UNIVERSITÀ DEGLI STUDI DI TRENTO

CIMeC - Center for Mind/Brain Sciences

Master's Degree in Cognitive Science

Emotional Language in Persuasive Communication

Tutor
Dr. Raffaella Bernardi
Co-Tutor
Dr. Carlo Strapparava

Student
Felicia Oberarzbacher

Academic Year 2014/2015

UNIVERSITY OF TRENTO
Center for Mind/Brain Sciences (CIMeC)

Abstract

Felicia Oberarzbacher

This work is concerned with the relation between emotion-evoking words and audience reactions in persuasive communication. It merges the research that has been done on computational emotional analysis in *natural language processing (NLP)* and the automatic analysis and detection of persuasive attempts in political speeches, namely in CORPS (CORpus of Political Speeches). The aim of this study is to gain insight into the impact of emotional words on audience reactions and to provide indications for further research by showing statistics of emotions in political speeches, by giving an overview of emotional lexical resources and by setting a baseline for distinguishing audience reactions based on emotions.

A computational, knowledge-based approach has been taken to detect emotional words, i.e. unigrams, before audience reactions using NRC Word-Emotion Association Lexicon (EmoLex). Quantitative analyses have been made that illustrate the high frequency of the positive emotions *trust* and *joy* in the corpus. They also show that negative emotions like *anger* are present before negative audience reactions as well as before positive ones, but being especially frequent before the audience reaction *booing*. In comparison to BNC the rate of emotional words in CORPS is higher, especially for positive emotions like *joy* and *trust*. A Machine learning experiment has been conducted with support vector machines (SVM) trying to distinguish between audience reactions, using nothing but the emotions occurring before them as features. The results show that there is a strong relation between emotional words and the audience reactions *booing* and *laughter*, and a weak one for *cheers* and *applause*.

Acknowledgements

At first, I must express my very profound gratitude to *Lorenzo Gatti* of HLT-NLP at Fondazione Bruno Kessler, who gave me invaluable advice and without whom I would have started and finished this thesis months later.

I would like to thank my thesis supervisor *Dr. Raffaella Bernardi* of the CIMeC at University of Trento for responding promptly to emails and being very supportive.

I would like to thank my co-supervisor *Dr. Carlo Strapparava* of HLT-NLP at Fondazione Bruno Kessler for providing the corpus, which was the basis of the work and for being very positive and encouraging.

I would also like to thank *Lisa Freise* of CIMeC at University of Trento to have read the thesis and given constructive comments.

Last but not least I would like to thank my parents, who have always supported me. Thank you.

Felicia Oberarzbacher

Contents

1. Introduction

The objective of this thesis is to understand the distribution of emotions and emotion-evoking words in persuasive communication, with special emphasis on the relation between emotional language and audience reactions, using the CORpus of tagged Political Speeches (CORPS) [Guerini et al., 2013b].

As persuasion is widely used in politics, advertising and in everyday human interactions, it is useful to understand the factors leading to successful persuasion. Although persuasion is also possible using logical argumentation only, in practice persuasive communication usually takes advantage of emotions and attempts to evoke them in the audience [Petty and Cacioppo, 1986]. Therefore, it is interesting to explore the relation between conveyed emotions and audience reactions.

Previous work investigating CORPS focused on the polarity (valence) before audience reactions, i.e. *positive*, *negative* [Guerini et al., 2008] or on the prediction of a *positive-ironical* audience reaction versus no audience reaction [Strapparava et al., 2010]. However, psychological research on emotions and persuasion indicates that distinct emotions have a different impact on the persuasiveness of an argument [Griskevicius et al., 2009], calling for a more fine-grained, beyond-sentiment approach which is taking into account specific emotions like *anger*, *fear*, *joy* or *trust*, which is exactly the scope of this study.

As a computational approach to an analysis of specific emotions in persuasive communication has not been taken yet, this work provides a baseline for determining the relation between emotions and audience reactions. Furthermore, it also indicates possible directions for future computational research on emotions and persuasiveness.

In this thesis, the absolute frequencies of words associated to different emotions before audience reactions, the emotional words in the whole corpus and in text passages not followed by an audience reaction will be compared, as well as frequencies normalized by window length. Three different emotional lexical resources have been tested for building frequency lists of emotional words, and the most suitable one has been chosen for the rest of the analysis.

The assumption has been made that before audience reactions the distribution of emotional words should be different from the distribution in text passages not followed by audience reactions. Given that there is no clear boundary between text passages preceding an audience reaction and text passages that are not followed by an audience reaction, audience reaction windows of different sizes have been extracted. In the following, some figures will be shown for the varying window sizes.

Furthermore, the emotional frequencies of CORPS will be confronted with the emotional frequencies of non-persuasive texts, sampled from the BNC (British Na-

TIONAL CORPUS), in order to observe differences in their emotional distribution. Besides, for different audience reactions, the words with the highest persuasive impact with their associated emotion will be shown, which have been extracted using a method introduced by Guerini et al. [2008].

Finally, the results of a machine learning experiment trying to predict different audience reactions using emotions only will be presented.

2. Related Work

2.1 Emotions and Persuasion

2.1.1 Emotions

When making a computational analysis of emotions, a set of emotions has of course to be chosen. This raises the question of which emotions actually exist, and whether some of them are more important than others.

In the 1980s, psychologists debated whether emotions are preceded by cognition [Lazarus, 1984] or whether they are instinctual [Zajonc, 1984]. To organize the field of emotions several psychologists, e.g. Ekman [1992] and Plutchik [1980] presented a few criteria that can be used to define a set of "basic" emotions. One such criterion is, for example, whether an emotion elicits a facial expression that is universally recognized across different cultures. In particular, *anger*, *disgust*, *fear*, *joy*, *sadness* and *surprise* were identified as basic, and in the case of Plutchik also *anticipation* and *trust*. Emotions of longer duration have been considered as "moods", and other emotions that are not recognized as basic were defined as mixtures or variations in intensity of the basic emotions [Plutchik, 1980].

Although the notion of basic emotions is not universally accepted amongst psychologists [Ortony and Turner, 1990], it gives at least an indication of which ones could be emotional categories that are reasonably universal amongst large groups of people.

2.1.2 Persuasion

Although there is no universally-accepted definition of persuasion, most theories agree that persuasion changes the mental state of the receiver by means of communication [Guerini et al., 2011]. One possible definition of persuasion is the following:

> "Persuasion is a form of attempted influence in the sense that it seeks to alter the way others think, feel, or act, but it differs from other forms of influence. [...] It affects their sense of what is true or false, probable or improbable; their evaluations of people, events, ideas, or proposals; their private and public commitments to take this or that action; and perhaps even their basic values and ideologies." [Simons, 1976]

Thus, persuasion is neither coercion, like "Confess or I'll fire you.", nor an inducement, like "If you print the article, I'll pay you $10.000." It predisposes others but does not impose, as it is addressed to individuals having some degree of freedom of decision.

Besides, it has to be differentiated between rational argumentation and persuasion, as persuasion exceeds the mere presentation of facts. Whereas argumentation is primarily concerned with stating valid arguments, persuasion is primarily concerned with convincing by making use of arguments [Blair, 2012]. Although persuasion can be misleading by favoring some arguments and disregarding others, already Aristotle has noted that – in matters of judgement where uncertainty is involved – persuasion is legitimate, as in controversial issues even experts can disagree on the facts, their relevance and their implications [Simons, 1976].

2.1.3 Emotions and Persuasiveness

In practice, in everyday life people have to make decisions without having the time, ability or motivation to understand all the relevant facts and implications as a basis of their decisions. Consequently, there are different routes to persuasion: the content-oriented "systematic processing", and "heuristic processing" [Chaiken, 1980, 1987], also referred to as the "central route to persuasion" and the "peripheral route to persuasion" [Petty and Cacioppo, 1984]. Whereas earlier work on persuasion was based mainly on the view that recipients of persuasion are cognitively processing the content of the persuasive message, in the 1970s an increasing number of psychological studies found that often judgements are made based on extrinsic persuasive cues involving only minimal information processing [Chaiken, 1987]. One of the factors determining how the recipient of a persuasive attempt processes its content, and whether he takes the central or peripheral route, is his emotional state.

There are different models regarding the impact of emotions on persuasion techniques:

- a general arousal model predicts the effectiveness of a persuasive attempt depending on the level of arousal, e.g. in a state of strong arousal the capacity to process complex arguments should decrease and peripheral cues should become more important [Sanbonmatsu and Kardes, 1988];
- an affective valence model predicts the effectiveness of a persuasive attempt depending on valence (good or bad mood), i.e. recipients in a bad mood should be more analytical and therefore less easily persuaded by weak arguments [Schwarz and Bless, 1991];
- an evolutionary model predicts that persuasiveness depends on both the specific emotion evoked and the kind of persuasion technique applied. According to the evolutionary model emotions have been an evolutionary advantage and have a specific purpose. Therefore each emotion induces a different behavior: in a context of *fear* it should be more effective to promote a product as appreciated by many people, whereas in a context of romantic desire it should be more effective to promote a product as special and rare [Griskevicius et al., 2009].

2.2 Automatic Analysis of Emotions in NLP

There has been a lot of work in the area of sentiment analysis and opinion mining that concentrated on *positive*, *negative* and *neutral* sentiment which has been summarized by Pang and Lee [2008]. Only in the course of the last decade interest in the area of automatic detection of specific emotions has been growing.

Aman and Szpakowicz [2007] took features from WordNet-Affect and General Inquirer [Stone et al., 1966] (both in isolation and together with non-linguistic features, such as punctuation and emoticons) for testing whether a Naive Bayes and an SVM classifier can distinguish between emotional and non-emotional sentences, in a corpus created from manually annotated blog posts.

A shared task on emotion recognition in news headlines was organized at SemEval 2007 [Strapparava and Mihalcea, 2007], and 3 system participated: UPAR7, UA and SWAT. UPAR7 [Chaumartin, 2007] identifies the opinions associated with the topic of the article by exploiting syntactic information, and then detects their emotion by using a combination of SentiWordNet [Baccianella et al., 2010], WordNet-Affect and the training data released for the task. UA-ZBSA [Kozareva et al., 2007] uses the Pointwise Mutual Information (PMI) between content words in a headline and an emotion to determine which emotion is associated to each particular article. The PMI is calculated using the number of retrieved results from three different search engines, using a technique similar to the one presented by Turney [2002]. SWAT-MP [Katz et al., 2007] is a supervised system trained on the data provided for the SemEval task (250 headlines), plus 1,000 additional headlines manually annotated. The emotional score of each word is calculated by averaging over the scores of the headlines where that word is present, while the score of a headline is given by the average of the scores of its content words. The system also uses synonym expansion through the Roget's Thesaurus, to mitigate the problem of data sparsity.

Strapparava and Mihalcea [2008] have implemented and compared five different approaches to automatically identify the six Ekman emotions *anger*, *disgust*, *fear*, *joy*, *sadness* and *surprise* in sentences, i.e. news headlines. As a baseline approach they were checking the presence of words in WordNet-Affect. Moreover they measured the similarity between words and emotional categories using a vector space model built with a variation of Latent Semantic Analysis (LSA) using three different ways for representing emotions of WordNet-Affect. In contrast to the knowledge-based approaches mentioned before – knowledge-based because they are relying on an emotional lexicon – they also tried a corpus based approach deploying a Naive Bayes classifier that has been trained on blog data annotated for emotions. In the same work, the authors compare their results with those of the systems that participated in the SemEval 2007 task, mentioned before.

More recently, Ghazi et al. [2010] presented an approach using hierarchical classification to detect emotions. The authors first discriminated between emotive and non-emotive texts, then tried to assign an emotion label to the emotive texts. They also used different methods and features, from a simple bag-of-word approach to lexical features taken from WordNet-Affect and the Roget's Thesaurus.

Also Brynielsson et al. [2013] use a bag-of-word approach. In this work the authors tried to discriminate between few different emotions (*anger*, *fear*, *positive*, *other*) and measured the effect of different parameters, such as using stemming or removing stopwords.

Another study was focused on determining the importance of syntactical and semantical information for emotion detection [Özbal and Pighin, 2013]. To do so, the authors used an SVM approach with a Tree Kernel function, that was enriched with linguistic information from WordNet-Affect and SentiWordNet.

Finally, in addition to works trying to detect emotions in a given text, other works used emotions as features for a higher-level task. For example, in a very recent work by Poels and Hoste [2015], it was proposed that a system for emotion detection in tweets could be very useful during organizational crisis, as they could help in adapting corporate crisis response strategies, while Ullah et al. [2016] built a predictor of the rating of product reviews, where emotional bigrams (in addition to word polarities) are taken into consideration.

3. Corpus and Lexical Resources

3.1 The Corpus CORPS

The corpus has been constructed by Marco Guerini, Carlo Strapparava and Oliviero Stock from FBK-Irst and its first version has been released in 2008 [Guerini et al., 2008], with a second version – extended and revised – was released in 2013 [Guerini et al., 2013b]. CORPS (CORpus of tagged Political Speeches) was built in order to automatically produce and analyse persuasive communication.

There are about 3,600 speeches in the corpus, about 7.9 million words, and more than 67,000 tags. It is a corpus of monological political speeches collected from the Internet, delivered at mass gatherings by native English speakers. The corpus is annotated with seven different tags that indicate the reactions of the audience, namely *applause*, *laughter*, *cheers*, *booing*, *spontaneous-demonstration*, *standing-ovation* and *sustained applause* (*applause* and *laughter* being the most frequent ones). Additionally the corpus has been annotated with meta-data regarding the speech (title, event, speaker, date, description).

Rather than that indicating that the audience has actually been persuaded successfully, the audience reaction tags show that a persuasive attempt was made by the speaker and has been recognized by the audience, e.g. they reacted to keywords or themes [Guerini et al., 2013b].

3.2 Emotional Lexical Resources

3.2.1 Overview of Emotional Lexical Resources

Emotional lexicons are lexicons containing entries that can be lemmata or stems of unigrams or n-grams associated to different emotional categories like *anger*, *fear*, *joy* or *sadness*, which are more fine-grained than just sentiment, i.e. *positive* and *negative*.

There are several resources that differ in many aspects, such as their number of entries, the emotional categories considered, the way they were created (manually vs. automatically) and consequently their reliability, the type of annotation (simple binary values vs. a more fine-grained value indicating the strength of the association or the intensity of the emotion), the choice of expressions considered (e.g. unigrams vs. n-grams, stems or lemmata, parts of speech), the included linguistic information (e.g. PoS tag, synset), etc.

The resources also differ in the range of emotional association they considered, as some contain only a strictly *direct* emotional association, others a more *indirect* one. For example, "terrified" or "anxious" directly refer to *fear* are *direct* emotional lemmata, whereas "slaughter" or "earthquake" don't denote an emotion but evoke emotions; thus they are an *indirect* emotional lemmata. *Direct* emotional lexicons have the disadvantage of sparseness, as *direct* emotional lemmata are far less frequent than *indirect* ones, but the advantage of precision, as the *direct* emotional lemmata have the same emotional association in different contexts, whereas the emotions associated with *indirect* emotional lemmata depend highly on the context. Given that the available resources are so diverse, it should be considered carefully which one(s) to deploy for a specific task.

A set of emotional categories that is used in some lexicons is the Ekman set of emotions [Ekman, 1992], which contains the six emotions *anger*, *disgust*, *fear*, *joy*, *sadness*, *surprise*, although it is unbalanced, having more negative categories than positive ones. Thus, other lexicons adopted the Plutchik categories [Plutchik, 1980], which include also *anticipation* and *trust*. While these two sets are probably the most commonly used, some lexical resources use different categories, based on other definitions of what emotions are. Although the emotional categories have a psychological foundation, it is questionable whether all of them are reasonable categories for linguistic emotional lexical resources, e.g. it is questionable whether there are lemmata that are universally associated with *surprise* – aside from the ones directly denoting surprise – or whether their association is highly dependent on the context.

There are several lexical resources for the English language that contain emotions. A brief description of the most frequently used follows, while the next subsections will provide more details on the lexicons that were used in this work.

The General Inquirer (GI) [Stone et al., 1966] is one of the oldest resources developed specifically for NLP tasks. The latest version provides manual annotations for 11,789 lemmata, over 182 different dimensions. In particular, it contains 4,206 words with a polarity rating (*positive*/*negative*) and 311 words that are classified as being related to emotions, albeit without mentioning which particular one.

The Affective Norms for English Words (ANEW) [Bradley and Lang, 1999] provides normative emotional ratings for approximately 1,000 tokens. This lexicon is based on the PAD emotional state model [Mehrabian and Russell, 1974], where emotions can be decomposed in three main dimensions: *valence*, *arousal* and *dominance*. For each entry it provides a continuous rating over these three dimensions, calculated by averaging the scores of different human annotators.

AffectNet [Cambria et al., 2010] is a semantic network containing about 10,000 items. It was created by blending entries from ConceptNet [Havasi et al., 2007] and the emotional labels of WordNet-Affect [Strapparava and Valitutti, 2004], thus assigning emotion values to everyday-life concepts like "meet people" or "watch TV".

DepecheMood [Staiano and Guerini, 2014] is the only automatically-constructed resource among these. It contains 37,771 lemma-PoS, aligned with WordNet 3.0 [Fellbaum, 1998], and for each of these it provides the strength of association with the following emotional labels: *afraid*, *amused*, *angry*, *annoyed*, *don't care*, *happy*, *inspired*, *sad*. These values were derived starting from human ratings of articles on social news networks.

Other resources that appeared in the literature are the Fuzzy Affect Lexicon [Subasic and Huettner, 2001] (about 4,000 entries with 80 emotion labels) and the Affect database [Neviarouskaya et al., 2010] (2,500 lemma-PoS taken from WordNet-

Affect, and manually enriched by adding the strength of association with Izard's basic emotions [Izard Carroll, 1977]).

3.2.2 Linguistic Inquiry and Word Count (LIWC)

The Linguistic Inquiry and Word Count Dictionary (LIWC) [Pennebaker et al., 2001] was explicitly developed

> "to provide an efficient and effective method for studying the various emotional, cognitive, structural, and process components present in individuals verbal and written speech samples." [Pennebaker et al., 2001]

It was constructed starting from several sources, including Roget's Thesaurus, several English dictionaries and other specialized psycholinguistic resources. The LIWC2015 contains almost 6,400 stems – consequently the number of lemmata is bigger – , and their (binary) association to several dimensions, indicating for example emotions, social and psychological processes, personal concerns or relativity, and their respective subcategories.

All the labels were manually annotated by multiple judges, and those not used frequently enough in general English texts were discarded entirely.

In particular, the dimensions relevant to this work are *anger*, *anxiety*, *posfeel* and *sadness*.

3.2.3 WordNet-Affect

The WordNet-Affect Lexicon (WAL) [Strapparava and Valitutti, 2004] is a resource containing roughly 1,000 emotional lemma-PoS entries of the PoS noun, adjective, verb and adverb that can consist of more than one token, i.e. not only unigrams. It has entries in the following format: `PoS#number_of_synset list_of_lemmata`. For example, `a#01734691 disgusted fed_up sick sick_of tired_of` is an entry of the emotional category *disgust*.

As WAL extends affective labels to WordNet synsets, it allows for word-sense disambiguation, its format is fully consistent with WordNet and it is hierarchically organized. Thus, very fine-grained emotional categories like *dolor* are contained in broader ones like *sorrow*, which are contained in even broader ones like *sadness* up to very broad ones like *negative-emotion*. The number of affective categories amounts to 311, of which some are affective categories that are not emotions in a strict sense, i.e. some of them correspond to a mood or a cognitive state [Strapparava and Valitutti, 2004].

Despite the huge number of emotional categories the number of lemmata in each category is rather small, e.g. the numbers of lemmata in the categories *anger*, *disgust*, *fear*, *joy*, *sadness*, *surprise* range from 52 in *disgust* to 412 in *joy*. The most fine-grained categories in some cases contain only one lemma, which can be explained by the fact that the lemmata in WAL are direct emotional lemmata.

3.2.4 NRC Word-Emotion Association Lexicon (EmoLex)

EmoLex version 0.92 [Mohammad, 2013] contains 14,182 lemmata and eight emotions, that correspond to the Plutchik set of emotions: *sadness* (1,191), *anger* (1,247), *fear* (1,476), *surprise* (534), *joy* (689), *trust* (1,231), *anticipation* (839) and *disgust* (1,058) (in brackets the number of lemmata associated with each category). It also contains *positive* (2,312) and *negative* (3,324) sentiment. Interestingly, the values of the categories *negative* and *positive* do not correspond to the sum of the values of the respective negative and positive emotions.

Every unigram has been assigned a binary association value for each category, i.e. associated or not associated with a particular emotion. That means that there are unigrams in the lexicon that are not associated with any emotion, as well. It contains unigrams with the PoS adjective, noun, verb and adverb. In contrast to WordNet-Affect most of the lemmata of EmoLex are indirect emotional lemmata. The downloadable version of the resource does not contain n-grams, provide PoS information or word-sense information.

EmoLex has been constructed through crowdsourcing with Amazon's Mechanical Turk[1] from annotations of at least five different annotators. As a basis they took entries of the Roget's Thesaurus[2] that occurred more than 120,000 times in the Google n-gram corpus[3]. The word-level emotion association lexicon has been constructed from 24,200 word-sense pairs, which have been merged by taking the union of all emotions associated with the different senses of the word.

[1] Mechanical Turk: `http://www.mturk.com/mturk/welcome`

[2] Roget's Thesaurus: `http://www.gutenberg.org/ebooks/10681`

[3] The Google n-gram corpus is available through the Linguistic Data Consortium: `https://www.ldc.upenn.edu`

4. Quantitative Analysis of Emotional Unigrams before Audience Reactions

4.1 Retrieving Unigrams and Emotions before Audience Reactions

In order to observe whether there is a relation between audience reactions and emotional unigrams, it is a necessary basic step to check the emotions associated with each single unigram. This might even be sufficient to observe such a relation, although it would be more precise and meaningful to compute the prevalent emotion of a whole sentence taking into account linguistic information, e.g. n-grams, negation, sentence structure, the context, semantics, etc. As this would exceed the scope of this thesis, only the occurrences of emotional unigrams belonging to different emotional categories in windows of one to three sentences before an audience reaction have been counted.

4.1.1 Preprocessing of the Data

For the preprocessing of the corpus and lexicons, small Python programs have been built: `tag_corpus.py`, `format_emo_lists.py`.

- At first CORPS has been PoS-tagged and lemmatized with TreeTagger[1], so that tokens of the same lemma are all recognized as the same type, and can be looked up more conveniently in different emotional lexicons. Furthermore, PoS can be synchronized with the emotional lexicon later, in case the lexicon disposes of PoS information.
- Then, the emotional lexicons have been converted into an equal format: `lemma PoS emotion value` (e.g. `aggression NN anger 1`).

  The value of `lemma`
  : can be a lemma or a stem alternatively (e.g. `gratef*`, where asterisk indicates several possible endings of the stem referring to different lemmata).

  The value of `PoS`
  : can become `NN` (nouns), `JJ` (adjectives), `VV` (verbs), `RB` (adverbs) or `**` (lexicon does not contain PoS tags).

[1] TreeTagger is available at: `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/`

The value of `emotion`

takes the values of the emotional categories of the respective lexicon, where `value` can be `0` (not associated) or `1` (associated). It should be noted that, depending on the lexicon, it is possible that a lemma is not associated with any emotion, associated with exactly one emotion or associated with more than one emotion. It is also possible that every lemma is listed for each emotion in the lexicon or that it is only listed for the emotions it is associated with, which means that `value` is always `1`.

4.1.2 Extracting Sentence Windows and Emotional Unigrams

Further programs which take the formatted lexicons and tagged CORPS files as input have been created in Python. These programs were built to obtain windows of sentences before every audience reaction, to get frequency lists of lemmata for each audience reaction and to get frequency lists of occurrences of lemmata in each emotional category for every audience reaction and for different lexicons:

- `tag_freq.py` extracts a window of an adjustable number of sentences before an audience reaction tag or negative windows, i.e. windows distant from audience reactions (denoted by *notag*) and creates a frequency list of lemmata and their PoS tags;
- `merge_tags.py` takes the frequency lists, merges similar PoS tags and their lemmata;
- `assign_emotions.py` takes frequency lists of lemmata before audience reactions and a formatted emotional dictionary and creates frequency lists of lemmata for audience reaction windows ordered by emotional categories and their frequencies.

The overall process is summarized in Figure 4.1.

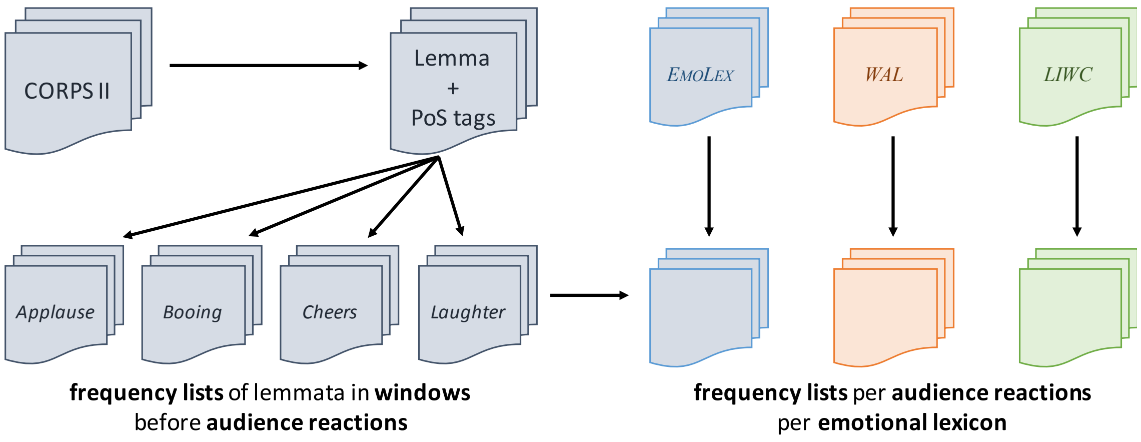
CORPS II
Lemma + PoS tags
EmoLex
WAL
LIWC
Applause
Booing
Cheers
Laughter
frequency lists of lemmata in windows before audience reactions
frequency lists per audience reactions per emotional lexicon

Figure 4.1: Process of obtaining frequency lists for each audience reaction tag for different lexicons

4.2 Counts of Unigrams and Emotions with Different Lexicons

This section shows frequency lists of unigrams and associated emotions when using different lexicons to give an impression of the corpus and how the lexicons influence the results. The frequency lists shown are for sentence windows of size two before the audience reactions (*applause*, *laughter*, *cheers* and *booing*) and for sentence windows of size two that are twenty sentences away from audience reactions denoted by "no audience reaction" – also referred to as *notag* in this work. Twenty sentences is a rather arbitrary number of sentences from the next audience reaction tag that has been chosen because it is considerably distant so that it should not have a direct impact on the audience reaction, but enough sentences could still be extracted. There is also a frequency list for the whole corpus. For these frequency lists only a period (i.e. ".") was considered as a sentence boundary.

4.2.1 Frequencies of Emotional Unigrams with EmoLex

The following tables show absolute frequencies of occurrences of unigrams before audience reactions. As table 4.1 shows, the most frequent emotional unigrams in the whole corpus are the ones associated with *trust*, *anticipation* and *joy*. Positive emotions are more than twice as frequent as negative ones. When considering tables 4.3 to 4.6 (showing the emotions before audience reactions) *trust*, *anticipation* and *joy* in exactly this order are still the most frequent ones. The only exception is *booing* (see table 4.6) where *anger* and *fear* are more frequent than *joy* and negative emotions are almost as frequent as positive ones.

whole corpus		
#occurrences	%	emotion
379,056	26%	*Trust*
259,568	18%	*Anticipation*
218,650	15%	*Joy*
171,173	12%	*Fear*
126,230	9%	*Anger*
114,749	8%	*Surprise*
113,865	8%	*Sadness*
52,540	4%	*Disgust*
#occurrences	%	sentiment
571,066	70%	*Positive*
246,006	30%	*Negative*

Table 4.1: No. of lemmata per emotion with EmoLex (whole corpus)

no audience reaction		
#occurrences	%	emotion
8,765	25%	*Trust*
6,181	18%	*Anticipation*
5,000	14%	*Joy*
4,603	13%	*Fear*
3,328	9%	*Anger*
3,091	9%	*Sadness*
2,723	8%	*Surprise*
1,425	4%	*Disgust*
#occurrences	%	sentiment
13,416	67%	*Positive*
6,551	33%	*Negative*

Table 4.2: No. of lemmata per emotion with EmoLex (2-sentences window not followed by a reaction)

Applause		
#occurrences	%	emotion
83,369	27%	*Trust*
56,125	18%	*Anticipation*
50,817	17%	*Joy*
33,186	11%	*Fear*
24,956	8%	*Surprise*
24,850	8%	*Anger*
21,570	7%	*Sadness*
10,245	3%	*Disgust*
#occurrences	%	sentiment
122,273	72%	Positive
47,955	28%	Negative

Table 4.3: No. of lemmata per emotion with EmoLex (2-sentences window followed by *applause*)

Laughter		
#occurrences	%	emotion
16,663	25%	*Trust*
13,384	20%	*Anticipation*
11,172	17%	*Joy*
7,078	11%	*Surprise*
5,667	9%	*Fear*
5,175	8%	*Sadness*
4,962	7%	*Anger*
2,407	4%	*Disgust*
#occurrences	%	sentiment
24,488	70%	*Positive*
10,368	30%	*Negative*

Table 4.4: No. of lemmata per emotion with EmoLex (2-sentences window followed by *laughter*)

Cheers		
#occurrences	%	emotion
2,051	26%	*Trust*
1,355	17%	*Anticipation*
1,255	16%	*Joy*
842	11%	*Fear*
677	9%	*Anger*
631	8%	*Surprise*
586	8%	*Sadness*
357	5%	*Disgust*
#occurrences	%	sentiment
2,859	67%	*Positive*
1,378	33%	*Negative*

Table 4.5: No. of lemmata per emotion with EmoLex (2-sentences window followed by *cheers*)

Booing		
#occurrences	%	emotion
1,417	22%	*Trust*
1,013	16%	*Anticipation*
949	15%	*Anger*
882	14%	*Fear*
699	11%	*Joy*
697	11%	*Sadness*
537	8%	*Surprise*
268	4%	*Disgust*
#occurrences	%	sentiment
1,902	57%	*Positive*
1,410	43%	*Negative*

Table 4.6: No. of lemmata per emotion with EmoLex (2-sentences window followed by *booing*)

4.2.2 Frequencies of Emotional Unigrams with Selected Emotions from WordNet-Affect

As WAL has many categories, not all of them could be considered for this analysis. To have categories that are more comparable to the EmoLex categories, and in order not to have to consider too sparse categories, only the Ekman subset has been used. The emotional category *trust* doesn't exist in WAL, and *anticipation* contain very few entries.

As there are less lemmata, i.e. only *direct* emotional ones, in each emotional category the frequency counts are less. Similarly to the results obtained with EmoLex, the most frequent emotional unigrams belong to a category of a positive emotion, i.e. *joy*. Also agreeing with the results from EmoLex, the order of the most frequent emotions *joy*, *surprise* and *sadness* is the same in almost all the audience reaction frequency lists and in the whole corpus frequency list. Also *disgust* remains almost consistently the least frequent emotion.

whole corpus		
#occurrences	%	emotion
75,034	57%	*Joy*
34,100	26%	*Surprise*
9,297	7%	*Sadness*
6,853	5%	*Fear*
4,531	3%	*Anger*
771	1%	*Disgust*

Table 4.7: No. of lemmata per emotion with WAL (whole corpus)

no audience reaction		
#occurrences	%	emotion
2,330	58%	*Joy*
802	20%	*Surprise*
488	12%	*Sadness*
210	5%	*Fear*
162	4%	*Anger*
19	0%	*Disgust*

Table 4.8: No. of lemmata per emotion with WAL (2-sentences window not followed by a reaction)

Applause		
#occurrences	%	emotion
18,282	63%	*Joy*
6,988	24%	*Surprise*
1,588	5%	*Sadness*
1,410	5%	*Fear*
727	2%	*Anger*
209	1%	*Disgust*

Table 4.9: No. of lemmata per emotion with WAL (2-sentences window not followed by *applause*)

Laughter		
#occurrences	%	emotion
4,421	50%	*Joy*
3,575	40%	*Surprise*
473	5%	*Sadness*
235	3%	*Anger*
182	2%	*Fear*
36	0%	*Disgust*

Table 4.10: No. of lemmata per emotion with WAL (2-sentences window not followed by *laughter*)

Cheers		
#occurrences	%	emotion
509	62%	*Joy*
218	26%	*Surprise*
48	6%	*Sadness*
26	3%	*Anger*
18	2%	*Fear*
4	0%	*Disgust*

Table 4.11: No. of lemmata per emotion with WAL (2-sentences window not followed by *cheers*)

Booing		
#occurrences	%	emotion
114	49%	*Joy*
58	25%	*Surprise*
38	16%	*Fear*
16	7%	*Sadness*
4	2%	*Anger*
1	0%	*Disgust*

Table 4.12: No. of lemmata per emotion with WAL (2-sentences window not followed by *booing*)

4.2.3 Frequencies of Unigrams and Emotions with Selected Emotions from LIWC

The negative emotional categories of LIWC correspond to the ones of the Ekman subset of Emotions but the definition of the positive ones "posfeel" and "posemo" couldn't be found. LIWC doesn't provide other more fine-grained positive emotional categories. As a positive emotion only "posfeel" was taken into account here, which shows how the sparseness of the lexicon as well as the choice of its user to consider certain categories and disregard others can distort the results making one emotion look like the most frequent one, i.e. *anger*.

whole corpus		
#occurrences	%	emotion
41,221	37%	*Anger*
37,125	33%	*Posfeel*
18,353	16%	*Sadness*
14,913	13%	*Anxiety*

Table 4.13: No. of lemmata per emotion with LIWC (whole corpus)

no audience reaction		
#occurrences	%	emotion
1,060	39%	*Anger*
793	29%	*Posfeel*
508	18%	*Sadness*
392	14%	*Anxiety*

Table 4.14: No. of lemmata per emotion with LIWC (2-sentences window not followed by a reaction)

Applause		
#occurrences	%	emotion
8,980	38%	*Anger*
8,342	35%	*Posfeel*
3,386	14%	*Sadness*
2,888	12%	*Anxiety*

Table 4.15: No. of lemmata per emotion with LIWC (2-sentences window not followed by *applause*)

Laughter		
#occurrences	%	emotion
2,821	57%	*Posfeel*
1,042	21%	*Anger*
615	12%	*Sadness*
455	9%	*Anxiety*

Table 4.16: No. of lemmata per emotion with LIWC (2-sentences window not followed by *laughter*)

Cheers		
#occurrences	%	emotion
191	42%	*Anger*
167	36%	*Posfeel*
81	18%	*Sadness*
20	4%	*Anxiety*

Table 4.17: No. of lemmata per emotion with LIWC (2-sentences window not followed by *cheers*)

Booing		
#occurrences	%	emotion
226	59%	*Anger*
61	16%	*Anxiety*
49	13%	*Sadness*
44	12%	*Posfeel*

Table 4.18: No. of lemmata per emotion with LIWC (2-sentences window not followed by *laughter*)

4.2.4 Analysis of Normalized Emotions

This section gives a more fine grained analysis of emotional unigrams before each audience reaction. EmoLex has been chosen for this and all further analysis, since it has more balanced emotional categories than LIWC and more entries in each category than WAL.

Having computed the average value of emotional words per token for each audience reaction (see Table 4.19) as it is described in Chapter 5.1, it can be observed that *booing*, having a score of ≈ 0.238, is the audience reaction with the most emotional words, closely followed by *applause* with ≈ 0.223 and *cheers* with ≈ 0.219. *Laughter* is, with ≈ 0.176, the least emotional category.
Looking at the averages of each emotional category for each audience reaction, it can be noticed that the negative categories *anger*, *disgust*, *fear*, *sadness* and *negative* are at least twice as high for *booing* as for the other categories. The values for *sadness* are below the average of all the categories, and remain rather constant over the classes *applause*, *laughter*, *cheers*, but are about twice as high for *booing*. The values for *disgust*, i.e. ≈ 0.015, are much lower than the values for the other emotional categories, which could be due to *disgust* being rather rare in the English language in general, being rare in persuasive communication or just being rare in political speeches. *Laughter* is the emotional category with a much lower value for *fear* than the other categories, as there is a difference of ≈ 0.012 to the next lowest value, which is the one of *applause*. *Booing* has the highest value in the *anticipation* category and *laughter* the lowest, which is consistent with the results for emotional words per lemma. All the emotional values

for the category *joy* are a bit above the average of all emotional categories (excluding *positive*, *negative* and *no emotion*), i.e. ≈ 0.044. This means that – despite being one of the emotions with least lemmata in EmoLex – joy is always frequent, even in a negative class like *booing*, which could be due to more positive words in the English language in general [Dodds et al., 2015] or due to the elating characteristic of political speeches. The highest value in the *joy* category is ≈ 0.061 for *applause*, the lowest ≈ 0.046 for *booing*.
It is very interesting that the values for *trust* are extremely high in all the classes, i.e. in average ≈ 0.086, for all the audience reactions, which is more than twice as high as the average over all the classes, which could be a characteristic for persuasive communication in general. The value for *trust* has a distinctly lower value for *laughter*, i.e. ≈ 0.700 than for the other classes, though still being high.
When comparing the total of emotional lemmata for *notag*, i.e. no audience reaction, to the average of the audience reactions it can be observed that the windows not followed by an audience reaction contain on average more emotional words than the windows before an audience reaction. Thus, it can't be said that the windows before an audience reaction are more emotional than the ones not followed by an audience reaction. In general, the values for *notag* are very close to the average of all the audience reactions (*avg*). Differences can be observed when comparing each audience reaction individually with *notag*, e.g. *trust* and *joy* have higher values in the *applause*-window than in the *notag*-window and negative emotions have higher values in the *booing*-window.

The following table (4.19) shows the average of all the sentence windows of size 2 normalized by the number of lemmata in the window. *No emotion* denotes the lemmata that are present in EmoLex but not linked to any emotion, and *avg* is the average over the audience reactions *laughter*, *booing*, *applause*, *cheers*.

Emotion	*Laughter*	*Booing*	*Applause*	*Cheers*	*Avg*	*Notag*
All emotions (avg)	0.176	0.238	0.223	0.219	0.214	0.246
Negative	0.045	0.093	0.051	0.063	0.063	0.065
Positive	0.102	0.115	0.137	0.122	0.119	0.138
No emotion	0.218	0.251	0.228	0.234	0.233	0.274
Anger	0.021	0.064	0.027	0.033	0.036	0.034
Anticipation	0.058	0.068	0.065	0.061	0.063	0.064
Disgust	0.011	0.021	0.011	0.017	0.015	0.015
Fear	0.024	0.058	0.036	0.038	0.039	0.045
Joy	0.050	0.046	0.062	0.057	0.054	0.053
Sadness	0.023	0.043	0.023	0.024	0.028	0.030
Surprise	0.033	0.035	0.029	0.028	0.031	0.029
Trust	0.070	0.092	0.096	0.089	0.086	0.091

Table 4.19: Emotional categories normalized by lemma, averaged over all the sentence windows of size 2 with the emotional values of EmoLex in blue: the lowest value of each line, in red: the highest value of each line

4.3 Comparison of Emotional Unigrams in a Persuasive and a Non-Persuasive Corpus

To assess if there is a difference in emotional word usage between the persuasive communication of CORPS and non-persuasive communication, it is interesting to confront the sentences used until now with others obtained for another corpus. Although CORPS is a corpus of American speeches, and therefore might be compared better to an American corpus, the British National Corpus (BNC)[2] [Burnard and Aston, 1998] has been chosen instead, since it was easily obtainable and it is a very balanced corpus that can also represent the English language in general.

The prediction is that CORPS will be more emotional than BNC, given that the BNC contains emotional texts like novels but also many factual texts like newspapers and journals. Taking BNC as a representative sample of non-persuasive communication, CORPS should also be more emotional due to the elating character of political speeches. Of course, even a non-persuasive corpus, e.g. a corpus of lyrics, can be just as emotional as a persuasive one. However, in that case the difference could be in the specific distribution of each emotion, with some being more or less frequent in persuasive communication.

The normalized values for 50,000 randomly extracted 2-sentence windows from the BNC have been computed and confronted with the values of 3,879 2-sentence windows of CORPS labeled *notag*, to observe whether even the "neutral" parts of CORPS (i.e. the ones not preceding an audience reaction) are different from the ones in a non-persuasive corpus.

Table 4.20 shows the average emotional word usage over all the sentences for BNC and CORPS, and the percentage difference between the two corpora for each category.

The results in the table show that indeed in CORPS there are 22% more emotional unigrams than in BNC, taking into account all the categories. This makes sense, because a more emotional speech might work in favor of the speaker, as according to the *general arousal model* [Sanbonmatsu and Kardes, 1988] a strongly aroused state of the audience should decrease their ability to deeply process all the arguments, and therefore make it easier for the speaker to persuade the audience. Although the speech passages considered here are the ones distant from the audience reactions, the speaker might already try to get the audience into a certain mood.

The most striking difference can be observed in the positive emotional categories: *joy* is 53% more frequent in CORPS than in BNC, *trust* 41% and *anticipation* 37%. It also makes sense that the audience should be in a joyful state, as they would be less critical being in a good mood, according to the *affective valence model* [Schwarz and Bless, 1991]. In case that the trust is attributed to the speaker, or his party, the high values for *trust* make sense, as persuadees are often guided by heuristics like whether they trust the speaker [Chaiken, 1987]. Moreover, the high values for *joy* increase *trust*, as *trust* is influenced significantly by other co-occurring emotions (e.g. *happiness* boosts *trust* as found by Dunn and Schweitzer [2005]).

Although the number of unigrams associated with the category *negative* does not change much, an increase of the particular negative emotions *anger* and *fear* can be observed. This can be explained by the finding that fearful recipients of persuasive attempts should tend to follow the crowd more, according to the *evolutionary model*

[2] BNC is available here: `http://www.natcorp.ox.ac.uk/`

[Griskevicius et al., 2009]. *Anger* is an essential emotion for political campaigns, as it encourages political participation and action in general, more than other emotions (like for example *enthusiasm* [Valentino et al., 2011]). Therefore, it might be an important emotion in political speeches, especially when aimed at the opponent.

Emotional avg.	BNC	CORPS (NOTAG)	Increase
Positive	0.101	0.138	36%
Negative	0.062	0.065	5%
Anger	0.026	0.034	31%
Anticipation	0.047	0.064	37%
Disgust	0.016	0.015	-8%
Fear	0.036	0.045	24%
Joy	0.035	0.053	53%
Sadness	0.029	0.030	3%
Surprise	0.023	0.029	29%
Trust	0.065	0.091	41%
Tot. emotions	0.202	0.246	22%
No emotion	0.277	0.274	-1%

Table 4.20: Emotional word usage in CORPS and BNC (normalized by the number of lemmata in the window)

4.4 Emotional Unigrams with highest Persuasive Impact

Whereas in preceding sections the occurrences of unigrams have been counted, the focus of this section is to retrieve the most specific unigrams for each audience reaction and then to see which emotions they are associated with. This way, it is possible to observe which emotions are occurring in the passages leading to one of the audience reaction *cheers*, *booing*, *applause* and *laughter* and are rare before whole the rest of the corpus (including the the passages before the other audience reactions). [Guerini et al., 2008] have introduced the notion of *persuasive impact (pi)* and have described a method to use a weighted tf-idf to calculate it:

$$tf_i = \frac{n_i * \sum_{n_i} s_i}{\sum_k n_k} \qquad\qquad idf_i = \log \frac{|D|}{|\{d : d \ni t_i\}|}$$

With n_i being the number of times word t_i appears in the document, $\sum_{n_i} s_i$ the sum of the scores of the word, $\sum_k n_k$ the number of total occurrences of all words, $|D|$ the number of speeches in the corpus and $|\{d : d \ni t_i\}|$ the number of documents where t_i is present. The set of documents contains a virtual document of all the audience reaction sentences of one type of audience reaction type and all the documents from the corpus, where the audience reaction windows have been subtracted.

$$s_{im} = \frac{1}{l_{im} - p_{im}} \qquad\qquad s_i = \frac{\sum_1^m s_{im}}{n_i}$$

Where l_{im} denotes the number of lemmata in the audience reaction window, i.e. the length of the window, p_{im} denotes the position of t_i in the window in a range from (0 to $l-1$, where $l-1$ is the one occurring directly before the audience reaction tag). Hence, s_{im} is the score for one particular occurrence m of t_i. The closer t_i is to the audience reaction tag, the higher the score. S_i is simply the average of s_{im}. In contrast to how Guerini et al. [2008] have computed the tf-idf, for this study a window of two sentences has been used, instead of a fixed number of lemmata.

After having determined the tf-idf for every lemma of each category, the entries that were not present in the EmoLex lexicon (mainly named entities) were discarded, since there is no information on their emotional values. The results of the analysis for the top-ranking terms (the top 25% for each category) can be seen in Table 4.21, while an example for the words (the top 25) is shown in Table 4.22. Words occurring only once in the corpus were discarded from the analysis.

Despite the small numbers that make the statistics less reliable, the results for most classes are very intuitive. They are consistent with the findings for the frequency analysis in section 4.2.4: *booing*, has also a higher prevalence of *negative* emotions, in particular *fear* (among the top-scoring words are "bomber", "blame", "opponent", "brutal"). All other reactions have a larger proportion of *positive* emotions, and this is particularly true for *cheers* (examples are "gentleman", "pretty", "agree", "spirit") and *laughter* (e.g. "triumph", "civilization", "intellectual"). Despite being mostly *positive*, also *applause* is also connected to *anger* ("hostility") and *fear* ("hatred"), like it happened for *booing*.

Emotion	*Applause*	*Booing*	*Cheers*	*Laughter*
Positive	25%	19%	27%	25%
Negative	20%	22%	12%	14%
Anger	10%	13%	3%	6%
Anticipation	11%	12%	12%	10%
Disgust	6%	6%	8%	4%
Fear	11%	13%	3%	8%
Joy	9%	6%	15%	10%
Sadness	9%	9%	4%	8%
Surprise	5%	3%	9%	6%
Trust	17%	14%	23%	16%
Total lemmata	1,645	149	74	906

Table 4.21: Distribution of emotions for the most relevant words of each reaction

Applause	*Booing*	*Cheers*	*Laughter*
Entity	Head	Number	Year
Separation	Bother^{1}	Ground^{9}	Lawn
Phase	High	Gentleman$^{0\ 9}$	Participation^{0}
Hatred$^{1\ 2\ 4\ 5\ 7}$	School^{0}	Lady	Solidarity^{9}
Independent	Service	Remind	Answer
Rule$^{5\ 9}$	Thing	Member	Question
Close	Remind	President$^{0\ 9}$	Ink
Weep$^{1\ 7}$	Day	Pretty$^{0\ 3\ 6\ 9}$	Outlook
Agreement$^{0\ 1}$	Million	Agree^{0}	Triumph$^{0\ 3\ 6}$
Major$^{1\ 2\ 4\ 5\ 7}$	Show^{9}	Part	Singularly^{8}
Step	Put	Inaugural^{3}	Move
Respect$^{0\ 3\ 6\ 9}$	Opponent$^{1\ 2\ 3\ 4\ 5}$	Set	Turn
Compromise	Deficit^{1}	Wonderful$^{0\ 6\ 8\ 9}$	Relay
Sign	Dealer	Day	Presentation
State	Kill$^{1\ 5\ 7}$	Hear	Long^{3}
Incline^{9}	Blame$^{1\ 2\ 4}$	Good$^{0\ 3\ 6\ 8\ 9}$	Intellectual^{0}
Empathy^{0}	Brutal$^{1\ 2\ 5}$	Work	Exception
Today	Pornography$^{1\ 4}$	Time^{3}	Civilization$^{0\ 9}$
Reach	Today	Rash$^{1\ 4}$	Sport
Land^{0}	Education	Present$^{0\ 3\ 6\ 8\ 9}$	Label^{9}
Pay$^{0\ 3\ 6\ 9}$	Ballot$^{0\ 3\ 9}$	Conquer	Tradition
Hostility$^{1\ 2\ 4}$	Bomber$^{5\ 7}$	Spirit^{0}	Scheme^{1}
Coexistence	Serve$^{1\ 9}$	Highway	Dialogue
Credit$^{0\ 9}$	Supreme^{0}	Catch^{8}	Artist
Side	Life	Chance^{8}	Basis

Table 4.22: Top 25 words for each reaction. Superscripts indicate the emotions associated to each word (0. Positive - 1. Negative - 2. Anger - 3. Anticipation - 4. Disgust - 5. Fear - 6. Joy - 7. Sadness - 8. Surprise - 9. Trust). Words without a superscript have an entry in EmoLex, but not an association.

5. Experiment: Predicting Audience Reactions using Emotions

5.1 Computing Normalized Emotional Values

This section describes the creation of the normalized values that have been used for the machine learning experiment. The frequency lists in the section 4.2 have given an impression of the corpus and its emotions and have as well illustrated the necessity of more comparable data. The raw frequencies for each category could be distorted by especially long sentences before a certain audience reaction with emotional counts that are not characteristic for that reaction.

The program `normalize.py` takes as input the lemmata before every audience reaction tag for each sentence window and a formatted emotional lexicon. It computes a normalized output value for each window before an audience reaction. The program calculates:

- the *total of emotions per lemma*, which is the total of occurrences of emotional lemmata of all emotional categories in the window, divided by all lemmata in the window
- the *emotional category per token*, which is the sum of emotional lemmata of one category, divided by all the lemmata in the window
- the *emotional category per emotion*, which is the sum of emotional lemmata of one category, divided by the total of occurrences of emotional lemmata of all emotional categories in the window

The idea behind this is that, since this data will be used as features for *SVM*, the double normalization might allow the classifier to view the same information from different angle, and this might make the reactions more easy to differentiate.

For the values normalized by emotions some values have been put by default to avoid division by 0. For the case that there is no emotional lemma in an audience reaction window, 0 was set as a value; for the case that there is no emotional lemma but neutral ones, i.e. *no emo*, -1 was chosen. For the case that there are emotional lemmata in the sentence window, but not of that specific category (e.g. 0 lemmata associated with *sadness*, divided by 4 emotional lemmata), the feature was set to -2 instead of 0. Again, the rationale is that differentiating between these cases could be useful information for *SVM*.

5.2 Parameters for ML Experiment

To get a clearer understanding of the relation between emotions and audience reactions, a supervised ML experiment has been conducted using WEKA 3.7[1] and *SVM* as the classifying algorithm (in particular, the LibSVM implementation with its default parameters). The experiment is to observe whether and to what extent two classes of audience reactions can be distinguished, using only the normalized emotions (described in section 4.2.4) in the sentence windows before the reaction as features (the audience reactions being the labels, i.e. classes).

In particular, only *applause*, *laughter*, *cheers* and *booing* have been used, as they are the most distinctive and frequent reactions. The emotions have been taken from EmoLex, as it contains more lemmata than LIWC and WAL. All the emotional categories of EmoLex have been taken into account, namely *anticipation*, *joy*, *surprise*, *sadness*, *fear*, *anger*, *disgust*, *trust*, *positive*, *negative*. Also *no emotion* was used, for lemmata that are in the lexicon but aren't associated to any emotion, i.e. their value is 0.

At first, one file has been created for each pair of audience reactions and for three different window sizes (1, 2 and 3 sentences). To account for the differences in the size of each class during the pairwise comparisons, random subsampling has been used to reduce the size of larger classes to that of the smaller. The size of each dataset is shown in Table 5.1.

LibSVM has been run with 10-fold cross-validation to use as much of the data as possible for the training, given the rather small dataset.

Comparison	Elements per class	Total elements
Applause vs. *laughter*	15,444	30,888
Applause vs. *cheers*	1,083	2,166
Applause vs. *booing*	773	1,546
Booing vs. *cheers*	773	1,546
Booing vs. *laughter*	773	1,546
Cheers vs. *laughter*	1,083	2,166

Table 5.1: Dataset size for each pairwise comparison of the ML experiment

5.3 Results of ML Experiment

The values on the lower side of the diagonal show the F1 score[2] for each pair of classes and the values on the upper side of the diagonal show the percentage of correctly classified instances for each pair of classes. The F1 score for each pair of classes is the average of the F1 score of the two classes, e.g. if the F1 score of *applause* is 0.632 and the F1 score of *laughter* is 0.612 for the pair *applause-laughter*, the F1 score is computed as $0.632/0.612 = 0.623$.

The tables show that the two classes that can be distinguished best from each other for all the sentence windows are *booing* and *laughter*, then *booing* and *applause*,

[1] WEKA is available here: `http://www.cs.waikato.ac.nz/ml/weka/index.html`

[2] F1 score = precision $*$ recal / (precision + recall)

then *booing* and *cheers*. The F1 score for *booing* is much higher than for the other classes, which means that *booing* is more distinctive. An explanation for this could be that although there can be negative as well as positive emotions in sentences before all audience reactions, there are much more negative emotions before *booing*. Additionally, the other three reactions are rather reactions of approval, i.e. the audience agrees with the speaker, whereas *booing* is rather - though not always - a reaction of disapproval.

	Applause	*Cheers*	*Laughter*	*Booing*	*Notag*
Applause		55.96%	62.38%	69.85%	58.06%
Cheers	0.558		58.86%	66.41%	57.76%
Laughter	0.624	0.586		70.54%	63.09%
Booing	0.698	0.662	0.704		66.62%
Notag	0.581	0.578	0.630	0.666	

Table 5.2: Prediction results for each pair of classes with a window of 1 sentence. Upper half: percentage of correctly classified instances. Lower half: average of F1-scores.

	Applause	*Cheers*	*Laughter*	*Booing*	*Notag*
Applause		55.24%	62.58%	70.67%	57.85%
Cheers	0.551		60.43%	70.05%	55.56%
Laughter	0.626	0.604		72.66%	66.51%
Booing	0.706	0.701	0.726		68.34%
Notag	0.574	0.556	0.665	0.683	

Table 5.3: Prediction results for each pair of classes with a window of 2 sentences. Upper half: percentage of correctly classified instances. Lower half: average of F1-scores.

	Applause	*Cheers*	*Laughter*	*Booing*	*Notag*
Applause		56.20%	62.74%	70.33%	57.14%
Cheers	0.552		61.12%	68.96%	58.27%
Laughter	0.627	0.611		71.09%	68.73%
Booing	0.703	0.689	0.711		68.89%
Notag	0.568	0.582	0.686	0.688	

Table 5.4: Prediction results for each pair of classes with a window of 3 sentences. Upper half: percentage of correctly classified instances. Lower half: average of F1-scores.

	Applause	*Cheers*	*Laughter*	*Booing*	*Notag*
Applause		55.80%	62.57%	70.28%	57.68%
Cheers	0.554		60.14%	68.48%	57.20%
Laughter	0.625	0.600		71.43%	66.11%
Booing	0.702	0.684	0.714		66.95%
Notag	0.574	0.572	0.660	0.79	

Table 5.5: Prediction results for each pair of classes (average over all the window sizes). Upper half: percentage of correctly classified instances. Lower half: average of F1-scores.

Different window sizes, i.e. different numbers of sentences preceding the reaction have been extracted, to see which window size works best for distinguishing between audience reactions. Comparing the results of Tables 5.2, 5.3 and 5.4, it can be seen that, while there is often a general improvement using longer sentence windows, this improvement is not particularly significant. This might mean that either the most important emotional content (i.e. the one that affects the audience most) is usually in the sentence directly preceding the reaction, or, simply, that the emotional distribution in all three preceding sentences is more or less constant.

Looking at the results more in detail, it can be seen that *booing* is the reaction that can be distinguished best. Intuitively this makes sense, as it is the only negative reaction and thus the most conceptually dissimilar from the others.

At the same time, also for *laughter* the results are overall good. It is worth noting that these two emotions are the easiest to tell apart from *notag*, i.e. windows not followed by an audience reaction. This most likely means that these reactions appear after sentences with both strong and diverse emotional content, while the sentences of *cheers* and *applause* are more similar to each other, but also to *notag*. This also seems reasonable, considering that the former are stronger reactions, while the latter are more generic manifestations of approval.

6. Ideas for Future Work

As a lexical approach for emotional analysis has been taken, the results of any research can only be as good as the lexicons used. The lexicons have many problems like sparseness, insignificant categories or errors. Consequently, for further research the specific problems of the lexicon(s) used should be taken into account more like correcting false entries, adding PoS tags, balancing categories and discarding irrelevant ones or normalizing the results by the number of lemmas that are in a category, avoiding sparseness by combining lexicons, etc. Besides, the emotions that each lemma is associated with in different lexicons could be compared.

Moreover, to have more meaningful results, a sophisticated approach should be used to determine the emotion of a lemma, e.g. determining the emotion of a lemma depending on its respective meaning in the context, as one lemma can be associated to a positive emotion in one context and to a negative one in another context, e.g. "cold beer" in the sense of "having a low temperature" and "cold person" referring to "being emotionless" [Guerini et al., 2013a]. Similarly to the problem of retrieving the prior polarity in sentiment analysis, it is a problem in emotional analysis to get the prior emotion of an indirect emotional lemma. Actually the prior emotion of the lemma in the lexicon should match the one that is relevant for the corpus. Aside from improving on the level of the lemma, a way to represent the emotion of whole audience reaction windows better than just counting emotional lemmas could be explored, e.g. using a vector model.

Furthermore, it could be interesting to examine which emotions are co-occurring in the audience reaction windows. Looking at speeches, it seems that contrasting emotions like *anger* and *joy* are often used together in the same window. Additionally, the ratios of the normalized frequencies of different emotions to each other should be computed. It has already been found by Guerini et al. [2013b] that if there are any valence dynamics before audience reactions, there is a valence crescendo, which could be tested further on the level of emotions taking into account the intensity of the emotions (being careful not to confound the intensity and the degree of association). Also, the emotions of the lemmas with the highest impact on audience reactions, which have already been retrieved by Guerini et al. [2013b], could be examined. Besides, another Machine Learning experiment should be conducted trying to distinguish between windows not followed by an audience reaction and windows followed by an audience reaction. Of course, also the statistical significance of the results has to be tested, which hasn't been done yet.

7. Conclusions

This thesis was aimed at studying the distribution of emotion-evoking words and the relation between emotions and audience reactions in persuasive language in the CORPS political speeches. Several existing lexical resources for emotion detection have been thoroughly examined, so that a suitable one, NRC Word-Emotion Association Lexicon, has been chosen for the task.

Using computational and statistical methods it was then shown that, persuasive language is rich in emotions, as even the less persuasive sentences of CORPS contain a much larger amount of emotion-evoking words than the ones of a non-persuasive corpus, in particular those referring to *joy* and *trust*.

Moreover, it was found that the distribution of emotional words is different for different audience reactions., *booing* being the one with the largest proportion of *negative* words, with the other reactions being mostly *positive*. Although some emotions are frequent in general, e.g. *trust* is consistently the prevalent emotion, independently of the reaction. In addition to that, also the most relevant words for each reaction were extracted and analyzed, showing numbers that are consistent with the overall analysis.

Finally, a machine learning experiment demonstrated that, using only the emotional words in a single sentence preceding an audience reaction, it is possible to discriminate that reaction with above-chance accuracy. Good results can be obtained in particular with *booing* and *laughter*, probably indicating that these reactions are connected to a characteristic "emotional signature", i.e. there is a strong presence of emotional terms before these reactions, and the distribution of the different emotions is peculiar of these two categories.

Appendix A. Python programs

A.1 Tag corpus

```
# This program takes the corpus files as input
# and creates a tagged output file for each file

import treetaggerwrapper, os, codecs

tagger = treetaggerwrapper.TreeTagger(TAGLANG='en',TAGDIR='C:/TreeTagger')

#get txt file names from directory and save them in a list
filenames = [f for f in os.listdir('./test/') if f.endswith('.txt')]

#for each txt file
for f in filenames:

    #to open, read and tag txt file
    inputPath = os.path.join('./test/', f)
    try:
        with codecs.open(inputPath, 'r', encoding='utf-8') as input:
            text = input.read()
            tags = tagger.TagText(text)

        #to write tagged files
            outputPath = os.path.join('./test_tagged/', f + ".ttr")
            with codecs.open(outputPath, 'w', encoding='utf-8') as output:
                for tag in tags:
                    output.write(tag + "\n")
    except IOError:
      print "There is an IOError in " + inputPath
```

A.2 Format emotion lists

```
# This program takes (a) file/s of an emotion lexicon as input
# (EmoLex, WAL or LIWC )
# and creates a standardized output file

import os, codecs, re

input_dir = 'C:/Users/Felicia/Documents/Praktikum/Lexicons/'
filenames = [f for f in os.listdir('./emo-lists/') if f.endswith('.txt')]
outfile = 'emolist2.txt'

infile2 = os.path.join(input_dir, 'NRC-Emotion-Lexicon-v0.92/NRC-emotion-
    lexicon-wordlevel-alphabetized-v0.92_plain.txt')
outfile2 = 'NRC-Emotion-Lexicion-v0.92_formated.txt'

filenames3 = [f for f in os.listdir('./LIWC/') if f.endswith('.txt')]
outfile3 = 'LIWC_formated.txt'
```

```
def format_direct_emotions(filenames):
    # dictionary to save tokens and associated emotions of all the files
    tok_tag_emo = {}
    output = codecs.open(outfile, 'w', encoding='utf-8')
    for f in filenames:
        emotion = f[:-4]
        print emotion
        inputPath = os.path.join('./emo-lists/', f)
        try:
            with codecs.open(inputPath, 'r', encoding='utf-8') as input:
                lines = input.readlines()
                for line in lines:
                    tokens = line.split('␣')
                    matchTag = re.match(r'(\w{1})#\d+', tokens[0])
                    if matchTag.group(1) == 'n':
                        tag = 'NN'
                    elif matchTag.group(1) == 'r':
                        tag = 'RB'
                    elif matchTag.group(1) == 'a':
                        tag = 'JJ'
                    elif matchTag.group(1) == 'v':
                         tag = 'VV'

                    for token in tokens:
                        token = token.rstrip('\n')
                        # we don't do anything with the first element of
                            tokens, i.e. the synset number
                        if token is tokens[0]:
                           pass
                        else:
                            token_tag = token, tag
                            tok_tag_emo[token_tag] = emotion # okay because
                                each word is assigned exactly to one emotion
                            print token_tag[0] + "␣" + token_tag[1] + "␣" +
                                str(tok_tag_emo[token_tag])

                for token_tag in sorted(tok_tag_emo):
                    output.write(token_tag[0] + '\t' + str(tok_tag_emo[
                        token_tag])+ '\t1\n')

        except IOError:
            print "Could␣not␣open␣file␣" + f
    output.close()
    return

def format_NRC_emotions(filename):
    output2 = codecs.open(outfile2, 'w', encoding='utf-8')
    try:
        with codecs.open(filename, 'r', encoding='utf-8') as input2:
            lines = input2.readlines()
            for line in lines:
                tokens = line.split('\t')
                tokens.insert(1, '**')
                #tokens[3] = tokens[3].rstrip('\n')
                output2.write(tokens[0] + '\t' + tokens[1] + '\t' + tokens
                    [2] + '\t' + tokens[3])
                print tokens
            output2.close()
    except IOError:
        print "Could␣not␣open␣file␣" + infile2
    return

def format_LIWC_emotions(filenames3):
    # dictionary to save tokens and associated emotions of all the files
    # the value is a list because one token can be associated with several
        emotions
    tok_emotions = {}
    output3 = codecs.open(outfile3, 'w', encoding='utf-8')
    for f in filenames3:
        emotion = f[:-4]
        print emotion
        inputPath = os.path.join('./LIWC/', f)

        try:
            with codecs.open(inputPath, 'r', encoding = 'utf-8') as input:
                lines = input.readlines()
                for token in lines:
                    token = token.rstrip('\r\n')
                    # add token, emotion to the dictionary
                    if token in tok_emotions:
                        tok_emotions[token].append(emotion) #several
                            emotions for one token
                        print token + ' ' + emotion
                    else:
                        tok_emotions[token] = [emotion]
                        print token + ' ' + emotion

        except IOError:
            print "Could not open file " + f

    for tok in sorted(tok_emotions):
        emotions = tok_emotions[tok]
        for emo in emotions:
            output3.write(tok + '\t' + '**' + '\t' + emo + '\t' + '1\n')

    output3.close()
    return

format_NRC_emotions(infile2)
```

A.3 Tag freq

```
# This program creates a frequency list of lemmas and their pos tags
# from the tagged corpus CORPS
# it extracts a window of [x] sentences before an audience reaction tag,
#  e.g. {APPLAUSE}
# and writes it to a "[audience reaction tag]_freq.txt" file.
# Additionally, it creates a file per input file in the corpus showing
# the extracted windows of lemmas
#
    ##############################################################################


import os, codecs, re

freq = {}

# tag_name: audience reaction tag like LAUGHTER, CHEERS, APPLAUSE, BOOING or
     NOTAG (no audience reaction)
# input_dir: directory of the tagged files of the corpus
# window_size: number of sentences that should be considered before an
    audience
# reaction tag
def tag_freq(tag_name, input_dir, window_size):

    count_tokens = 0

    # the input files are read form a folder "tagged" and have the ending ".
        ttr"
    filenames = [f for f in os.listdir(input_dir) if f.endswith('.ttr')]
    output2 = codecs.open(tag_name.lower() + '_freq_' + str(window_size) + '
        .txt', 'w', encoding='utf-8')

    if not os.path.exists('./'+ tag_name.lower() + '_sentences_' + str(
        window_size) + '/'):
```

```
        os.makedirs('./'+ tag_name.lower() + '_sentences_' + str(window_size
            ) + '/')

    for f in filenames:
        tokens = []
        tags = []
        lemmas = []
        i = 0

        out_filename = f[:-8] + '_' + tag_name.lower() + '_' + str(
            window_size) + '.txt'
        outputPath = os.path.join('./' + tag_name.lower() + '_sentences_' +
            str(window_size) + '/', out_filename)
        output = codecs.open(outputPath, 'w', encoding='utf-8')
        inputPath = os.path.join(input_dir, f)
        try:
            with codecs.open(inputPath, 'r', encoding='utf-8') as input:
                lines = input.readlines() #get a list of lines in the file

                for line in lines:
                    line = line.rstrip('\n')
                    # save the current token, tag and lemma in a list
                    tok_tag_lem = line.split('\t')
                    # use only complete tokens that have a tag and a lemma
                    if len(tok_tag_lem) == 3:
                        #create aligned lists for tokens, tags and lemmas
                        tokens.append(tok_tag_lem[0])
                        tags.append(tok_tag_lem[1])
                        lemmas.append(tok_tag_lem[2])

                        # the condition basically checks if an audience
                            reaction tag is reached
                        if tokens[i] == tag_name or (tag_name == 'NOTAG' and
                                ((tokens[i]=='APPLAUSE')\
                                 or (tokens[i]=='LAUGHTER')\
                                 or (tokens[i]=='CHEERS')\
                                 or (tokens[i]=='BOOING')\
                                 or (tokens[i]=='SPONTANEOUS_DEMONSTRATION')
                                    \
                                 or (tokens[i]=='STANDING_OVATIONS'))):
                            # 'i' keeps track of the position of the token
                                running through the file
                            # while 'j' keeps track of the position while
                                extracting the sentence windows
                            j = i

                            # only in case of 'NOTAG' (negative windows)
                            if tag_name == 'NOTAG':
                                j = go_x_sentences_back(20, tokens, j)
                                # go to the beginning of the loop if there
                                    is another tag before x is reached
                                if j == 0:
                                    i += 1
                                    continue

                            sentence_count = 0
                            lem_in_window = []
                            #array contains audience reaction tag(s)
                            #e.g. ['{', 'APPLAUSE', ';', 'LAUGHTER', '}'] or
                                ['{', 'APPLAUSE', '}']
                            taglist = []

                            #if audience reaction tag is reached, run
                                forwards, to see if there are multiple tags
                            y = j # y is an index to run forward in the
                                audience reaction tags without modifying 'j'
                            while (y+1) < len(tokens) and tokens[y+1] != "}"
                                :
                                y += 1
                                taglist.append(y)
```

```
  taglist.append("}")
  y = j # y is an index to run forward in the
      audience reaction tags without modifying 'j'
  # run backwards to see if there are multiple
      audience reaction tags
  while y > 0 and tokens[y-1] != "{":
      #lem_in_window.insert(0, tokens[j])
      taglist.insert(0, tokens[y])
      y -= 1
  taglist.insert(0, tokens[y])
  taglist.insert(0, "{")

  # if there is more than one audience reaction in
       taglist, e.g. ["{", "APPLAUSE", "}"]
  if len(taglist) > 3:
      pass
  #!!! Sentences with multiple audience reactions
      are not considered in the frequency list
  # anymore
  else:
# is only executed in case that there is only one
      audience reaction
#  multiple reactions are discarded
      # add tags to lem_in_window
      lem_in_window = lem_in_window + taglist
      # decrement j in order not to have twice "{"
      j-= 1

      # from TAG consider lemmas within a window
          of
      # [window_size] sentences before
      # '!= "}"' ensures that the window doesn't
          extend
      # to the scope of the tag before or EOF #and
           j >= 0
      while sentence_count <= window_size and
          lemmas[j] != "}":
          j -= 1
          # create a list of lemmas in the window
          # (to display sentences in window)
          lem_in_window.insert(0, lemmas[j])
          # create tuple of a lemma and its tag
          # (needed as key for frequency
              dictionary)
          lem_tag = lemmas[j], tags[j]

          # create dictionary of frequencies of
              lemmas
          # with tuple of lemma and tag as key and
          # the number of occurrence as value
          if lem_tag in freq:
              freq[lem_tag] += 1
              count_tokens += 1
          else:
              freq[lem_tag] = 1
              count_tokens += 1

          #. , : , !, ?, ; are used as the
              boundary of the window
          #update counter if a dot is found
          if (lemmas[j] == '.'or lemmas[j] == ':'
              or lemmas[j] == '?' or lemmas[j] =='!
              ' or lemmas[j] ==';'):
              sentence_count += 1
      #remove dot at the first position of the
          list
      lem_in_window.pop(0)

      #write window of lemmas before TAG to a file
      for lem in lem_in_window:
```

```
                                    if lem != '}':
                                        output.write(lem + '␣')
                                    else:
                                        output.write(lem)
                                output.write('\n')

                                ###End of 'else' (block excluding multiple
                                    audience reactions)
                        i += 1 #count of tagged tokens in file
            output.close()
            print f
        except IOError:
            print "Could␣not␣open␣file␣" + inputPath

    #write a frequency list of the occurrences of lemmas in the window to a
        file
    for lem_tag in sorted(freq, key = freq.get, reverse = True):
        output2.write(lem_tag[0] + '␣' + lem_tag[1] + ':␣' + str(freq[
            lem_tag]) + '\n')
    print count_tokens
    output2.close()
    return

# this function returns the position j that is x sentences before
# from the last audience reaction tag
# 'sentences' is the number of sentences, you want to go back
# 'tokens' is the list of tokens that has been read so far
def go_x_sentences_back(sentences, tokens, j):
    sentence_count = 0
    # loop until the specified number of sentences is reached or EOF
    while sentence_count < sentences and j > 0:
        # decrement j to go one token backwards
        j -= 1
        # if '}' is reached (indicating the end of a tag)
        # before the specified number of sentences, exit the loop
        if tokens[j] == '}':
            return 0 # is it a good idea to return 0?
        elif (tokens[j] == '.' or tokens[j] == '!' or tokens[j] == '?' or
            tokens[j] == ':' or tokens[j] == ';'):
            sentence_count += 1
    return j

tag_freq('APPLAUSE', 'C:/Users/Felicia/Documents/Praktikum/test_tagged/', 2)
```

A.4 Merge tags

```
# This program takes a frequency list

import os, codecs

f = 'spontaneous_demonstration_freq.txt'
inputPath = os.path.join('./', f)
out_f = f[:-4] + '_merged.txt'
output = codecs.open(out_f, 'w', encoding='utf-8-sig')
freq = {}
count_tokens = 0

def merge_tags( tag ):
  # merge adjective tags
  if tag in ('JJ', 'JJR', 'JJS'):
    return u'JJ'
  # merge adverb tags
  elif tag in ('RB', 'RBR', 'RBS'):
    return u'RB'
  # merge noun tags
  elif tag in ('NN', 'NNS'):
    return u'NN'
  # merge proper noun tags
```

```
    elif tag in ('NP', 'NPS'):
      return u'NP'
    # merge verb tags
    elif tag in ('VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ',\
           'VD', 'VDD', 'VDG', 'VDN', 'VDZ', 'VDP',\
           'VG', 'VHD', 'VHG', 'VHN', 'VHZ', 'VHP',\
           'VV', 'VVD', 'VVG', 'VVN', 'VVP', 'VVZ'):
      return u'VV'
    # merge determiner tags:
    elif tag in ('DT', 'PDT'):
      return u'DT'
    else:
      return tag

try:
  with codecs.open(inputPath, 'r', encoding='utf-8') as input:
    lines = input.readlines()

    for line in lines:
      line = line.rstrip('\n')
      lem_tag_freq = line.split(' ')
      current_tag = lem_tag_freq[1]
      if current_tag[-1] == ':':
        current_tag = current_tag[:-1]

      new_tag = merge_tags(current_tag)
      new_lem_tag = lem_tag_freq[0], new_tag # save new tuple of lemma and
          tag in lem_tag
      #print new_lem_tag

      # fill the dictionary with a tuple as key and the frequency as value
      # and unify double entries resulting from tag merging

      if new_lem_tag in freq:
        #print(int(freq[new_lem_tag]))
        freq[new_lem_tag] += int(lem_tag_freq[2]) # frequencies of
            previously different tags are added
        #print "+ %s = %s\n" % (int(lem_tag_freq[2]), int(freq[new_lem_tag])
            )
        count_tokens += int(lem_tag_freq[2])
      else:
        freq[new_lem_tag] = int(lem_tag_freq[2])
        count_tokens += int(lem_tag_freq[2])

    for new_lem_tag in sorted(freq, key = freq.get, reverse = True):
      output.write(new_lem_tag[0] + ' ' + new_lem_tag[1] + ': ' + str(freq[
          new_lem_tag]) + '\n')

    print count_tokens

except IOError:
  print "Could not open file " + inputPath
```

A.5 Assign emotions

```
# This program connects lemmas of a frequency list
# (as created by tag_freq.py) to emotions of an
# emotions lexicon and creates two output text files
#
# output files:
# [...]_emotions.txt:
# frequency list ordered by the frequency of each lemma with associated emo-
# tions
# example line: "strength  NN  507:  trust"
#
# [...]_ordered_emo.txt:
# frequency list ordered by the emotions associated with the lemmas and
   their
# frequency, then by the frequency of the lemma
#
# input files:
# -a frequency list with lines like "country NN: 5311"
# -an emotion lexicon with lines like: "abandonment NN   joy  0"
#                                      "abandonment NN    sadness 1"
#  if the POS are unknown:             "abandonment **   joy  0"
#  The tokens are separated by \t.
#
#

import os, codecs

f = 'LIWC_emotions.txt'
inputPath = os.path.join('C:/Users/Felicia/Documents/Praktikum/Lexicons/', f
    )

f2 = 'freq_corpus_merged.txt'
inputPath2 = os.path.join('./', f2)

out_f = f2[:-10] + 'emotions.txt'
output = codecs.open(out_f, 'w', encoding='utf-8')

out_f2 = f2[:-10] + 'ordered_emo.txt'
output2 = codecs.open(out_f2, 'w', encoding='utf-8')

lem_tag_emotions = {}  # dictionary of lemmas, POS tags, emotions and values
# dictionary of emotions and the sum of lemmas connected to it
count_lem_per_emotion = {}
emotions_in_lex = []  # to keept track of of the emotions contained in the
    lexicon
freq_lem_emo = {}  # a frequency list of lemmas with pos and emotions
tag_flag = 1  # 1 if POS tags should be considered

freq = {}  # lemmas, POS tags as keys and frequency as value


# read the emotion lexicon into a dictionary with a tuple of
# lemma and tag as
# key and a list of emotions and value tuples as value.
# One lemma can be associated with several emotions.
# The value of the emotion can be 0 or 1 depending on whether the emotion is
# associated with the lemma

try:
    with codecs.open(inputPath, 'r', encoding='utf-8') as input:
        lines = input.readlines()  # get a list of lines in the file
        for line in lines:
            line = line.rstrip('\n')
            # save current word, emotion and value of the emotion in a list
            lem_tag_emo_val = line.split('\t')

            # will be the key of the dictionary
            lem_tag = lem_tag_emo_val[0], lem_tag_emo_val[1]
```

```
                # will be an item of the emotions values list
                emo_val = lem_tag_emo_val[2], lem_tag_emo_val[3]

                # create a dictionary with the lem_tag as key
                # and a list of emotion-value tuples as value
                if lem_tag in lem_tag_emotions:
                    lem_tag_emotions[lem_tag].append(emo_val)
                else:
                    lem_tag_emotions[lem_tag] = [emo_val]

                # make list of emotions contained in the emotions lexicon
                emotions_in_lex.append(emo_val[0])  # NEEDED?
            emotions_in_lex = set(emotions_in_lex)

except IOError:
    print "Could not open file " + inputPath


# read frequency list of lemmas, POS and number of occurrence from file
# into dictionary

try:
    with codecs.open(inputPath2, 'r', encoding='utf-8') as input2:
        lines = input2.readlines()  # get a list of lines in the file

        for line in lines:
            line = line.rstrip('\n')
            lem_tag_freq = line.split(' ')
            tag = lem_tag_freq[1]
            if tag[-1] == ':':  # remove ':' from the end of the tag
                tag = tag[:-1]

            # save current tuple of lemma and tag in lem_tag
            lem_tag = lem_tag_freq[0], tag
            # fill the dictionary with a tuple as key and the frequency as
                value
            freq[lem_tag] = lem_tag_freq[2]
            # this flag is 2 if the lemma is not contained in
                lem_tag_emotions

            contained = 2

            # match lemma and tag of frequency list
            #with lemma and tag of emotion lexicon

            # the tag_flag should be 0, if the emotions lexicon
            # has no pos tags (but '**' as a place holder)
            # in that case the emotions dictionary can't be accessed
            # with the lem_tag tuple from the frequency list and therefore
            # has to be changed to lemma, '**'
            if tag_flag == 0:
                lem_tag2 = lem_tag[0], '**'
            else:
                lem_tag2 = lem_tag

            # the entries of the corpus frequency list are matched with
                entries of the
            # emotions dictionary. The entries of the emotions dictionary
                can be like
            # 'gratef*' : representing lemmas starting with 'gratef'
            # or like 'love' : all lemmas equal to 'love'

            for key in lem_tag_emotions:
                if lem_tag2[0] == key[0]\
                    or lem_tag2[0].startswith(key[0].rstrip('*')) and key
                        [0].endswith('*'):

                    # to access the key of the emotions dictionary, lem_tag2
                        [0] has to be set to
                    # the format in the emotions dictionary, i.e. 'gratef'
```

```
                                        instead  of  'grateful'

                                    # this  flag  is  0  if  the  word  is  contained  in
                                        words_emotions
                                    # but  no  emotions  are  associated  with  the  word
                                    contained = 0
                                    # run  through  the  list  of  emotion  value  tuples
                                    # that  are  associated  with  the  lemma
                                    # (the  value  of  the  dictionary  lem_tag_emotions  is  a
                                        list)
                                    for emo_val in lem_tag_emotions[key]:
                                       # print  emo_val[0]  +  ' '  +  lem_tag2[0]  +  ' '  +
                                           lem_tag2[1]
                                        # check  if  value  of  emotion  is  1
                                        # (i.e.  emotion  is  associated  with  lemma)
                                        if emo_val[1] == "1":
                                            # set  flag  to  1  because  there  is  an  associated
                                                emotion
                                            contained = 1
                                            emo = emo_val[0]

                                            # add  the  frequency  of  each  lemma  to  the
                                                frequency  of  each  emotion
                                            # WARNING:  if  one  lemma  is  linked  to  several
                                                emotions,  the  frequency  of  the  lemma
                                            # is  used  for  BOTH  emotions  (no  distinction  is
                                                made  if  in  a  particular  case
                                            # it's  one  or  the  other  emotion
                                            if emo in count_lem_per_emotion:
                                                count_lem_per_emotion[emo] += int(freq[
                                                    lem_tag])
                                            else:
                                                count_lem_per_emotion[emo] = int(freq[
                                                    lem_tag])

                                            # create  a  new  frequency  list  with  a  tuple  of
                                                lemma,  pos  and  emotion  as  key
                                            # and  the  frequency  as  value
                                            # the  purpose  of  the  frequency  list  is  to  be
                                                able  to  sort  primarily  by  emotion  and
                                                secondarily  by  frequency

                                            # WARNING:  the  frequency  is  only  the  frequency
                                                of  the  lemma,  e.g.  if  there  is  an  entry  '(
                                                aggressive,  JJ,  anger)  :  1'
                                            # and  '(aggressive,  JJ,  fear)  :  1',  it  means
                                                that  the  lemma  'aggressive'  occurs  once  and
                                                it  can  be  associated  with
                                            # 'fear'  and  'anger'
                                            lem_tag_emo = lem_tag[0], lem_tag[1], emo

                                            if lem_tag_emo in freq_lem_emo:
                                                freq_lem_emo[lem_tag_emo] += int(freq[
                                                    lem_tag])
                                            else:
                                                freq_lem_emo[lem_tag_emo] = int(freq[lem_tag
                                                    ])

                                            output.write(lem_tag[0] + '␣␣' + lem_tag[1] + '␣
                                                ␣' + freq[lem_tag] + ':')
                                            output.write('␣␣' + emo_val[0] + '\n')

                            if contained == 0:
                                output.write(lem_tag[0] + '␣␣' + lem_tag[1] + '␣␣' + freq[
                                    lem_tag] + ':')
                                output.write('␣␣␣␣neutral\n')

                except IOError:
                    print "Could␣not␣open␣file␣" + inputPath2
```

```
emotions_in_lex = []

output2.write('
    **************************************************************\n')
output2.write('*␣␣␣Total␣count␣of␣␣lemmas␣per␣emotion:␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
    *\n')
output2.write('
    **************************************************************\n')
for e in sorted(count_lem_per_emotion, key=count_lem_per_emotion.get,
    reverse=True):
    emotions_in_lex.append(e)
    output2.write(e + '␣' + str(count_lem_per_emotion[e]) + '\n')

if tag_flag == 0:
    output2.write('
        **************************************************************\n')
    output2.write('*␣␣␣Warning:␣the␣emotions␣lexicon␣didn\'t␣contain␣POS␣
        tags.␣␣␣*\n')
    output2.write('*␣␣␣They␣are␣just␣the␣ones␣from␣the␣corpus.␣␣␣␣␣␣␣␣␣␣␣␣␣␣
        ␣␣␣␣*\n')


output2.write('
    **************************************************************\n')
# sort the frequency list first by emotions and then by frequency of the
    lemmas
# the order of the emotions is defined by the total count of the emotion
    stored in 'count_lem_per_emotion'
for lem_tag_em in sorted(freq_lem_emo, key=lambda key: (-
    count_lem_per_emotion[key[2]], -freq_lem_emo[key])):
    output2.write(
        lem_tag_em[0] + '␣' + lem_tag_em[1] + '␣' + lem_tag_em[2] + ':␣' +
            str(freq_lem_emo[lem_tag_em]) + '\n')
```

A.6 Normalize

```
# This program computes for lines of lemmas (that are the files from
    tag_frey.py:
# -the ratio of emotional words
#  (i.e. associated to an emotion) to all words
# -the ratio of words associated to
#   a specific emotion (e.g. sadness) to all words
# -the ratio of words associated to
#  a specific emotion (e.g. sadness)
# to number of emotional words

import codecs, os

def read_lexicon(emo_lex_name):

    emoLexPath = os.path.join('../Lexicons/NRC-Emotion-Lexicon-v0.92/',
        emo_lex_name)

    #lemmas-emotions dictionary
    # dictionary format:
    #{(lemma, pos_tag) : [(emotion1, value), (emotion2, value)...] , ...}
    lem_emo = {}

    try:
        with codecs.open(emoLexPath, 'r', encoding = 'utf-8') as input:
            lines = input.readlines()
            for line in lines:
                line = line.rstrip('\n')
                tokens = line.split('\t')
                #tuple of lemma and pos tag
                lem_tag = tokens[0], tokens[1]
                #tuple of lemma emotion and value
                emo_val = tokens[2], tokens[3]
                #fill the lemmas-emotions dictionary
                if lem_tag in lem_emo:
                    lem_emo[lem_tag].append(emo_val)
                else:
                    lem_emo[lem_tag] = [emo_val]
    except IOError:
        print 'could␣not␣open␣file:␣' + emoLexPath
    return lem_emo

def read_and_count(filename):

    emoSentPath = os.path.join('../audience_reaction_sentences/', filename)
    outfile = filename[:-4] + '_emo.txt'
    output = codecs.open(outfile, 'w', encoding = 'utf-8')

    lem_emo = read_lexicon('NRC-Emotion-Lexicon-v0.92_formated.txt')
    stopwords = read_stopwords('conservative_stopwords.txt')

    try:
        with codecs.open(emoSentPath, 'r', encoding = 'utf-8') as input:

            lines = input.readlines()
            for line in lines:
                line = line.rstrip('\n')
                line = line.rstrip('␣')
                tokens = line.split('␣')
                # for the number of occurences of each emotion in a window
                    before an audience reaction
                emo_in_window = {}
                tokens_count = 0
                emotional_count = 0
                line_to_print = ''

                for token in tokens:
                    if token in stopwords:
                        pass
                    elif token in reaction_tags:
                        pass
                    elif token in reaction_tags_lower:
                        pass
                    else:
                        #count total of tokens in window before reaction tag
                        tokens_count += 1
                        #if flag1 is still 0 after running through the
                            entries of the emo lexicon
                        #the lemma is not contained in the lexicon
                        flag = 0
                        for key in lem_emo:
                            if key[0] == token:
                                flag = 1
                                emotional_count += 1
                                # if flag is still 1 after running through
                                    the associated emotions
                                # a lemma is in the emotional lexicon but
                                    not associated with an emotion
                                flag2 = 0
                                # run through the values of the dictionary
                                    to get the associated emotion
                                for emo_val in lem_emo[key]:
                                    if int(emo_val[1]) == 1:
                                        line_to_print += token + '[' +
                                            emo_val[0] + ']␣'
                                        if emo_val[0] in emo_in_window:
                                            emo_in_window[emo_val[0]] += 1
                                        else:
                                            emo_in_window[emo_val[0]] = 1
                                        flag = 2
                                #don't count lemma as emotional if it's in
                                    the lexicon but no association
                                if flag == 1:
                                    line_to_print += token + '[no_emo]␣'
                                    #but add 'no emotion' to dictionary of
                                        emotions in window
                                    #emo_val has to be changed to ('no_emo
                                        ',1)
                                    emo_val = 'no_emo', 1
                                    if emo_val[0] in emo_in_window:
                                        emo_in_window[emo_val[0]] += 1
                                    else:
                                        emo_in_window[emo_val[0]] = 1

                                    emotional_count -= 1
                        if flag == 0:
                            line_to_print += token + '␣'
                calc_normaliz(tokens_count, emotional_count, emo_in_window)

                line_to_print = line_to_print + '\n'
                # if after reading one line tokens_count is still 0, don't
                    print the line
                if tokens_count > 0:
                    output.write(line_to_print)
        output.close()
    except IOError:
        print 'could␣not␣open␣file:␣' + filename
    return 0

def calc_normaliz(tokens_count, emotional_count, emo_dico):

    #don't consider lines, where tokens_count is 0, i.e. they are composed
        by stopwords and tags
    if tokens_count == 0:
        return 1

    #ratio of emotion evoking lemmas to all lemmas
    emo_all_avg = float(emotional_count)/tokens_count
    output.write(str(emo_all_avg) + '\t||')

    for emo in emolist:
        if emo in emo_dico:
            count = emo_dico[emo]
            #ratio of occurence of lemmas of an emotional category to all
                lemmas
            emo_cat_avg1 = float(count)/tokens_count
            output.write(str(emo_cat_avg1) + '\t')
        # emotion that doesn't occur in the window
        else:
            emo_cat_avg1 = float(0)
            output.write(str(emo_cat_avg1) + '\t')

    output.write('||')

    for emo in emolist:
        if emo in emo_dico:
            count = emo_dico[emo]
            #ratio of occurence of lemmas of an emotional category to
                emotional lemmas

            # Avoid division by 0
            # For the case that there are no emotional lemmas in the window
                but a neutral lemma
            if emotional_count == 0 and count >= 1:
                emo_cat_avg2 = float(-1)
                #output.write(str(count) + '/' + str(emotional_count) + '\t
                    ')
                output.write(str(emo_cat_avg2) + '\t')
            # For the case that there ARE emotional lemmas in the window
            else:
                emo_cat_avg2 = float(count)/emotional_count
                output.write(str(emo_cat_avg2) + '\t')
        # emotion doesn't occur in the window
        else:
             # Avoid division by 0
            # For the case that there are no emotional lemmas in the window
            if emotional_count == 0:
                emo_cat_avg2 = float(0)
                output.write(str(emo_cat_avg2) + '\t')
            # For the case that there ARE emotional lemmas in the window
            # but not this emotion
            elif emotional_count >=1:
                emo_cat_avg2 = float(-2)
                output.write(str(emo_cat_avg2) + '\t')
    output.write('||\n')
    return 0

def read_stopwords(stopfile):
    stopwords = []
    stopwordsPath = os.path.join('../Lexicons/', stopfile)
    try:
        with codecs.open(stopwordsPath, 'r', encoding='utf-8') as inp_stop:
            lines = inp_stop.readlines()
            for line in lines:
                line = line.rstrip('\n')
                stopwords.append(line)
    except IOError:
        print 'Can\'t open file: ' + stopwordsPath + '\n'
    return stopwords


filename = 'applause1.txt'
#values have to be adapted for each dictionary
emolist = ['anger', 'anticipation', 'disgust', 'fear', 'joy', 'sadness', '
    surprise', 'trust', 'negative', 'positive', 'no_emo']
reaction_tags =['APPLAUSE', 'LAUGHTER', 'CHEERS', 'BOOING', 'STANDING-
    OVATION', 'SUSTAINED_APPLAUSE', 'NOTAG']
# To remove audience reaction tags that are not capitalized
reaction_tags_lower = ['Applause', 'Laughter', 'Cheers', 'Booing']

# concatenate the emotions of emolist
emotions = ''
for emo in emolist:
    emotions += '\t' + emo

outfile = filename[:-4] + '_avg.txt'
output = codecs.open(outfile, 'a', encoding = 'utf-8')
output.write('divided_by_tokens: emotional\t|| devided_by_tokens: ' +
    emotions + '\t'
             '|| divided_by_emotional: ' + emotions + '\n')
output.flush()
read_and_count(filename)


#comments:
#
#Counts each token as emotional, if it is associated with an emotion.
#If one token is associated to several emotions, count doesn't get more than
     one.
#Counts also tokens that are not associated with an emotion but in the
    lexicon.
```

A.7 Create virtual documents

```
# This program extracts a window of [x] sentences before an
# audience reaction tag,
#  e.g. {APPLAUSE}, from the tagged corpus CORPS
# and writes the extracted windows of lemmas to
# to "[audience reaction tag]_sentences_[window size].txt"
# It writes the remaining content of the files, i.e. the parts
# of each file that don't occur before an audience reaction to
# another file

import os, codecs

freq = {}

# tag_name: audience reaction tag like LAUGHTER, CHEERS, APPLAUSE, BOOING or
     NOTAG (no audience reaction)
# input_dir: directory of the tagged files of the corpus
# window_size: number of sentences that should be considered before an
    audience
# reaction tag
def tag_freq(tag_name, input_dir, window_size):

    count_tokens = 0

    # the input files are read form a folder "tagged" and have the ending ".
        ttr"
    filenames = [f for f in os.listdir(input_dir) if f.endswith('.ttr')]
    output2 = codecs.open(tag_name.lower() + '_freq_' + str(window_size) + '
        .txt', 'w', encoding='utf-8')

    if not os.path.exists('./'+ tag_name.lower() + '_sentences_' + str(
        window_size) + '/'):
        os.makedirs('./'+ tag_name.lower() + '_sentences_' + str(window_size
            ) + '/')

    if not os.path.exists('./' + tag_name.lower() + '_remaining_sentences/')
        :
        os.makedirs('./' + tag_name.lower() +'_remaining_sentences/')

    for f in filenames:
        tokens = []
        tags = []
        lemmas = []

        out_filename = f[:-8] + '_' + tag_name.lower() + '_' + str(
            window_size) + '.txt'
        out_filename2 = f[:-8] + '_rest.txt'
        outputPath = os.path.join('./' + tag_name.lower() + '_sentences_' +
            str(window_size) + '/', out_filename)
        outputPath2 = os.path.join('./' + tag_name.lower() +'
            _remaining_sentences/', out_filename2)
        output = codecs.open(outputPath, 'w', encoding='utf-8')
        output3 = codecs.open(outputPath2, 'w', encoding='utf-8')
        inputPath = os.path.join(input_dir, f)
        try:
            with codecs.open(inputPath, 'r', encoding='utf-8') as input:
                lines = input.readlines() #get a list of lines in the file

                for line in lines:
                    line = line.rstrip('\n')
                    # save the current token, tag and lemma in a list
                    tok_tag_lem = line.split('\t')
                    # use only complete tokens that have a tag and a lemma
                    if len(tok_tag_lem) == 3:
                        #create aligned lists for tokens, tags and lemmas
                        tokens.append(tok_tag_lem[0])
                        tags.append(tok_tag_lem[1])
                        lemmas.append(tok_tag_lem[2])
```

```
#at this point, tokens, tags and lemmas are 'full'
# save position of tokens that were in the audience reaction
     window
index_tokens_in_window = []
# save indices of tags, headers, so that they can be
    discarded form printing
index_tags = []
for i in range(0, len(tokens) ):
        # add indices of the header to index_tags
        if tokens[i] == 'speech' and tokens[i-1] == '{':
            index_tags.extend(range(0, i+2))
            print index_tags
        elif tokens[i] == tag_name:
            # the condition basically checks if an audience
                reaction tag is reached
            # 'i' keeps track of the position of the token
                running through the file
            # while 'j' keeps track of the position while
                extracting the sentence windows
            j = i

            sentence_count = 0
            lem_in_window = []
            lem_out_window = []
            #array contains audience reaction tag(s)
            #e.g. ['{', 'APPLAUSE', ';', 'LAUGHTER', '}'] or
                 ['{', 'APPLAUSE', '}']
            taglist = []

            #if audience reaction tag is reached, run
                forwards, to see if there are multiple tags
            y = j # y is an index to run forward in the
                audience reaction tags without modifying 'j'
            while (y+1) < len(tokens) and tokens[y+1] != "}"
                :
                y += 1
                taglist.append(y)
            taglist.append("}")

            y = j # y is an index to run forward in the
                audience reaction tags without modifying 'j'
            # run backwards to see if there are multiple
                audience reaction tags
            while y > 0 and tokens[y-1] != "{":
                taglist.insert(0, tokens[y])
                y -= 1
            taglist.insert(0, tokens[y])
            taglist.insert(0, "{")
            print taglist
            # add tags to lem_in_window
            lem_in_window = lem_in_window + taglist
            # decrement j in order not to have twice "{"
            j-= 1

            # from TAG consider lemmas within a window of
            # [window_size] sentences before
            # '!= "}"' ensures that the window doesn't
                extend
            # to the scope of the tag before or EOF #and j
               >= 0
            while sentence_count <= window_size and lemmas[j
                ] != "}":
                j -= 1
                # create a list of lemmas in the window
                # (to display sentences in window)
                if len(taglist) > 3:
                    index_tags.append(j)
                else:
                    lem_in_window.insert(0, lemmas[j])
```

```
                                    index_tokens_in_window.append(j)
                                # create tuple of a lemma and its tag
                                # (needed as key for frequency dictionary)
                                lem_tag = lemmas[j], tags[j]

                                # create dictionary of frequencies of lemmas
                                # with tuple of lemma and tag as key and
                                # the number of occurrence as value
                                if len(taglist) <= 3:
                                    if lem_tag in freq:
                                        freq[lem_tag] += 1
                                        count_tokens += 1
                                    else:
                                        freq[lem_tag] = 1
                                        count_tokens += 1
                                #. , : , !, ?, ; are used as the boundary of
                                     the window
                                #update counter if a sentence boundary is
                                     found
                                if (lemmas[j] == '.'or lemmas[j] == ':' or
                                   lemmas[j] == '?' or lemmas[j] =='!' or
                                   lemmas[j] ==';')\
                                        and (lemmas[j-1] != 'Mr' and lemmas[
                                            j-1] != 'Mrs' and lemmas[j-1] !=
                                            'Ms'):
                                    sentence_count += 1

                            #write window of lemmas before TAG to a file
                            #if len(taglist)<=3:
                            for lem in lem_in_window:
                                if lem != '}':
                                    output.write(str(lem) + '␣')
                                else:
                                    output.write(lem)
                            output.write('\n')

            output.close()
            for i in range( 0 , len(tokens) ):
                if i not in index_tokens_in_window and i not in index_tags:
                    #then it was never in a window of applause
                    output3.write(tokens[i] + '␣')
                    #then i take it (unless the token at position i is {
                        LAUGHTER})
            output3.close()
            print f
        except IOError:
            print "Could␣not␣open␣file␣" + inputPath

    #write a frequency list of the occurrences of lemmas in the window to a
        file
    for lem_tag in sorted(freq, key = freq.get, reverse = True):
        output2.write(lem_tag[0] + '␣' + lem_tag[1] + ':␣' + str(freq[
            lem_tag]) + '\n')
    print count_tokens
    output2.close()
    return

tag_freq('CHEERS', 'C:/Users/Felicia/Documents/Praktikum/tagged/', 2)
```

A.8 Persuasive impact

```
# This program takes as input the files created with
# create_virtual_ documents.py
# (i.e. files of only lemmas not followed by audience reactions
# and one virtual document containing all audience reaction windows
# for an audience reactions )
# It calculates the persuasive impact for one an audience reaction

import os, codecs, re, math, sys

# this function reads the stopwords from a file and returns an array of
    stopwords
def read_stopwords(lex_file):
    stopwords = []
    try:
        with codecs.open(lex_file, 'r', encoding='utf-8') as input_lex:
            lines = input_lex.readlines()
            print "Reading stopword lexicon: " + lex_file + "..."
            for line in lines:
                token = line.rstrip('\n')
                stopwords.append(token)
    except IOError:
        print "Could not read stopword file: " + lex_file
        sys.exit()
    return stopwords

# this function removes stopwords and unwanted characters
# and returns an array of arrays of lemmas in an audience reaction window
def remove_stopwords(inp_file, stopwords):

    windows = []

    try:
        with codecs.open(inp_file, 'r', encoding='utf-8') as input:
            lines = input.readlines()
            print "Reading file of audience reaction windows: " + inp_file +
                "..."
            for line in lines:
                tokens = line.split(' ')
                #copy tokens to another array, because tokens can't be
                    modified while looping through it
                lemmas = list(tokens)
                lemmas = [lem.lstrip('}') for lem in lemmas]
                for i in range(0, len(tokens)):
                    tokens[i] = tokens[i].lstrip('}')
                    token = tokens[i]
                    #remove stopwords from lemmas
                    if token in stopwords and token in lemmas:
                        matchObj = re.match(r'.*\{.*', token)
                        # if the token is the start of a tag, remove the
                            whole tag from lemmas
                        if matchObj:
                            j = lemmas.index(token)
                            del lemmas[j:]
                        # if the token is a stopword, remove stopword form
                            lemmas
                        else:
                            lemmas.remove(token)
                #create an array of arrays containing lemmas of each
                    sentence
                windows.append(lemmas)
    except IOError:
        print 'Could not open file: ' + inp_file
        sys.exit()
    return windows

# this function takes the array of arrays of lemmas in an audience reaction
    window as input
```

```
# and returns the number of occurrences of each lemma in the document,
# its weight and the number occurrences of all lemmas
def get_values_for_tf(windows):
    # to save the number of occurrences of each lemma
    # in the document before an audience reaction (n_i)
    lemma_freq = {}
    # sum of the scores of the lemma (the closer to the tag, the higher the
        score)
    lemma_weight = {}
    total_lemmas = 0
    print "Computing values for tf_i from audience reaction windows..."
    for line in windows:
        win_length = len(line)
        for i in range(0, win_length):
            lemma = line[i]
            total_lemmas += 1
            #print '#' + line[i] + '#',
            weight = 0.0
            # save number of occurrence of each token in the window in
                lemma_freq
            if lemma in lemma_freq:
                lemma_freq[lemma] += 1
            else:
                lemma_freq[lemma] = 1
            # the weight can be computed using the number of tokens in the
                window
            # the closer to the audience reaction, the higher the weight
                should be
            weight = float(1 / float(len(line) - i) )

            if lemma in lemma_weight:
                lemma_weight[lemma] += weight
            else:
                lemma_weight[lemma] = weight

        # the weight of each term/lemma is the average of its weights
        for lemma in lemma_weight:
            lemma_weight[lemma] = lemma_weight[lemma] / lemma_freq[lemma]

    return [lemma_freq, lemma_weight, total_lemmas]

# this function takes the input directory of the documents with the passages
     that don't occur before an
# audience reaction and the array of windows of arrays of lemmas before
    audience reactions
def get_values_for_idf(inp_dir, windows):

    # total number of documents in the corpus
    doc_count = 0
    # number of documents in the corpus where the current lemma (t_i)
        appears
    lemma_docs = {}
    # take the list of windows of lists of lemmas and create one list of
        unique lemmas
    # that correspond to the terms of the audience reaction windows
    my_terms = []
    for window in windows:
        my_terms += window
    set(my_terms)
    print "Computing values for idf_i from all documents..."
    # for each term of audience reaction window, look in each file of corpus
         whether it occurs:

    # for each word in each document check if it is in the list of terms, if
         yes, update
    # lemma_docs{lemma} += 1

    # IMPORTANT: the document of audience reaction windows
    # should be included in the directory!!!
    # otherwise there will be key errors for lemmas/terms
    # that occur only before audience reactions!!
    filenames = [f for f in os.listdir(inp_dir) if f.endswith('.txt')]
    for f in filenames:
        print "Reading␣file␣" + inp_dir + f + '...'
        doc_count += 1
        terms_in_doc = []
        inputPath = os.path.join(inp_dir, f)
        try:
            with codecs.open(inputPath, 'r', encoding = "utf-8") as input:
                text = input.read()
                if text.startswith(u'\ufeff'):
                    text = line[1:]
                tokens = text.split('␣')
                for token in tokens:
                    if token in my_terms:
                        # create a list of lemmas/terms that are in the
                            document
                        # and in the list of terms before an audience
                            reaction
                        terms_in_doc.append(token)
                # merge occurrences of same lemma/term in a document as we
                    want to count them only once per document
                terms_in_doc = set(terms_in_doc)
                # for each document, update each term that occurs
                for term in terms_in_doc:
                    if term in lemma_docs:
                        lemma_docs[term] += 1
                    else:
                        lemma_docs[term] = 1
        except IOError:
            print "Can't␣open␣file:␣" + inp_file
            sys.exit()

    return [doc_count, lemma_docs]

def compute_tf_idf(lemma_freq, lemma_weight, total_lemmas, doc_count,
    lemma_docs):

    #this dictionary stores each term and its tf_idf score
    tf_idf = {}
    print "Computing␣tf_idf␣for␣each␣term/lemma..."
    # for each term in lemma_freq, i.e. each term before an audience
        reaction
    for term_i in lemma_freq:

        # compute tf_i:
        # multiply the number of occurrences of the term with its score
        score_i = lemma_freq[term_i] * lemma_weight[term_i]
        # divide by the total_lemmas
        tf_i = score_i / total_lemmas

        # compute idf_i:
        # divide the number of documents by the number of documents, in
            which the term occurs
        idf_i = float(doc_count) / float(lemma_docs[term_i])
        idf_i = math.log(idf_i)

        # to get tf_idf for term i:
        tf_idf_i = tf_i * idf_i

        # store term and score in dico
        tf_idf[term_i] = tf_idf_i

    return tf_idf

input_folder = './booing_sentences_2/'
input_file = 'all_booing.txt'
input_directory = input_folder + input_file
stopwords = read_stopwords('../Lexicons/conservative_stopwords.txt')
print "Stopword␣lexicon␣has␣been␣read."
```

```
windows = remove_stopwords(input_directory, stopwords)
print input_directory + ' has been read, stopwords and crap have been 
    removed.'

# IMPORTANT: the document of audience reaction windows
# should be included in the directory of remaining sentences!!!
# otherwise there will be key errors for lemmas/terms that occur only before
     audience reactions!!
doc_count, lemma_docs = get_values_for_idf('./booing_remaining_sentences/',
    windows)
print "All documents have been read and values for idf have been calculated.
    "

lemma_freq, lemma_weight, total_lemmas = get_values_for_tf(windows)
print "Values for tf have been calculated."

tf_idf = compute_tf_idf(lemma_freq, lemma_weight, total_lemmas, doc_count,
    lemma_docs)

print "Writing tf_idf of each lemma to output file..."
output = codecs.open('tf_idf_' + input_file, 'w', encoding = 'utf-8')
for term in sorted(tf_idf, key = tf_idf.get, reverse = True):
    output.write(term + '\t' + str(tf_idf[term])+ '\t' + str(lemma_freq[term
        ]) + '\t' + str(lemma_weight[term]) + '\t' + str(lemma_docs[term]) +
        '\n')
print "Done"
```

7. Bibliography

Aman, S. and Szpakowicz, S. (2007). Identifying expressions of emotion in text. In *Proceedings of the 10th International Conference on Text, Speech and Dialogue (TSD 2007)*, pages 196–205.

Baccianella, S., Esuli, A., and Sebastiani, F. (2010). SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 10)*, pages 2200–2204.

Blair, J. A. (2012). Argumentation as rational persuasion. *Argumentation*, 26(1):71–81.

Bradley, M. M. and Lang, P. J. (1999). Affective norms for English words (ANEW): Instruction manual and affective ratings. Technical report, The center for research in psychophysiology, University of Florida.

Brynielsson, J., Johansson, F., and Westling, A. (2013). Learning to classify emotional content in crisis-related tweets. In *Proceedings of the 11th IEEE International Conference on Intelligence and Security Informatics (ISI 2013)*, pages 33–38.

Burnard, L. and Aston, G. (1998). *The BNC handbook: exploring the British National Corpus*. Edinburgh University Press.

Cambria, E., Hussain, A., Havasi, C., and Eckl, C. (2010). Senticspace: visualizing opinions and sentiments in a multi-dimensional vector space. In *Knowledge-Based and Intelligent Information and Engineering Systems*, pages 385–393. Springer.

Chaiken, S. (1980). Heuristic versus systematic information processing and the use of source versus message cues in persuasion. *Journal of personality and social psychology*, 39(5):752.

Chaiken, S. (1987). The heuristic model of persuasion. In *Proceedings of the 5th Ontario Symposium on Personality & Social Psychology: Social Influence*, volume 5, pages 3–39.

Chaumartin, F.-R. (2007). UPAR7: A knowledge-based system for headline sentiment tagging. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval 2007)*, pages 422–425.

Dodds, P. S., Clark, E. M., Desu, S., et al. (2015). Human language reveals a universal positivity bias. *Proceedings of the National Academy of Sciences*, 112(8):2389–2394.

Dunn, J. R. and Schweitzer, M. E. (2005). Feeling and believing: the influence of emotion on trust. *Journal of personality and social psychology*, 88(5):736.

Ekman, P. (1992). An argument for basic emotions. *Cognition & emotion*, 6(3-4):169–200.

Fellbaum, C. (1998). *WordNet*. Wiley Online Library.

Ghazi, D., Inkpen, D., and Szpakowicz, S. (2010). Hierarchical versus flat classification of emotions in text. In *Proceedings of the NAACL-HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text (CAAGET 2010)*, pages 140–146.

Griskevicius, V., Goldstein, N. J., Mortensen, C. R., Sundie, J. M., Cialdini, R. B., and Kenrick, D. T. (2009). Fear and loving in Las Vegas: Evolution, emotion, and persuasion. *Journal of Marketing Research*, 46(3):384–395.

Guerini, M., Gatti, L., and Turchi, M. (2013a). Sentiment analysis: How to derive prior polarities from SentiWordNet. In *Proceedings of the 18th Conference on Empirical Methods on Natural Language Processing (EMNLP 2013)*, pages 1259–1269.

Guerini, M., Giampiccolo, D., Moretti, G., Sprugnoli, R., and Strapparava, C. (2013b). The new release of CORPS: A corpus of political speeches annotated with audience reactions. In *Multimodal Communication in Political Speech. Shaping Minds and Social Action*, pages 86–98. Springer.

Guerini, M., Stock, O., Zancanaro, M., OKeefe, D. J., Mazzotta, I., de Rosis, F., Poggi, I., Lim, M. Y., and Aylett, R. (2011). Approaches to verbal persuasion in intelligent user interfaces. In *Emotion-Oriented Systems*, pages 559–584. Springer.

Guerini, M., Strapparava, C., and Stock, O. (2008). CORPS: A corpus of tagged political speeches for persuasive communication processing. *Journal of Information Technology & Politics*, 5(1):19–32.

Havasi, C., Speer, R., and Alonso, J. (2007). ConceptNet 3: a flexible, multilingual semantic network for common sense knowledge. In *Proceedings of the 12th International Conference on Recent Advances in Natural Language Processing (RANLP 2007)*, pages 27–29.

Izard Carroll, E. (1977). *Human emotions*. Plenum Press.

Katz, P., Singleton, M., and Wicentowski, R. (2007). SWAT-MP: the SemEval-2007 systems for task 5 and task 14. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval 2007)*, pages 308–313.

Kozareva, Z., Navarro, B., Vázquez, S., and Montoyo, A. (2007). UA-ZBSA: a headline emotion classification through web information. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval 2007)*, pages 334–337.

Lazarus, R. S. (1984). On the primacy of cognition. 39(2):124–129.

Mehrabian, A. and Russell, J. A. (1974). *An approach to environmental psychology*. MIT Press.

Mohammad, Saif M. ; Turney, P. D. (2013). Crowdsourcing a word-emotion association lexicon. *Computational Intelligence*, 29:436–465.

Neviarouskaya, A., Prendinger, H., and Ishizuka, M. (2010). EmoHeart: conveying emotions in second life based on affect sensing from text. *Advances in Human-Computer Interaction*, 2010:1.

Ortony, A. and Turner, T. J. (1990). What's basic about basic emotions? *Psychological review*, 97(3):315.

Özbal, G. and Pighin, D. (2013). Evaluating the impact of syntax and semantics on emotion recognition from text. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing 2013)*, pages 161–173.

Pang, B. and Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Pennebaker, J. W., Francis, M. E., and Booth, R. J. (2001). Linguistic inquiry and word count: LIWC 2001.

Petty, R. E. and Cacioppo, J. T. (1984). The effects of involvement on responses to argument quantity and quality: Central and peripheral routes to persuasion. *Journal of personality and social psychology*, 46(1):69.

Petty, R. E. and Cacioppo, J. T. (1986). The Elaboration Likelihood Model of persuasion. *Advances in Experimental Social Psychology*, 19:123–205.

Plutchik, R. (1980). A general psychoevolutionary theory of emotion. *Theories of emotion*, 1:3–31.

Poels, K. and Hoste, V. (2015). Automatic emotion detection in social media for on the fly organizational crisis communication. In *Proceedings of the 1st International Conference on Human and Social Analytics (HUSSO 2015)*, pages 49–50.

Sanbonmatsu, D. M. and Kardes, F. R. (1988). The effects of physiological arousal on information processing and persuasion. *Journal of Consumer Research*, 15(3):379–385.

Schwarz, N. and Bless, H. (1991). Happy and mindless, but sad and smart? the impact of affective states on analytic reasoning. In Forgas, J., editor, *Emotion and social judgments*, pages 55–71. Psychology Press.

Simons, H. W. (1976). *Persuasion: Understanding, practice, and analysis*. Addison-Wesley Publishing Company.

Staiano, J. and Guerini, M. (2014). DepecheMood: a lexicon for emotion analysis from crowd-annotated news. *Proceeding of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 427–433.

Stone, P., Dunphy, D., and Smith, M. (1966). *The General Inquirer: A Computer Approach to Content Analysis*. MIT press.

Strapparava, C., Guerini, M., and Stock, O. (2010). Predicting persuasiveness in political discourses. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)*, pages 1342–1345.

Strapparava, C. and Mihalcea, R. (2007). SemEval-2007 task 14: Affective text. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval 2007)*, pages 70–74.

Strapparava, C. and Mihalcea, R. (2008). Learning to identify emotions in text. In *Proceedings of the 23rd 2008 ACM Symposium on Applied Computing (SAC 2018)*, pages 1556–1560.

Strapparava, C. and Valitutti, A. (2004). WordNet-Affect: an affective extension of WordNet. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, pages 1083–1086.

Subasic, P. and Huettner, A. (2001). Affect analysis of text using fuzzy semantic typing. *IEEE Transactions on Fuzzy Systems*, 9(4):483–496.

Turney, P. D. (2002). Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACM 2002)*, pages 417–424.

Ullah, R., Amblee, N., Kim, W., and Lee, H. (2016). From valence to emotions: Exploring the distribution of emotions in online product reviews. *Decision Support Systems*, 81:41–53.

Valentino, N. A., Brader, T., Groenendyk, E. W., Gregorowicz, K., and Hutchings, V. L. (2011). Election night's alright for fighting: The role of emotions in political participation. *The Journal of Politics*, 73(01):156–170.

Zajonc, R. B. (1984). On the primacy of affect. 39(2):117–123.