derived from statistical studies to automatically classify and analyze data. The main idea of these algorithms is that they need to automatically *learn* from current data how to do a specific task. According to [Mitc97, S.2], "*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E*".

Following the definition given by Mitchell, the expected pipeline for a ML system that will be used for the identification of sentiment of tweets is that the algorithm will be given the development tweets and trained to properly identify which are positive, negative and neutral, and that it will get better at the task according to the measure and the problem formulation in section 3.1. The trained model is able to predict the polarities for unknown tweets of a test dataset. As the task of training the ML model for the SA task is based on a set of labeled training data it is called *supervised learning*.

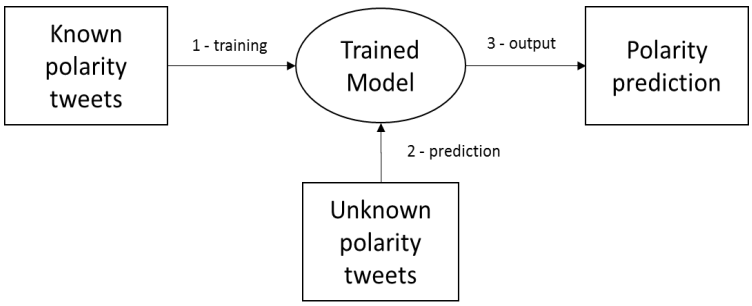

Figure 3-1 Basic ML pipeline

The simplified flow explained before is a high-level view of the process. Usually, the process is more complex. First, training data is gathered and a preparation step is done where data is cleaned, analyzed and understood. Next, features are created so that the ML algorithm can use the training data as the input to its learning process. Most of the algorithms have parameters that can be set in order to make a better learning process. The process of finding the best parameters and the best models is called *model selection*. Finally, to make sure that the model has learned well and generalize properly to unseen data, a new known dataset is

used to properly evaluate the performance of the classifier. This is an iterative process and after all the steps are done, it may start again from the beginning with the goal to make the model even better.

Due to the amount of possible features created as input the most used algorithms are the ones capable of analyzing high dimensions vectors like Support Vector Machines [CoVa95] and Logistic Regression [Cram02], and the one based on probabilistic rules, i.e. Naïve Bayes [Zhan04], that given a word *w* it tries to find the probability of this word belong to class A or B. This work will not dive into how these algorithms work as they are not the main object of the study and each of them are complex enough to be subject of individual studies.

3.3. Common features

Research of automatic SA started mainly with the work by [Turn02] and [PaLV02]. The first used mutual information between phrases and words "excellent" and "poor" to classify documents, while the second developed what is the base of most of research on this area today.

Initially Pang et al. (2002) did an experiment on how important words could be for sentiment classification and how human perceive them. Two humans selected what they considered good and bad words in a corpus. The agreement score achieved was only 69%. Later they applied machine learning algorithms to classify reviews from the IMDB[1], a movie reviews website, and used as features only words that appeared at least 4 times in the corpus. They also negated words that were placed between a negation term and a punctuation mark using the prefix NEG_ on them. The use of bigrams, part of speech (the classification adjective, noun, verb, etc.), using only adjectives did not improve the final results. A good consideration is that using only the 2633 most common words yield almost the same result of using the full set of features. Table 3-2 is a reproduction of Pang's summary table showing the type of features and the number of features created, if the features used frequency or just presence of the tokens and finally the results achieved for each of the tested algorithms.

[1] http://www.imdb.com/

	Features	# of features	Frequency or Presence?	NB	ME	SVM
1	unigrams	16165	freq.	**78.7**	N/A	72.8
2	unigrams	"	pres.	81.0	80.4	**82.9**
3	unigrams + bigrams	32330	pres.	80.6	80.8	**82.7**
4	bigrams	16165	pres.	77.3	**77.4**	77.1
5	unigrams + POS	16695	pres.	81.5	80.4	**81.9**
6	adjectives	2633	pres.	77.3	**77.7**	75.1
7	top 2633 unigrams	2633	pres.	80.3	81.0	**81.4**
8	unigrams+position	22430	pres.	81.0	80.1	**81.6**

Table 3-2 Results extracted from [PaLV02]

As the initial work done by Pang, most of the current machine learning approaches for sentiment analysis rely on the Bag of Words (BOW) model. The idea is straight forward. Given a text, split the text in small chunks called *tokens* and represent their presence or absence in some form. As an example, the tweet "*Chuck and Cheese is #awesome*" and *"Algorithms don't have to be BIG to be #AWESOME"* would be represented as the number 1 and 2 in Table 3-3.

Another alternative to represent the words is to use a direct count of the occurrences in the tweet. This would give a notion of how often a word appear. This approach is shown as the indexes 1-c and 2-c in Table 3-3.

Not utilized much in the Twitter's SA but used in Information Retrieval is a measure called TF-IDF (*term frequency–inverse document frequency*). This metric capture how frequent a word is in a document. The idea is that a word that is related to the main topic of the document will be frequent inside this document but it should not be frequent in the whole corpus. To capture it, weight the document frequency with the ratio of the number of documents in the corpus and the number of documents that uses this term. The equation (3-5) depicts the TF-IDF where t_j is a term, d_i is a document, N is the total number of documents and N_t is the number of documents that contains the term t_j. This approach is represented as 1-tf and 2-tf in Table 3-3.

$$TF - IDF(t_j, d_i) = tf(t_j, d_i).idf(log\frac{N}{N_t}) \tag{3-5}$$

#	Chuck	and	Cheese	is	#awesome	Algorithms	don't	have	to	be	big
1	1	1	1	1	1	0	0	0	0	0	0
2	0	0	0	0	1	1	1	1	1	1	1
1-c	1	1	1	1	1	0	0	0	0	0	0
2-c	0	0	0	0	1	1	1	1	2	2	1
1-tf	0.3	0.3	0.3	0.3	0	0	0	0	0	0	0
2-tf	0	0	0	0	0	0.3	0.3	0.3	0.6	0.6	0.3

Table 3-3 BOW representation and the frequency, count and TF-IDF representation

There are other metrics that are used in Information Retrieval, but for sentiment analysis they do not seem to be too relevant as shown in [DDLL03]. They include Odds Ratio, Fisher discriminant, Information gain, Jaccard Distance. [PaLV02] obtained better performance using boolean than frequency representation of terms and this is used in SA almost like a norm.

Some concepts may not be represented with a single word. For example, the name "San Francisco" cannot be separated or it will lose completely its sense. Phrasal verbs, ("cheer up"), idiomatic expressions ("hands down") and commonly used word combinations ("third world countries") are other examples of words that commonly occur together called *Collocations* and together represent a concept. The *n-word model* tries to address these situations representing tokens with more words instead of a single word as in the simple BOW representation. Bigrams and trigrams are the terms used for the two-word and three-word model, respectively.

Yet another variation to the BOW model, *Stemming* is the step to reduce words to their radical (stem), representing variations of the words as a single token. The stem "compute" is the stem for "computer", "computed", "computing" and other word variations. Stemming is normally used as a feature reduction step, as it acts as a mapping reducing many words to just one.

Considered a lexical feature, Part of Speech (POS) determines the function of the word in the sentence. For the English language, the common POS are noun, adjective, verb, adverb, conjunction, interjection, pronoun and preposition. In some studies [HuLi04] adjectives are considered to carry sentimental weight and are the base for the SA systems. The automatic identification of the part of speech is considered a task with high accuracy with

systems that are able to correctly access the classification with 93% of accuracy [OODG13]. The features created from this step may be a token added with the POS tag (i.e. Verb-be) or the statistical information like the number of verbs, adjectives, personal pronouns, etc.

Another important set of features for NLP are the linguistic features such as negation, intensification and modality. How a system deal with sentences such as “I don’t like” can vary. In [TBTV11] positive words as *like* will only have their positive score decreased and intensification situations such as “like a lot” will have the positive score increased. Differently, [PaLV02] system, in the previous negation example, would have simply created a new token “NEG_like” while the intensification case would not have any special treatment.

3.4. Feature Selection

Feature selection, as the name implies, try to select the best features from the whole set that can help the classifier to better to its job. The main objective of this is to reduce the so called “course of dimensionality” and to reduce the overall processing power needed to analyze data.

In [PaLV02] selecting the 2633 most common words had almost no effect in performance and made the feature number an order smaller. [DDLL03] tried to substitute low frequency words by the term *_unique* or words that occurred only in some products like the “the focus is poor” and “sound is poor” with the word product type but that did not help. In [ShDe12] the author compares five feature selection metrics used in IR with 7 different classifiers in the sentiment analysis of movies reviews. He finds that Gain Ratio used with SVM had the best performance. [RiPW06] use subsumption hierarchy to create more complex features and to remove other unused features. [AbCS08] mixed entropy with Genetic Algorithm to create a feature selection algorithm that perform better than standard reduction techniques on web forums.

Most of the work presented in the following section 3.6 uses a simple frequency count to reduce the number of features.

3.5. Twitter Specific

Sentiment analysis in texts, blogs, emails and product reviews may suffer from a problem of many opinions in the same text. As they may contain many sentences, each of

them can be related to one specific feature and have a different sentiment regarding its subjects. Twitter sentiment analysis usually do not suffer from this problem.

As the maximum number of characters allowed to be written is 140, the number of topics that can be found in a tweet is usually one. This can be considered as an advantage as it is less likely that opposing sentences occur. On the other side, the lack of text and mainly the exaggerated use of informal text, emoticons and misspelled words make the reuse of dictionaries and lexicons created in different sets of texts more difficult.

A tweet is a 140 characters message, a post, that express someone ideas and feelings or just an information communication like news. Due to the small size, Twitter is considered a micro-blogging tool like Tumblr, Facebook, Instagram and many others that came after them.

The main characteristics of a Twitter post are:

- Short text: with only 140 characters the ideas are expressed in a very condensed form.
- Acronyms and abbreviations: for example, *for your information* becomes *fyi*;
- Emoticons: character sequences that bring emotions like :) for happiness and :(for sadness;
- Elongated words and misspellings: *cool* becomes *cooooool* or *angel* become *angle* or *cologne* becomes *colon*;
- Hashtags: in Twitter #something means a tag to make a short summary of the post or to put it inside a bigger public conversation context;
- Mentions: the @ symbol denotes that the post is directed at some other user that will receive the post as well;
- RT: retweet

Due to its high use, Twitter has an increasing interest in the scientific community as denoted by [MTAR12]. The analysis of the possible applications, content analysis and how tweets are used are among the most studied themes. Disaster prevention [HaPK14, MTAR12], health care analysis [ViBo12] and politics [TSSW10] are some examples of applications studies. Content analysis, sentiment analysis, spam detection and tag analysis are good examples of research areas. The variety of topics that are discussed is also a great advantage as past efforts on SA are based on a single domain such as movie reviews.

3.6.Sentiment Analysis on Tweets

In this section, we will detail some previous work on sentiment analysis specifically for Twitter. We start with the SemEval competition of 2013, 2014 and 2015, analyzing the first 3 best systems of each year. Later we analyze works done on other datasets not related to SemEval. This revision is not meant to be exhaustive, as the topic has many papers, but a list of the latest papers and the state-of-the-art. Other reviews can be found in [HaPK14, MeHK14, MTAR12].

3.6.1. SemEval 2013

SemEval 2013 edition [NaKR13] had 34 teams that created 48 submissions. Due to its popularity, this competition is a great place to find techniques for the task of short text sentiment analysis and compare them, as they use the same dataset and the same scoring measure presented in Section 3.1.

In 2013, the team NRC (National Research Council) Canada won the competition with 2 automatic built sentiment lexicons in conjunction with 3 manually created lexicons [MoKZ13]. They used their own NRC Emotion Lexicon, NRC Hashtag Sentiment Lexicon, Sentiment 140 Lexicon, Bing Liu's Lexicon, and the MPQA Subjectivity Lexicon along with regular features like word and char n-grams, POS, Brown word clustering creating features used as input to a SVM classifier. To find the negated words, they used the same logic as Christopher Potts' Sentiment Symposium Tutorial[1].

In their ablation experiments they concluded that the most important features for detecting the polarity on Twitter were the features created from the automatic generated lexicons. Another important aspect of their work is that they used their automatic lexicons on Movie Reviews and achieved a better score than the state of the art best result, that used a very complex Neural Network and many manually tagged sentences.

System	Accuracy
a. Majority baseline	50.1
b. SVM-unigrams	71.9
c. Previous best result (Socher et al. 2013)	85.4
d. Our System	**85.5**

Table 3-4 Message-level task performance on Movie Reviews. Extracted from [MoKZ13]

[1] http://sentiment.christopherpotts.net/

Another great contribution of NRC Canadas's work is that the authors also demonstrated the creation of automatic sentiment lexicons that uses no previous annotations. Besides choosing some twitter tags and the treatment done with negative words and phrases, the technique created a very powerful lexicon with high coverage results and good polarity of words.

[GüFu13] developed the second-place system. A much simpler one but that achieved good results. As a preprocessing step, they used a simple regular expression to split the tweet text into words. The phrase "#liiike:)" was split into the tokens #, *liiike* and :). Then they were normalized lowercasing the words and substituting digits by 0. This tokens were stemmed using porter stemming [Port80]. Elongated tokens such as "liiike" were converted to "like". The id of the Brown Clustering [Brow92] for raw, normalized and collapsed tokens were fetched. The lexicon based features were summed positive and negative for the words found in the SentiWordNet lexicon.

Finally, as input to a SGD classifier, they used the normalized tokens, stems, word cluster ids and the lexicon features. This relatively simple system was able to score 65.27 achieving the second place in the SemEval 2013 competition.

A different approach to sentiment analysis is shown on [RBCC13] where the prediction is fully done using rules and no machine learning algorithm. The team at SAS used their previously created set of rules used to track sentiment around brands, entities and topics and adapted it to the context of the SemEval task. The job took around 2 person-week and shows that the system can be generalized to other kind of sentiment texts.

Rules prediction in a high level can be though as a lexicon but instead of just words that reference a value of positivity or negativity they contain a rule like "If negation word found before adjective, polarity is negative with value -1". In the SAS rule system, that would be translated as "def{Negation} def{PositiveAdjectives} = -1".

With this set of rules, the system was able to achieve the third place in the contest with a measurement of 64.86.

3.6.2. SemEval 2014

The 2014 edition [RRNS14] had 50 submissions from 44 teams, a raise from the previous edition. Some of last year participants submitted revised work from the previous

edition. To better understand and compare the systems, the dataset used in the 2013 edition also appeared in the final results, but were not used to declare the final winner. The 2014 edition had regular tweets, sarcastic tweets and LiveJournal sentences and only the scores achieved on these datasets were considered the final scores.

The winner of this edition was called TeamX and was composed of Fuji Xerox's employees [MSHO14]. They improved the NRC Canada system that won the previous year with more sentiment lexicons and better treatment of the words, making sure the words could be found inside the lexicons. Figure 3-2 shows their system and its modules composition.

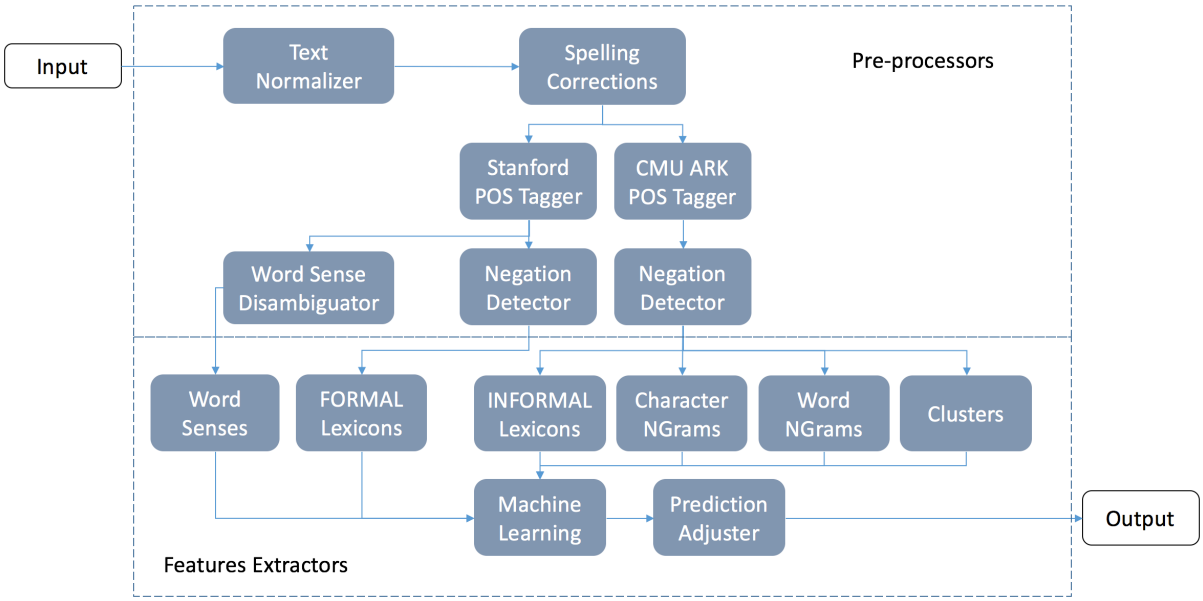

Figure 3-2 Overview of Team X system. Extracted from [MSHO14].

The text normalizer did Unicode normalization, upper case letters were converted to lower case and URLs were replaced with the token 'URL'. The Spelling Corrector module will search in a dictionary called Jazzy to correct misspelled words. The POS part uses 2 tagging systems, Stanford POS Tagger and CMU ARK POS tagger. The first one is meant to deal with “formal” tokens and extract word sense features while the second one is directed to the “informal” tokens and extracted ngram features and cluster features. Word sense disambiguation was done using UKB. Finally, on the pre-processing part, the negation used Christopher Potts logic the same as NRC's 2013 winning solution.

The final features were word and character ngrams, lexicons based features, cluster and word senses created using the UKB word senses disambiguation. These features were

used as an input to a logistic regression classifier. Additionally, after the classification, weights were associated to the positive and to the negative classes and tuned to give the best F1-score according to the competition rules.

The submission scored 72.12 in the 2013 dataset and 70.96 in the newly created dataset for the 2014 dataset achieving the first place.

The second best scoring submission came from the work of [TWQL14]. As the first scoring system, NRC's 2013 system was used as base. Additionally to most of the features from the previous year winner, they added a sentiment-specific word embedding (SSWE) created from a neural network and unsupervised learning method. This work latter generated a sentiment lexicon called TS-Lex, presented in section 2.2.7.

The third place this year was the second place of 2013. Team RTRGO [GVJT14] basically improved their system changing the lexicon dictionary from SentiWordNet to Bing Liu's, MPQA, Twitrratr wordlist and NRC hashtag sentiment lexicon. This helped raise their system final performance in about 5% achieving 69.95.

In their ablation experiments, they concluded that the lexicon created features were the most significant as features for a machine learning algorithm but alone are not sufficient for good prediction. A basic machine learning algorithm with bag of words had better performance surpassing with only 700 examples a majority-vote strategy classifier that uses Bin Liu's sentiment lexicon only.

3.6.3. SemEval 2015

The SemEval 2015 competition [RNKM15] was composed of two reruns from previous years, sentiment of just a phrase and sentiment of the whole tweet, and three new tasks, sentiment towards a topic in a single tweet, the overall sentiment towards a topic in a set of tweets and finally the degree of prior polarity of a phrase. Forty teams competed on the task sentiment analysis of a tweet.

Rather than creating more features or different algorithms, the 2015 winner adopted a strategy that is very used by winners at Kaggle[1], a popular Machine Learning competition website. The model proposed is an ensemble of previous year's winners [HPBS15]. They reproduced the systems created by NRC Canada (2013-1st), GU-MLT-LT (2013-2nd), Klue (2013-5th) and TeamX (2014-1st) but instead of using the majority voting of the predictions

[1] www.kaggle.com

done by the classifier, the confidence of the four classifiers given in each class was averaged and then the class with the biggest confidence was chosen.

According to the authors, they handpicked these systems first because of the good performance but also because they had different creation methods and that would guarantee different views for the same problem. They also emphasize that their reproduction of the systems is not exactly the same as the original, but a best effort copy. Even with some errors in implementation, they managed to score 64.84 points achieving first place.

The second best system in 2015 [SeMo15] was called Unitn and achieved a close score of 64.59. Using neural networks and word embedding the system was one of the most robust in the whole classification path. Although not scoring the first place in the sentiment analysis task with the 2015 dataset, it achieved compared results for it and very good results for 2013 and 2014 making it the first place if the 3 years' results were averaged.

The system created uses a single convolutional layer followed by a non-linearity using ReLU activation, max pooling and a soft-max classification layer. Training the neural network with random weights and biases, the score achieved for 2014 dataset was 63.69. To improve the performance, word embedding retraining [MCCD13] and distant supervision for sentiment capture [GoBH09] raising the F1-score to 71.07 and 73.60, respectively.

Achieving the bronze medal of 2015 competition, team Lsislif [HaBB15] used a combination of features generated from sentiment lexicons, word cluster, topic analysis, semantic role labeling and a standardization of the term frequency using multinomial distribution called Z score.

The system achieved 64.27 points and the authors shown that although the Z-score helped the system, the lexicons are the strongest features and have the biggest impact in the performance.

SemEval 2016 works were not considered in this work as the logic changed in this competition. The task of classifying positive, negative and neutral became a regression problem were the systems had to predict a score between 0 and 5, being 0 a negative Tweet while 5 a positive Tweet. This does not enable a direct comparison with the works presented in 2013, 2014 and 2015.

Table 3-5 summarizes the top 3 systems for each year the SemEval's competition, Twitter sentiment analysis. It shows the position achieved in the competition, the machine learning algorithm, lexicons and other features used and finally, the F1-score for each of the

datasets they were tested. It can be seen that with exception of TeraGram (rule based) and UNITN (deep learning based) all the models rely on some lexicon, corroborating the importance of this method on sentiment analysis for Twitter.

* contain or not contain ** number of

Nr	Team	Competition Rank	Learning Method	Lexicons	Other Features	Results
1	NRC CANADA (Kiritchenko et al, 2014)	SemEval 13-1	SVM	NRC Hashtag Sentiment (A) NRC Sentiment 140 (A) NRC Emotion Lexicon (M) Bing Liu (M) MPQA Subjectivity Lexicon (M)	Word ngrams (1,2,3,4) * Character ngrams (1,2,3,4) * ALL-CAPS * POS * #hashtags * NEG_negation * !punctuation ** Emoticons :) * Elooooongated ** Brown Cluster 10010001 *	SemEval -2013 – 69,02
2	GU-MLT-LT (Gunther et al, 2013)	SemEval 13-2	SGD	SentiWordNet	normalized token * Stems * Brown Cluster 10010001 * Negated normalized *	SemEval -2013 – 65,27
3	TeraGram (Reckman, H., Baird, C., Crawford, J.,et.al. 2013)	SemEval 13-3	Rules Based	---	---	SemEval -2013 – 64,86
4	TeamX (Miura et al., 2014)	SemEval 14-1	Logistic Regression	All dictionaries from 1 AFINN-111 General Inquirer SentiWordNet	Same as NRC Word Sense UKB	SemEval-2013 - 72.12 SemEval-2014 - 70.96
5	Coooollll (Tang et al,.2014)	SemEval 14--2	SVM	Sentiment word embeddings	Same as NRC (1) minus POS, #hashtags Emoticons :) from SentiStrengh	SemEval-2013 - 70.40 SemEval-2014 - 70.14
6	RTRGO (Gunther et al, 2014)	SemEval 14-3	SGD	Bing Liu MPQA Twitrratr NRC hashtag sentiment	unigrams, bigrams - stopwords and punctuation * stems - Porter * Brown Cluster 100010001 * #hashtags - * URL * Question Mark * negated normalized * exluded names and user mention	SemEval-2013 - 69.10 SemEval-2014 - 69.95
7	Webis (Hagen et al, 2015)	SemEval 15-1	Ensemble	Ensemble of other models NRC (1), GU-MLT-LT (2), TeamX(3) and Klue	Ensemble of other models NRC (1), GU-MLT-LT (2), TeamX(3) and Klue	SemEval-2013 - 68.49 SemEval-2014 - 70.86 SemEval-2015 - 64.84
8	UNITN (Severyn et al, 2015)	SemEval 15-2	Neural Nets	---	Word embeddings (word2vec), Sentiment embeddings	SemEval-2013 - 72.79 SemEval-2014 - 73.60 SemEval-2015 - 64.59
9	Lsislif (Hamdam et al., 2015)	SemEval 15-3	LogisticRegression	NRC Hashtag Sentiment (A) NRC Sentiment 140 (A) SentiWordNet (A) Bing Liu (M) MPQA Subjectivity Lexicon (M)	unigrams, bigrams * Potts' negated * Own Created Twitter Dictionary * Z Score ** Brown Cluster 100010001 * Topic Features ** Semantic Role Labeling Features	SemEval-2013 - 71.34 SemEval-2014 - 71.54 SemEval-2015 - 64.27

Table 3-5 Summary of SemEval's works.

3.6.4. Other Works

One of the first works to focus specifically in sentiment analysis on Twitter is presented in [GoBH09]. By the time, as there was no big corpora of tweets, they created one using emoticons like ":)" and ":(" as the key of Twitter's search queries. It leads to a total of

1.6M tweets for training and 359 manually labeled for testing. For the learning part, they followed [PaLV02] and used unigrams and bigrams as input for SVM, Max Entropy and Naive Bayes. The best result came from the use of Unigram and Bigram and the Max Entropy classifier. As they were only predicting positive and negative tweets, no neutral, the accuracy achieved was 83.0%. The thing that needs to be noted here is that the measure used on their work is not the same as the ones used in the SemEval task. This makes comparison impossible.

In [PaPa10] they created a new dataset with 300,000 tweets, but they evolved Go's model adding posts from 44 newspapers as the neutral. The idea is that objective texts, like news, do not contain sentiment attached to them as journalists search for neutral sentiments towards the news presented. An analysis of the dataset collected showed that the distribution of the words follows Zipf's law [Powe98] that some words appear many times and many appears just a few times. Another analysis made was based on the POS of the words. Subjective texts use more first and second person pronouns, past tense, superlative adjectives and adverbs while objective texts use third person, past participle and comparative adjectives. Comparing positive and negative contexts, positive use more superlative adverbs like “most” and “best” while negative use more verbs in the past tense like “missed”, “gone” and “lost”.

The system created uses SVM as the classifier and as the input, unigrams, bigrams, trigrams and negated bigrams linking the negative terms with previous and next words. It creates tokens like “I do+not”, “do+not like” and “not+like fish”. The accuracy achieved was not explicitly informed, but by the graphs shown seemed around 0.65 in Go's test database. Some conclusions drawn in the paper is that bigrams worked better than trigrams and unigrams, bigger datasets help the classifiers until some point, the negation of words helped and that choosing features using salience rather than entropy is better.

[ZGDH11] were the first to use lexicons on Twitter SA. The system was meant to perform entity-level sentiment-analysis and it was composed of multiple steps that would prepare the input features for a machine learning classifier. In a preprocessing step, retweets were removed, abbreviations were set back to the original form and external links and user names were removed. Questions were also removed and co-reference resolution is done finding the entity associated with pronouns like “it”. The lexicon dictionary used [DSLY08] was an expansion of WordNet added with idioms like “cost (somebody) an arm and a leg”. The authors further specialized it to Twitter context with emotions hashtags. Additional steps

include detecting comparative and opinion sentences and giving polarity to them. To raise recall after detecting possible polarity, using chi-square metrics they identified possible opinionated tweets and words that may be indicators of opinion and added them to the lexicon as well. The final step is a SVM classifier using unigrams and negation presence and the orientation for the entity. Their system was able to achieve and average accuracy of 0.854 in three classes classification, using a dataset that was created by them. The average F1-score is 0.749.

[JYZL11] discussed the target-dependent sentiment analysis. They argue that although sometimes the tweet is positive towards a subject, towards another in the same post it is not. In the example “Windows 7 is much better than Vista” there is a positive feeling regarding Windows 7 but not for Vista. They developed a system that is composed of three parts. First the subjective and objective posts are detected. For the subjective, polarity classification occurs and later a graph-based optimization is performed comparing the post with related posts. As input for the polarity classifier they used features “target-independent” like General Inquirer lexicon, words, hashtags, punctuations, emoticons and “target-dependent” that take into consideration the target and the words surrounding them (extended target) and the verbs influence over the target.

They did not report their results very clearly but the accuracy achieved for 2 class classification seems to be 85.6% and for 3 classes is 68.3%, with and average F1-score for the 3 classes of 67.6. The biggest contributions in this paper is that they show that subjectivity and objective classification is harder to do than polarity and that target dependent features and graph analysis may help to achieve better results.

[MaFu11] analyze polarity of tweets related to politics. Their experiment is not the typical positive, negative and neutral classification problem but a classification whether the tweet is pro or con a political party and which party. For that they have created a corpus of 4 million tweets collected over the UK pre-election of 2010 selected using hashtags like #election2010 and #bbcqt (BBC Question Time). The system was composed of many modules that included affect annotations using the WordNet as base and hashtags, detection of opinionated statements that included question marks, negatives, entity recognition to find political parties and finally a subject detection to identify the opinion holder. The performance of the system is calculated to be around 62% precision 37% recall. This result is not

comparable with other works seen so far as they do not deal exactly with the same kind of problem.

The work closest to ours is in [LiCC15] where the use of lexicons were also done. The difference is that they used emoticons constructed from messengers from Yahoo, Msn, Google and SentiStrength and words from ANEW, PANAS-X, SENTICNET, LIWC, Sentiment Strength and SentiWordNet to create a knowledge base used in a first stage classifier. Only the tweets that did not have any tokens from the emoticons and affective words list would be sent to a second stage Machine Learning classifier. Although the first stage classifier had a mean accuracy of 82.88% in different datasets, its average coverage was only 11.53%. The final classifier that uses the 2-step procedure achieved a maximum F1-Score of 72.4.

Paper	**Learning Method**	**Lexicons**	**Other Features**	**Results**
Go et al. 2009	SVM, NB, Maximum Entropy	---	Unigrams, bigrams	Accuracy (2 classes) 78.8% - 83.0%
Pak and Paroubek 2010	SVM	---	Unigrams, bigrams, trigrams, negation	Accuracy (3 classes) around 0.65
Zhang et al. 2011	SVM	Wordnet Expanded	Unigrams, negation	Accuracy (3 classes) 85% F1-Score (2 classes) 0.749
Jiang et al 2011	SVM	General Inquirer	Unigrams, emoticons, punctuation, prior subjectivity, polarity	Accuracy (2 classes) 85.6% Accuracy(3 classes) 68.3%
Maynard and Funk (2012)	Rules	Own dictionary	base and hashtags, detection of opinionated statements that included question marks, negatives, entity recognition, subject definition	---
Lima et Al 2015	NB SVM J48 KNN	ANEW PANAS-X SENTICNET LIWC SentiWordNet Sentiment Strength	n-grams, POS, lemmas	F1-score 0.724

Table 3-6 Summary of the other works on Sentiment Analysis using affective lexicons.

3.7. Foundations Summary

In this chapter, it was presented the problem of Sentiment Analysis and why it is still relevant today, specially in the context of short messages like Twitter's. It also described the techniques commonly used in the task of sentiment analysis of texts as well as the works done in the SemEval competitions that represent the state of the art in the area.

The next chapter will cover the creation of a base model and the study of each lexicon individual contribution to the SA pipeline.

Chapter 4

Base pipeline and Lexicon's features

This study will use a machine learning approach to the SA task. That means that the original tweet will become a feature matrix fed into a Machine Learning algorithm creating a model that maps the input data to the output value of the twitter sentiment analysis. The performance of each generated model will be measured using the F1-macro score, as explained in section 3.1.

To create a more robust Sentiment Analysis pipeline and to understand what are the best features that can contribute to the model performance, the following macro procedure was used:

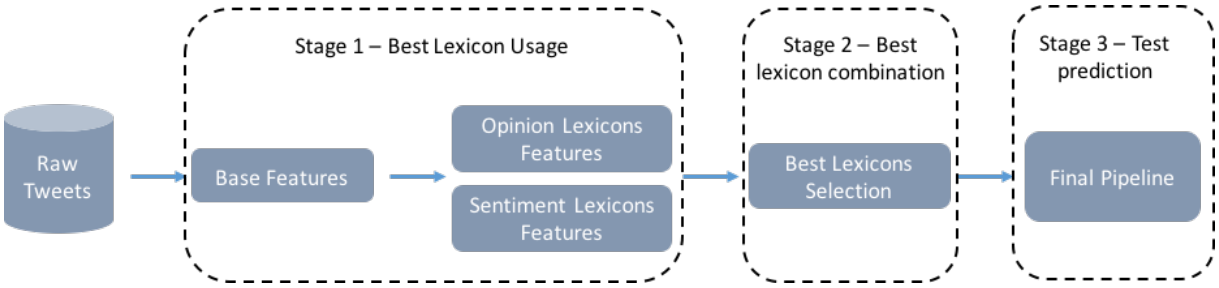

Figure 4-1 Overview of the study pipeline.

As shown in Figure 4-1, the pipeline is a 3-step process. First, using the training and development dataset provided by SemEval 2013 some non-lexicon based features are created (BOW, negated words, etc.) forming a base feature set. Then the opinion and sentiment lexicons are tested individually. For each opinion and sentiment lexicon, different ways of using the lexicon is tested merging the newly created features with the base features created in

the previous step. This features dataset is then tested with many of the machine learning algorithms to find the best features/algorithms for each of the lexicons.

On stage 2, a Genetic Algorithm is used to search heuristically for the most performant combinations of lexicons. The final result of this step is a list of lexicon combination and their individual performance. This enables to pick the best combination that maximizes the F1-score for this dataset.

Finally, in the third step, the training and development datasets are merged and the features are recreated using the same procedures used in the first step and the best combination found in step 2. This newly created train dataset is used to make the final model that will predict the sentiment of the test dataset provided.

The afore mentioned text processing and machine learning tasks were performed using well-known Python libraries namely NLTK[1] [BiKL09] and Scikit-learn[2] [PeVa11]. All the code can be found in a Github[3] repository provided by the author.

In the next sections the creation process of the base features and each lexicon best features will be presented. Chapter 5 will demonstrate the search procedure used to find the best combination of the lexicons and how they perform in the test dataset of SemEval 2014 and 2015 dataset and Chapter 6 will present the final conclusions.

4.1. Raw Tweets

The SemEval 2013 task B dataset was used as the base to develop the whole pipeline. The main reason for this is that the SemEval works done in 2013, 2014 and 2015 all use this dataset, enabling comparison between works.

Although the dataset is open and free for use, the text content of each tweet is property of Tweeter and was not given by the competition organizer. It is up to the interested user to download the content directly from the website using a download script provided.

Three dataset files were given containing the *uid* and *sid* of individual tweets and the sentiment associated to each of them. The first is a train file, that was used to train the model, a development (dev) file used to evaluate the performance of the trained model and finally, a test file that should have its sentiment predicted. By the time of the competition, the test file

[1] http://www.ntlk.org
[2] http://scikit-learn.org/
[3] https://github.com/AdrianoW/Twitter-sentiment-analysis-master

did not have the sentiment given as they were kept secret so that the organizers could assess the teams' models independently.

In this work, we have used the test file only at the final pipeline to guarantee that the decisions were not biased towards best results on the test set. This emulates a real-world scenario where the polarities of the tweets are unknown and the SA pipeline is used to predict it.

The biggest problem with downloading content directly from Twitter is that users may exclude their posts. This causes the dataset final size to change over time and make the comparison between models created in different works not precise as each of them have different dataset sizes. The tables below present the original dataset constitution and the number of tweets used in the top 3 works of SemEval 2013 as well as this work. This will influence the final result but it is impossible to say whether it affects positively or negatively as the dataset used in the previous works are not available.

Work	Train	Dev	Test
SemEval 2013 Task 2 (Nakov, P. et al, 2013)	9728	1654	3814
NRC CANADA (Kiritchenko et al, 2014)	8258	1654	3813
GU-MLT-LT (Gunther et al, 2013)	10368	---	3813
TeraGram (Reckman, H., Baird, C., Crawford, J.,et.al. 2013)	---	---	3813
This Work	8171	1405	3239

Table 4-1 Dataset sizes compared

The final constitution of this work dataset and the difference to the original proposed dataset is shown on table 4-2. The lost in average was around 15% but the negative set of tweets had the biggest lost. This can hurt the model performance if compared to other works as the short number of negative examples make harder for the classifier to properly learn this class.

Corpus	Positive	Negative	Neutral	Total
Training	3045	1197	3929	8171
Dev	489	284	632	1405
Test	1313	488	1438	3239

Corpus	Positive	Negative	Neutral	Total
Training	17%	18%	15%	16%
Dev	15%	16%	14%	15%
Test	17%	19%	12%	15%

Table 4-2 Final dataset distribution and the percent lost

4.2. Base Features

As explained in Chapter 3, the Sentiment Analysis models that achieve better performance at this task are the hybrid ones. It means that the features created are a mix of regular semantic features like BOW, number of POS tags, count of positive and negative emoticons, and lexicon created features like the polarity of individual words or the total count or sum of all individual polarities. This is not a solved problem as each study uses the lexicon in different ways, creating different features.

The consensus as shown in the many works done for the SemEval competition is that after having a base set of features, the lexicon created features add power to the models. This section describes the procedure used to test for a lexical set of features that enable a good starting point for the model. The final result of this part is what will be called the *Base Features*.

For all the following tests a Stratified Cross Validation of 10 folds was used. Figure 4-2 displays the process in more details using as example a dataset with two classes. The same idea could be extrapolated to a 3-class dataset.

The dataset is split in 10 pieces and for each fold, a different piece is chosen as the validation dataset being left out of the training dataset. The main objective is to do data augmentation having 10 independent results that can capture the behavior of the model in different datasets.

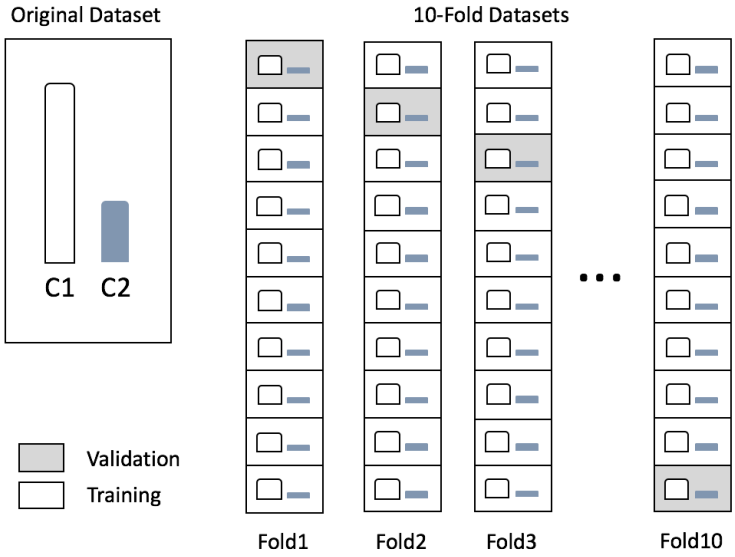

Figure 4-2 Explanation of the 10-fold cross validation

Each fold was scored according to the F1-Score used in the SemEval 2013 task B. The results shown are the average and the standard deviation of the 10-fold models.

4.2.1. Bag of words model

The first characteristic tested was the use of a BOW (bag of words) with count of words (frequency) or a simple Boolean indicating the presence or absence of a word.

Notice in Figure 4-3 for example the word *calling*. The first table has a value of 2 in the first line, meaning that there was on the first tweet 2 appearances of this word while the second line has a value of 0, indicating that this word was not present in this tweet. In contrast, the second table has a value of 1 for the first tweet in the same word *calling*, indicating that this word was present but the number of times is not important.

The SemEval training dataset generated a BOW model of 21.143 tokens where each token is a feature that will be used by the machine learning model.

```
id:264187448853151744 | sentiment:0 | text: Calling all voters, I'm cal
ling on Nevada,Florida,Ohio,Tuesday night is the night u guys will i re
peat will shine for the president lets go!

id:254941790757601280 | sentiment:-1 | text: They may have a SuperBowl
in Dallas, but Dallas ain't winning a SuperBowl. Not with that quarterb
ack and owner. @S4NYC @RasmussenPoll
```

	ain	all	and	but	calling	dallas	florida	for	go	guys	...	shine	superbowl	that	the	they	tuesday	voters	will	winning
264187448853151744	0	1	0	0	2	0	1	1	1	1	...	1	0	0	2	0	1	1	2	0
254941790757601280	1	0	1	1	0	2	0	0	0	0	...	0	2	1	0	1	0	0	0	1

	ain	all	and	but	calling	dallas	florida	for	go	guys	...	shine	superbowl	that	the	they	tuesday	voters	will	winning
264187448853151744	0	1	0	0	1	0	1	1	1	1	...	1	0	0	1	0	1	1	1	0
254941790757601280	1	0	1	1	0	1	0	0	0	0	...	0	1	1	0	1	0	0	0	1

Figure 4-3 Two example tweets with frequency and boolean versions of the features

To check the difference between the two types of BOW, the features created were used as the input dataset of Scikit-learn's Linear SVC with a 10-fold cross validation. The results on Table 4-3 demonstrated a better performance on the train dataset for the Boolean version while the frequency version had a better performance on the development dataset making its value even better than the one presented on the train dataset.

	train score	dev score	f1_dev_neg	f1_dev_neu	f1_dev_pos
Freq-LinearSVC	53.343 +- 3.181	**53.949 +- 1.449**	**45.151 +- 2.203**	**66.011 +- 0.562**	**62.747 +- 1.055**
Bool-LinearSVC	**54.213 +- 3.777**	52.658 +- 1.076	43.234 +- 1.355	65.671 +- 0.825	62.082 +- 1.047

Table 4-3 Frequency and Boolean BOW

An interesting consideration is that the F1 for the negated part of the development dataset is where the biggest hit is. That means that our pipeline is not able to handle as well negative tweets as it is able to handle positive and neutral ones. Another consideration is that the frequency BOW resulted in a better average F1-score.

Section 4.2.3 will dive deeper and test the BOW model in conjunction with different machine learning models to decide which one will be best.

4.2.2. Tokenizer

The tokenizer has also an importance on the SA. Properly separate and clean the tokens is important because they will serve as the index to find the polarities of the words in the lexicons.

Scikit-learn tokenizes the text automatically when doing the BOW using frequency or boolean. The default behavior strips important information in the twitter SA pipeline like hashtags. To properly separated the words into tokens without losing information, the following tokenizers were tested:

- Word Tokenizer – parse the text using a regular expression that identifies words. It is the default behavior of the counter tokenizer previously shown;
- NLTK Casual – uses a set of regular expressions that were tuned to properly parse tweets with its particular properties like emoticons, hashtags, etc.
- CMU ARK Twitter POS Tagger – This is the tokenizer created by the Carnegie Mellon. Additional details can be found in [OODG13]

The Figure 4-4 shows a fake test tweet created to show the main differences between the tokenizers. In Table 4-4 the tokens create by each tokenizer is shown. Notice how different tokens where created. The simple *word* tokenizer has no mentions (@mention) and hashtags (#hashtext) and it splits the URL address, while *NLTK Casual* kept the hashtag and

the url and deleted the mention and the CMU *ARK* kept the whole tweet respecting mentions, emoticons, hashtags and urls.

```
This is a TEST tweet #hashtext @mention http://www.testurl.com $89.99 :
) >:(
```

Figure 4-4 Sample tweets

ARK	NLTK Casual	Word
#hashtext	#hashtext	89
$89.99	$	99
:)	89.99	com
>:(	:)	hashtext
@mention	>:(	http
a	a	is
http://www.testurl.com	http://www.testurl.com	mention
is	is	test
test	test	testurl
this	this	this

Table 4-4 Tokenization of the example tweet

Finally, to check if the different tokenization would influence the machine learning model, each of the three tokenizers was used to create a dataset that was used to create a different model. The word tokenizer generated 21.143 features, the NLTK generated 19.835 and the ARK created 23.387 tokens.

Table 4-5 shows that the ARK and the NLTK tokenizer had almost the same performance, with a small advantage in the development dataset to the ARK tokenizer, while the Word tokenizer was behind. Due to be more complete and maintain more varied tokens, the ARK tokenizer was chosen to be part of the baseline. An independent t-test demonstrates that the results of the 10-fold cross-validation over the development dataset were significant when comparing the values achieved using the tokenizers ARK and NLTK against tokenizing using the words only. The p-values shown in Table 4-6 are below the significant level of 0.05, indicating that the differences are meaningful and very unlikely to happen by chance.

	train score	dev score	f1_dev_neg	f1_dev_neu	f1_dev_pos
Word-LinearSVC	53.343 +- 3.181	53.949 +- 1.449	45.151 +- 2.203	66.011 +- 0.562	62.747 +- 1.055
NLTK-LinearSVC	**55.890 +- 3.338**	55.862 +- 1.106	**48.127 +- 1.874**	66.753 +- 0.668	63.597 +- 0.926
ARK-LinearSVC	55.663 +- 3.295	**56.187 +- 0.875**	47.785 +- 1.467	**67.401 +- 0.484**	**64.589 +- 0.826**

Table 4-5 Comparision of three tokenizers

	t-value	p-value	significant
NLTK-LinearSVC	3.317140	0.003833	True
ARK-LinearSVC	4.180524	0.000562	True

Table 4-6 T-test to check if the changes were significant

4.2.3. Testing Algorithms

Another important step in any pipeline is to choose the algorithm that can produces the model with the best performance in the given dataset.

Twelve different algorithms were tested using a 10-fold cross validation as described in section 0. They were chosen due to good performance presented in previous paper or due to the different nature of its composition, like trees and boost. All algorithms were part of the Scikit-learn library with exception of the XGBoost presented on [ChGu16] that is a standalone library.

To make a second check on how the creation of the features presented in 4.2.1 would affect this problem, 3 different features creation methods were used: Boolean, counting and counting then normalizing using the Max Absolute scaler. The scaler dives all the values in a feature by its biggest value, making the values between -1 and 1.

Tables Table 4-7, Table 4-8 and Table 4-9 demonstrate the performance of the algorithms using the Ark Tokenizer with Boolean, count and count plus normalization as described above. The field “train score” and “dev score” represents the average performance of the model in the 10-fold cross-validation while the “f1_dev_neg”, “f1_dev_neu” and “f1_dev_pos” show the performance in the negative, neutral and positive classes. The first

column is the name of the algorithm class in the Scikit-learn library. For details of the individual implementation of each, please refer to the library's documentation[1].

The best performance was achieved using the ARK tokenizer and count only. The XGBoost that is a great contender in many machine learning challenges (Kaggle[2]) performed very poorly in this task. A simple perceptron had also a poor performance.

	train score	dev score	f1_dev_neg	f1_dev_neu	f1_dev_pos
KNeighborsClassifier	24.936 +- 3.172	24.712 +- 1.363	17.415 +- 1.011	60.939 +- 0.459	32.009 +- 2.205
DecisionTreeClassifier	46.668 +- 3.215	47.340 +- 1.443	35.584 +- 1.861	63.036 +- 1.376	59.097 +- 1.377
RandomForestClassifier	40.537 +- 2.735	38.533 +- 1.416	21.432 +- 1.916	66.640 +- 0.776	55.634 +- 1.724
AdaBoostClassifier	50.949 +- 4.006	50.621 +- 1.001	39.764 +- 1.459	69.035 +- 0.666	61.479 +- 0.962
GaussianNB	38.211 +- 1.881	35.138 +- 0.528	27.413 +- 0.925	45.689 +- 0.893	42.862 +- 0.869
LogisticRegression	**55.758 +- 4.135**	54.741 +- 0.849	44.421 +- 1.560	**70.210 +- 0.401**	**65.062 +- 0.590**
SGDClassifier	52.474 +- 3.863	53.398 +- 1.626	44.046 +- 3.611	68.595 +- 1.728	62.750 +- 0.824
RidgeClassifier	53.585 +- 2.763	53.517 +- 1.068	44.984 +- 1.696	67.105 +- 0.802	62.051 +- 0.747
MultinomialNB	40.501 +- 1.849	37.604 +- 0.796	15.630 +- 1.423	64.306 +- 0.658	59.579 +- 0.461
LinearSVC	55.253 +- 3.359	**55.268 +- 1.125**	**46.546 +- 2.026**	67.277 +- 0.390	63.990 +- 0.864

Table 4-7 Comparision of 10 algorithms using Ark Tokenizer and boolean features.

	train score	dev score	f1_dev_neg	f1_dev_neu	f1_dev_pos
KNeighborsClassifier	28.729 +- 3.452	31.225 +- 0.630	23.214 +- 1.244	60.215 +- 0.543	39.236 +- 1.428
DecisionTreeClassifier	45.993 +- 3.652	46.509 +- 1.739	33.666 +- 2.510	63.347 +- 1.055	59.353 +- 1.730
RandomForestClassifier	40.029 +- 3.800	37.236 +- 1.677	19.248 +- 2.856	66.140 +- 0.975	55.224 +- 1.753
AdaBoostClassifier	51.587 +- 3.647	50.383 +- 0.746	39.479 +- 1.398	68.778 +- 0.580	61.287 +- 0.941
GaussianNB	38.221 +- 1.926	35.119 +- 0.592	27.566 +- 1.021	45.492 +- 0.865	42.672 +- 0.814
LogisticRegression	55.161 +- 3.987	55.356 +- 0.901	45.893 +- 1.562	**69.922 +- 0.485**	**64.818 +- 0.580**
SGDClassifier	54.432 +- 2.162	54.256 +- 0.956	44.547 +- 2.226	67.595 +- 0.952	63.964 +- 0.945
RidgeClassifier	53.572 +- 2.490	54.398 +- 1.158	45.887 +- 1.742	67.947 +- 0.797	62.908 +- 1.038
MultinomialNB	40.160 +- 1.271	37.503 +- 0.746	16.879 +- 1.202	62.836 +- 0.788	58.128 +- 0.571
LinearSVC	**55.663 +- 3.295**	**56.187 +- 0.875**	**47.785 +- 1.467**	67.401 +- 0.484	64.589 +- 0.826

Table 4-8 Comparision of 10 algorithms using Ark Tokenizer and count

[1] http://scikit-learn.org/stable/modules/classes.html
[2] www.kaggle.com

	train score	dev score	f1_dev_neg	f1_dev_neu	f1_dev_pos
KNeighborsClassifier	13.018 +- 2.119	12.405 +- 1.388	9.555 +- 2.687	62.704 +- 0.255	15.256 +- 1.445
DecisionTreeClassifier	45.993 +- 3.652	46.509 +- 1.739	33.666 +- 2.510	63.347 +- 1.055	59.353 +- 1.730
RandomForestClassifier	40.029 +- 3.800	37.236 +- 1.677	19.248 +- 2.856	66.140 +- 0.975	55.224 +- 1.753
AdaBoostClassifier	51.587 +- 3.647	50.383 +- 0.746	39.479 +- 1.398	68.778 +- 0.580	61.287 +- 0.941
GaussianNB	38.208 +- 1.965	35.064 +- 0.618	27.450 +- 0.978	45.761 +- 0.854	42.678 +- 0.770
LogisticRegression	54.424 +- 2.292	54.153 +- 0.748	42.924 +- 1.578	**70.099 +- 0.531**	**65.381 +- 0.460**
SGDClassifier	54.432 +- 2.162	54.256 +- 0.956	44.547 +- 2.226	67.595 +- 0.952	63.964 +- 0.945
RidgeClassifier	52.903 +- 2.249	53.830 +- 1.043	44.646 +- 2.052	67.418 +- 0.880	63.013 +- 0.692
MultinomialNB	40.160 +- 1.271	37.503 +- 0.746	16.879 +- 1.202	62.836 +- 0.788	58.128 +- 0.571
LinearSVC	**55.488 +- 2.588**	**55.618 +- 0.911**	**46.658 +- 1.828**	67.302 +- 0.687	64.578 +- 0.624

Table 4-9 Comparision of 10 algorithms using Ark Tokenizer and count features and normalization.

The t-test shows that there is no statistical difference between using the boolean or the frequency while the normalization has worst performance. Considering the count of words, Table 4-10 shows that the algorithms LogisticRegression and LinearSVC have statistical similar values (significant column is False) while the others are worst.

	t-value	p-value	significant
KNeighborsClassifier	-73.238714	9.726461e-24	True
DecisionTreeClassifier	-15.722295	5.853280e-12	True
RandomForestClassifier	-31.681587	3.056330e-17	True
AdaBoostClassifier	-15.967045	4.511517e-12	True
GaussianNB	-63.083690	1.414042e-22	True
LogisticRegression	-2.093742	5.070794e-02	False
SGDClassifier	-4.714273	1.729578e-04	True
RidgeClassifier	-3.899778	1.049935e-03	True
MultinomialNB	-51.391433	5.552300e-21	True
LinearSVC	-0.000000	1.000000e+00	False

Table 4-10 T-test between the Ark-LinearSVC (best model in section 4.2.3) and 10 algorithms

4.2.4. Dimensionality reduction

One of the biggest issues with working with Machine Learning on NLP is that the number of features that are created are usually very big. This happens as the BOW model creates one feature for each of the words it finds in the text.

Using the ARK tokenizer, the training dataset has a dimension of 8171 lines by 23387 features. In other words, the number of features is almost four times bigger than the number tweets.

Four different approaches were tested to reduce the number of features:

- Create a list of stop words to be removed from the dataset.
- Filter some part of speech tags as in [KSTA15] keeping only the noun, verb, adverb, interjection, emoticon, abbreviations, foreign words, possessive endings.
- Normalizing tokens. Mentions, hashtags, urls, numbers were substituted by fixed words. For example, the mentions @user1 and @user2 were replaced by MENTION word.
- Automatic Token selection using correlation between the labels and the features. ANOVA f-value, mutual information for classification and chi-squared were tested.

The approach of removing stop words was done in two steps: first a list of how many times each word appeared was done and then the 300 most common were used as stop words. That did not yield better results. In a second step, some stop words were handpicked according to how meaningful they were. The final list is composed of the words shown in the Figure 4-5. A list of 153 stop words present in the NTLK library was also used as a benchmark. Table 4-11 depicts the results with manually selected tokens while Table 4-13 shows the results with the NTLK's big list of stop words. Notice that the smaller selection yielded better results.

Stop words:

the	to	in	on	and	of	a	for	at	with	be	it	that	-	this	,	.	an

Figure 4-5 Stop words selected

	train score	dev score	f1_dev_neg	f1_dev_neu	f1_dev_pos
KNeighborsClassifier	30.350 +- 2.812	25.991 +- 1.001	14.758 +- 1.844	60.597 +- 1.022	37.224 +- 1.321
DecisionTreeClassifier	48.724 +- 3.603	49.293 +- 1.271	37.898 +- 2.165	64.701 +- 0.958	60.688 +- 1.246
RandomForestClassifier	42.926 +- 3.654	41.762 +- 2.150	25.070 +- 3.961	67.334 +- 0.993	58.454 +- 1.505
AdaBoostClassifier	51.261 +- 3.679	50.063 +- 0.927	39.099 +- 1.558	68.897 +- 0.883	61.026 +- 0.680
GaussianNB	38.221 +- 1.926	35.119 +- 0.592	27.566 +- 1.021	45.492 +- 0.865	42.672 +- 0.814
LogisticRegression	55.218 +- 3.311	55.666 +- 0.778	46.229 +- 1.320	**70.041 +- 0.526**	**65.104 +- 0.594**
SGDClassifier	54.373 +- 2.983	54.563 +- 1.155	45.097 +- 2.277	67.476 +- 0.917	64.029 +- 0.472
RidgeClassifier	53.260 +- 2.853	54.501 +- 0.948	45.966 +- 1.667	67.836 +- 0.576	63.036 +- 0.697
MultinomialNB	41.579 +- 1.469	39.345 +- 0.732	20.486 +- 1.505	62.675 +- 0.730	58.203 +- 0.589
LinearSVC	**55.333 +- 3.294**	**56.084 +- 0.917**	**47.742 +- 1.806**	67.470 +- 0.792	64.426 +- 0.651

Table 4-11 Results using a set of manually picked stopwords.

	train score	dev score	f1_dev_neg	f1_dev_neu	f1_dev_pos
KNeighborsClassifier	27.941 +- 3.278	26.574 +- 0.867	18.788 +- 1.006	59.858 +- 0.670	34.360 +- 1.506
DecisionTreeClassifier	48.415 +- 4.294	49.736 +- 1.461	37.601 +- 2.345	65.881 +- 0.749	61.872 +- 1.398
RandomForestClassifier	45.691 +- 4.500	44.181 +- 1.428	28.500 +- 3.085	68.006 +- 0.638	59.862 +- 1.125
AdaBoostClassifier	49.788 +- 3.839	49.596 +- 0.992	37.272 +- 1.776	68.973 +- 1.563	61.920 +- 0.854
GaussianNB	38.153 +- 1.951	35.324 +- 0.659	27.983 +- 1.123	45.332 +- 0.881	42.664 +- 0.824
LogisticRegression	53.046 +- 2.625	54.393 +- 0.915	43.548 +- 1.493	**70.388 +- 0.596**	**65.237 +- 0.606**
SGDClassifier	51.987 +- 1.647	54.043 +- 1.298	44.381 +- 2.467	67.717 +- 0.831	63.706 +- 0.734
RidgeClassifier	52.686 +- 2.456	53.479 +- 1.128	44.656 +- 1.825	68.520 +- 0.501	62.303 +- 0.621
MultinomialNB	42.563 +- 1.664	39.354 +- 1.140	20.454 +- 2.141	63.005 +- 0.813	58.255 +- 0.861
LinearSVC	**54.281 +- 3.258**	**55.161 +- 0.893**	**46.352 +- 1.781**	67.761 +- 0.404	63.969 +- 0.784

Table 4-12 Results of using the NTLK stop words

Following the work done in [KSTA15], some POS tags were filtered but that did not improve the performance as shown in Table 4-13.

	train score	dev score	f1_dev_neg	f1_dev_neu	f1_dev_pos
KNeighborsClassifier	20.277 +- 3.471	17.781 +- 3.599	11.503 +- 1.144	56.571 +- 4.659	24.060 +- 6.664
DecisionTreeClassifier	45.998 +- 2.720	44.848 +- 1.285	35.271 +- 1.838	60.626 +- 1.180	54.425 +- 1.787
RandomForestClassifier	45.413 +- 3.359	44.997 +- 1.527	34.198 +- 2.586	62.791 +- 1.096	55.797 +- 1.225
AdaBoostClassifier	43.661 +- 2.957	42.162 +- 1.737	33.104 +- 3.508	66.952 +- 0.384	51.221 +- 0.871
GaussianNB	36.041 +- 2.175	36.988 +- 0.631	30.537 +- 0.816	32.127 +- 0.901	43.440 +- 0.815
LogisticRegression	48.953 +- 2.192	49.745 +- 0.746	40.665 +- 1.241	**67.307 +- 0.469**	**58.824 +- 0.428**
SGDClassifier	**49.287 +- 2.530**	50.186 +- 1.164	**43.089 +- 1.785**	65.207 +- 0.529	57.284 +- 0.920
RidgeClassifier	48.615 +- 1.807	49.810 +- 1.056	42.727 +- 1.980	65.479 +- 0.615	56.892 +- 0.471
MultinomialNB	44.752 +- 2.601	39.324 +- 0.863	23.075 +- 1.646	62.012 +- 0.575	55.573 +- 0.372
LinearSVC	49.232 +- 1.518	**50.447 +- 0.877**	43.057 +- 1.619	63.932 +- 0.731	57.838 +- 0.583

Table 4-13 Results filtering the POS tags.

Another step taken in the direction of dimensionality reduction was to substitute (normalize) the mentions (@username), url, numbers and hashtags(#text) for a single token that would represent the element type. For example, all the mentions were substituted by the token *@MENTION* while all the urls were replaced by *URL*. As it can be seen in Table 4-14, this operation helped the model and raised a little its performance to *0.565* in the development dataset.

	train score	dev score	f1_dev_neg	f1_dev_neu	f1_dev_pos
KNeighborsClassifier	28.586 +- 1.958	30.102 +- 0.891	21.255 +- 1.031	60.705 +- 0.482	38.949 +- 1.780
DecisionTreeClassifier	47.763 +- 3.141	47.901 +- 1.770	36.609 +- 2.427	63.945 +- 1.069	59.194 +- 1.636
RandomForestClassifier	40.327 +- 5.081	38.286 +- 2.147	21.649 +- 3.783	66.493 +- 0.966	54.923 +- 2.483
AdaBoostClassifier	51.146 +- 3.768	49.689 +- 0.895	37.885 +- 1.668	68.650 +- 0.325	61.493 +- 0.481
GaussianNB	37.900 +- 1.693	34.999 +- 0.710	27.200 +- 0.945	45.644 +- 0.993	42.798 +- 0.925
LogisticRegression	**55.906 +- 3.494**	55.850 +- 0.921	46.269 +- 1.438	**69.979 +- 0.452**	**65.430 +- 0.806**
SGDClassifier	54.577 +- 2.127	55.302 +- 1.375	46.257 +- 2.320	68.085 +- 0.623	64.347 +- 0.878
RidgeClassifier	53.178 +- 2.507	53.573 +- 1.083	45.009 +- 1.901	67.036 +- 0.561	62.136 +- 0.708
MultinomialNB	43.777 +- 1.811	40.376 +- 0.480	22.444 +- 0.978	62.906 +- 0.720	58.308 +- 0.566
LinearSVC	55.361 +- 2.788	**56.535 +- 1.390**	**48.277 +- 2.195**	67.052 +- 0.443	64.793 +- 1.056

Table 4-14 Results normalizing tokens.

So far, using stop words and normalizing tokens improved by a small margin the train and dev scores. In all the test so far, the best models achieved came from the use of the LinearSVC, LogisticRegression and SGDClassifier algorithms.

The last step in the features reduction was to test for automatic feature selection. We expand the feature selection idea presented in [PCEP16] using three different correlation metrics, ANOVA, Mutual Information and Chi-squared.

To have a better idea of what is the best selection, an incremental reduction plot was used. For each of the measures, 10 different percentage levels of the most important features were selected: 1%, 3%, 5%, 15%, 20%, 30%, 40%. 60%, 80% and 100%. This enables a good view of how the number of features affects the final performance. The features used as base were the ones created on the normalized tokens steps with the stop words removed and the algorithm was the Logistic Regression as it had a better performance on some of the previous step.

The figures 4-6 to 4-8 shows the 3 metrics and how they behave with different percentages of feature selection. The results in Train and Dev dataset had similar behavior when varying the number of features. Notice the peak of performance using 3% and 5% with a drop on performance around 20% and 40% with a constant raise proportional with the raise in the number of features. We hypothesize that the feature reduction first chooses the important features, then adds noise to the selection and finally, with addition of a lot features, starts to overfit for the specific dataset.

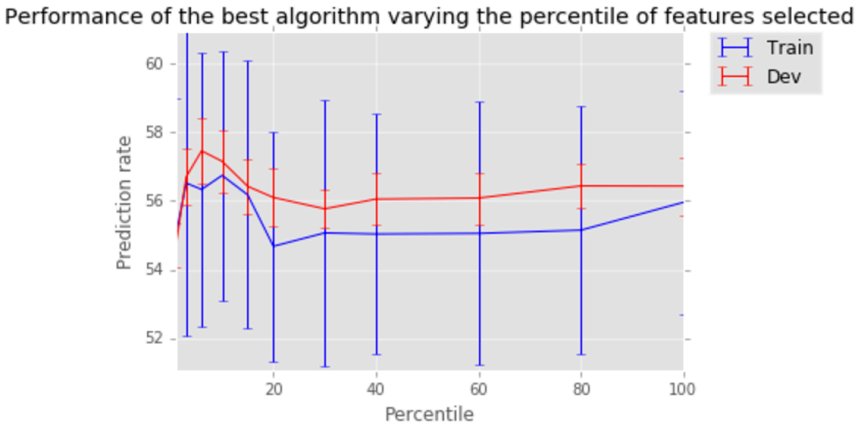

Figure 4-6 Performance varying feature selection percent – Metric ANOVA

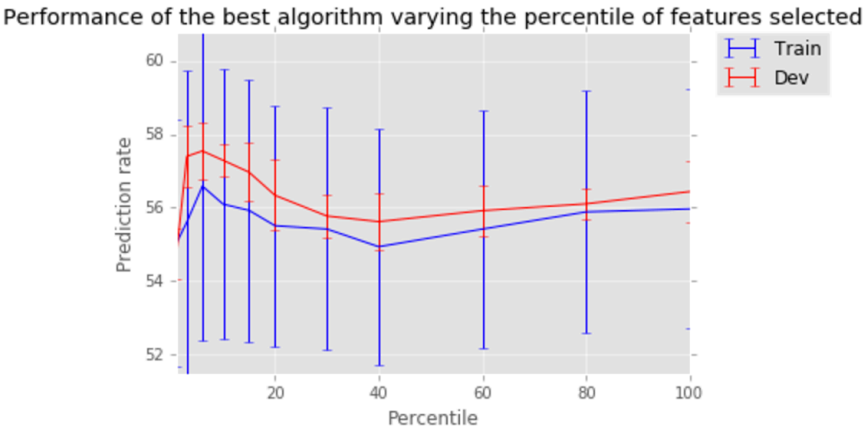

Figure 4-7 Performance varying feature selection percent – Metric Mutual Information

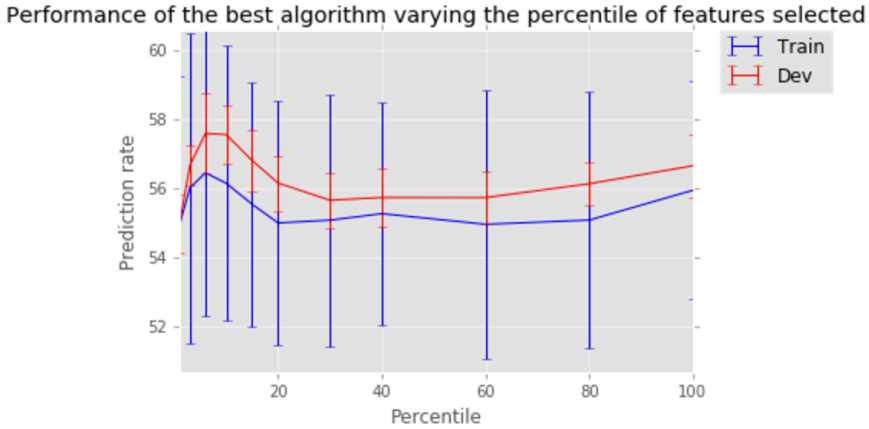

Figure 4-8 Performance varying feature selection percent – Metric Chi-squared

Table 4-15 to Table 4-17 show the numerical performance of the algorithms with the different levels of feature selection. It is presented the 4 best algorithms because in some of the selections a forth algorithm (RidgeClassifier) outperformed one of the 3 best so far. The *nan* values indicate that the algorithm was not in the top 3 for the specific percentage.

	1 train score	**1 dev score**	**3 train score**	**3 dev score**	**5 train score**	**5 dev score**	**15 train score**	**15 dev score**	**30 train score**	**30 dev score**
LinearSVC	54.9051	54.3207	**57.6365**	**57.2156**	**57.4029**	**57.5909**	55.3662	54.8462	53.3326	53.4439
LogisticRegression	**55.0241**	**54.7248**	56.5215	56.7042	56.343	57.4323	**56.1764**	**56.425**	**55.069**	**55.7713**
RidgeClassifier	53.7117	52.8605	nan	nan	nan	nan	nan	nan	nan	nan
SGDClassifier	nan	nan	55.6739	55.2222	55.482	56.0848	54.8955	55.1497	53.7305	54.8038

Table 4-15 Top 3 results – varying percentage – metric Anova

	1 train score	1 dev score	3 train score	3 dev score	5 train score	5 dev score	15 train score	15 dev score	30 train score	30 dev score
LinearSVC	**54.8079**	**54.69**	**56.419**	57.539	**56.9161**	56.8903	55.8865	56.0734	53.4075	54.3607
LogisticRegression	54.7833	54.6564	55.8968	**57.6001**	55.725	**57.3799**	**56.0486**	**56.8877**	**55.4322**	**55.841**
RidgeClassifier	53.6813	53.1529	54.3472	55.4842	54.3857	55.4698	nan	nan	nan	nan
SGDClassifier	nan	nan	nan	nan	nan	nan	55.4044	55.161	54.0491	54.6821

Table 4-16 Top 3 results – varying percentage – metric Mutual Information

	1 train score	1 dev score	3 train score	3 dev score	5 train score	5 dev score	15 train score	15 dev score	30 train score	30 dev score
LinearSVC	54.5841	54.6174	**56.7399**	**56.8383**	**57.4336**	**58.1078**	55.4298	54.9299	53.6294	53.6973
LogisticRegression	**54.9625**	**55.0023**	56.0229	56.6766	56.5265	57.3536	**55.5482**	**56.8154**	**55.0885**	**55.6663**
RidgeClassifier	53.2899	53.226	54.7664	54.9019	55.0198	56.0844	nan	nan	nan	nan
SGDClassifier	nan	nan	nan	nan	nan	nan	55.148	55.6794	53.5037	54.4632

Table 4-17 Top 3 results – varying percentage – metric Chi-squared

The best result found for the features selection was using the metric Chi-squared at 5% selection. From the results achieved so far, it can also be seen that 3 algorithms work better for this dataset: Linear SVC, LogisticRegression and SGDClassifier. The RidgeClassifier is this last 3 tests were close to the SGDClassifier but when the dimensionality grows it loses its prediction power.

In summary, the best dimensionality reduction setup is the removal of selected stop words followed by the normalization of the url, mentions, numbers and hash ending with the automatic selection using the chi-squared measure. The t-test presented in Table 4-18 shows that there is a statistically significant difference in the model's performance before and after the feature selection process.

	t-value	p-value	significant
LinearSVC	4.878484	0.000121	True
LogisticRegression	5.148076	0.000067	True
RidgeClassifier	3.733438	0.001521	True
SGDClassifier	3.471746	0.002722	True

Table 4-18 T-test for the dimensionality reduction

As the result achieved in this test was and improvement from previous step, this setup will be used as the base for the rest of the SA pipeline experiments. Also, only the 3

classifiers with best results will be compared for the remaining sections as the rest of the classifier do not add much to the experiment due to its low performance.

4.2.5. Bigrams and Trigrams

Bigrams and trigrams were tested to check if they could improve the pipeline. The number of features grows a lot when using bigrams (103.150) and trigrams (244.480). By the previous section, it could be seen that the results were better when less features were used. So auto feature selection was applied to the bigger dataset creating a dataset with 5% of the original size.

The performance demonstrated in Table 4-19 and Table 4-20 shows that the bigrams and trigrams can indeed improve performance in the *train* dataset. Comparing the performance of the train and development dataset, it can be seen that the bigrams and trigrams did not influence the performance of the *development* creating a big gap between both datasets. This means that these features only made the model to overfit and did not provide any real gain.

	train score	dev score
LogisticRegression	64.049465	56.421652
LinearSVC	66.271249	56.325996
SGDClassifier	65.471807	55.012354

Table 4-19 - Bigrams and automatic feature selection

	train score	dev score
LogisticRegression	61.020669	50.933941
AdaBoostClassifier	52.231244	50.153715
SGDClassifier	61.310259	48.297990

Table 4-20 - Trigrams and automatic feature selection

4.2.6. Finding the Best Parameters

So far, the best algorithms have been LogisticRegression, LinearSVC and SGDClassifier using feature selection with the metric Chi-square as shown on table Table 4-17. All the algorithms have been used with their default parameters.

In this step, a brute force method known as *grid search* is used in conjunction with cross validation of 10-fold (as shown in Figure 4-2) to find the best parameters for the three previously mentioned algorithms.

The idea is that all the parameters combinations are tested and the one with the best average f1-score in the development dataset is chosen as the configuration of the algorithm.

	train score	**dev score**	**f1_dev_neg**	**f1_dev_neu**	**f1_dev_pos**
LogisticRegression	66.298 +- 2.892	62.564 +- 0.645	56.416 +- 1.032	71.125 +- 0.582	68.711 +- 0.518
SGDClassifier	66.290 +- 3.265	**62.977 +- 0.684**	**56.492 +- 1.146**	**72.477 +- 0.260**	**69.463 +- 0.590**
LinearSVC	**66.624 +- 3.285**	61.988 +- 0.648	55.302 +- 1.166	72.279 +- 0.407	68.674 +- 0.430

Table 4-21 Newly achieved results, using the new parameters found with grid search.

As it can be seen in Table 4-22, tuning the parameters had a big improvement in the algorithm performance achieving an average gain of around 8.5%. This is a good raise if compared to the previous steps in the SA pipeline. The t-test in Table 4-23 shows that the improvement was statistically significant. Table 4-24 lists the parameters found by the grid search.

	Train			**Development**		
	No Tuning	Tuned	Difference	No Tuning	Tuned	Difference
LinearSVC	**57.434**	**66.624**	9.190	**58.107**	61.988	3.881
LogisticRegression	56.526	66.298	9.772	57.353	62.564	5.211
SGDClassifier	55.695	66.290	10.595	55.932	**62.977**	7.045

Table 4-22 Comparison before and after tuning

	t-value	**p-value**	**significant**
LogisticRegression	12.865161	1.632389e-10	True
SGDClassifier	16.500632	2.588051e-12	True
LinearSVC	14.122152	3.520324e-11	True

Table 4-23 T-test comparing the results before and afeter optimization

LogisticRegression	LinearSVC	SGDClassifier
C: 100 class_weight: balanced dual: False fit_intercept: True intercept_scaling: 1 max_iter: 300 multi_class: ovr penalty: l2 random_state: 9000 solver: sag	C: 0.5 class_weight: balanced dual: False fit_intercept: True intercept_scaling: 1 loss: squared_hinge max_iter: 1000 multi_class: ovr penalty: l2 random_state: 9000	alpha: 0.0001 average: False class_weight: balanced epsilon: 0.1 fit_intercept: True loss: hinge n_iter: 100 random_state: 9000 shuffle: True

Table 4-24 The best algorithm's parameters discovered by the search

4.2.7. Negation of words

Negation of words is another pre-treatment that is tested in many works. As it can be seen in [Coun10] the negation is an area of study itself in SA and there is not consensus where the negation can improve or decrease the performance of the SA pipeline.

Following Pott's tutorial[1], as used in [Pott11, ReSB15, TBTV11] negation tokens were created. The basic idea is that if a negation word is found, all the next tokens (words) are appended with a prefix "NEG_" until a stop punctuation is found. As a consequence, words that are in a negation phrase create a different token than when these words are in a positive sentence. This would enable the classifier to distinguish both situations.

Figure 4-9 shows the different tokens created using the negation token. Notice that 2 new tokens (*NEG_know* and *NEG_nothing*) were created and the token *know* has a small number of counts in the negated version.

[1] http://sentiment.christopherpotts.net/lingstruc.html#negation

Phrase: I know that I do not know nothing, but I try.

	I	know	that	do	not	nothing	but	try
no negation:	3	2	1	1	1	1	1	1

	I	know	that	do	not	NEG_know	NEG_nothing	but	try
w/ negation:	3	1	1	1	1	1	1	1	1

Figure 4-9 Tokens generated using and without using the negation context

With the new set of negated tokens, all the pipeline was run: normalizing, stop words removal, feature reduction with chi-squared as the metric and test with the best performant tuned algorithms.

	train score	dev score	f1_dev_neg	f1_dev_neu	f1_dev_pos
LogisticRegression	65.271 +- 3.896	59.838 +- 0.711	**50.796 +- 1.516**	71.400 +- 0.324	68.880 +- 0.377
SGDClassifier	65.505 +- 3.425	59.900 +- 0.595	50.354 +- 1.207	**73.150 +- 0.217**	69.446 +- 0.457
LinearSVC	**65.673 +- 3.932**	**60.005 +- 0.810**	50.520 +- 1.655	72.607 +- 0.245	**69.491 +- 0.292**

Table 4-25 Results with negation tokens

The results of the negated dataset (Table 4-25) are worse than the ones not using negation (Table 4-21). The biggest hit happened in the negation part of the development dataset where for all the algorithms used, the *f1_dev_neg* column had a decrease in value.

4.2.8. Brown Clustering

Following the work done by [GüFu13, MoKZ13], features related to the Brown Cluster were also tested. The idea is simple, for all word that is part of Brown's list, a new token with the cluster number is created. For example, the word *gas* will generate an additional token *11110101011011* that is its cluster number. More information related to Brown is found in Section 2.1.4.

In this work we use the Tweet NLP Ark from Carnegie Mellon that relies on the word cluster created in [Brow92] using 50M tweets collected from September of 2008 and August 2012. The use is relatively simple where words serve as keys to find the cluster number associated to them.

The final feature vector generated increased the number of features from 927 to 19021. As it can be seen in the previous experiments, highest dimensionalities hurt the model's performance. A reduction using the Chi-Squared was also applied to the 19k features creating a final vector of 951 features. Comparing Table 4-26 and Table 4-27 there is an improvement using features selection but comparing with the results from section 4.2.6, it did not improve the overall performance of the model using the development or training dataset.

	train score	dev score	f1_dev_neg	f1_dev_neu	f1_dev_pos
LogisticRegression	**59.044 +- 4.192**	**59.126 +- 0.870**	**51.566 +- 1.551**	68.380 +- 0.600	**66.687 +- 0.681**
SGDClassifier	56.871 +- 2.766	56.682 +- 1.729	47.781 +- 2.653	67.734 +- 0.534	65.583 +- 1.077
LinearSVC	58.279 +- 3.647	57.899 +- 1.400	49.586 +- 2.119	**68.536 +- 0.746**	66.212 +- 0.883

Table 4-26 Results with Brown clusters, no feature selection.

	train score	dev score	f1_dev_neg	f1_dev_neu	f1_dev_pos
LogisticRegression	**66.726 +- 3.165**	**61.081 +- 0.536**	56.056 +- 0.764	69.683 +- 0.555	66.106 +- 0.523
SGDClassifier	65.101 +- 4.176	60.937 +- 0.957	**56.526 +- 1.552**	71.958 +- 0.499	65.348 +- 0.689
LinearSVC	66.336 +- 3.511	60.899 +- 0.474	55.689 +- 0.809	**72.206 +- 0.429**	**66.110 +- 0.243**

Table 4-27 Results with Brown clusters, with feature selection.

4.2.9. Other pre-processing techniques

A final set of features was created that capture the overall number of occurrences of capitalization words, exclamations, question and exclamations phrases. This added 4 features to the original 927 features from section 0.

Once more, the changes did not improve the scores so they will not be used in the pipeline.

	train score	dev score	f1_dev_neg	f1_dev_neu	f1_dev_pos
LogisticRegression	66.275 +- 2.835	62.569 +- 0.683	56.384 +- 1.108	71.123 +- 0.614	68.754 +- 0.479
SGDClassifier	66.310 +- 3.103	**62.919 +- 0.699**	**56.573 +- 1.142**	**72.407 +- 0.698**	**69.264 +- 0.626**
LinearSVC	**66.624 +- 3.285**	61.988 +- 0.648	55.302 +- 1.166	72.279 +- 0.407	68.674 +- 0.430

Table 4-28 Result of the creation of 4 new features

4.2.10. Final best combination

In this topic, it was demonstrated the many features that were created and tested in the SA pipeline. As it can be seen, most of them did not improved the final performance. The Table 4-29 summarizes the features tested, the best scores achieved in the train and development dataset. The column Best Model summarizes the best final combination, the "+/-" signalizes whether this step helped or not the pipeline. If -, this step is ignored, if + this step is kept.

Step	Best Model (dev)	Algorithm	+/-	Best Train	Best Dev
Bag of words	Token Count	LinearSVC		53.343	53.949
Tokenizer	ARK Tokenizer	LinearSVC	+	55.663	56.187
Testing Algorithms	ARK Token - Token Count	LinearSVC	+	55.663	56.187
Dim. Reduction	Chi2 5% - Ment, url, number removal	LinearSVC	+	57.433	58.108
Bigrams & Trigrams	Bigram - Chi2 5%	LinearSVC	-	64.049	56.422
Finding the Best Parameters		**SGD Classifier**	**+**	**66.290**	**62.977**
Negation of Words	Linear SVC	LinearSVC	-	65.673	60.005
Brown Clustering	Chi2 5% - Log. Reg. - Feat. Selection	LogisticRegression	-	66.726	61.081
Other		SGD Classifier	-	66.310	62.919

Table 4-29 The summary of base feature creation.

Notice that the biggest differences came from the dimensionality reduction, the cleaning of the tokens and the tuning of the models. All these steps were responsible for nearly 16% of improvement over the original bag of words model with no tuning. The final model of the base features is able to generate and average F1-score of **66.296** in the training dataset and **62.977** in the development dataset.

For the next part of this chapter, we will use this newly created based features and pipeline as the base to test the lexicon features.

4.3. Opinion Lexicon's Features

Each of the Opinion Lexicon has a different set of words and the values associated with them. For example, the word *great* may have a value of 1 in one lexicon and have the value 5 in another.

Each of the 7 lexicons presented in the Chapter 2 were tested to find the best way to use them with the SemEval 2013 dataset. Three set of features were created:

- Tokens' Polarities: Each token found in the lexicon creates a feature with the lexicon name and the token. The value found in the Opinion lexicon is set as the value for this created feature. For example, the word *hate* was found in the Bing Lexicon, and has a value of -1. The feature created is BING_hate with value -1.
- Statistical Features:
    - Mean value of all the tokens found in a tweet (avg).
    - The minimum value found in a tweet (min).
    - The maximum value found in a tweet (max).
    - The sum of all values found in a tweet (sum).
    - The amount of negative words in a tweet (negamt).
    - The amount of positive words in a tweet (posamt).
- Final polarity: If the sum is positive, this feature receives the value +1, if the sum is 0, this value receives 0 else it receives the value -1 (fpol).

These features were created following the work done by [GüFu13]. Figure 4-10 demonstrate a set of statistical features generated for a single tweet using the Bing lexicon. Notice that in the example, the word like is negated and that generated a reversed polarity. The use of this technique will also be tested in the process described next.

To identify the features that adds more predictive power to the predictive model, the same machine learning pipeline with 10-fold cross validation was used as presented in session 0.

Figure 4-11 explains the lexicon verification process. The first step, the *Features Sets* part, features for the lexicon are created as shown in Figure 4-10. Each feature is tested alone merged with the best base features summarized in section 4.2.10 or together with the other features and the base features.

Tweet: Iranian general says Israel's iron dome can't deal with their missiles (keep talking like that and we may and up finding out)

Base Features:

iranian	general	says	israel's	iron	dome	can't	NEG_deal	NEG_with	NEG_their	NEG_missiles	NEG_(	NEG_keep	NEG_talking	NEG_like	NEG_that	NEG_and	NEG_we	NEG_may	NEG_end	NEG_up	NEG_finding	NEG_out	NEG_)
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Tokens Polarities

BING_NEG_like
-1

Lex Statistical:

BING_avg	BING_max	BING_min	BING_negamt	BING_posamt	BING_sum
-1	-1	-1	1	0	-1

Final Polarities:

BING_fpol
-1

Figure 4-10 Example of the features created for one tweet using Bing Lexicon with the polarity values found in it.

The next step, the *Machine Learning Pipeline*, the features matrixes created in the previous step are used to train a 10-fold cross validation model explained in section 0. The algorithms that will be used in the pipeline will still be only the 3 best from the previous sections.

Finally, in the *ML Pipeline Best Results* step, the results from the models are compared generating and *Overall Best* pipeline that makes use of the Lexicon.

Notice that due to the number of features generated when the *tokens polarities* feature set is used, automatic dimensionality reduction will also be used to find the best features generated by the Lexicon. For each of the 5%, 10%, 15%, 30% best features, a 10-fold cross validation will create the average F1-score of the models.

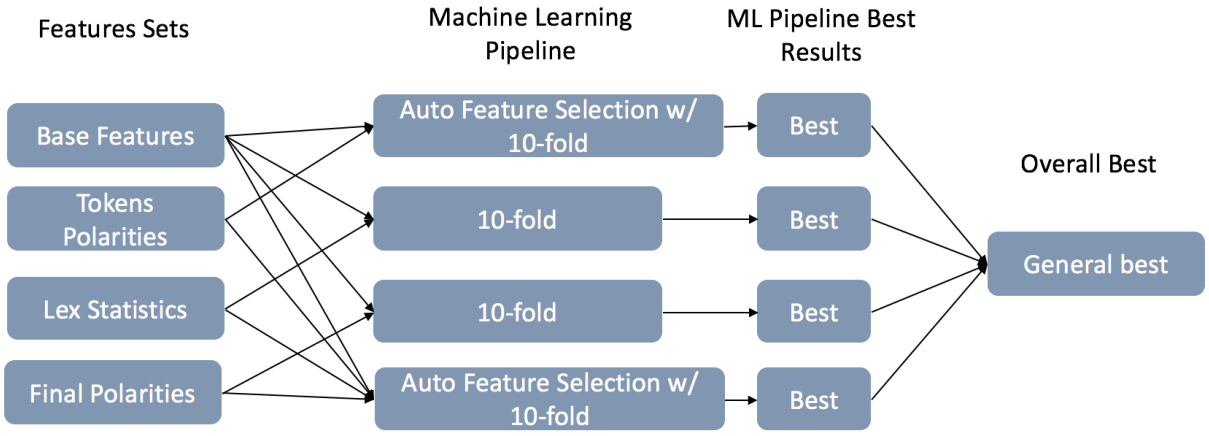

Figure 4-11 Pipeline used to identify the best features for each lexicon

To make the process clearer, take a hypothetical lexicon. Each of the tokens from the training dataset will be checked if they have a polarity value in the lexicon and the features are created as explained in Figure 4-10. Hypothetically, this created another 1500 features that will be joined with the 927 base features creating a training dataset of 8171 tweets with 2427 features. The same 1500 features will be created for the development dataset, generating a feature matrix of 1405 per 2427. Next, a 10-fold cross validation will be used to check which of the 3 algorithms produce a better average F1-score. Then, the development dataset is reduced to its 5% and checked against the 10-fold 3 algorithms pipeline, then the 10% and so on. Finally, all these results are compared creating the best pipeline for the tested lexicon. The best result can then be compared with the base results produced with just the base features and it can be stated that the lexicon helps or not in the SA task.

The next sessions will describe the best features found in each of the Opinion Lexicons and the performance achieved using the individual lexicon. The Bing Lexicon will be used as an example of the whole pipeline described in the hypothetical case above. For the rest of the lexicons, it will be shown only the results of the best parameters.

4.3.1. Bing Lexicon

This opinion lexicon is composed of just positive and negative values. The distribution displayed in Figure 4-12 shows that the lexicon is composed of almost twice the number of negative values than positive values.

The first step tested was to use just the statistic features (min, max, sum, avg, posamt and negamt) and Final Polarity without the Tokens' individual polarities. The negation of words was ignored. This created a training and development dataset of 933 features (927 original plus 6 lexical). Table 4-30 shows the results achieved.

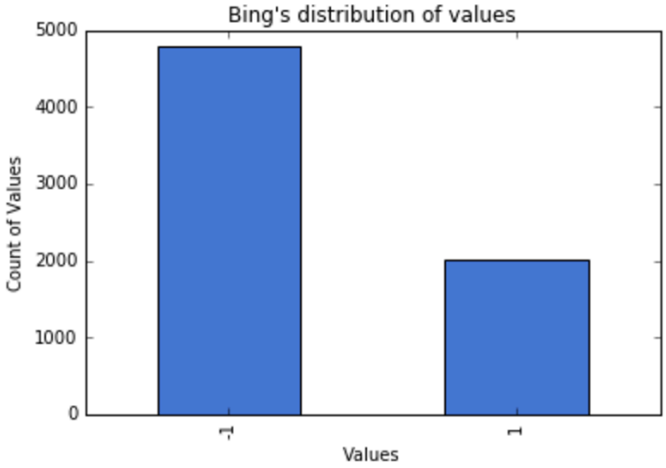

Figure 4-12 Distribution of the word values found inside the Bing Lexicon

	train score	dev score	f1_dev_neg	f1_dev_neu	f1_dev_pos
LogisticRegression	67.116 +- 2.826	64.703 +- 0.532	**60.752 +- 0.846**	71.451 +- 0.528	68.655 +- 0.712
SGDClassifier	**68.171 +- 2.824**	**65.031 +- 0.706**	60.691 +- 1.042	**72.823 +- 0.879**	**69.371 +- 0.725**
LinearSVC	67.837 +- 3.268	64.394 +- 0.470	59.991 +- 0.918	72.733 +- 0.471	68.797 +- 0.583

Table 4-30 Results of the 10-fold training for statistical features only for the Bing lexicon

Next, the same procedure is repeated but now considering the negation of the tokens when creating the tokens' polarities. A small improvement was achieved in both datasets.

	train score	dev score	f1_dev_neg	f1_dev_neu	f1_dev_pos
LogisticRegression	67.665 +- 2.939	64.149 +- 0.549	59.981 +- 0.693	70.962 +- 0.520	68.317 +- 0.683
SGDClassifier	**68.517 +- 3.297**	**65.591 +- 0.652**	**61.872 +- 1.116**	72.495 +- 0.663	**69.310 +- 0.517**
LinearSVC	68.487 +- 3.299	65.160 +- 0.381	61.336 +- 1.007	**72.534 +- 0.302**	68.983 +- 0.371

Table 4-31 Results of the 10-fold training using the negation to reverse token's polarities

Testing the statistical features individually or the tokens individual polarities with negation or nor did not yield better results as it is shown by Table 4-32 and Table 4-33.

Features	Best Algorithm	train score	dev score
Amt pos/neg Only	Logistic Regression	**66.928 +- 2.321**	**64.701 +- 0.535**
Sum Only	Logistic Regression	67.019 +- 2.605	64.002 +- 0.873
Polarities Only	Logistic Regression	66.814 +- 2.601	63.689 +- 0.820
Average Only	Logistic Regression	67.112 +- 2.476	63.411 +- 0.799
Max Only	Logistic Regression	66.663 +- 2.929	63.259 +- 0.533
Min Only	Logistic Regression	66.550 +- 2.662	63.235 +- 0.774
Lex Tokens	Logistic Regression	64.860 +- 2.758	63.025 +- 0.820

Table 4-32 Best results for the individual features generation. No negation.

Features	Best Algorithm	train score	dev score
Amt pos/neg Only	SGDClassifier	**68.186 +- 3.381**	**64.959 +- 0.440**
Average Only	LinearSVC	67.837 +- 3.181	64.006 +- 0.826
Final Polarities Only	LinearSVC	67.859 +- 3.165	63.995 +- 0.600
Sum Only	LinearSVC	67.756 +- 3.296	63.893 +- 0.517
Min Only	LinearSVC	67.355 +- 3.224	63.696 +- 0.749
Max Only	Logistic Regression	67.098 +- 3.096	63.291 +- 0.775
Lex Tokens	SGDClassifier	66.755 +- 2.816	62.459 +- 0.758

Table 4-33 Best results for the individual features generation. With negation.

Next, the selection of the token polarities features is done with all the other types of features, without negation and with negation of the tokens. Although being close to the scores of the model created without using token polarities, no model using token polarities was better.

	5 train score	5 dev score	10 train score	10 dev score	15 train score	15 dev score	30 train score	30 dev score	100 train score	100 dev score
LogisticRegression	67.1154	**64.6564**	67.1506	64.5806	67.1065	**64.6839**	67.2422	**64.8103**	66.1611	**64.6774**
SGDClassifier	**68.0389**	64.3969	**67.9435**	**64.6064**	**67.9571**	64.5075	**67.8748**	64.5995	**66.8373**	64.5254
LinearSVC	67.6669	64.3211	67.5644	64.186	67.3737	64.1095	67.6049	64.3794	66.462	64.4527

Table 4-34 Token Polarities, Final Polarities, Statistical features.

	5 train score	5 dev score	10 train score	10 dev score	15 train score	15 dev score	30 train score	30 dev score	100 train score	100 dev score
LogisticRegression	67.8635	64.3451	67.9479	63.8421	67.8207	63.6908	67.5703	64.0412	66.6941	64.1741
SGDClassifier	**68.6577**	**65.4088**	**68.7964**	**65.1748**	**68.5126**	**64.9768**	**68.1209**	**64.7755**	**67.753**	**64.7955**
LinearSVC	68.4477	65.0483	68.3876	64.5696	68.3079	64.3286	67.8131	64.2815	67.2649	64.2042

Table 4-35 Token Polarities, Final Polarities, Statistical features and negation of tokens.

As it can be seen from the models created, the best use of the Bing's Lexicon came from the use of Statistical and Final Polarity features and using the negation of the words to reverse the individual polarities of the tokens before creating the statistical features. The score achieved by the best model is **65.591** using the statistical and polarities features and not using the tokens. According to the t-test, this result was statistically significant when it was compared with the base results (from section 4.2.10).

4.3.2. SentiWordNet Lexicon

As explained in Chapter 2.2.2. the words in this lexicon are made of positive, negative and objective values that complement each other summing to 1. The histogram of the words according to its values is shown in Figure 4-13. By the histograms, it can be seen that the extreme words, the ones that carry the highest values, are the minority. Most of the words are close to 0 and that the values vary from 0.0 to 0.9.

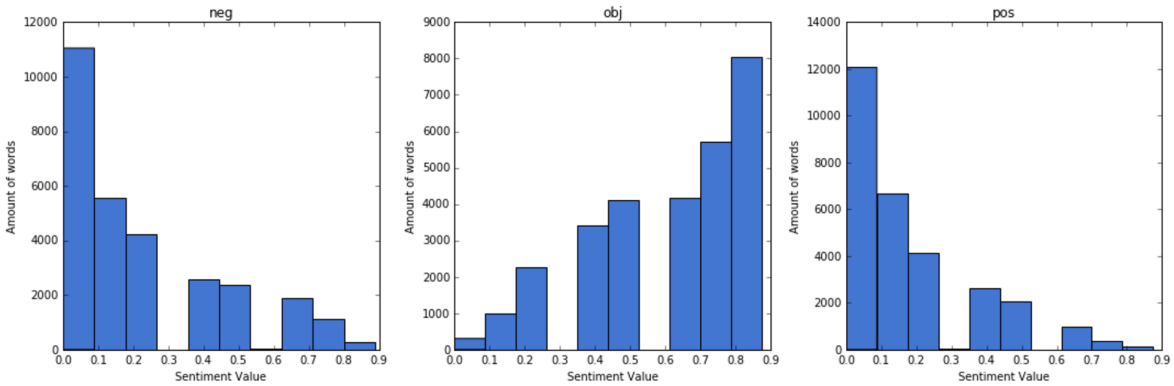

Figure 4-13 Distribution of the word values found inside the SentiWordNet Lexicon

The features tested for the Lexicon were:

- For each token in a tweet, create a Positive, Objective and Negative score
- Create a single Positive, Objective and Negative score for each tweet that is the sum of all tokens' values
- Create a single Positive, Objective and Negative score for each tweet that is the average of all tokens' values

- Consider a minimum value (threshold) to consider a word in the statistics. For these tokens, create a single Positive and Negative threshold value for the tweet that is the sum their values and another that is the average of this values
- Create a decision that the tweet is positive or negative. If the average of the positive is bigger than the average of negatives, the tweet is classified as positive. Otherwise, it is considered negative. It is marked as neutral when they are the same.

Following the work of [GüFu13, KiZM14] all the above features were considered using two types of values for the positive, negative and neutral scales:

- Get the value of the first occurrence of the word in the WordNet
- Create an average of all the related synsets' values

To better illustrate the differences noted above, the word *pure* as an *adjective* has 6 synsets: *arrant*, *saturated* and 4 synsets with different meaning of pure. If used the first synset, the positive, negative and objective values would be [0.32, 0.16, 0.52] respectively while using the average of all synsets, it becomes [0.375, 0.125, 0.5].

In the end, the best features parameters were a combination of using a threshold of 0.75, making the average, sum and the decision features using the value of the first synset only. The real big difference though came from the negated tokens. When a token with the *NEG_* prefix was found, the negative and positive values were swapped before doing the statistics. The final model had a F1-score of **63.390** using the SGDClassifier and was not statistically significant. This means that as the base score has 62.97 for the same classifier, the difference is too small to conclude that it does not happen by chance.

4.3.3. MSOL Lexicon

Like Bing's Lexicon, the MSOL is composed of two possible values for the tokens polarities and the distribution is made of more negative than positive values. Figure 4-14 displays its values distribution. On the other side, the amount of words is more than 10 times the amount of words found in the Bing's Lexicon. MSOL also has bigrams and the words are joined by an "_" character so this will be accounted when searching for tokens polarities.

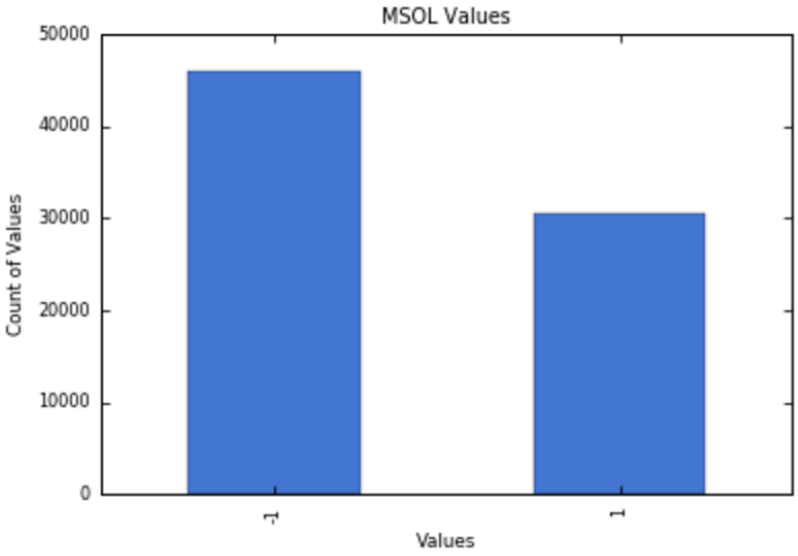

Figure 4-14 Distribution of the word values in MSOL Lexicon

The methods used to create models and check the predictive power of MSOL were the same used for Bing's lexicon: the statistical features and final polarity used individually or together, tokens' polarities with automatic selection and use of negation to create the tokens.

None of the techniques were able to achieve a better score than the base process or Bing, confirmed by the t-test.

4.3.4. SentiStrength Lexicon

As a manual created lexicon, SentiStrength has only 2,6k words in its lexicon. Its distribution of values is different from the previous lexicons as the majority of the tokens are negative but are not in the extremes of the values range. Figure 4-15 displays the values distributions.

The features created were the same 3 types used in the others, but the results were different from the ones seen so far. From all the features created, the one with the biggest impact in the model was the max value of all the tokens in the tweet when using the negated version of the tokens. This feature was able to give a gain of **0.9** to the model and surpassed all other techniques tried in this lexicon.

The final result achieved in the development dataset was **63.892** and it was statistically relevant according to the t-test.

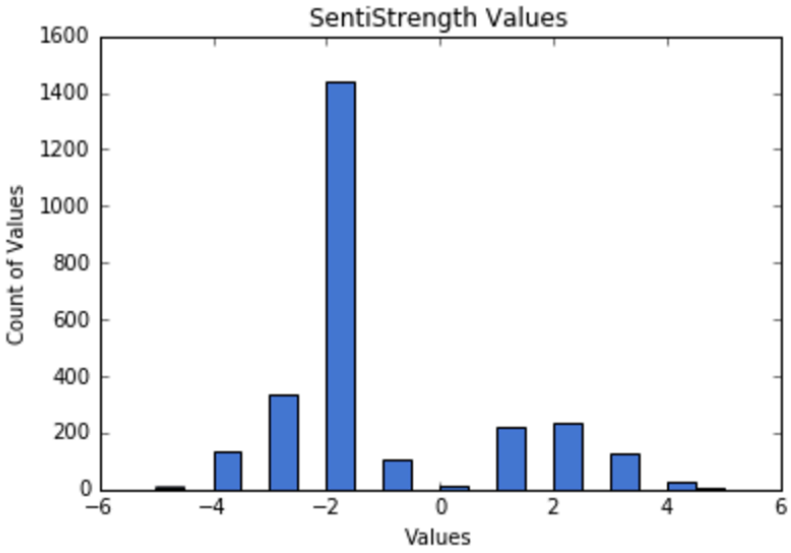

Figure 4-15 Distribution of the word values in SentiStrength Lexicon

4.3.5. NRC Hashtag Sentiment Lexicon

Like all the lexicons so far, the NRC Hashtag present more negative values than positive as it can be seen on its distribution shown in Figure 4-16. The values in this lexicon should range from -5 to +5 indicating how strong the word represent the positive or negative sentiment but the graph shows there is a value that was not properly annotated and has a value close to -9.

As explained in section 2.2.5, the lexicon's tokens and their polarities were created in a semi-automatic manner. This generated bigrams, trigrams and non-contiguous words tokens. For Example, *be---life* is a non-contiguous token that could represent "be *my* life", "be *your* life", etc. To properly test this lexicon, the use of bigrams, trigrams, non-contiguous was also done.

The individual use of the statistical features produced 5 features that added a little to the original base model: the total average, total sum, minimum, the total polarity and the final polarity. Using the tokens' polarities did not helped the model as it created 44k new tokens. Using auto selection of features did not improve the model too.

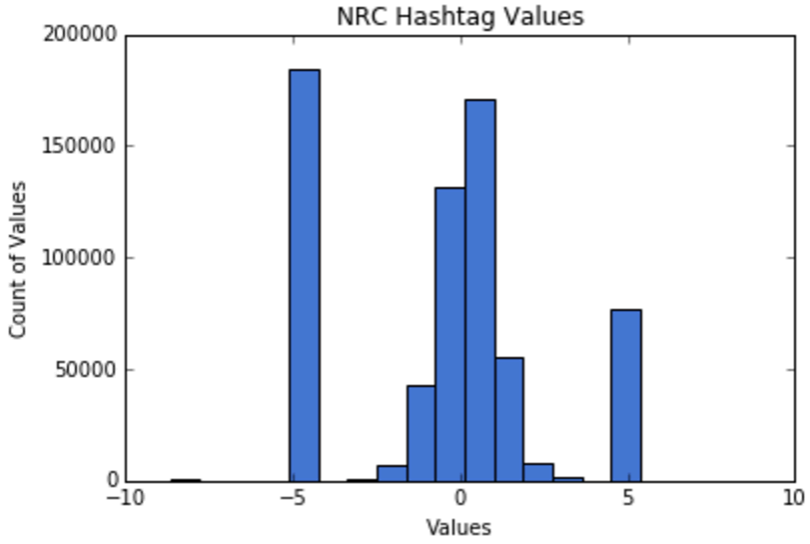

Figure 4-16 Distribution of the word values in NRC Hashtag Lexicon

The best model came when all the features were put together except the tokens' polarities. The model achieved in the development dataset a not statistically relevant F1-score of **63.699**, an increase of **0.7** from the base model but did not raise the score for the training dataset. This could mean that the model has overfit the development dataset.

Another observation is that when bigrams, trigrams and non-contiguous tokens were not used, the model had a hit in performance.

4.3.6. Sentiment 140 Lexicon

Sentiment 140 lexicon was developed in a similar way as the previous lexicon, the NRC Hash. The biggest difference is that the Twitter dataset used was bigger, containing 1.6 Million tweets. As it can be seen in Figure 4-17 distribution of values is similar to the ones found in its sibling with peeks of values in the lowest part of the scale and in the middle of the scale.

Following the test procedures, the use of individual statistic and polarities created 7 features that used together increased the average score in the development database to **64.544**, a very significant score according to the t-test**.** The training dataset F1-Score also had a raise so that could mean a real gain in the pipeline with the use of this lexicon. Negation of the tokens values did not help.

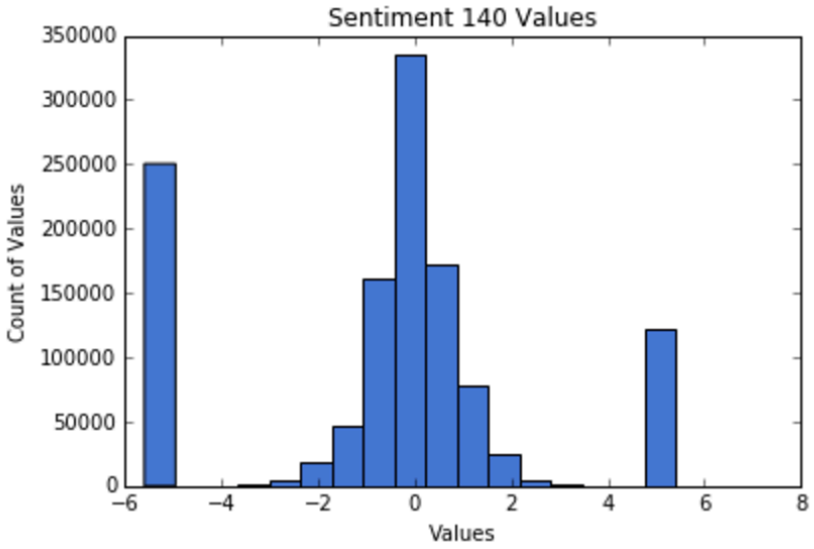

Figure 4-17 Distribution of the word values in NRC Sentiment 140 Lexicon

The use of all the token's polarities created 57k new features that did not help the base model even if auto selection of features was used. In pair with the other automated created lexicon NRC Hash, removing the bigrams, trigrams and contiguous tokens had a hit in performance.

4.3.7. TS-Lex Lexicon

The TS-Lex is also an automatic created lexicon and because of that has a number an order higher of words than the manually created lexicons. Its values are normalized between -1 and 1 and like the other lexicons, it has the most values in the extremes of the value range as it can be seen in its histogram in Figure 4-18.

The same procedure and features were created to check if this lexicon can help with the SA pipeline: test of individual and mutual statistical features, final polarities and token's polarities. Using all statistics showed again to be the best features for the model as it achieved in the development dataset and increase of **1.883** from the base Logistic Regression model, one of the biggest so far, achieving a F1-Score of **64.447**. It was very statistically significant when comparing with the base values using the t-test. The increase also occurred in the training dataset, meaning that it can be a sustainable growth.

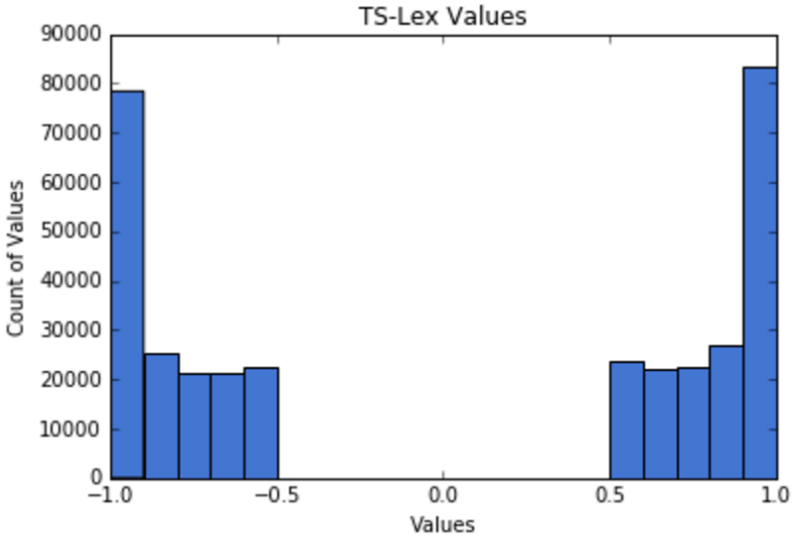

Figure 4-18 Distribution of the word values in TS-Lex lexicon

Once more, using the individual tokens created 16k new features and did not help to improve the model although this time the score was very close to the score achieved by the model created with only the statistical features.

4.3.8. MPQA Lexicon

The MPQA Lexicon is a manually created lexicon so the number of terms is a lot smaller than the last lexicons demonstrated before. It contains around 8k words so it coverage on the dataset was smaller. According to Figure 4-19, besides the negative and the positive tokens, this lexicon contains words that are considered neutral.

The procedure used to check the contribution of this lexicon to the SA pipeline is the same of most opinion lexicons with the creation of the statistical features, final polarities and token's polarities creation.

The best model was created using only the statistical features all together while negating the tokens. The Logistic Regression model achieved a statistically significant score of **63.539**.

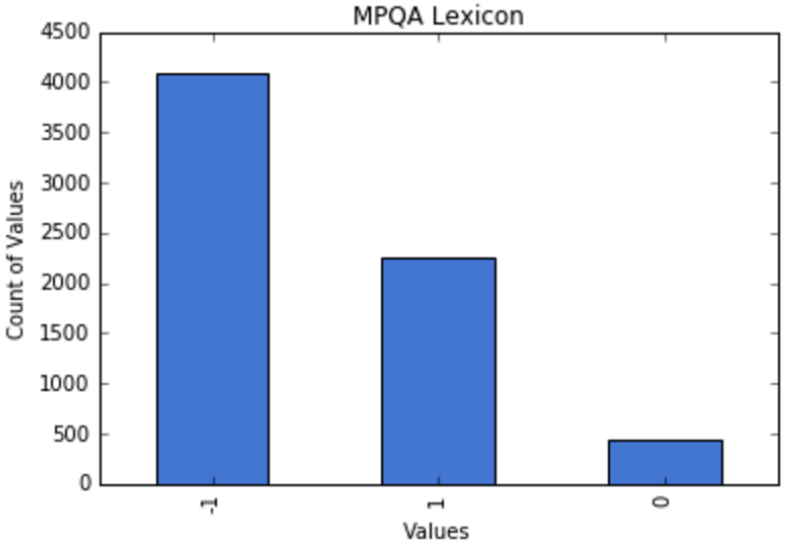

Figure 4-19 Distribution of the tokens' values on MPQA Lexicon

4.4. Emotional Lexicon's Features

In theory, the use of Emotional Lexicons is not very different from the use of Opinion Lexicons. Given a token (word), search inside the lexicon the values associated with it and create the features based on the values found.

The biggest changes come from what the values represent. If in the first kind of lexicons, the value was simply a measurement of how positive or negative a term was, in this second, the Emotional, almost every Lexicon represent the human feelings with different measurements that sometimes are not straight forward to understand and certainly impossible to compare.

For example, WNA (WordNet-affect, section 0) has a set of labels like *emotion* that is made of substantives like *anger* and *fear*, or labels related to *traits* that are related to the personality like *aggressiveness* or *competitive*. In these both cases they are Boolean values indicating the presence or not of that label.

On the other side, ANEW (section 2.3.1) is composed of only 3 values: *dominance* (controlled/in-control), *arousal* (excitement/calm) and *valence* (happy/unhappy) in a scale from 0 to 9. Dominance is related to how diminished or empowered a person feel when thinking about the word, valence is about how happy the person feels about that word and arousal is about being excited or calm.

Due to not having a direct opposite relation of value to represent the opposite terms, in most of the tests there was no test using negation of the words if they were found inside a negation phrase. The Final Polarity was also not used as it is not straight to define it.

In resume, the features created for the Emotional Lexicons were:

- Tokens' Emotional Values: Each token found in the lexicon creates a feature with the lexicon name and the token and all the emotions found inside the lexicon. For example, the DAL lexicon uses the dimensions *activation*, *imagery* and *pleasantness* to define emotions. For the token *yellow*, the features *DAL_yellow_activation, DAL_yellow_imagery* and *DAL_yellow_pleasantness* would be created with values 2.11, 3.0 and 2.3 respectively.
- Statistical Features:
    - For each emotion, the mean value of all the tokens found in a tweet (avg).
    - For each emotion, the minimum value found in a tweet (min).
    - For each emotion, the maximum value found in a tweet (max).
    - For each emotion, the sum of all values found in a tweet (sum).

In the remaining of this subsection it will be presented the use of the 9 different Emotional Lexicons and how they affected the Sentiment Analysis pipeline performance.

4.4.1. ANEW Lexicon

As previously stated, the ANEW has 3 values for each token present in it. The *Valence* can be considered an individual mapping to the opinion/sentiment so this is the single value used as a feature. Because of that, the same process used with the Opinion Lexicons was used here where a set of statistical, polarity and individual tokens is created and each of them is tested together and individually. Automatic feature reduction is applied when using the individual lexicons in the hope to help improve the Pipeline performance.

Looking at the values distribution in Figure 4-20 there can be seen that unlike the opinion lexicons, ANEW has a much more balanced distribution amongst its scale of 1 to 9 than the previous Opinion Lexicons and the extreme values are not the biggest concentration.

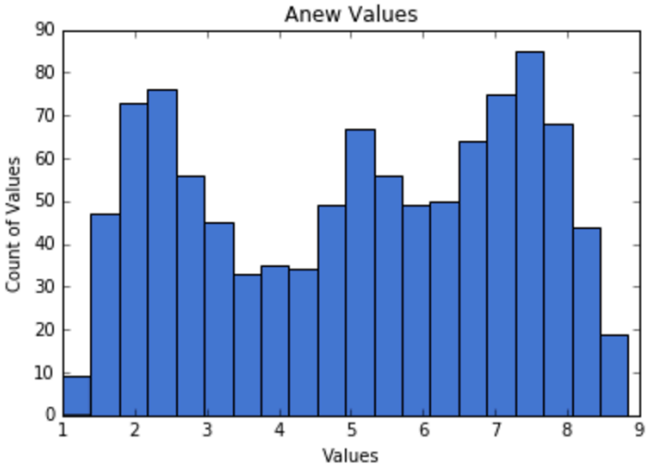

Figure 4-20 Distribution of the word values found inside the ANEW Lexicon

No individual statistic promoted a boost in performance and this time, the individual feature with automatic selection of 15% produced the best model with a performance of **63.631**, a statistically significant result according to the t-test.

The selection process produced 31 new features. Not surprisingly the statistical features were considered the best but some other words also appeared (in order of importance): good, love, happy, fun, hope, birthday, sad, hate and others. As these are words that usually depicts strong emotions, the valence associated with them was captured as a good sign by the classifiers.

4.4.2. WordNet-affect Lexicon

This lexicon is a mapping of 1339 words to 103 tags that represent emotions. In Figure 4-21 there can be seen that there are just a few tags that are used a lot while many of them are used just once. The typical use of the tags is around 4 to 7 tags per word. Investigating further the co-occurrence of the tags, most of the pairs are the tag *emotion* and *positive-emotion* or *emotion* and *negative-emotion*.

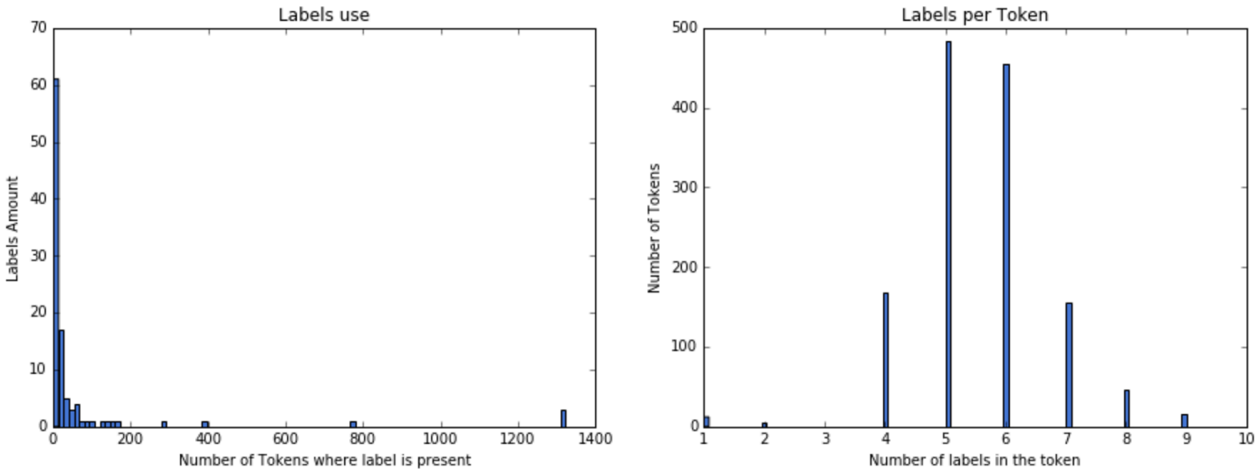

Figure 4-21 Left the histogram of labels used, right the histogram of number of labels in one token

The best model achieved a small but statistically significant increase achieving **63.342**. It was achieved creating the features for all the tokens and then later using feature reduction to 5%. As an example, figure shows the tags created for 1 tweet. Individual use of statistics per each tag or use of just some tags as positive-emotion or negative-emotion did not help the model to achieve better results.

```
{u'WNA_hopefully_affective-state': 1,
 u'WNA_hopefully_emotion': 1,
 u'WNA_hopefully_mental-state': 1,
 u'WNA_hopefully_positive-emotion': 1,
 u'WNA_hopefully_positive-hope': 1,
 u'affective-state_avg': 1.0,
 u'affective-state_maxn': 1,
 u'affective-state_minn': 1,
 u'affective-state_summ': 1,
 u'emotion_avg': 1.0,
 u'emotion_maxn': 1,
 u'emotion_minn': 1,
 u'emotion_summ': 1,
 u'mental-state_avg': 1.0,
 u'mental-state_maxn': 1,
 u'mental-state_minn': 1,
 u'mental-state_summ': 1,
 u'positive-emotion_avg': 1.0,
 u'positive-emotion_maxn': 1,
 u'positive-emotion_minn': 1,
 u'positive-emotion_summ': 1,
 u'positive-hope_avg': 1.0,
 u'positive-hope_maxn': 1,
 u'positive-hope_minn': 1,
 u'positive-hope_summ': 1}
```

Figure 4-22 Features created for one tweet using WNA

4.4.3. DAL Lexicon

The Dictionary of Affect in Language is another example of a manually created lexicon. It uses a 3-dimensions system to represent the emotions in each of the 8842 words contained inside its dictionary. As it can be seen in Figure 4-23 for the activation and pleasantness, the values are more present in the middle of the scale while for imagery, the values are well distributed inside the scale.

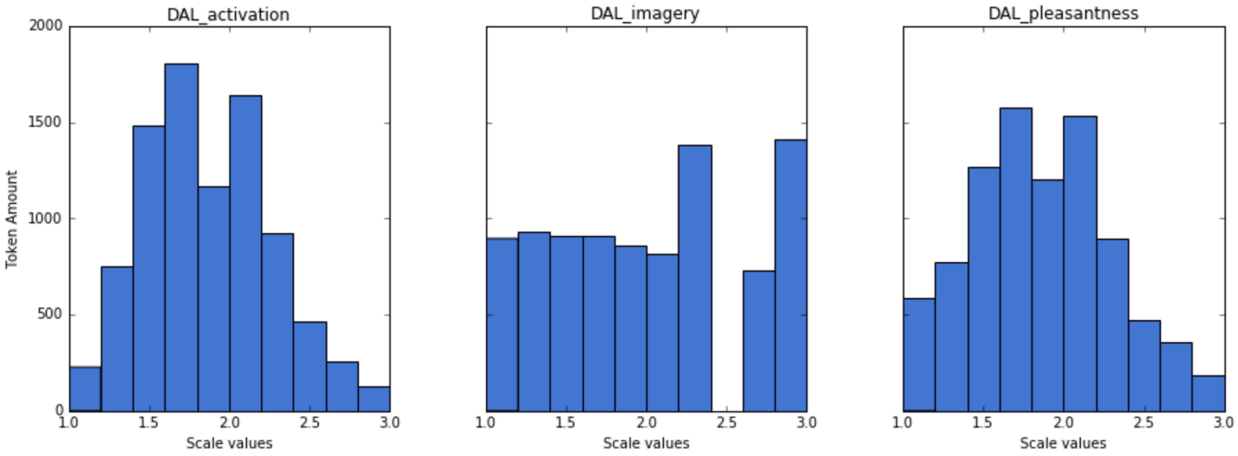

Figure 4-23 Distribution of values on the 3 emotions in DAL

Creating the statistics features for individual tokens or for the full tweet did not yield much improvements. The best model created made a small improvement in the base model using only the *pleasantness* scale and creating full tweet statistics. The final score achieved was **63.304** but was not statistically significant according to the t-test.

4.4.4. SenticNet Lexicon

SenticNet is composed of 30,000 concepts with 4 different emotions: *pleasantness, attention, polarity and aptitude.* As shown in Figure 4-24, the values are distributed among the -1 and +1 values with a higher concentration in the 0, the center of the scale. As this is a semi-automatically created lexicon, it has a big number of tokens and it is composed of tokens like *a lot of flowers*, composed of 4 different words.

The models created used a combination of features based on token's emotional values and the tweet's statistical features. Figure 4-25 displays and example of the statistical features only. The best model achieved a F1-Score of **63.889** and it was created with the tweet's

statistics for the 4 dimensions and the token's emotional values. The t-test indicates the result was statistically significant.

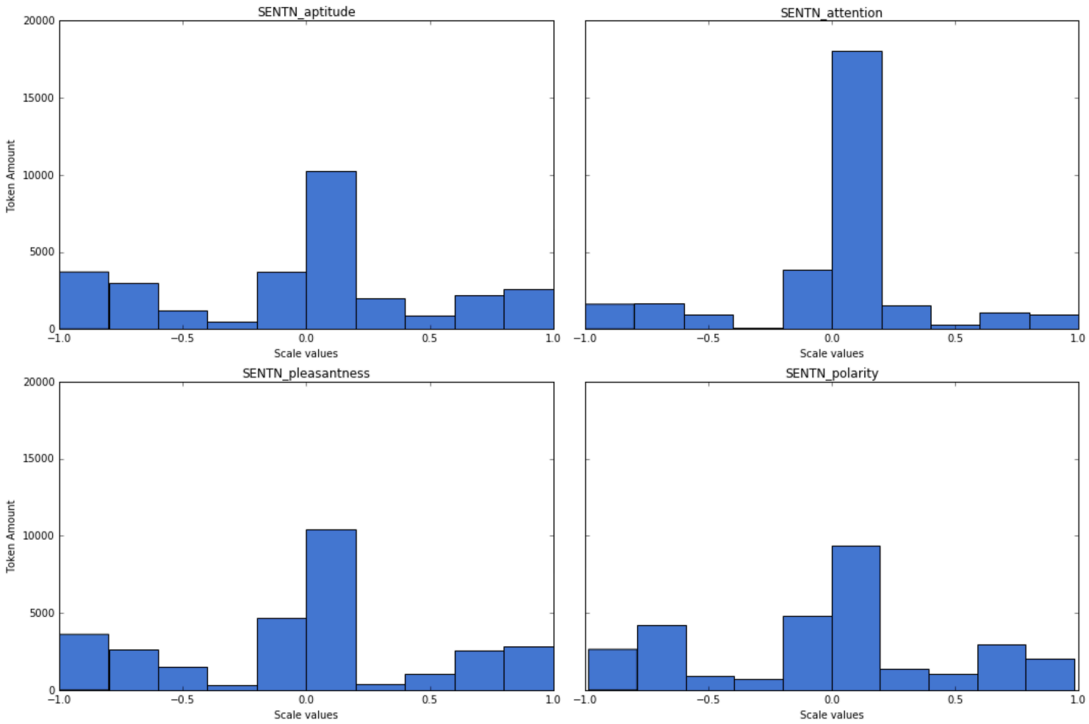

Figure 4-24 Distribution of the SenticNet values

```
{u'aptitude_avg': -0.14580000000000001,
 u'aptitude_count': 5,
 u'aptitude_maxn': 0.86199999999999999,
 u'aptitude_minn': -0.997,
 u'aptitude_summ': -0.72900000000000009,
 u'attention_avg': 0.42859999999999998,
 u'attention_count': 5,
 u'attention_maxn': 0.996,
 u'attention_minn': 0.0,
 u'attention_summ': 2.1429999999999998,
 u'pleasantness_avg': 0.17680000000000001,
 u'pleasantness_count': 5,
 u'pleasantness_maxn': 0.96699999999999997,
 u'pleasantness_minn': -0.82899999999999996,
 u'pleasantness_summ': 0.88400000000000001,
 u'polarity_avg': 0.077199999999999991,
 u'polarity_count': 5,
 u'polarity_maxn': 0.94199999999999995,
 u'polarity_minn': -0.89700000000000002,
 u'polarity_summ': 0.38599999999999995}
```

Figure 4-25 A tweet's features created with SenticNet Lexicon

4.4.5. EmoLex Lexicon

EmoLex was created using the Amazon Turk, where many people have classified the tokens according to 8 emotional categories plus 2 if they are considered positive or negative. Figure 4-26 shows the distribution of the tokens into the 8 categories.

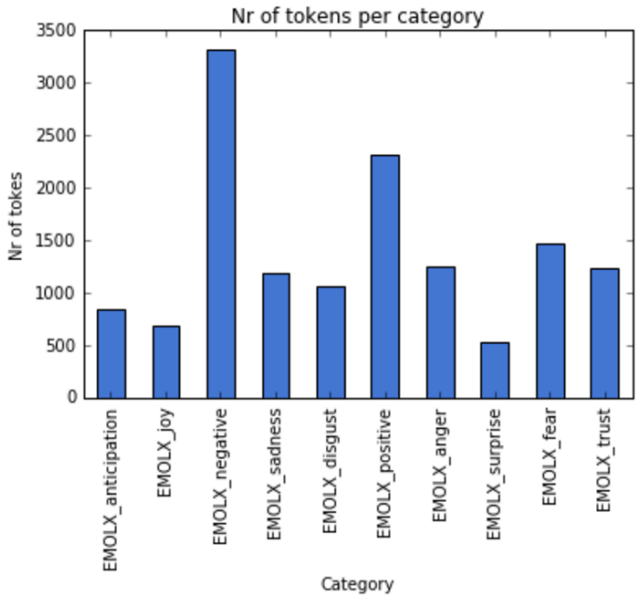

Figure 4-26 Distribution of tokens per emotional category

The best model was created using only the tweet's statistical features and it achieved an average F1-Score of **64.545,** tested to be statistically significant. The use of individual statistics and categories or the creation of individual tokens with the 10 categories did not helped to create a good model.

4.4.6. SentiSense Lexicon

SentiSense is a lexicon that has the same base idea of WNA where the WordNet is enriched with emotional tags. It is composed of 1581 of up to 14 tags. Figure 4-27 displays the tags distribution amongst the tags. Differently from WNA, just one tag is allowed per concept/word.

For this lexicon, the token's part of speech (POS) was considered so that the value is assigned only if the token is in the correct POS. For example, the word *artificial* has a tag of

disgust when it is used as adjective so it will only generate a feature if the word *artificial* is an adjective in the sentence.

This lexicon presented a very poor performance in the SA pipeline. All the statistics features were tested together or individually and the creation of all dimensions for each of tokens. The single pipeline that could improve the base pipeline was the use of *sadness* dimension alone. The model achieved a weak score of **63.168**, not considered statistically significant.

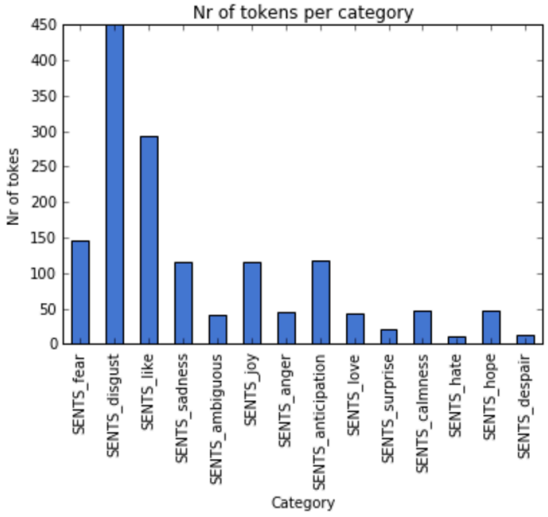

Figure 4-27 Distribution of SentiSense tokens per category

4.4.7. LEW Lexicon

The LEW Lexicon is composed of 1736 tokens that have 3 main dimensions and 92 emotional categories. In this study, we split the use of the lexicon in 2 parts: the first one uses only the 3 main dimensions and the second one uses the 92 emotional categories.

The 3 main dimensions (Activation, Evaluation and Power) use a scale from 1 to 8 and most of the words are distributed in the middle of the scale as can be seen in Figure 4-28.

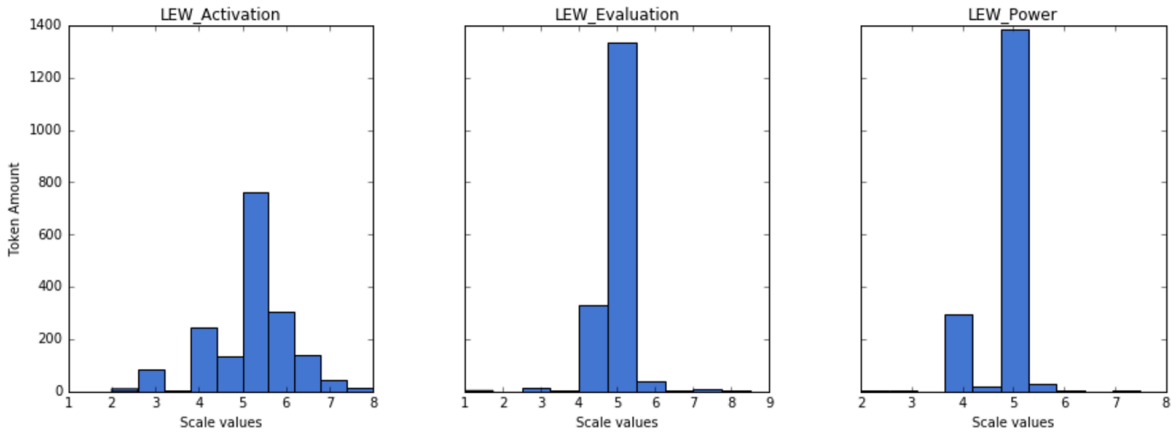

Figure 4-28 Distribution of LEW values per dimension

From all the models generated using LEW, the best one uses only the *Evaluation* dimension and full tweet statistics and achieves a not statistically significant F1-Score of **63.259**.

In a second step, the use of emotions was tested. There are 92 emotional categories, Statistical features were created summarizing the tokens of each tweet with measures of minimum, maximum, etc. and creating 92 features for each token found was also tested. The end result was that no model had a good performance, even when the features were auto selected.

4.4.8. EmoSenticNet Lexicon

EmoSenticNet is a lexicon that is an expansion of SenticNet where 6 emotional categories were added to 13188 words. Differently from most of the opinions lexicons, most of the words are tagged as Joy and not with a negative tag as shown in Figure 4-29.

Individual features, automatic feature selection, statistical values per tweet were all tested. The best model performance used only the *Joy* or *Disgust* emotions. Both achieved individually a F1-score higher than **63.2.** Combining both created the best model with a score of **63.431**, not statistically significant according to the t-test.

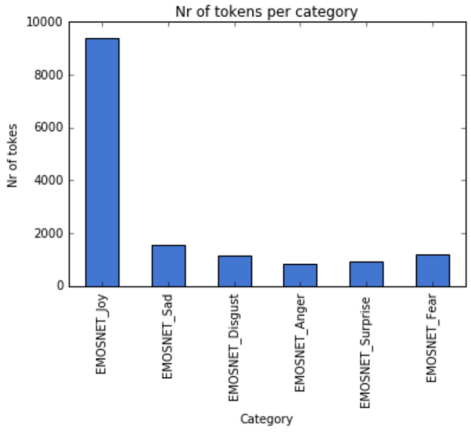

Figure 4-29 Distribution of the EmoSenticNet values per category

4.4.9. LIWC Lexicon

LIWC is more than a lexicon, it is a full pipeline of textual analysis that creates 88 different features as shown in section 2.3.9. All the characteristics are of statistical type like counts, minimum and maximum.

Among the features created there are 3 that describe positive and negative emotions, positive feelings. Their histogram in Figure 4-30 shows that most of the values are in the beginning of the scale.

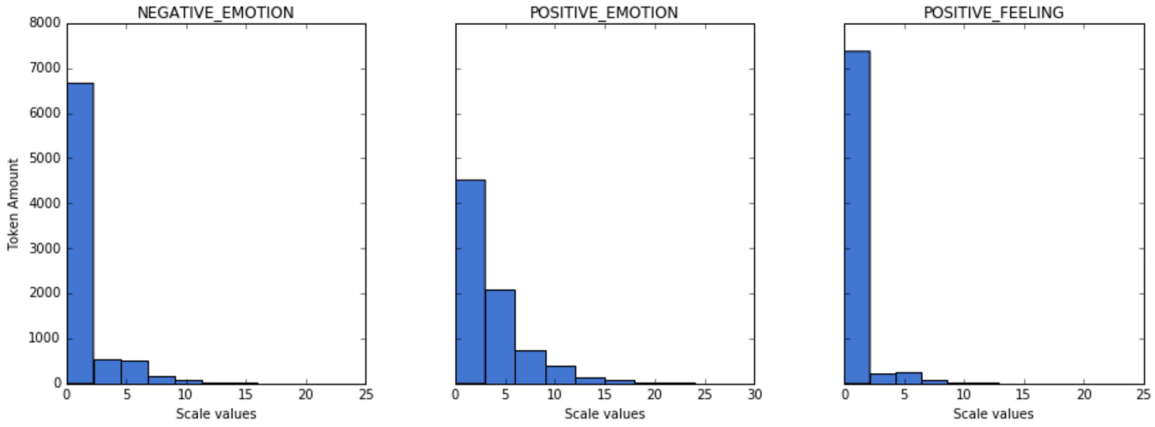

Figure 4-30 Distribution of the values generated from LIWC for 3 categories

The LIWC feature creation added 88 new features to the base pipeline reaching a total score of **63.977** in the development dataset, tested as statistically significant. The use of just some features lowered the predictive power of the model.

4.5. Lexicon Creation Summary

This chapter showed the creation process of the non-lexicon related features, called base features, as well tests performed with the 8 Opinion and 9 Sentiment Lexicons individually.

A first conclusion that is obviously shown in the Table 4-36 is that the performance of the model in the training dataset is bigger than in the development. The cases where the model is a lot better in the training than in the development show that an overfitting situation is occurring.

Lexicon	Type	Best Algorithm	cov %	negated	Stat. Sign	train score	dev score	f1_dev_neg	f1_dev_neu	f1_dev_pos
Bing	**Sent**	**SGDClassifier**	**100**	**Yes**	**Yes**	**68.517 +- 3.297**	**65.591 +- 0.652**	**61.872 +- 1.116**	**72.495 +- 0.663**	**69.310 +- 0.517**
EmoLex	Emo	LogisticRegression	98.42	No	Yes	66.412 +- 3.048	64.545 +- 0.823	59.135 +- 1.392	70.190 +- 0.539	69.955 +- 0.444
Sentiment 140	Sent	SGDClassifier	100	No	Yes	67.595 +- 3.248	64.544 +- 0.508	59.412 +- 0.954	72.739 +- 0.587	69.675 +- 0.361
TS-Lex	Sent	LogisticRegression	100	No	Yes	66.728 +- 2.886	64.447 +- 0.428	59.365 +- 0.670	69.532 +- 0.414	69.529 +- 0.559
LIWC	Emo	SGDClassifier	100	No	Yes	67.953 +- 3.457	63.977 +- 0.601	58.193 +- 0.793	71.918 +- 0.435	69.761 +- 0.657
SentiStrength	Sent	SGDClassifier	100	Yes	Yes	66.618 +- 3.124	63.892 +- 0.615	57.728 +- 1.366	72.532 +- 0.345	70.056 +- 0.497
SenticNet	Emo	LogisticRegression	99.60	No	Yes	66.861 +- 2.506	63.884 +- 0.714	58.121 +- 1.104	70.677 +- 0.595	69.647 +- 0.532
NRC Hashtag	Sent	SGDClassifier	100	No	No	66.609 +- 2.785	63.699 +- 0.859	57.856 +- 1.450	72.572 +- 0.497	69.541 +- 0.669
ANEW	Emo	LogisticRegression	100	No	Yes	64.743 +- 3.179	63.631 +- 0.634	58.494 +- 1.013	68.307 +- 0.542	68.768 +- 0.490
MPQA	Sent	LogisticRegression	100	Yes	No	67.835 +- 2.898	63.539 +- 0.543	58.033 +- 0.759	70.962 +- 0.682	69.046 +- 0.667
EmoSenticNet	Emo	SGDClassifier	98.24	No	No	66.631 +- 3.378	63.431 +- 0.745	57.645 +- 1.151	72.490 +- 0.570	69.217 +- 0.463
SWN	Sent	SGDClassifier	100	Yes	No	66.994 +- 3.383	63.390 +- 0.815	57.579 +- 1.332	72.234 +- 0.460	69.201 +- 0.557
WNA	Emo	LogisticRegression	36.78	No	Yes	66.083 +- 3.173	63.342 +- 0.518	57.567 +- 0.723	71.259 +- 0.668	69.116 +- 0.632
DAL	Emo	SGDClassifier	99.90	No	No	66.754 +- 3.209	63.304 +- 0.513	57.209 +- 1.054	72.408 +- 0.379	69.400 +- 0.653
LEW	Emo	SGDClassifier	93.10	No	No	66.779 +- 2.712	63.259 +- 0.692	57.158 +- 1.065	72.351 +- 0.288	69.361 +- 0.689
SentiSense	Emo	SGDClassifier	1.44	No	Yes	66.496 +- 3.172	63.168 +- 0.578	56.780 +- 1.046	72.490 +- 0.323	69.555 +- 0.506
Base Features	--	SGDClassifier				66.290 +- 3.265	62.977 +- 0.684	56.492 +- 1.146	72.477 +- 0.260	69.463 +- 0.590
MSOL	Sent	SGDClassifier	99.99	No	Yes	66.753 +- 3.000	62.611 +- 0.744	56.872 +- 1.252	71.695 +- 0.423	68.351 +- 0.708

Table 4-36 Best results for each Lexicon. Results are in *dev score* descending order.

From all the Lexicon tested, MSOL not only was not able to generate a model that improved the Base Model created but created a set of features that decreased the performance. From the top 5 models, 3 used sentiment lexicons and 2 used emotional lexicons. Analyzing the top 10, 5 models for each. The conclusion here is that the type of the lexicon may not be a determinant for the performance of the model as we have good models created with both types.

Another thing to consider is that some models were created with the SGDClassifier while others were created using LogisticRegression. SGDClassifier appeared 3 times as the best algorithm when considering the top 5 and 5 times if considering the top 10 results. There

is no evidence that one is better than the other. Later this can be a problem as some features may not help the model achieve the best performance as it is not the best algorithm for it. For example, when using the SGDClassifier, Bing will top the performance but EmoLex won't.

The coverage (number of tweets with a feature created/total tweets) of the lexicons or the distribution of the values seen on them do not show a trend that could explain this phenomenon. Using the context to negate the tokens did not improve the models most of the time but helped in the best model. Looking at the performance of the models, it can be seen that negation (f1-dev_neg) was always the weakest value and it brought the general F1-Score down. SentStrength for example had a very good performance in the positive and neutral tweets but was very bad with the negative ones. Ordering the results in negative best performance did not show any trends.

Using the Table 2-13, as base, we can summarize that amongst the top 10, there are:

- 8 Semi-Automatic Lexicons and 2 Manually created;
- 5 Corpus Based and 5 Dictionary Based Lexicons;
- 6 Lexicons with less than 10k. The best lexicon has approximately 6700 terms;

Analyzing the above statistics there may be an indication that Semi-automatic creation can produce good lexicons and that there is no need to make a very big lexicon to have a good predictive power.

One important observation is that the performance of the models may vary a lot in another dataset. The coverage of the lexicons, the style of the text and the subject they address have great influence in the final performance of the model. The method to find the better use of the lexicons is valid though regardless of the dataset used.

Finally, the reader may think a model that use all the features created from all the lexicons would produce a superior result and that should be the sum of all of the performance delta from the base model and these models. If that was the case, the final model would score a F1-Score in the development dataset of approximately **75.1** (the base 62.977 plus the sum of the deltas between the base model and the lexicon created models).

Sadly, it is not the case. Features will influence each other so when 2 set of features are put together with the base features the final model won't be the direct sum of the individual contribution of each lexicon's created features.

The next chapter will show the process used to join the features set and discover the best model.

Chapter 5

Method: Best Features Selection – Genetic Algorithm and Final Pipeline

As there are 17 different lexicons, there are 2^{17} different lexicon combinations that can be done. This means that if it were to test all of the possibilities, 131,072 combinations would need to be tried. Considering an average of 1 minute for each combination, it would take around 91 days to find the best lexicons mix.

Although for this problem, the total time was not something prohibitive, it is easy to see that the total time would easily become unmanageable if new lexicons were added or if another bigger dataset was used. The Sentiment Analysis with multiple lexicons can be considered an Optimization Problem with discrete variables. That means that we know all the possibilities that are available and we want to find the best solution.

These combinations are called the search space and there are many methods in the literature to search for the combination that provide the best performance. The simplest way is to try every single combination but as previously described, that would require a lot of computational time.

This work proposes the use of a Genetic Algorithm (GA) to search for a good solution created after a controlled search. The solution may not be optimal as the genetic algorithm may not provide one but it is still acceptable and with a reduced time if compared to a brute force search.

This chapter will provide a primer on genetic algorithms and will describe the method applied in this particular problem of Sentiment Analysis with multiple lexicons.

5.1. Genetic Algorithm Primer

Genetic algorithms were first mentioned in the work of John Holland in the 60s and further developed in the next decades. His work [Holl75] on Genetic Algorithm (GA) created a theoretical framework that represents problems and the search of possible solutions as a typical biological environment where the individuals survive if they are most fit and the species evolution is done through mating and mutation.

In this framework, an individual represents a possible solution for a problem. His representation is done through a DNA of the individual (genes), usually a string of ones and zeros (bits) where each position (locus) represents the presence or absence of an element of the solution.

At a given time, there will exist a number of P individuals each representing one or more solutions to the same problem creating a *population* of solutions.

A *fitness function* is created to model the environment and the condition that leads an individual to survive. In practical terms, this function will distinguish good and bad solutions for the problem in place.

The search for the best solution mimics the species behavior in an environment. The individuals go through a *selection* process where they are ranked using the fitness function. The most fit have better chance to create its descendants mating with another individual. Their genes are exchanged using a *crossover* process generating two *offspring* that will replace a set of the previous generation, usually the less fit to the environment. Finally, the individuals may go through mutation, adding another layer of variability to the population's specimens.

Like in nature, this procedure is repeated many times leading to "stronger", most fit, individuals in each generation. In computing terms, the solutions will become more robust and will present a better performance in solving the problem that was modeled.

During the years, many other aspects and variations of this framework were proposed. They evolve specific parts like the selection, the crossover, the mutation, the survival of the individuals, etc. Figure 5-1 demonstrate two variations of the *crossover* step in a GA process.

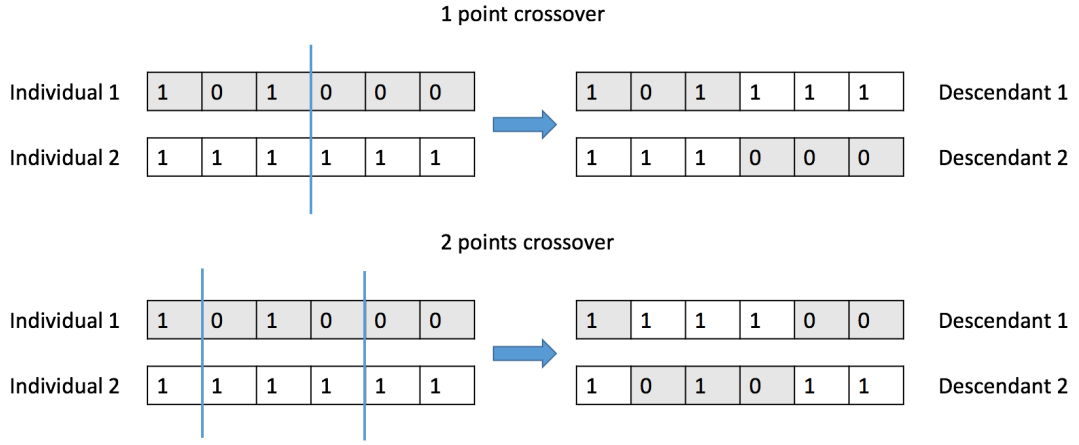

Figure 5-1 An example of variations that can exists in GA. In this case, the crossover is done in 1 or 2 points resulting in different offspring

In the remaining of this chapter, it will be described the modeling of the problem of finding the best lexicon along with the parameters used and the results achieved.

5.2. Genetic Algorithm Setup

The Genetic Algorithm was done using a Python Library called DEAP [RFGP12] that handles the base steps of the genetic algorithm leaving for the user to implement the specific parts that varies according to the problem. Two tutorials[1][2] were used to understand how to use the library as well as the documentation[3]. DEAP uses the concepts explained in [BaFM00] making the genetic algorithms usable out of the box, without much setup or the need to program the evolutionary steps.

The biggest advantage in using this library is that the user needs only to program the *individuals* and the *fitness* function. The library handles the population creation, the evolution, the mating and the mutation processes according to parameters set and it has some helpers like a Hall of Fame and statistics that enable the user to properly track the evolution as well as the best individuals in each generation.

[1] https://github.com/lmarti/evolutionary-computation-course/blob/master/AEC.02%20-%20Elements%20of%20Evolutionary%20Algorithms.ipynb
[2] https://github.com/DEAP/notebooks/blob/master/OneMax.ipynb
[3] http://deap.readthedocs.io/en/master/

The following definitions were used for the problem proposed in this work. They were selected empirically to produce small increase in the number of lexicons, avoiding to add too fast all the genes (all lexicons) to the individuals without exploring the search space:

- **Genes**: each lexicon is considered one gene inside the individual definition.
- **Individuals**: A individual is characterized by 17 genes indicating the absence or presence of the lexicon's features set.
- **Population Size (P)**: the population size is 30.
- **Fitness function**: The fitness function is going to test the lexicons combinations and find the best result for the F1-Score macro according to the SemEval 2013.
- **Selection (μ)**: For each generation, the top P/5 (6) (population size) will be selected to the next round.
- **Lambda (λ)**: The number of offspring to be created P- μ (24).
- **Genetic variation**: *crossover* (mating) using 1 point and 20% probability of a mutation in 1 gene.

The flow of the genetic evolution is showed in Figure 5-2. First, the population with size P (30) is created with random individuals. Then each of the individuals is tested using the fitness function, creating an evaluation of how fit this individual is. The fitness output is the 10-fold average F1-score macro as proposed by the SemEval 2013 for the development database. Third, the selection process occurs, where only the P/5 (6) most fit are selected to create descendants.

The next step is where the genetic variation occurs. The individuals will mate (exchange genes) with a probability of 50% and they will mutate with a probability of 20% generating and offspring of 24 new individuals.

The same process of evaluation and selection occurs but this time the parents and offspring will be put together, generating again a population of 30 individuals, and the top 6 most fit will survive to the next generation to mate and mutate. This procedure is repeated for 30 generations.

The procedure described above is the evolutionary algorithm called $(\mu + \lambda)$ (Mu plus Lambda). It was chosen because for every interaction it keeps the best results. This guarantee that the best individual will be kept until the end.

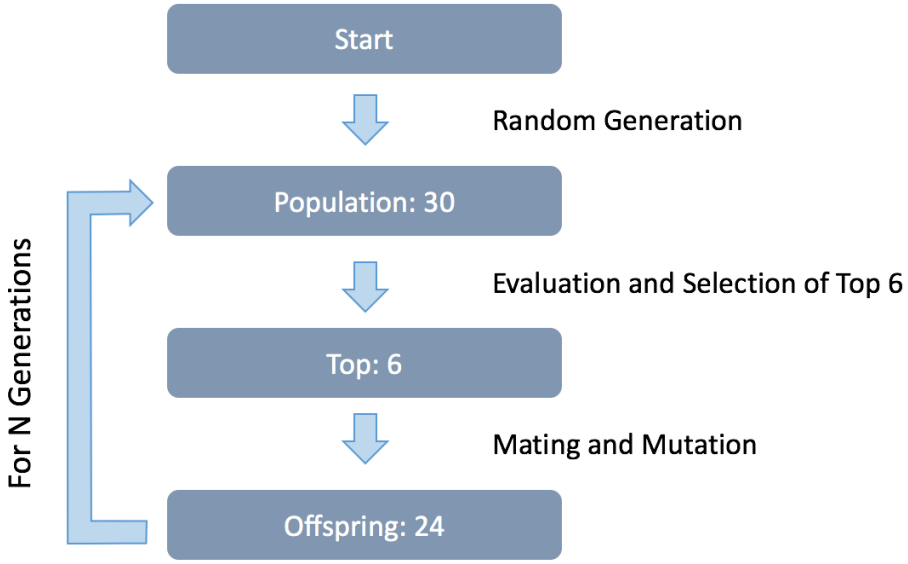

Figure 5-2 Flow of the genetic algorithm

Some points of the flow need to be clarified. The individuals are the Sentiment Analysis pipeline defined by 1 gene of 17 positions. Figure 5-3 illustrates this concept where each position (locus) represents one of the lexicons that should or should not be used in the pipeline. In the example, the first 2 positions of the sequence are set while the other 15 are not. That means this individual, or this pipeline, should use Bing and SWN lexicon's features only.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
BING	SWN	MSOL	SSTREN	NRCHASH	SENT140	TSLEX	MPQA	ANEW	WNA	DAL	SENTN	EMOLX	SENTS	LEW	EMOSNET	LIWC
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Position 1: Bing Lexicon Features
Position 2: SWN Lexicon Features
…

Figure 5-3 An individual gene and the meaning of 2 position set

The fitness function is very similar to all of the Sentiment Analysis pipeline used so far. The flow is shown in Figure 5-4. The validation procedures check the individual genes and load the appropriate lexicons' features for the train and development dataset, merging them with the base features described in section 4.2. Then the full pipeline of machine learning, with the 3 best algorithms with 10-fold cross validation, is run over the train dataset and tested in the development dataset. This will create 3 average F1-score. The maximum value for the dev dataset is considered the fit value and will be used to rank this individual against the other of this generation.

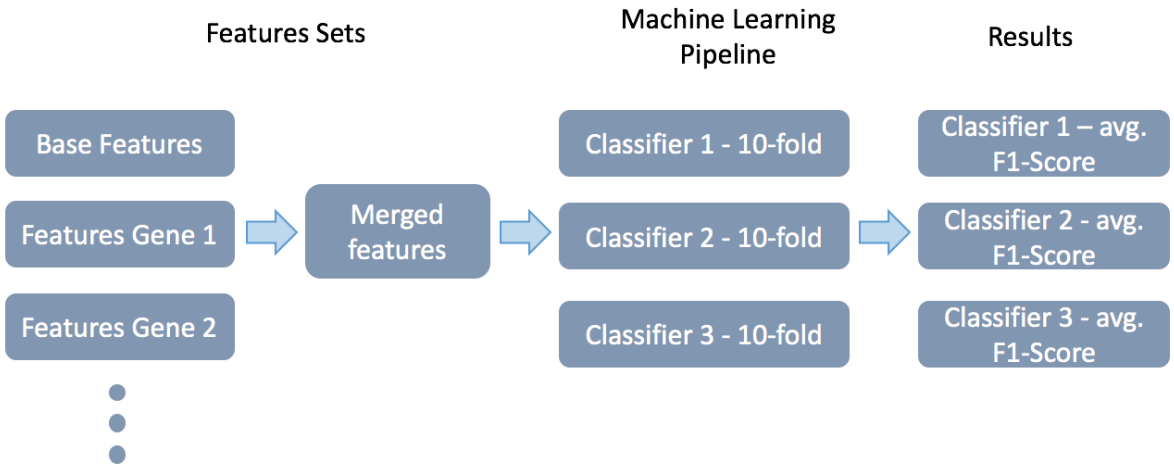

Figure 5-4 Flow that occurs inside the fitness function

5.3. Genetic Algorithm Results

The GA took around 4h to calculate the best individuals on the setup described in the previous section. A trick used to optimize the search is that each individual is calculated only once and have its results stored. If an offspring has the same gene of his parent, it would not be recalculated. This reduced the search time considerably as the individuals may appear multiple times in the same or in the next generations. Figure 5-5 shows one example of the output made in each individual fitness function.

```
New Individual      [1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0]   (67.888261177016929, 0.63881472812958484, 'LogisticRegression')
New Individual      [1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0]   (66.651649425826909, 0.63865380457600396, 'LogisticRegression')
Calculated already  [1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0]   (66.75858845266206, 0.66492310010450351, 'LogisticRegression')
```

Figure 5-5 Examples of the fitness function's output

Figure 5-6 displays the minimum, maximum and average fitness during the Genetic Algorithm search. Note that the difference between the worst combination and the best combination is around 6.5 points or around 10%, a very statistically relevant result confirmed with the t-test. Another point that calls the attention is that the Genetic algorithm converge to its maximum around the 20th generation. From the 20 to the 30, it became stable and did not create any individual that were more fit than its parents.

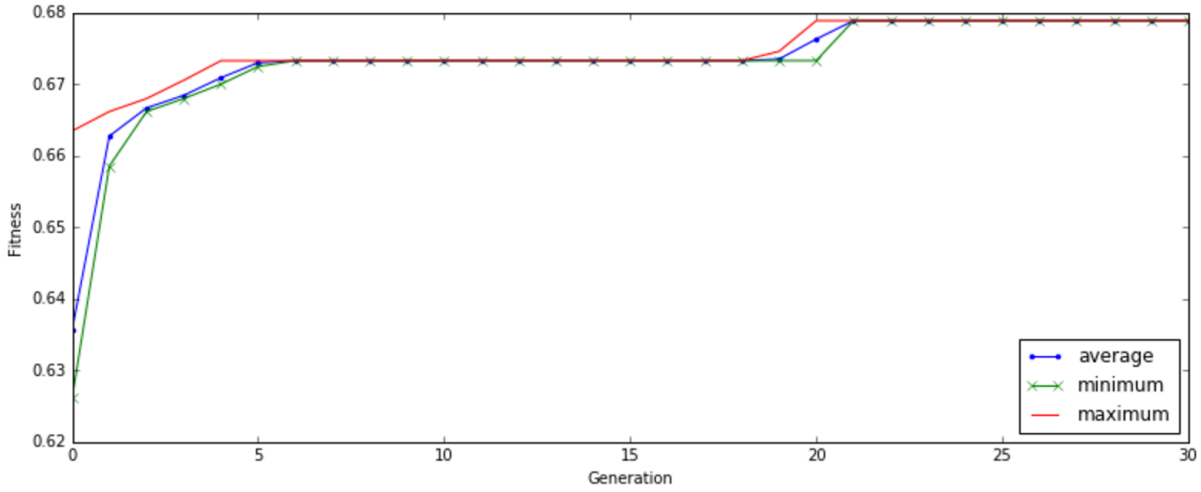

Figure 5-6 Fitness function performance evolution

The top 3 combinations are shown in table 5-1 and the bottom 3 are shown in table 5-2. Notice that the best combination does not use all the Lexicons available, just 7, and that third place share many genes with the most fitted individual.

An important matter is that most of the works done so far using many lexicons usually shows the combination of all of the lexicons and then each one of them is removed to find the importance of them to the performance of the prediction. The analysis of the generated combinations shows that this may not be the best procedure as each lexicon inserted or removed from the pipeline influence the contribution that is made by other lexicons.

Another conclusion is that not all lexicon help the sentiment analysis pipeline to be better as the individual with all genes (or with all the lexicons) is not present in the top 3. The bottom 3 have some lexicons but have a performance below the base pipeline, with no lexicons.

	Train F1	Dev F1	Agl F1
Gene			
[1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0]	68.429600	67.991538	LogisticRegression
[1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0]	68.614322	67.946938	LogisticRegression
[1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0]	67.823926	67.893985	LogisticRegression

Table 5-1 Top 3 individuals from the Genetic Algorthm search

	Train F1	Dev F1	Agl F1
Gene			
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]	66.612523	62.649224	SGDClassifier
[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	66.752715	62.611457	SGDClassifier
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	66.752715	62.611457	SGDClassifier

Table 5-2 Bottom 3 individuals from the Genetic Algorthm search

With the best individual in place, the next step is to create the final pipeline using the training and the development dataset as the training dataset and check the performance in the test dataset.

5.4. Final Sentiment Analysis Pipeline

With the best combination found heuristically, the final pipeline can be created. The steps required are:

- Concatenate the train and dev datasets creating a new bigger train dataset.
- Create the base features for the new train dataset.
- Create all the opinion and lexicon features found through GA and join them with the new base features
- Train a model using the full dataset
- Predict the values on the test set with the new model

Following the steps above, the train and development datasets are concatenated creating a bigger train dataset and the automatic feature selection (reduction using chi-squared) is done creating the base features.

Next, translating the Gene of the best individual found using the GA (Table 5-3) shows that the best combination is comprised of 8 lexicons: Bing, MSol, SentiStrength, NRC

Hash, TSLex, DAL and SenticNet. These lexicon's features were created for both the bigger train and test dataset and joined with the base features. The first dataset will be used to create the final predictive Machine Learning model while the second dataset will have its polarity predicted, enabling a benchmark with other works.

	BING	SWN	MSOL	SSTREN	NRCHASH	SENT140	TSLEX	MPQA	ANEW	WNA	DAL	SENTN	EMOLX	SENTS	LEW	EMOSNET	LIWC
0	1	0	1	1	1	0	1	0	0	0	1	1	0	0	0	0	0

Table 5-3 Gene translation of the best individual

The results of the Genetic Algorithm search stated that the best run was produced using the Logistic Regression algorithm. These results were found analyzing the performance of the model over the development dataset. To validate that this is still the case, the 10-fold is rerun but now over the bigger training dataset and its performance is tested again the test dataset. Table 5-4 displays the results and the Logistic Regression is still the best algorithm for the final train dataset created.

	train score	test score	f1_test_neg	f1_test_neu	f1_test_pos
LogisticRegression	68.459 +- 2.117	**66.516 +- 0.296**	**61.042 +- 0.604**	**73.968 +- 0.314**	**71.991 +- 0.495**
SGDClassifier	67.073 +- 2.536	61.584 +- 1.332	53.590 +- 1.972	71.789 +- 0.623	69.577 +- 2.031
LinearSVC	**68.531 +- 2.385**	62.775 +- 0.340	55.580 +- 0.555	72.458 +- 0.299	69.970 +- 0.431

Table 5-4 10-fold cross validation on the final train and test dataset

To produce the 10-fold cross validation, the train dataset is split as shown in Figure 4-2. To have the final F1-Score of the model, the full train dataset should be used, without any splitting. The *test score* in Table 5-5 had a marginal improvement over it former result.

	train score	test score	f1_test_neg	f1_test_neu	f1_test_pos
LogisticRegression	**76.427 +- 0.016**	**66.904 +- 0.019**	**61.719 +- 0.000**	**74.289 +- 0.017**	**72.089 +- 0.037**

Table 5-5 The final values using the whole train dataset, no split

As the top 3 models found in GA had very close results, we also ran the procedure on the second and third place to check if the order still holds. Table 5-6 shows that in the final

results, the order of the best individuals changes a little. The second best became the best method. As a comparison, the final pipeline is also done with all the lexicons and with no lexicons. In both situations, the model generated was not better than the combination discovered by the GA.

The best lexicon combination the Genetic Algorithm found achieved a score **66.904**. If compared to the original values in the SemEval 2013 competition, shown in Table 3-5, this model would have achieved the second place.

Another interesting observation is that from the 5 lexicons that created the best pipeline in 2013 (Table 3-5), only 2 are present in this work's final pipeline: BING and NRCHash. The winning combination is comprised of 4 sentiment based lexicons and 2 emotional based lexicons. This shows that emotional lexicons can help a Twitter sentiment analysis pipeline.

	train score	**test score**	**f1_test_neg**	**f1_test_neu**	**f1_test_pos**
[1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0]	76.427 +- 0.016	66.904 +- 0.019	61.719 +- 0.000	74.289 +- 0.017	72.089 +- 0.037
[1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0]	76.357 +- 0.015	**67.018 +- 0.032**	**61.947 +- 0.064**	74.217 +- 0.011	72.090 +- 0.000
[1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0]	75.531 +- 0.011	66.357 +- 0.016	60.516 +- 0.040	**74.308 +- 0.012**	**72.199 +- 0.015**
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]	70.415 +- 0.017	65.516 +- 0.015	59.291 +- 0.023	73.569 +- 0.012	71.742 +- 0.017
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	**77.780 +- 0.002**	60.137 +- 0.000	52.066 +- 0.000	71.392 +- 0.000	68.209 +- 0.000

Table 5-6 Results of the top combination and a comparison with using all lexicons and no lexicon

As a comparison, the final pipeline was generated adding a lexicon at a time. Table 5-7 shows a list of scores similar to what is seen in other works [GVJT14, MoKZ13] where in the ablation experiments, all the features are added to the pipeline and then one by one they are removed to see how important they were. Notice in Figure 5-7 how the gradual addition of different lexicons keeps improving the pipeline until a point where the lexicons were not able to make the pipeline better.

	train score	test score	f1_test_neg	f1_test_neu	f1_test_pos
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	78.783 +- 0.007	62.109 +- 0.037	54.977 +- 0.078	72.122 +- 0.011	69.240 +- 0.009
[1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	78.905 +- 0.014	61.978 +- 0.030	54.925 +- 0.044	72.037 +- 0.018	69.032 +- 0.023
[1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	**80.538 +- 0.018**	62.783 +- 0.043	56.376 +- 0.080	71.620 +- 0.007	69.190 +- 0.019
[1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	80.479 +- 0.025	63.359 +- 0.026	57.010 +- 0.024	71.668 +- 0.020	69.708 +- 0.029
[1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	76.152 +- 0.023	64.859 +- 0.016	58.923 +- 0.032	72.691 +- 0.014	70.795 +- 0.000
[1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]	72.723 +- 0.007	64.984 +- 0.015	59.239 +- 0.000	72.235 +- 0.023	70.729 +- 0.031
[1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]	72.564 +- 0.025	66.042 +- 0.024	60.685 +- 0.038	73.054 +- 0.045	71.399 +- 0.015
[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0]	72.700 +- 0.025	65.689 +- 0.055	60.045 +- 0.078	72.798 +- 0.033	71.333 +- 0.045
[1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]	73.838 +- 0.022	65.636 +- 0.039	59.782 +- 0.061	72.715 +- 0.029	71.489 +- 0.022
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]	73.907 +- 0.016	65.489 +- 0.040	59.297 +- 0.062	72.500 +- 0.031	71.681 +- 0.035
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]	73.781 +- 0.012	65.748 +- 0.028	60.235 +- 0.056	73.389 +- 0.012	71.261 +- 0.017
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0]	74.948 +- 0.014	66.143 +- 0.051	60.875 +- 0.063	73.202 +- 0.022	71.411 +- 0.041
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0]	75.220 +- 0.019	66.212 +- 0.028	**61.117 +- 0.062**	73.208 +- 0.028	71.307 +- 0.027
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0]	75.197 +- 0.012	66.169 +- 0.042	60.967 +- 0.093	73.275 +- 0.029	71.372 +- 0.014
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]	74.719 +- 0.011	**66.242 +- 0.043**	61.011 +- 0.079	73.199 +- 0.013	71.473 +- 0.014
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]	74.869 +- 0.014	66.026 +- 0.038	60.729 +- 0.090	72.731 +- 0.012	71.324 +- 0.026
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]	70.415 +- 0.017	65.516 +- 0.015	59.291 +- 0.023	**73.569 +- 0.012**	**71.742 +- 0.017**

Table 5-7 Adding/removing the features one by one did not produce a better result

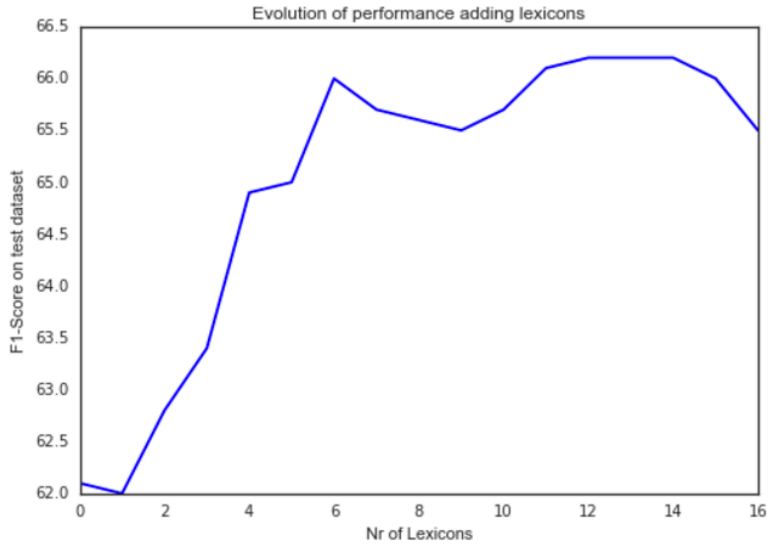

Figure 5-7 Evolution of performance when adding lexicons

In a final experiment, the individuals were “artificially" bred. The lexicons are added one by one following the rank of best to worst lexicons shown in Table 4-36. Notice in Table 5-8 that the test score is gradually improved until a certain point (line 7) where it falls back.

	Individual	train score	test score
0	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	78.7828	62.1087
1	[1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	75.4252	64.8557
2	[1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]	74.919	65.7575
3	[1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]	71.6054	65.4273
4	[1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1]	66.8649	64.9746
5	[1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1]	67.943	65.4135
6	[1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1]	68.2051	65.6203
7	[1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1]	68.2229	**65.8258**
8	[1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1]	69.2724	65.0412
9	[1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1]	69.3751	65.1055
10	[1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1]	69.5514	65.3488
11	[1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1]	69.6908	65.0394
12	[1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1]	69.6797	64.9333
13	[1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1]	69.7853	65.0446
14	[1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1]	69.8339	65.3532
15	[1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]	69.8395	65.2426
16	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]	70.4146	65.5164

Table 5-8 Adding the lexicons in the order of best individual performance

The LIWC (line 4) seems to be the detractor in this case. Still in the search for a best manual model, LIWC is not used and the pipeline is run again improving the scores. Notice that the pipeline improved almost achieving the best performance so far.

	train score	test score
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	78.7828	62.1087
[1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]	75.4252	64.8557
[1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0]	74.919	65.7575
[1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0]	71.6054	65.4273
[1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0]	73.3882	66.9122
[1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0]	73.4792	**66.9767**
[1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0]	73.0075	66.1762
[1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0]	74.1542	65.694
[1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0]	74.1801	65.2401
[1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0]	74.264	65.1439
[1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0]	74.2614	65.5168
[1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1]	74.6078	65.4697
[1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1]	74.4071	65.09
[1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1]	74.3143	65.2949
[1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]	74.3154	65.2268
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]	74.8694	66.0263

Table 5-9 Adding the lexicons in the order of best individual performance, without LIWC

In a third attempt, all the lexicons that decreased the values in Table 5-9 were removed, LIWC included. Table 5-10 shows that although some of the lexicons caused a performance drop when combined with others, they were responsible for an increase of the overall performance of the pipeline.

	train score	test score
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	78.7828	62.1087
[1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]	75.4252	64.8557
[1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0]	74.919	65.7575
[1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0]	76.6937	66.3101
[1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0]	76.8468	66.4536
[1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0]	76.3078	66.3366
[1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0]	75.0875	65.1043
[1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0]	75.2401	64.8799
[1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0]	75.4231	65.2366
[1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0]	76.1567	**66.4588**

Table 5-10 All detractors removed

In a summary, Figure 5-8 draws the progression of all the manually runs comparing the performance of them. Notice that the performance varies and removing all the detractors did not yield the best performance while manual addition without the LIWC reached the best performance. This demonstrates that sometimes the lexicon does not have great power alone but when used in conjunction with another it may add value to the pipeline. This set of manually created experiments illustrate the difficulty to do a proper lexicon combination and how the GA method enables to find a good combination of models easily.

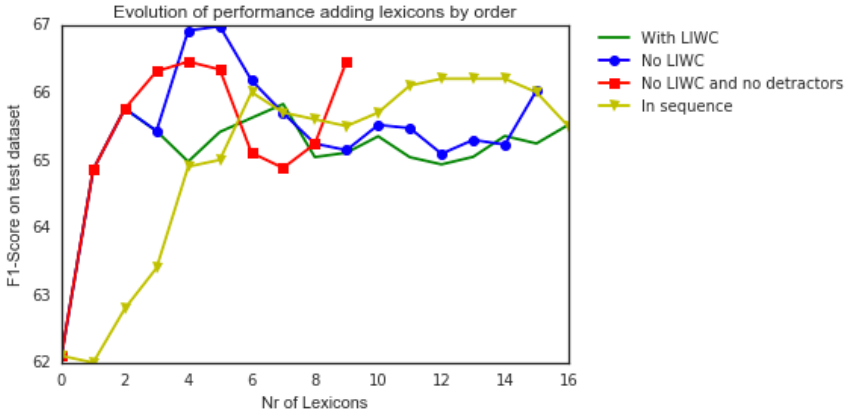

Figure 5-8 Progression curve of the 3 runs using lexicon addition by order of best individual performance

5.5. SemEval Comparison

Following the SemEval tradition, where many of the competitors submitted works in many years, the model created was also tested in the test dataset provided for the 2014 and 2015 versions. In both years, the train and dev datasets where the same used in 2013 while the test dataset was different in each year.

In 2014 [RRNS14], the test dataset of SemEval task 9, subtask B, was compromised of texts from "regular" tweets, sarcastic tweets, Live Journal and SMS. As it happened with the 2013 dataset, some of the test tweets did not exists so it was impossible to use the model to predict them. Table 5-11 shows the results achieved using the full test dataset, predicting the label “neutral” for the unknown tweets. In Table 5-12 the unknown tweets are ignored and not used in the F1-score calculus.

LiveJournal2014	SMS2013	Twitter2013	Twitter2014	Twitter2014Sarcasm
71.11	65.89	58.51	56.21	42.05

Table 5-11 Scores achieved using neutral labels for the unknown tweets

	LiveJournal2014	SMS2013	Twitter2013	Twitter2014	Twitter2014Sarcasm
0	71.11	65.89	67.12	64.07	42.05

Table 5-12 Scores achieved ignoring tweets that did not have their labels predicted

Comparing the results achieved by the best model created in this work with the other competitors of 2014, it can be seen that the model lost some positions in the rank from second in 2013 to 10th in the Twitter 2013 dataset and for the 2014 dataset it was worst achieving the 19th place. The model performed well in the SMS dataset and very poorly in the Sarcasm dataset. Table 5-13 compares the results achieved with the 3 best models created in the 2014 competition and rank the model for each of the datasets.

	SMS2013	Twitter2013	Twitter2014	Twitter2014Sarcasm	LiveJournal2014
1st	70.28	72.12	70.96	58.16	74.84
2nd	67.68	70.75	70.14	57.26	74.46
3rd	67.51	70.40	69.95	56.50	73.99
this work	65.89	67.12	64.07	42.05	71.11
ranking	7/50	10/50	19/50	35/50	11/50

Table 5-13 Results compared with 2014 competitors

The 2015 Subtask B's dataset [RNKM15] is composed of only tweets. Amongst the dataset, some were identified as having sarcasm and were ranked separately from the rest of the tweets. Table 5-14 displays the result of the model in this dataset and ranked it against 2015's competitors. Once more many of the test tweets were not available so the F1-score values displayed are considering the prediction of only the available ones.

	Twittter2015	Twittter2015Sarcasm
1st	64.84	65.77
2nd	64.59	64.91
3rd	64.27	63.62
this work	59.67	52.22
ranking	19/40	26/40

Table 5-14 Results compared with 2015 competitors

As it can be seen by the comparison with other methods, the method developed is competitive but is not a top performer. It handles well the 2013 dataset but is not able to generalize well with the 2014 and 2015 datasets. It did well with SMS achieving the 7th place but failed with the sarcasm dataset. This raises an important question that was presented previously. In sarcasm, the words meanings represent the opposite of what it means in lexicons making it hard for the methods that use BOW and Lexicons to properly measure this fact in their sentiment analysis.

5.6.Summary

In this chapter, it was presented a method that used Genetic Algorithm to search heuristically for a combination of lexicons that could create a good performing model. It is shown that this method is a viable solution as it was able to find a combination in good time and with superior results if compared with the traditional all-in or remove-some techniques used in other works.

Finally, in this chapter, the final pipeline created in this work was compared with other works presented in SemEval 2014 and 2015.

Next, chapter 6 will conclude this work with the final conclusions and future works.

Chapter 6

Conclusion

As it was shown, affective lexicons are a highly studied topic and it still contribute to the state of the art of Twitter sentiment Analysis. Just in this work it is presented seventeen different lexicons that have direct or indirect relation to affective properties and are used in affective computing in Sentiment Analysis, Emotional Analysis and Personality Recognition.

Most of the works done so far take little further the study on the lexicons power, having used few of them and in a simple manner. This work is unique as it explored each of these lexicons individually and in conjunction with others proposing the successful use of Genetic Algorithm to accelerate the search for the best combination of lexicons.

As it was hypothesized, adding lexicons improve a SA pipeline performance and emotional lexicons commonly used for personality extraction from text can indeed help to raise the predictive power on the sentiment analysis task.

As the features created by one lexicon can influence the performance of another lexicon, the task of adding lexicons and features is not linear or logic. Manual and automatic mix of lexicons were studied and, for the first time in our knowledge, Genetic Algorithm was used and shown to be a viable solution to search for good combinations of the lexicons.

Additionally to the hypothesis, the following questions are answered:

> *Q1: Are there lexicons with better performance? Is there a way to identify them?*
>
> In Table 4-36 it can be seen that there are some lexicons that have a better individual performance but it could not be found an individual characteristic that could identify in advance the best lexicons for the dataset.

Q2: Can different features created from the lexicons produce better performance than the simple count of the words and or sum of word polarities?

Sections 4.3 and 4.4 described the many tests done and demonstrated that feature construction is dependent of the lexicon. Most of them worked fine using statistical features only like minimal and max values, the sum of all polarities, etc. but in some cases using the token polarities or just some of the emotions contained in the lexicon was the best approach.

Q3: What lexicons combinations can yield the best performance? Can they all be put together to create a model with the best performance?

Sections 5.4 and 5.5 demonstrated that the combination found using the Genetic Algorithm achieved good results in SemEval's 2013 test dataset and that adding all the lexicons together is not a guarantee of the best performance. It also showed that manually doing the combination is a tricky process that may not achieve best performance.

Q4: Is there a pattern that can identify the best lexicons?

In this work it was not possible to identify any pattern. No characteristic proved to be a predictor of good lexicon performance.

A detail that was shown clearly is that the choice of algorithm, feature selection and the base pipeline matters a lot for a good model. The base model, as the name says, is the starting point that determines if the model will perform well. The features added increased the performance in 8% but the biggest part came from the feature creation of the base model and from the algorithm tuning.

Another interesting characteristic of the approach is that the lexicons features can all be calculated in parallel and the processing is very fast. That would make the process suitable for the production environment where the response time is important.

6.1. Limitations

One of the explicit limitations on this work is that the BOW model is completely dependent on the vocabulary present in the training dataset. The base model created had 927 features that are words present in the text vocabulary. If a new test dataset is presented with completely different subject, probably many of these words won't be in the new dataset and the predictive power of the model would be mainly based on the features created by the lexicons.

Another limitation is that although we have presented many different lexicons, there are others that were not used in this work as they were not referenced in the base models used in the SemEvals competition or the base work for personality extraction. This includes the works later than beginning of 2016, date the research was done.

6.2. Future Work

An immediate direct work is how the use of anthologies to identify words that carry the same meanings could lead to a higher coverage of words found in the lexicons. If a word cannot be found in any lexicon, additional searches could be done on its synonyms, raising the possibilities to have a sentiment-carrying token.

This work presented the use of Genetic Algorithms to find the model with best performance. Instead of using all the lexicons to create a search space, another direction of study would be to previously identify lexicons characteristics that could predict the good or bad performance of the models, making the search space smaller.

A third future work approach could be to use ensemble of different models to try to improve the pipeline. The GA generates many individuals with different characteristics and an ensemble of these models would be a good way to boost the performance of the pipeline.

A study like this could be done with Portuguese lexicons, identifying which ones could help a SA pipeline for short messages in Portuguese.

Finally, a broader direction of study is how to use and combine the lexicons with other Sentiment Analysis techniques that enable the pipeline to extract the best of all worlds.

Bibliography

[AbCS08] ABBASI, AHMED ; CHEN, HSINCHUN ; SALEM, ARAB: Sentiment analysis in multiple languages: Feature selection for opinion classification in Web forums. In: *ACM Transactions on Information Systems ...* Bd. 26 (2008), Nr. 3, S. 1–34 — ISBN 1046-8188

[AlPG12] ALBORNOZ, JORGE CARRILLO DE ; PLAZA, LAURA ; GERVÁS, PABLO: SentiSense: An easily scalable concept-based affective lexicon for sentiment analysis. In: *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)* (2012), S. 23–25 — ISBN 978-2-9517408-7-7

[BaES08] BACCIANELLA, STEFANO ; ESULI, ANDREA ; SEBASTIANI, FABRIZIO: Sentiwordnet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining Bd. 0 (2008), S. 2200–2204

[BaFM00] BACK, THOMAS ; FOGEL, D.B ; MICHALEWICZ, Z: Evolutionary Computation 1: Basic Algorithms and Operators. In: *Evolutionary Computation* (2000), S. 378 — ISBN 0750306645

[Bern86] BERNARD, JOHN R L: *The Macquarie thesaurus : the book of words*. Rev. ed. Aufl. : Dee Why, NSW : Macquarie Library Pty. Ltd, 1986. — 1999 reprint has subtitle: Australia's national thesaurus — ISBN 0949757144

[BiKL09] BIRD, STEVEN ; KLEIN, EWAN ; LOPER, EDWARD: *Natural language processing with Python* : „ O'Reilly Media, Inc.", 2009

[BoOs69] BOUCHER, JERRY ; OSGOOD, CHARLES E: The pollyanna hypothesis. In: *Journal Of Verbal Learning And Verbal Behavior* Bd. 8 (1969), Nr. 1, S. 1–8 — ISBN 0022-5371

[BrLa94] BRADLEY, M ; LANG, PETER J: Measuring Emotion: The Self-Assessment Semantic Differential Manikin and the. In: *Journal of Behavior Therapy and Experimental Psychiatry* Bd. 25 (1994), Nr. I, S. 49–59 — ISBN 0005-7916 (Print)n0005-7916 (Linking)

[BrLa99] BRADLEY, MARGARET MM ; LANG, PJ PETER J: Affective Norms for English Words (ANEW): Instruction Manual and Affective Ratings. In: *Psychology* Bd. Technical (1999), Nr. C-1, S. 0 — ISBN Technical report C-1

[Brow92] BROWN, PETER F.: Class-Based n-gram Models of Natural Language. In: *Computational Linguistics* (1992), Nr. 1950

[CHHE10a] CAMBRIA, ERIK ; HUSSAIN, AMIR ; HAVASI, CATHERINE ; ECKL, CHRIS: Sentic computing: Exploitation of common sense for the development of emotion-sensitive systems. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Bd. 5967 LNCS, 2010 — ISBN 3642123961, S. 148–156

[CHHE10b] CAMBRIA, ERIK ; HUSSAIN, AMIR ; HAVASI, CATHERINE ; ECKL, CHRIS: SenticSpace: Visualizing opinions and sentiments in a multi-dimensional vector space. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Bd. 6279 LNAI, 2010 — ISBN 3642153836, S. 385–393

[COCC89] COLBY, B. N. ; ORTONY, ANDREW ; CLORE, GERALD L. ; COLLINS, ALLAN: The Cognitive Structure of Emotions. In: *Contemporary Sociology* Bd. 18 (1989), Nr. 6, S. 957 — ISBN 0521353645

[CoMc95] COSTA, P T ; MCCRAE, R R: Domains and facets: hierarchical personality assessment using the revised NEO personality inventory. In: *Journal of personality assessment* Bd. 64 (1995), Nr. 1, S. 21–50 — ISBN 0022-3891r1532-7752

[Coun10] COUNCILL, ISAAC G: What's Great and What's Not: Learning to Classify the Scope of Negation for Improved Sentiment Analysis (2010), Nr. July, S. 51–59

[CoVa95] CORTES, CORINNA ; VAPNIK, VLADIMIR: Support-vector networks. In: *Machine Learning* Bd. 20 (1995), Nr. 3, S. 273–297

[Cram02] CRAMER, J.S.: The Origins of Logistic Regression. In: *Tinbergen Institute Working Paper* Bd. 2002–119/4 (2002), Nr. june, S. 1–19

[CSHH10] CAMBRIA, ERIK ; SPEER, ROBERT ; HAVASI, CATHERINE ; HUSSAIN, AMIR: SenticNet: A Publicly Available Semantic Resource for Opinion Mining (2010), S. 14–18

[DaCh01] DAS, S ; CHEN, M: Yahoo! for Amazon: Extracting market sentiment from stock message boards. In: *Proceedings of the Asia Pacific finance ...* Bd. 35 (2001), S. 43

[DDLL03] DAVE, K. ; DAVE, K. ; LAWRENCE, S. ; LAWRENCE, S. ; PENNOCK, D.M. ; PENNOCK, D.M.: Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In: *Proceedings of the 12th international conference on World Wide Web* (2003), S. 519–528 — ISBN 1581136803

[DSLY08] DING, XIAOWEN ; STREET, S MORGAN ; LIU, BING ; YU, PHILIP S: A Holistic Lexicon-based Approach to Opinion Mining. In: *Proceedings of the 2008 International Conference on Web Search and Data Mining* (2008), S. 231–240 — ISBN 978-1-59593-927-2

[Ekma92] EKMAN, P: Are there basic emotions? In: *Psychological review* Bd. 99 (1992), Nr. 3, S. 550–553 — ISBN 1939-1471 (Electronic); 0033-295X (Print)

[Fell05] FELLBAUM, C: WordNet and wordnets. In: *Encyclopedia of Language & Linguistics* Bd. 13 (2005), S. 665–670 — ISBN 978-0-08-044854-1

[FrGe13] FRANCISCO, VIRGINIA ; GERVÁS, PABLO: EMOTAG: AN APPROACH TO AUTOMATED MARKUP OF EMOTIONS IN TEXTS. In: *Computational Intelligence* Bd. 29 (2013), Nr. 4, S. 680–721

[GoBH09] GO, ALEC ; BHAYANI, RICHA ; HUANG, LEI: Twitter Sentiment Classification using Distant Supervision. In: *Processing* Bd. 150 (2009), Nr. 12, S. 1–6 — ISBN 1012341234

[GSOD17] GRANATYR, JONES ; SCALABRIN, EDSON EMÍLIO ; OSMAN, NARDINE ; DIAS, JOÃO ; NUNES, MARIA AUGUSTA SILVEIRA NETTO ; MASTHOFF, JUDITH ; ENEMBRECK, FABRÍCIO ; LESSING, OTTO ROBERT ; U. A.: The Need for Affective Trust Applied to Trust and Reputation Models. In: *ACM Computing Surveys* Bd. 50 (2017), Nr. 4, S. 1–36

[GüFu13] GÜNTHER, TOBIAS ; FURRER, LENZ: Gu-mlt-lt: Sentiment analysis of short messages using linguistic features and stochastic gradient descent Bd. 2 (2013), Nr. 4, S. 328–332

[GVJT14] GÜNTHER, TOBIAS ; VANCOPPENOLLE, JEAN ; JOHANSSON, RICHARD ; TOBIAS, G: RTRGO: Enhancing the GU-MLT-LT System for Sentiment Analysis of Short Messages. In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)* (2014), Nr. SemEval, S. 497–502

[HaBB15] HAMDAN, HUSSAM ; BELLOT, PATRICE ; BECHET, FREDERIC: Lsislif : Feature Extraction and Label Weighting for Sentiment Analysis in Twitter. In: *SemEval2015* (2015), Nr. SemEval, S. 568–573

[HaPK14] HA, ILKYU ; PARK, HOHWAN ; KIM, CHONGGUN: Analysis of Twitter Research Trends based on SLR (2014), S. 774–778 — ISBN 9788996865025

[HHWW00] HATZIVASSILOGLOU, V. ; HATZIVASSILOGLOU, V. ; WIEBE, J.M. ; WIEBE, J.M.: Effects of adjective orientation and gradability on sentence subjectivity. In: *Proceedings of the 18th conference on Computational linguistics-Volume 1* (2000), S. 299–305 — ISBN 155860717X

[Holl75] HOLLAND, JOHN H.: *Adaptation in Natural and Artificial Systems*, 1975 — ISBN 0262581116

[HPBS15] HAGEN, MATTHIAS ; POTTHAST, MARTIN ; BÜCHNER, MICHEL ; STEIN, BENNO: Webis : An Ensemble for Twitter Sentiment Detection. In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 15)* (2015), Nr. SemEval, S. 582–589 — ISBN 978-3-319-16353-6

[HuLi04] HU, MINQING ; LIU, BING: Mining and summarizing customer reviews. In: *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining KDD 04* Bd. 4 (2004), S. 168 — ISBN 1581138889

[InDa10] INDURKHYA, NITIN ; DAMERAU, FRED J: *Handbook of Natural Language Processing*. Bd. 2, 2010 — ISBN 9781420085921

[JoDL91] JOHN, OLIVER P ; DONAHUE, E. M. ; L, KENTLE. R.: *The Big Five Inventory--Versions 4a and 54*. Bd. 37, 1991

[JYZL11] JIANG, LONG ; YU, MO ; ZHOU, MING ; LIU, XIAOHUA ; ZHAO, TIEJUN: Target-dependent Twitter Sentiment Classification. In: *Computational Linguistics* (2011), S. 151–160 — ISBN 9781932432879

[KiZM14] KIRITCHENKO, SVETLANA ; ZHU, XIAODAN ; MOHAMMAD, SAIF M.: Sentiment analysis of short informal texts. In: *Journal of Artificial Intelligence Research* (2014)

[KSTA15] KOLCHYNA, OLGA ; SOUZA, THARSIS T. P. ; TRELEAVEN, PHILIP ; ASTE, TOMASO: Twitter Sentiment Analysis: Lexicon Method, Machine Learning Method and Their Combination (2015)

[KuFr79] KUCERA, H ; FRANCIS, W NELSON: *A Standard Corpus of Present-Day Edited American English, for use with Digital Computers- MANUAL OF INFORMATION*, 1979

[LaKa50] LASSWELL, HAROLD D. ; KAPLAN, ABRAHAM: Power and Society: A Framework for Political Inquiry (1950) — ISBN 1412852803

[LiCC15] LIMA, ANA CAROLINA E S ; DE CASTRO, LEANDRO NUNES ; CORCHADO, JUAN M.: A polarity analysis framework for Twitter messages. In: *Applied Mathematics and Computation* Bd. 270, Elsevier Ltd. (2015), S. 756–767

[LiZh12] LIU, BING ; ZHANG, LEI: Mining Text Data. In: AGGARWAL, C. C. ; ZHAI, C. (Hrsg.): . Boston, MA : Springer US, 2012 — ISBN 978-1-4614-3223-4, S. 415–463

[MaFu11] MAYNARD, DIANA ; FUNK, ADAM: Automatic detection of political opinions in tweets. In: *CEUR Workshop Proceedings* Bd. 718 (2011), S. 81–92 — ISBN 9783642259524

[MCCD13] MIKOLOV, TOMAS ; CHEN, KAI ; CORRADO, GREG ; DEAN, JEFFREY: Distributed Representations of Words and Phrases and their Compositionality. In: *Nips* (2013), S. 1–9 — ISBN 2150-8097

[MeHK14] MEDHAT, WALAA ; HASSAN, AHMED ; KORASHY, HODA: Sentiment analysis algorithms and applications: A survey. In: *Ain Shams Engineering Journal* Bd. 5, Faculty of Engineering, Ain Shams University (2014), Nr. 4, S. 1093–1113

[Mitc97] MITCHELL, TOM M: *Machine Learning*. Bd. 1, 1997 — ISBN 0070428077

[MMSP14] MUNEZERO, MYRIAM ; MONTERO, CALKIN SUERO ; SUTINEN, ERKKI ; PAJUNEN, JOHN: Are they different? affect, feeling, emotion, sentiment, and opinion detection in text. In: *IEEE Transactions on Affective Computing* Bd. 5 (2014), Nr. 2, S. 101–111 — ISBN 1949-3045 VO - 5

[MoDD09] MOHAMMAD, SAIF ; DUNNE, CODY ; DORR, BONNIE: Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus. In: *EMNLP '09 Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing* Bd. 2 (2009), Nr. August, S. 599–608 — ISBN 978-1-932432-62-6

[Moha12] MOHAMMAD, SM: # Emotional tweets. In: *Proceedings of the First Joint Conference on Lexical ...* (2012), S. 246–255

[Moha15] MOHAMMAD, SAIF M: Sentiment Analysis: Detecting Valence, Emotions, and Other Affectual States from Text. In: *Emotion Measurement* (2015)

[MoKZ13] MOHAMMAD, SAIF M ; KIRITCHENKO, SVETLANA ; ZHU, XIAODAN: NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets. In: *Proceedings of the seventh international workshop on Semantic Evaluation Exercises (SemEval-2013)* Bd. 2 (2013), Nr. SemEval, S. 321–327

[MoTu13] MOHAMMAD, SAIF M. ; TURNEY, PETER D.: Crowdsourcing a word-emotion association lexicon. In: *Computational Intelligence*. Bd. 29, 2013 — ISBN 978-3-319-39930-0, S. 436–465

[MSHO14] MIURA, YASUHIDE ; SAKAKI, SHIGEYUKI ; HATTORI, KEIGO ; OHKUMA, TOMOKO: TeamX: A Sentiment Analyzer with Enhanced Lexicon Mapping and Weighting Scheme for Unbalanced Data. In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)* (2014), Nr. SemEval, S. 628–632

[MTAR12] MARTÍNEZ-CÁMARA, EUGENIO ; TERESA MARTÍN-VALDIVIA, M ; ALFONSO UREÑA-LÓPEZ, L ; RTURO MONTEJO-RÁEZ, A: Sentiment analysis in Twitter. In: *Natural Language Engineering Natural Language Engineering Natural Language Engineering* Bd. 201672135 (2012), Nr. 201, S. 1–28

[NaKR13] NAKOV, P ; KOZAREVA, Z ; RITTER, A: Semeval-2013 task 2: Sentiment analysis in twitter (2013)

[NiHu06] NIGAM, KAMAL ; HURST, MATTHEW: Computing Attitude and Affect in Text: Theory and Applications. In: SHANAHAN, J. G. ; QU, Y. ; WIEBE, J. (Hrsg.): . Dordrecht : Springer Netherlands, 2006 — ISBN 978-1-4020-4102-0, S. 265–279

[OlCA16] OLIVEIRA, NUNO ; CORTEZ, PAULO ; AREAL, NELSON: Stock market sentiment lexicon acquisition using microblogging data and statistical measures. In: *Decision Support Systems* Bd. 85, Elsevier B.V. (2016), S. 62–73 — ISBN 0167-9236

[OODG13] OWOPUTI, OLUTOBI ; O'CONNOR, BRENDAN ; DYER, CHRIS ; GIMPEL, KEVIN ; SCHNEIDER, NATHAN ; SMITH, NOAH A: Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters. In: *Proceedings of NAACL-HLT 2013* (2013), Nr. June, S. 380–390 — ISBN 9781937284473

[PaLe08] PANG, BO ; LEE, LILLIAN: Opinion Mining and Sentiment Analysis. In: *Foundations and Trends in Information Retrieval* Bd. 2 (2008), Nr. 1–2, S. 1–135 — ISBN 1601981503

[PaLV02] PANG, BO ; LEE, LILLIAN ; VAITHYANATHAN, SHIVAKUMAR: Thumbs up?: sentiment classification using machine learning techniques. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (2002), S. 79–86

[PaPa10] PAK, ALEXANDER ; PAROUBEK, PATRICK: Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In: *In Proceedings of the Seventh Conference on International Language Resources and Evaluation* (2010), S. 1320–1326 — ISBN 2951740867

[PBJB15] PENNEBAKER, JAMES W ; BOYD, R ; JORDAN, K ; BLACKBURN, K: The development and psychometric properties of LIWC2015 (2015), Nr. SEPTEMBER 2015, S. 1–22

[PCEP16] PAIM, ALDO M ; CAMATI, RICARDO S ; ENEMBRECK, FABRÍCIO ; PÓS-GRADUAÇÃO, PROGRAMA DE ; PPGIA, INFORMÁTICA: Inferência de Personalidade a partir de textos em Português utilizando Léxico Linguístico e Aprendizagem de Máquina (2016), Nr. i, S. 481–492

[PCHH16] PORIA, SOUJANYA ; CAMBRIA, ERIK ; HOWARD, NEWTON ; HUANG, GUANG BIN ; HUSSAIN, AMIR: Fusing audio, visual and textual clues for sentiment analysis from multimodal content. In: *Neurocomputing* Bd. 174, Elsevier (2016), S. 50–59 — ISBN 0925-2312

[PeVa11] PEDREGOSA, FABIAN ; VAROQUAUX, G: Scikit-learn: Machine learning in Python. In: *... of Machine Learning ...* Bd. 12 (2011), S. 2825–2830 — ISBN 9781783281930

[PGCH14] PORIA, SOUJANYA ; GELBUKH, ALEXANDER ; CAMBRIA, ERIK ; HUSSAIN, AMIR ; HUANG, GUANG BIN: EmoSenticSpace: A novel framework for affective common-sense reasoning. In: *Knowledge-Based Systems* Bd. 69, Elsevier B.V. (2014), Nr. 1, S. 108–123

[Plut01] PLUTCHIK, ROBERT: The nature of emotions: Human emotions have deep evolutionary roots. In: *American Scientist* Bd. 89 (2001), Nr. 4 — ISBN 0003-0996

[Port80] PORTER, M.F.: An algorithm for suffix stripping. In: *Program: electronic library and information systems* Bd. 14 (1980), Nr. 3 — ISBN 1558604545

[Pott11] POTTS, CHRISTOPHER: On the negativity of negation ∗ (2011), S. 636–659

[Powe98] POWERS, DAVID M W: Applications and Explanations of Zipf's Law. In: *NeMLaP3/CoNLL '98 Proceedings of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning* (1998), S. 151–160 — ISBN 0-7258-0634-6

[RaRo16] RANGEL, FRANCISCO ; ROSSO, PAOLO: On the impact of emotions on author profiling. In: *Information Processing and Management* Bd. 52, Elsevier B.V. (2016), Nr. 1, S. 73–92

[RBCC13] RECKMAN, HILKE ; BAIRD, CHEYANNE ; CRAWFORD, JEAN ; CROWELL,

RICHARD ; MICCIULLA, LINNEA ; SETHI, SARATENDU ; VERESS, FRUZSINA: teragram: Rule-based detection of sentiment phrases using SAS Sentiment Analysis. In: *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)* Bd. 2 (2013), Nr. SemEval, S. 513–519

[ReSB15] REITAN, JOHAN ; SCIENCE, INFORMATION ; BUNGUM, LARS: Negation Scope Detection for Twitter Sentiment Analysis (2015), Nr. Wassa, S. 99–108

[RFGP12] RAINVILLE, FRANÇOIS-MICHEL DE ; FORTIN, FÉLIX-ANTOINE ; GARDNER, MARC-ANDRÉ ; PARIZEAU, MARC ; GAGNÉ, CHRISTIAN: DEAP : A Python Framework for Evolutionary Algorithms. In: *Companion proc. of the Genetic and Evolutionary Computation Conference* (2012), S. 85–92 — ISBN 9781450311786

[RiPW06] RILOFF, ELLEN ; PATWARDHAN, SIDDHARTH ; WIEBE, JANYCE: *Feature subsumption for opinion analysis*, 2006 — ISBN 1932432736

[RiWi03] RILOFF, ELLEN ; WIEBE, JANYCE: Learning extraction patterns for subjective expressions. In: *Proceedings of the 2003 conference on Empirical methods in natural language processing* - Bd. 10 (2003), S. 105–112

[RNKM15] ROSENTHAL, SARA ; NAKOV, PRESLAV ; KIRITCHENKO, SVETLANA ; MOHAMMAD, SAIF M. ; RITTER, ALAN ; STOYANOV, VESELIN: SemEval-2015 Task 10: Sentiment Analysis in Twitter. In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (2015), Nr. SemEval, S. 451–463

[RRNS14] ROSENTHAL, SARA ; RITTER, ALAN ; NAKOV, PRESLAV ; STOYANOV, VESELIN: SemEval-2014 Task 9: Sentiment Analysis in Twitter. In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)* (2014), Nr. SemEval, S. 73–80

[RuMe77] RUSSELL, J A ; MEHRABIAN, A: Evidence for a Three Factor Theory of Emotions. In: *Journal of Research in Personality* Bd. 11 (1977) — ISBN 0092-6566

[Sche05] SCHERER, KLAUS R: What are emotions? And how can they be measured? In: *Social Science Information* Bd. 44 (2005), Nr. 4, S. 695–729 — ISBN 0539-0184

[SeMo15] SEVERYN, ALIAKSEI ; MOSCHITTI, ALESSANDRO: UNITN : Training Deep Convolutional Neural Network for Twitter Sentiment Classification (2015), Nr. SemEval, S. 464–469

[ShDe12] SHARMA, ANUJ ; DEY, SHUBHAMOY: A comparative study of feature selection and machine learning techniques for sentiment analysis. In: *Proceedings of the 2012 ACM Research in Applied Computation Symposium on - RACS '12*. New York, New York, USA : ACM Press, 2012 — ISBN 9781450314923, S. 1

[SHDS66] STONE, PHILIP J. ; HARTMAN, JOHN J. ; DUNPHY, DEXTER C. ; SMITH, MARSHALL S. ; OGILVIA, DANIEL M.: The General Inquirer: A Computer Approach to Content Analysis. In: *American Sociological Review* Bd. 32 (1966), Nr. 5, S. 859 — ISBN 9780262690119

[StVa04] STRAPPARAVA, CARLO ; VALITUTTI, A.: WordNet-Affect: an affective extension of WordNet. In: *Proceedings of the 4th International Conference on Language Resources and Evaluation* (2004), S. 1083–1086

[TaPe10] TAUSCZIK, YLA R ; PENNEBAKER, JAMES W: The Psychological Meaning of Words: LIWC and Computerized Text Analysis Methods (2010)

[TBPC10] THELWALL, MIKE ; BUCKLEY, KEVAN ; PALTOGLOU, GEORGIOS ; CAI, DI: Sentiment Strength Detection in Short Informal Text. In: *The American Society for Informational science and technology* Bd. 61 (2010), Nr. 12, S. 2544–2558 — ISBN 9783848215430

[TBTV11] TABOADA, MAITE ; BROOKE, JULIAN ; TOFILOSKI, MILAN ; VOLL, KIMBERLY ; STEDE, MANFRED: Lexicon-Based Methods for Sentiment Analysis. In: *Computational Linguistics* Bd. 37 (2011), Nr. 2, S. 267–307 — ISBN 0891-2017

[Tong01] TONG, RICHARD: An Operational System for Detecting and Tracking Opinions in On-line Discussions. In: *Working Notes of the SIGIR Workshop on Operational Text Classification*. New Orleans, Louisianna, 2001, S. 1–6

[TSSW10] TUMASJAN, ANDRANIK ; SPRENGER, TO ; SANDNER, PG ; WELPE, IM: 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 — ISBN 0894439310386, S. 178–185

[Turn02] TURNEY, PETER D: Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews (2002), Nr. July, S. 417–424

[TWQL14] TANG, DUYU ; WEI, FURU ; QIN, BING ; LIU, TING ; ZHOU, MING: Coooolll: A Deep Learning System for Twitter Sentiment Classification. In: *Semeval-2014* (2014), Nr. SemEval, S. 208–212 — ISBN 9781941643242

[TWQZ14] TANG, DUYU ; WEI, FURU ; QIN, BING ; ZHOU, MING ; LIU, TING: Building Large-Scale Twitter-Specific Sentiment Lexicon: a Representation Learning Approach. In: *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)* (2014), S. 172–182

[ViBo12] VIVACQUA, ADRIANA S. ; BORGES, MARCOS R.S.: Taking advantage of collective knowledge in emergency response systems. In: *Journal of Network and Computer Applications* Bd. 35 (2012), Nr. 1, S. 189–198 — ISBN 1084-8045

[ViMo00] VINCIARELLI, ALESSANDRO ; MOHAMMADI, GELAREH: A Survey of Personality Computing (2000), S. 1–20

[WaCa16] WANG, LU ; CARDIE, CLAIRE: A Piece of My Mind: A Sentiment Analysis Approach for Online Dispute Detection (2016)

[Whis09] WHISSELL, CYNTHIA: Using the Revised Dictionary of Affect in Language to quantify the emotional undertones of samples of natural language. In: *Psychological reports* Bd. 105 (2009), S. 509–521

[Wils88] WILSON, MICHAEL: MRC Psycholinguistic Database : Machine-usable dictionary , version 2 .00. In: *Behavior Research Methods, Instruments, & Computers* Bd. 20 (1988), Nr. 1, S. 6–10 — ISBN 0743-3808

[WiWC05] WIEBE, JANYCE ; WILSON, THERESA ; CARDIE, CLAIRE: Annotating expressions of opinions and emotions in language. In: *Language Resources and Evaluation* Bd. 39 (2005), Nr. 2–3, S. 165–210 — ISBN 1574-020X

[ZGDH11] ZHANG, LEI ; GHOSH, RIDDHIMAN ; DEKHIL, MOHAMED ; HSU, MEICHUN ; LIU, BING: Combining lexiconbased and learning-based methods for twitter sentiment analysis. In: *HP Laboratories, Technical ...* (2011)

[Zhan04] ZHANG, HARRY: The Optimality of Naive Bayes. In: *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference FLAIRS 2004* Bd. 1 (2004), Nr. 2, S. 1–6 — ISBN 978-1-57735-201-3