Polarity Analysis of Texts using Discourse Structure

Bas Heerschop[1]
basheerschop@gmail.com

Frank Goossen[1]
frank.goossen@xs4all.nl

Alexander Hogenboom[1]
hogenboom@ese.eur.nl

Flavius Frasincar[1]
frasincar@ese.eur.nl

Uzay Kaymak[1,2]
u.kaymak@ieee.org

Franciska de Jong[1,3]
f.m.g.dejong@utwente.nl

[1]Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, the Netherlands
[2]Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, the Netherlands
[3]Universiteit Twente, P.O. Box 217, 7500 AE Enschede, the Netherlands

ABSTRACT

Sentiment analysis has applications in many areas and the exploration of its potential has only just begun. We propose Pathos, a framework which performs document sentiment analysis (partly) based on a document's discourse structure. We hypothesize that by splitting a text into important and less important text spans, and by subsequently making use of this information by weighting the sentiment conveyed by distinct text spans in accordance with their importance, we can improve the performance of a sentiment classifier. A document's discourse structure is obtained by applying Rhetorical Structure Theory on sentence level. When controlling for each considered method's structural bias towards positive classifications, weights optimized by a genetic algorithm yield an improvement in sentiment classification accuracy and macro-level F_1 score on documents of 4.5% and 4.7%, respectively, in comparison to a baseline not taking into account discourse structure.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Linguistic processing*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Discourse*

General Terms

Algorithms, Experimentation, Performance

Keywords

Discourse Structure, Linguistics, Polarity, Sentiment, RST

1. INTRODUCTION

Sentiment analysis is a rather young research area focusing on how to determine the attitude or subjectivity of a text. It has many applications, such as mining social media like Facebook and Twitter for consumer opinions about products and brands. For companies, sentiment analysis on data obtained from stakeholders can provide highly valuable information. The importance of sentiment analysis in specific areas has been indicated in [1, 5, 15, 16, 18], for financial markets, politics, organizations, brand management, and economic systems, respectively.

The goal of sentiment analysis is typically to determine the polarity of a piece of natural language text. Sentiment analysis methods are mainly rooted in, among others, natural language processing, computational linguistics, and text mining. Several research directions are explored in recent literature, including word sentiment scoring (i.e., learning sentiment scores of single words), subject/aspect relevance filtering (i.e., determining the relevant subject and/or aspect for a sentiment-carrying word), sentiment negation and amplification, and subjectivity analysis (i.e., determining whether sentences are subjective or objective) [12].

A typical approach to sentiment analysis is to use frequencies of positive and negative words in order to determine whether a document is predominantly positive or negative [26, 31]. Such an approach ignores structural aspects of a document, whereas these aspects may contain valuable information [14, 26, 30]. Yet, using knowledge obtained from structural elements in texts is a relatively unexplored direction in sentiment analysis. When capturing a text's discourse structure, this knowledge could be used to improve sentiment analysis (e.g., by assigning different weights to conclusions and footnotes, as conclusions may be more important for the overall sentiment of a text than footnotes).

A popular model for analyzing a text's discourse structure is the Rhetorical Structure Theory (RST) [19]. RST describes how to split a text into spans, each representing a meaningful part of the text. These spans can be either a nucleus or a satellite. A nucleus is considered to be the span with the highest degree of importance with respect to its related spans. Satellites support the nuclei and can therefore be seen as less important spans. By splitting a text into important and less important parts, we can treat these parts differently from each other when determining the overall sentiment. We hypothesize that by weighting text spans in accordance with their importance for the overall document sentiment, the detected document sentiment can be more reliable. We aim to investigate whether the use of discourse structure in sentiment analysis has a significant added value. For this purpose, we propose Pathos, a sentiment analysis framework that can interpret a text using RST, and use this information to classify the text's polarity.

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.

The paper is structured as follows. Section 2 presents the related work. Subsequently, Section 3 elaborates on our proposed sentiment analysis framework, after which we present the results of our evaluation in Section 4. In Section 5, we draw conclusions and propose directions for future work.

2. RELATED WORK

In a recent literature survey on sentiment analysis, Pang and Lee [25] attribute the recent surge of research interest in systems that deal with opinions and sentiment to the fact that, despite today's users' hunger for and reliance upon online advice and recommendations, explicit information on user opinions is often hard to find, confusing, or overwhelming. As sentiment analysis tools may be particularly useful in analyzing, e.g., reviews, a widely used corpus for assessing sentiment analysis approaches is a collection of 2,000 English movie reviews, annotated for sentiment [23].

2.1 State of the Art

In many existing sentiment analysis approaches, a document is represented as a bag of words, i.e., an unordered collection of the words occurring in a document. Such an approach allows for vector representations of documents, enabling the use of machine learning techniques like Support Vector Machines for classifying documents. Features in such representations may be for instance words or parts of words. A binary representation of documents, indicating the presence or absence of specific words, has proven to be an effective approach, yielding an accuracy of 87.2% on the movie review data set [23]. Later research has focused on adding other features to the vector representations of documents. For instance, Whitelaw et al. [32] added features representing semantic distinctions between words based on the Appraisal Theory [20], thus yielding an accuracy of 90.2% on the movie review data set. Paltoglou and Thelwall [22] report a leave-one-out accuracy of 96.9% on this data set, obtained by using *tf-idf*-based weights for word features rather than using a binary representation of documents.

However, even though classifiers like the ones mentioned above may perform very well in the domain that they have been trained on, their performance drops tremendously when they are used in a different domain. In this light, lexicon-based methods, operating at a deeper level of analysis by incorporating the semantic orientation of individual words, can be used as an alternative [29]. A sentiment lexicon typically contains words and their associated sentiment, possibly differentiated by Part-of-Speech (POS) and/or meaning. A relatively straightforward lexicon-based sentiment analysis framework has been shown to have an accuracy up to 59.5% on the full movie review data set [11]. A more sophisticated lexicon-based sentiment analysis approach has been shown to have an accuracy of 59.6% to 76.4% on 1,900 documents from the movie review data set, depending on the sentiment lexicon used [29]. The latter lexicon-based approach is presented as a well-performing method, which is robust across domains and texts. Approaches like the one proposed by Taboada et al. [29] enable a more thorough linguistic analysis to be incorporated in the process of analyzing sentiment in natural language text. Yet, rather than just looking at semantic orientation of individual words or groups of words, one may also consider analyzing the role these textual elements play in conveying the overall sentiment by applying discourse analysis.

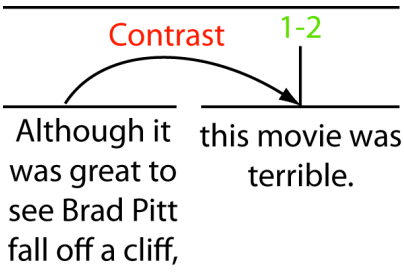

Figure 1: Example of an RST-structured sentence.

2.2 Rhetorical Structure Theory

According to UsingEnglish (`http://www.usingenglish.com`), discourse analysis can be defined as 'the area of linguistics that is concerned with how we build up meaning in the larger communicative rather than grammatical units; meaning in a text, paragraph, conversation, etc., rather than in a single sentence'. In our current endeavors, we apply discourse analysis in order to determine the parts of the text that are most relevant to the overall document sentiment. Intuitively, by splitting the text into parts with different levels of importance, sentiment analysis can be more reliable when weighting sentiment of parts of a text in accordance with their associated impact on a document's sentiment.

One of the leading discourse theories is RST [19], which can be used to split a text into spans which are rhetorically related to each other. There are two forms of relations: hypotactic and paratactic relations. In a hypotactic relation, one span is classified as nucleus, whereas the other spans are classified as satellite. RST claims nuclei to be more significant than satellites with respect to understanding and interpreting a text. In a paratactic relation, spans are equally significant, thus resulting in all spans to be classified as nuclei. RST identifies the smallest text spans that can hold rhetorical relations as Elementary Discourse Units (EDUs). Together, multiple EDUs can form a new text span, which again holds a rhetorical relation to another text span, thus yielding in a hierarchical structure of the text. RST also distinguishes several types of relations (e.g., elaboration, attribution, contrast, etc.). The authoritative paper on RST [19] defines 23 types of relations. In our framework, we differentiate among relation types by assigning them weights according to their importance, which we hypothesize to have a significant influence on the polarity of a document.

Figure 1 shows an example of an RST-structured sentence, where the text 'Although it was great to see Brad Pitt fall off a cliff, the movie was terrible.' is split into two segments. The first span is classified as a satellite and is related to the other span, the nucleus. The relation type is 'contrast', which indicates that the satellite segment of the sentence provides a contrast with the nucleus segment.

A human would typically interpret the specific sentence of Figure 1 as a negative review for the movie, as he would see the second span (the nucleus) as the most important span. However, in a classical (word-counting) sentiment analysis approach, all words would contribute equally to the total sentiment. Accordingly, a computer would count 'great' as very positive, and 'terrible' as very negative, which summed up makes a neutral review. When we exploit the information contained in the RST structure, the nucleus can be given a higher weight than the satellite, thus shifting focus to the

nucleus segment. In this case, a higher weight for the nucleus and a lower weight for the satellite would probably lead to a negative sentiment score for this sentence. By using the knowledge obtained from the RST structure, we can thus get a more reliable sentiment score.

2.3 Parsing Structure from Documents

In order to exploit a document's discourse structure and in an analysis of the sentiment conveyed by the text, one needs to first identify the discourse structure. Manual annotation of discourse structure is typically cumbersome, time-consuming, and not easily scalable, thus rendering automatic discourse parsing an attractive alternative. There are several discourse parsers publicly available which can parse an RST structure from a document. One of them is Sentence-level PArsing of DiscoursE (SPADE) [28], which creates RST trees for every sentence in a document. SPADE has been trained and tested on the train and test set of the RST Discourse Treebank (RST-DT) [7], where an F_1 score of 83.1% is reached on identifying the right rhetorical relations and their right arguments [28].

Another publicly available discourse parser is the HIgh-Level Discourse Analyzer [13] (HILDA), which applies statistical machine-learning techniques [8] to parse discourse structures from documents using the Rhetorical Structure Theory. In contrast with SPADE's sentence-level parsing, HILDA offers document-level parsing. HILDA has also been trained and tested on RST-DT, and achieved an F_1 score of 94.1% on identifying relations [13].

2.4 Rhetorical Structure Theory in Sentiment Analysis

Taboada et al. present an RST-based sentiment analysis approach with the Sentiment Orientation CALculator (SO-CAL) [30]. The assumption is that certain parts of a text are more relevant than others with respect to the overall sentiment expressed. Two methods are proposed to extract the relevant sentences: (1) extract nuclei within sentences using the SPADE discourse parser, and (2) extract sentences that are considered on-topic using a decision tree based on the ID3 algorithm [27]. Both approaches have proven to contribute to SO-CAL's performance in classifying sentiment.

However, SO-CAL merely differentiates between core elements of a text (nuclei) on the one hand, and any type of less important (satellite) element on the other hand. Yet, we hypothesize that the contribution of text elements to the overall sentiment of a document depends on their respective positions within the overall discourse structure and hence their relation to other elements. For instance, a contrasting text span may play a different role in conveying the overall sentiment than an elaboration on information in nuclei does. Therefore, we propose a more elaborate approach to utilizing RST in sentiment analysis by taking into account hypotactic relations between nuclei and satellites.

3. PATHOS

We present Pathos, a sentiment analysis framework that is able to interpret a text in terms of its discourse structure, and use this information to classify the text's polarity. First, we provide an overview of all components used in our framework. Then, we discuss the design of our sentiment classifier. Finally, we discuss the approaches to discourse parsing supported by this classifier.

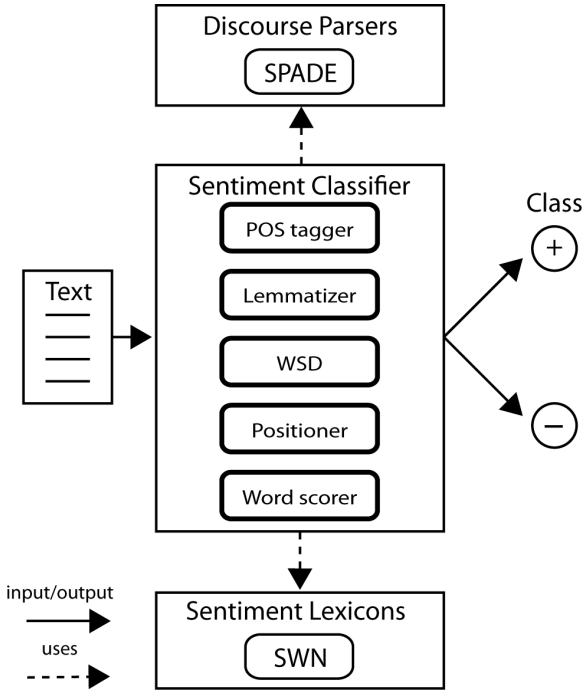

Figure 2: Overview of the Pathos framework.

3.1 Overview

The proposed framework consists of three parts, depicted in Figure 2. The central part is the sentiment classifier which classifies documents as either positive or negative. In order to do this, our framework first identifies the POS and the lemmas of all words and performs Word Sense Disambiguation (WSD). The positioner can employ a discourse parser (SPADE) to transform the input text into EDUs which are used to assign a weight to each individual word. The overall sentiment of a document is then computed as a weighted average of individual word scores, retrieved from a sentiment lexicon which differentiates on POS and word sense – SentiWordNet 3.0 [3, 9] (SWN), which has proven to be very useful for this purpose [11]. Such a lexicon-based sentiment scoring approach is in accordance with the work by Taboada et al. [30].

3.2 Sentiment Classifier

To investigate the merits of taking into account structural aspects of content in sentiment analysis, we propose a lexicon-based sentiment analysis approach, taking into account adjectives, adverbs, verbs, and nouns. Scoring a document for sentiment involves aggregating word-level sentiment scores, retrieved from a sentiment lexicon, after having initially determined each word's POS and lemma. A sentiment lexicon can contain entries for different senses for an arbitrary POS of an arbitrary word. In order to determine the appropriate sense of a word in a particular sentence, we use a similarity function proposed by Baazaoui Zghal et al. [2] and inspired by the Lesk algorithm [17]. We apply this algorithm because it is an unsupervised algorithm, able to compute adequate senses in a relatively small amount of time. Other unsupervised algorithms as SSI [21] and Lesk [17] require more computations which make the WSD a slow process.

Algorithm 1: **Word Sense Disambiguation.**

```
input : The word w to be disambiguated, its part-of-speech
        pos, and the sentence s that contains the word
output: The sense i of w with the highest semantic
        similarity to the words in the context
I = All words in s; // The context
senses = retrieveSenses(w, pos);
// All unique senses of w
mostSimilarS_i = 0;
// Most similar set of related synonyms
mostSimilarSynset = ∅;
// The synset with the best similarity to w
if |senses| <= 1 then
    return |senses|;
else
    foreach i in senses do
        containingSynsets =
        retrieveSynsetsContainingSense(i, pos);
        foreach synset in containingSynsets do
            S_i = ∅;
            foreach synonym in synset.Synonyms do
                S_i = {S_i, synonym};
            end
            foreach relation in synset.Relations do
                foreach synonym in relation.Synonyms do
                    S_i = {S_i, synonym};
                end
            end
            if |S_i ∩ I| > |mostSimilarS_i ∩ I| then
                mostSimilarS_i = S_i;
                mostSimilarSynset = synset;
            end
            else if |S_i ∩ I| = |mostSimilarS_i ∩ I| then
                if |S_i| > |mostSimilarS_i| then
                    mostSimilarS_i = S_i;
                    mostSimilarSynset = synset;
                end
            end
        end
        return i;
    end
end
```

In our applied WSD algorithm (described in Algorithm 1), the word sense with the highest semantic similarity to the word's context is selected. Here, the similarity $\mathrm{sim}(S_i, I)$ of a set S_i, denoting the semantic neighborhood of sense i of the word to be disambiguated, with the word's context I (i.e., the set denoting the sentence lexical neighborhood of the word to be disambiguated) can be defined as:

$$\mathrm{sim}(S_i, I) = |S_i \cap I|. \tag{1}$$

S_i contains the word to be disambiguated and all the synonyms, hyponyms, hypernyms, and gloss words of the WordNet synset. Furthermore, I contains all the words in the sentence without the word to be disambiguated. The set which has the highest similarity to I is then selected, and gives the most similar sense i. If there are more sets with the same similarity, the set S_i which has the maximum number of elements is chosen.

When having determined each word's POS, lemma, and its associated word sense, the score $\mathrm{eval}(d)$ of a document d can be computed as the sum of the scores of the individual sentences:

$$\mathrm{eval}(d) = \sum_{s_i \in d} \mathrm{score}(s_i), \tag{2}$$

Algorithm 2: **Scoring a document.**

```
input : A document d
output: A floating point number representing the sentiment
        score of document d
docScore = 0;
docScoreSentenceCount = 0;
foreach s in d.Sentences do
    sentenceScore = 0;
    foreach w in s.Words do
        pos = getPOS(w, s);
        lemma = getLemma(w, pos);
        sense = getWordSense(w, s, pos);
        score = getWordScore(lemma, sense, pos);
        weight = getWeight(w, s, document);
        sentenceScore = sentenceScore + (weight × score);
    end
    docScore = docScore + sentenceScore;
end
return docScore;
```

where $\mathrm{score}(s_i)$ is the score of the ith sentence s_i in d. The score of sentence s_i is computed by aggregating all sentiment scores $\mathrm{score}(w_j)$ of all words $w_j \in s_i$, multiplied with their respective weights $\mathrm{weight}(w_j)$:

$$\mathrm{score}(s_i) = \sum_{w_j \in s_i} \mathrm{score}(w_j) \times \mathrm{weight}(w_j), \tag{3}$$

where the weights are computed differently for our three positioners in accordance with the methods explained in Section 3.3. Here, word-level sentiment scores $\mathrm{score}(w_j)$ are assumed to be in the range $[-1, 1]$ (anywhere in between negative and positive, respectively).

Using (2), the classification $\mathrm{class}(d)$ of a document d can finally be determined as follows:

$$\mathrm{class}(d) = \begin{cases} 1 & \text{if } \mathrm{eval}(d) - \mathrm{offset} \geq 0, \\ -1 & \text{if } \mathrm{eval}(d) - \mathrm{offset} < 0, \end{cases} \tag{4}$$

where 1 denotes a positive document, and -1 denotes a negative document. The offset corrects a possible bias in the sentiment scores caused by people's tendency to write negative reviews with rather positive words, which can lead to a small negative sentiment score or sometimes even a positive score for a negative document, whereas a positive document usually gets a high positive sentiment score [30]. The offset can be calculated by taking the average sentiment scores of both positive and negative documents in the training set and subsequently computing the equidistant point of these scores.

Algorithm 2 is used to score a document. Each sentence is scored separately and its score is added to the overall document sentiment score. Sentence scores are essentially weighted averages of the sentiment scores of their individual words. The calculation for each word in a sentence (only non-stopwords) consists of five steps: (1) determining the POS, (2) retrieving the lemma of the word based on its POS-type, (3) determining which meaning of a word to use (using the WSD process described in Algorithm 1), (4) retrieving the score of the word from the sentiment lexicon, and (5) assigning a weight in accordance with the word's position in the document's discourse structure. Pathos supports several methods for parsing a document's discourse structure and assigning corresponding weights to individual words, as further detailed in Section 3.3.

Algorithm 3: Simple positioner.

```
input  : A document d
output: A weighted document d
wordCount = getNumberOfwords(d);
coefficient = 1/wordCount;
weight = 0;
foreach s in d.Sentences do
    foreach w in s.Words do
        w.setWeight(weight);
        weight = weight + coefficient;
    end
end
return d;
```

3.3 Discourse Parsing

Existing work suggests that sentiment analysis may benefit from a better understanding of discourse in texts [14, 26, 30]. Hence, we propose a discourse parsing module that first retrieves the discourse structure of a document and subsequently assigns discourse-based weights to sentiment-carrying words using a so-called positioner. Our framework supports three ways of accounting for discourse structure when determining a document's polarity.

A simple example of a discourse structure found in most sentiment-carrying documents (e.g., movie reviews) is the distinction of 'introduction', 'arguments', and 'conclusions'. Intuitively, one would think that an author typically puts his final and most important opinion towards the end of the text. To test this intuition, Pathos is able to assign weights to each word of a document, based on their respective position. The words are weighted uniformly in the range $[0, 1]$, where the first word is assigned a weight of 0, and the last word is assigned a weight of 1. This Simple positioner approach is described in Algorithm 3.

In order to obtain a more advanced discourse structure, Pathos uses SPADE [28] to extract sentence-level RST structures from texts. Subsequently, Pathos is able to assign a weight to each word based on its position in an RST structure. Table 1 shows all RST relation types handled by Pathos. These relations are a subset of the 23 standard relations defined in [19], encompassing only the relation types occurring in at least 10% of our considered set of reviews (see Section 4 for more details on our corpus).

Taboada et al. [30] hypothesize that adjectives found in nuclei of a document are more important for the overall sentiment, while adjectives found in satellites potentially interfere with the overall sentiment – the latter adjectives may be tangential or even irrelevant for a document's overall sentiment. Pathos can handle the notion of discourse structure thus introduced to the sentiment mining process by means of the SPADE positioner algorithm (Algorithm 4), which tags the words in the top-level nuclei of each sentence as nuclei, and the words in all other top-level elements as satellites. If a sentence only consists of a single text span, all words in that span are tagged as nuclei. Each word can be given a weight based on the RST element in which it resides. Following Taboada et al. [30], we consider two sets of weights for nuclei and satellites: (1) 1 and 0, and (2) 1.5 and 0.5, respectively. Yet, rather than only analyzing adjectives, we additionally handle adverbs, verbs, and nouns.

Algorithm 4: SPADE positioner.

```
input  : A document d
output: A weighted document d
foreach s in d.Sentences do
    foreach w in s.Words do
        rstElement = getRSTElement(w);
        weight = getWeight(rstElement);
        w.setWeight(weight);
    end
end
return d;
```

Building on the idea that some text spans can be more important for the overall sentiment of a document than other text spans, we propose a third approach which further explores satellites and their relation to the nuclei. We hypothesize that a hierarchy exists between the satellite relation types – some satellite relation types may contribute differently to the overall sentiment than others. By employing the SPADE Extended algorithm, shown in Algorithm 5, Pathos can give specific weights to the words in satellite elements based on their RST relation type. In addition to the positive weights considered by Taboada et al. [30] for their RST elements, we consider negative weights, as some text spans (e.g., contrasting text spans) may contribute negatively to the overall sentiment of a document. In order to be able to additionally intensify sentiment in certain text spans, we assume the weights to be in the range $[-2, 2]$. These weights can be optimized by means of a Genetic Algorithm (GA).

To this end, the sum of sentiment scores in the span of each RST relation type in a document (e.g., 'nucleus' in the case of a nucleus span, or 'attribution', 'elaboration', etc. in case of satellites) is first calculated. Subsequently, a set of potential solutions – chromosomes – can be generated, where each chromosome represents a set of weights for all considered RST relation types. These chromosomes are then subject to a process of simulated biological evolution according to the principle of survival of the fittest.

Table 1: Relation types handled by Pathos.

Relation	Description
Attribution	Clauses containing reporting verbs or cognitive predicates related to reported messages presented in nuclei.
Background	Information helping a reader to sufficiently comprehend matters presented in nuclei.
Cause	Information on the effects of causes presented in nuclei.
Condition	Hypothetical, future, or otherwise unrealized situations, the realization of which influences the realization of nucleus matters.
Contrast	Situations juxtaposed to situations in nuclei, where juxtaposed situations are considered as the same in many respects, yet differing in a few respects, and compared with respect to one or more differences.
Elaboration	Rhetorical elements containing additional detail about matters presented in nuclei.
Enablement	Rhetorical elements containing information increasing a reader's potential ability of performing actions presented in nuclei.
Explanation	Justifications or reasons for situations presented in nuclei.

Algorithm 5: SPADE Extended positioner.

```
input  : A document d
output: A weighted document d
foreach s in d.Sentences do
    foreach w in s.Words do
        rstElement = getRSTElement(w);
        if rstElement == 'Satellite' then
            relationType = getRelationType(w);
            weight = getWeight(relationType);
        else
            weight = getWeight(rstElement);
        end
        w.setWeight(weight);
    end
end
return d;
```

To optimize the weights, the GA applies tournament selection, one-point crossover, and two mutation functions, i.e., changing the sign for a random weight in the chromosome and switching weights in the chromosome.

The fitness of a chromosome is computed in terms of its performance in classifying document polarity, which is assessed as follows. For both the positive documents and the negative documents in our data set, we first compute precision, recall, and the F_1 measure. Precision is the proportion of the positively (negatively) classified documents which have an actual classification of positive (negative). Recall is the proportion of the actual positive (negative) documents which are also classified as such. The F_1 measure is the harmonic mean of precision and recall, i.e.,

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \quad (5)$$

The performance of a chromosome on the full corpus can then be assessed by means of the macro-level F_1 measure, which is the average of the F_1 scores of the positive and negative documents.

4. EVALUATION

In this section, we evaluate our different approaches to determine the polarity of texts. Evaluation is done by assessing the processing times, accuracy, precision, recall, and F_1 score of the different positioners for the sentiment analysis framework. We have evaluated our framework on a data set provided by Pang and Lee [24], which was introduced in [23]. The set is a collection of 1,000 positive and 1,000 negative movie reviews, which have been extracted from movie review websites. From the original set, we have extracted a subset of 500 positive and 500 negative reviews, because SPADE was not able to process all reviews due to problems with syntax in over 800 documents.

We have randomly split the dataset into a training and test set, consisting of 60% and 40% of the documents, respectively, with both sets encompassing a proportional number of occurrences of each considered relation type. Thus, our training set contains 300 positive reviews and 300 negative reviews, whereas the test set contains 200 positive reviews and 200 negative reviews. The training set is used to train the GA for the SPADE Extended positioner, as well as to compute an offset value for each individual positioner. The test set is used to measure and compare the performances of all approaches.

For running our experiments, we have built a Graphical User Interface (GUI), enabling us to select different options for an experiment as well as to select the directories containing positive and negative documents. This GUI also displays the results of an experiment, as demonstrated in Figure 3.

To evaluate Pathos' processing performance of a single document, we have created another GUI, which is depicted in Figure 4. The GUI offers the possibility to evaluate all positioners which have been implemented in Pathos. Additionally, this GUI provides insight into the sentiment analysis process. For example, in Figure 4, a document processed by the SPADE Extended positioner is shown. In our GUI, words are colored depending on their sentiment score (a positive word is colored green, and a negative word is colored red), and the underlying information about a word (lemma, score, weight, POS, RST relation) can be requested by selecting the word.

4.1 Experimental Setup

To test the performance of the considered positioners, we have evaluated them in our sentiment analysis framework, the core of which acts as a document processing pipeline (see Section 3.1). In this pipeline, we first determine the POS of words by using the OpenNLP [4] POS tagger, which has an accuracy of 98.7% [6]. We then determine the lemmas of words by means of a third party lemmatizer, based on WordNet using the Java WordNet Library (JWNL) API. Its estimated prediction accuracy is about 98%.

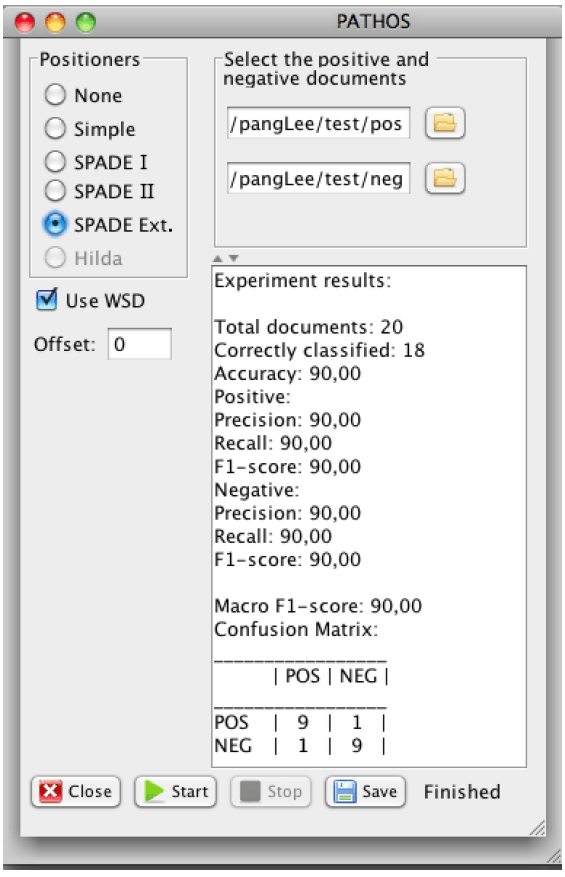

Figure 3: Experiment GUI.

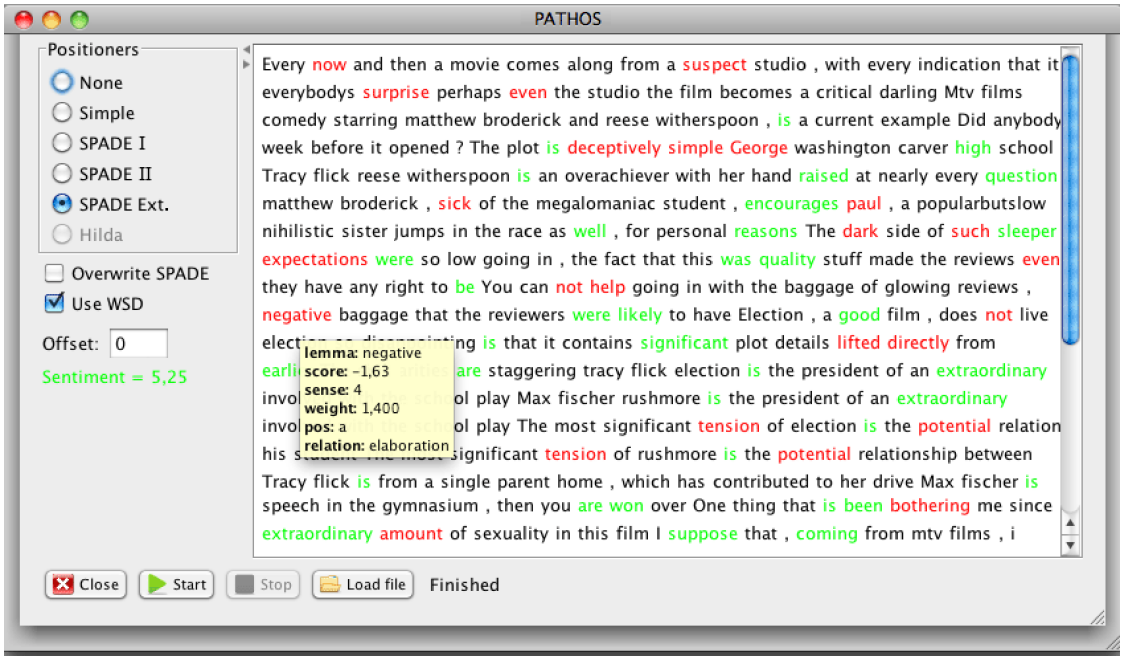

Figure 4: Document test GUI.

In order to subsequently determine which sentiment score of a word to select from our sentiment lexicon, we additionally employ a WSD algorithm (see Algorithm 1). A baseline for this would be always selecting the first sense from WordNet (which is the most common sense) for each word. We have evaluated our WSD algorithm on a test set of 100 sentences extracted from our corpus. The senses of the words in these sentences have been manually evaluated and annotated by three experts until they reached agreement. Our algorithm obtains an accuracy of 68% on this test set, whereas taking the first sense from WordNet yields an accuracy of 44%.

For each non-stopword, the POS, lemma, and sense thus determined are used for retrieving the associated sentiment score from the SentiWordNet 3.0 [3, 9] sentiment lexicon. SentiWordNet is based on WordNet [10], which is a (semantic) lexical resource, organized into sets of synonyms – synsets – which can be differentiated based on their POS type. Each synset expresses a distinct concept and is linked to other synsets through different kinds of relations (e.g., synonymy, antonymy, hyponymy, or meronymy). In SentiWordNet, each WordNet synset σ has been assigned scores in the range $[0, 1]$ on objectivity $Obj(\sigma)$, positivity $Pos(\sigma)$, and negativity $Neg(\sigma)$, the sum of which always equals 1. In our framework, we use SentiWordNet to compute the word sentiment score as a single number computed by subtracting $Neg(\sigma)$ from $Pos(\sigma)$, which results in a real number in the interval $[-1, 1]$, representing sentiment scores in the range from negative to positive, respectively.

The key of our framework is in weighting the retrieved word-level sentiment scores in accordance with the place of these words in the discourse structure of a document. As a baseline, we first assess the performance of our framework on our test set without any positioner (i.e., all words in the document are considered to equally contribute to the overall sentiment). An additional baseline is the Simple positioner, which weights the sentiment of words in accordance with their respective positions in the text. Then, we introduce two additional baselines derived from existing work [30] by evaluating the SPADE positioner with different weights for nucleus and satellite, i.e., 1 and 0 (SPADE I), and 1.5 and 0.5 (SPADE II), respectively. Finally, we assess the use of weights for distinct RST relation types, optimized by a GA, and compare the performance of this positioner (SPADE Extended) with the other considered approaches in an attempt to assess how information conveyed by rhetorical structure can be utilized in sentiment analysis of natural language texts.

4.2 Experimental Results

The optimal set of weights for the different RST elements in our training set is presented in Table 2. With an associated weight of 0.771, 'Nucleus' elements appear to contribute relatively much to the overall document sentiment in our training set. Yet, some satellite elements turn out to play an important role in conveying the overall document sentiment as well.

Table 2: Frequencies in training set of and optimized weights for relation types handled by Pathos.

Relation	Frequency	Weight
Nucleus	600	0.771
Attribution	461	0.451
Background	362	0.017
Cause	89	−0.271
Condition	176	0.304
Contrast	243	−0.660
Elaboration	531	1.400
Enablement	266	0.956
Explanation	134	−0.099

For instance, 'Elaboration' elements receive an even higher weight of 1.400, thus indicating that – in our training set – writers of reviews typically tend to express their sentiment in a more apparent fashion in elaborations on their core message. To a lesser extent, this also holds for rhetorical elements with a purpose of increasing a reader's potential ability of performing actions presented in the core of a review, as 'Enablement' elements are associated with a weight of 0.956. The information presented in 'Attribution', 'Condition', and especially 'Background' elements is clearly less relevant for the overall sentiment in reviews, as these elements receive weights of 0.451, 0.304, and 0.017, respectively.

Conversely, the sentiment conveyed by elements presenting matters contrasting with the information presented in the core of a review is typically inverted with a weight of -0.660. The sentiment conveyed by 'Cause' and 'Explanation' elements is slightly inverted as well, as these elements receive respective weights of -0.271 and -0.099. These minor inversions may however be artifacts of the relatively low frequencies of these elements in our training set, as they occur in only 89 and 134 out of 600 documents, respectively.

Tables 3 and 4 present the results for all considered approaches, respectively without and with offset, on our test set of 400 documents. Table 3 shows that the baseline approach has an overall accuracy of 0.585 and a macro-level F_1 of 0.569. The Simple positioner exhibits the largest improvement with respect to these measures, i.e., 3.9% and 4.8%, respectively. Two out of three considered SPADE positioners show an improvement in terms of these measures as well, albeit to a lesser extent. With respect to the baseline, SPADE I exhibits no change in overall accuracy and a 1.2% decrease in macro-level F_1. Conversely, applying discourse parsing by means of the SPADE II positioner yields a 2.1% increase in both accuracy and macro-level F_1. Yet, the best performing SPADE positioner is the SPADE Extended positioner using the weights presented in Table 2. This positioner exhibits an increase in overall accuracy and macro-level F_1 of 2.6% and 2.1%, respectively.

Yet, Table 3 clearly shows that all considered positioners exhibit a structural bias towards positive classifications. When controlling for each positioner's structural bias, the respective positioners exhibit a less biased performance, as detailed in Table 4. The baseline approach now yields an overall accuracy of 0.688 and a macro-level F_1 of 0.687. Compensation for the structural bias causes the Simple positioner, the SPADE I positioner, as well as the SPADE II positioner to perform below baseline. With respect to the baseline, overall accuracy and macro-level F_1 decrease with 5.8% and 5.5% for the Simple positioner, 1.9% and 1.8% for the SPADE I positioner, and 0.4% and 0.4% for the SPADE II positioner, respectively. Conversely, the SPADE Extended positioner exhibits a 4.5% increase in overall accuracy and a 4.7% increase in macro-level F_1 with respect to the baseline when controlling for each positioner's structural bias.

These results indicate that information contained in discourse structure of documents can improve the classification of sentiment conveyed by these documents. Additionally, the use of offsets improves the overall performance of each considered method, thus confirming the hypothesis of Taboada et al. [30] that a possible structural bias caused by negative documents carrying much positive sentiment can be corrected by applying an offset in the calculation of sentiment scores for documents. Moreover, we improve on existing work due to our more elaborate, optimized weighting scheme, which assigns distinct rhetorical elements different roles in conveying a document's overall sentiment. Our results indicate that both nuclei and satellites play an important role in conveying sentiment, whereas satellites have until now been deemed predominantly irrelevant.

However, these observed performance improvements come at a cost of increased processing time. On a standard 2,400 GHz Intel Core 2 Duo system with 2,048 MB physical memory, the average processing time for the baseline approach is approximately 2,786 milliseconds per document in our test set, with a standard deviation of approximately 1,176 milliseconds. Our Simple positioner has a similar performance, as it takes on average approximately 2,585 milliseconds to process a single document, with a standard deviation of about 1,121 milliseconds. Conversely, the SPADE positioners inspired by existing work [30], i.e., SPADE I and SPADE II, need on average about 45,862 milliseconds to process a document, with a standard deviation of 22,457 milliseconds. As the SPADE Extended positioner ignores less frequently occurring RST elements, it needs slightly less time than the SPADE I and SPADE II positioners for processing a document. On average, the SPADE Extended positioner processes a document in approximately 37,943 milliseconds, with a standard deviation of about 16,556 milliseconds.

The SPADE positioners spend a considerable amount of their processing time on a computationally intensive process of document parsing by means of the freely available SPADE discourse parser [28]. When considering only the time spent on activities other than using the SPADE discourse parser, the SPADE I and SPADE II positioners show an average document processing time of approximately 2,559 milliseconds, with a standard deviation of 1,062 milliseconds, whereas the SPADE Extended positioner needs on average about 2,571 milliseconds to process a document, with a standard deviation of 1,066 milliseconds. These results indicate that a major challenge lies in finding principal ways of efficiently and effectively extracting discourse structure from natural language texts.

5. CONCLUSIONS

While most research in sentiment analysis focuses on the main components of a sentiment classifier (e.g., word sentiment scoring, topic classification, negation, and intensifiers), little research has been done on analyzing the discourse structure of texts in order to identify text spans that are more important for the overall sentiment in a document. We compare three methods for dividing texts into important and less important parts. One method is based on the position of a word in a text. The other two methods exploit discourse structure in natural language text, either by distinguishing between (sentence-level) nuclei and satellites, or by identifying and exploiting (sentence-level) RST relation types. The objective of this paper is to give insights into how information can be harvested from structural aspects of content in order to improve the state-of-the art in sentiment analysis. Our results show that our method exploiting sentence-level RST relation types is the best performing approach, outperforming the baseline with a sentiment classification accuracy increased with 4.5% and a macro-level F_1 score increased with 4.7%, when controlling for each method's structural bias towards positive classifications.

Table 3: Experimental results for all positioners without offset.

		Positive			Negative			Overall	
Positioner	Offset	Precision	Recall	F_1	Precision	Recall	F_1	Accuracy	Macro F_1
Baseline	0.000	0.562	0.775	0.651	0.637	0.395	0.488	0.585	0.569
Simple	0.000	**0.581**	0.770	0.662	0.659	**0.445**	**0.531**	**0.608**	**0.597**
SPADE I (1, 0)	0.000	0.559	**0.810**	0.661	0.655	0.360	0.465	0.585	0.563
SPADE II (1.5, 0.5)	0.000	0.570	0.795	0.664	0.661	0.400	0.498	0.598	0.581
SPADE Extended	0.000	0.570	**0.810**	**0.669**	**0.672**	0.390	0.494	0.600	0.582

Table 4: Experimental results for all positioners with offset.

		Positive			Negative			Overall	
Positioner	Offset	Precision	Recall	F_1	Precision	Recall	F_1	Accuracy	Macro F_1
Baseline	2.016	0.687	0.690	0.688	0.688	0.685	0.687	0.688	0.687
Simple	1.290	0.647	0.660	0.654	0.653	0.640	0.647	0.650	0.650
SPADE I (1, 0)	1.560	0.668	**0.695**	0.681	0.682	0.655	0.668	0.675	0.675
SPADE II (1.5, 0.5)	2.796	0.683	0.690	0.687	0.687	0.680	0.683	0.685	0.685
SPADE Extended	2.562	**0.732**	**0.695**	**0.713**	**0.710**	**0.745**	**0.727**	**0.720**	**0.720**

A major bottleneck when accounting for discourse structure is the processing time required for identifying discourse structure in natural language text. Therefore, as future work, we aim to further explore other, scalable methods of identifying the discourse structure of texts. In addition, we would like to explore the applicability of our results in other types of sentiment mining approaches, e.g., methods making use of Support Vector Machines rather than sentiment lexicons. Furthermore, our currently used discourse parser SPADE only retrieves the RST structure on a sentence level, so it would be interesting to investigate the performance of a document level RST structure. Additionally, we aim to evaluate the performance of our methods on different corpora, which may contain other relation types or exhibit a different relation of discourse structure to the overall sentiment. Another interesting direction for future work would be to summarize a text and evaluate this summary using our sentiment classifier. Last, we would like to investigate how to best present a sentiment analysis system's results in order to suit a typical user's needs.

6. REFERENCES

[1] I. Arnold and E. Vrugt. Fundamental Uncertainty and Stock Market Volatility. *Applied Financial Economics*, 18(17):1425–1440, 2008.

[2] H. Baazaoui Zghal, M. Aufaure, and N. Ben Mustapha. A Model-Driven Approach of Ontological Components for On-line Semantic Web Information Retrieval. *Journal of Web Engineering*, 6(4):309–336, 2007.

[3] S. Baccianella, A. Esuli, and F. Sebastiani. SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. In *7th Conference on International Language Resources and Evaluation (LREC 2010)*, pages 2200–2204. European Language Resources Association, 2010.

[4] J. Baldridge and T. Morton. OpenNLP, 2004. Available online, `http://opennlp.sourceforge.net/`.

[5] D. Baron. Competing for the Public through the News Media. *Journal of Economics and Management Strategy*, 14(2):339–376, 2005.

[6] E. Buyko, J. Wermter, M. Poprat, and U. Hahn. Automatically Adapting an NLP Core Engine to the Biology Domain. In *Joint Linking Literature, Information and Knowledge for Biology and the 9th Bio-Ontologies SIG Meeting (ISMB 2006)*, pages 65–68. Oxford University Press, 2006.

[7] L. Carlson, D. Marcu, and M. Okurowski. Building a Discourse-Tagged Corpus in the Framework of Rhetorical Structure Theory. In *Current Directions in Discourse and Dialogue*, pages 85–112. Kluwer Academic Publishers, 2003.

[8] D. duVerle and H. Prendinger. A Novel Discourse Parser Based on Support Vector Machine Classification. In *47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL/AFNLP 2009)*, pages 665–673. Association for Computational Linguistics, 2009.

[9] A. Esuli and F. Sebastiani. SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining. In *5th Conference on Language Resources and Evaluation (LREC 2006)*, pages 417–422. European Language Resources Association, 2006.

[10] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[11] B. Heerschop, A. Hogenboom, and F. Frasincar. Sentiment Lexicon Creation from Lexical Resources. In *14th International Conference on Business Information Systems (BIS 2011)*, volume 87 of *Lecture Notes in Business Information Processing*, pages 185–196. Springer, 2011.

[12] B. Heerschop, P. van Iterson, A. Hogenboom, F. Frasincar, and U. Kaymak. Analyzing Sentiment in a Large Set of Web Data while Accounting for Negation. In *7th Atlantic Web Intelligence Conference (AWIC 2011)*, pages 195–205. Springer, 2011.

[13] H. Hernault, H. Prendinger, D. duVerle, and M. Ishizuka. HILDA: A Discourse Parser Using Support Vector Machine Classification. *Dialogue and Discourse*, 1(3):1–33, 2010.

[14] A. Hogenboom, F. Hogenboom, U. Kaymak, P. Wouters, and F. de Jong. Mining Economic Sentiment using Argumentation Structures. In *7th International Workshop on Web Information Systems Modeling (WISM 2010) at 29th International Conference on Conceptual Modeling (ER 2010)*, volume 6413 of *Lecture Notes in Computer Science*, pages 200–209. Springer, 2010.

[15] C. Holton. Identifying Disgruntled Employee Systems Fraud Risk Through Text Mining: A Simple Solution for a Multi-Billion Dollar Problem. *Decision Support Systems*, 46(4):853–846, 2009.

[16] B. Jansen, M. Zhang, K. Sobel, and A. Chowdury. Twitter Power: Tweets as Electronic Word of Mouth. *Journal of the American Society for Information Science and Technology*, 60(11):2169–2188, 2009.

[17] M. Lesk. Automatic Sense Disambiguation Using Machine Readable Dictionaries: How To Tell a Pine Cone from an Ice Cream Cone. In *5th Annual International Conference on Systems Documentation (SIGDOC 1986)*, pages 24–26. Association for Computing Machinery, 1986.

[18] S. Ludvigson. Consumer Confidence and Consumer Spending. *The Journal of Economic Perspectives*, 18(2):29–50, 2004.

[19] W. Mann and S. Thompson. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text*, 8(3):243–281, 1988.

[20] J. Martin and P. White. *The Language of Evaluation: Appraisal in English*. Palgrave Macmillan, 2005.

[21] R. Navigli and P. Velardi. Structural Semantic Interconnections: A Knowledge-Based Approach to Word Sense Disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):671–674, 2005.

[22] G. Paltoglou and M. Thelwall. A study of Information Retrieval weighting schemes for sentiment analysis. In *48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, pages 1386–1395. Association for Computational Linguistics, 2010.

[23] B. Pang and L. Lee. A Sentimental Education: Sentiment Analysis using Subjectivity Summarization based on Minimum Cuts. In *42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, pages 271–280. Association for Computational Linguistics, 2004.

[24] B. Pang and L. Lee. Polarity dataset v2.0, 2004. Available online, `http://www.cs.cornell.edu/People/pabo/movie-review-data/`.

[25] B. Pang and L. Lee. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2(1):1–135, 2008.

[26] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques. In *Empirical Methods in Natural Language Processing (EMNLP 2002)*. Association for Computational Linguistics, 2002.

[27] J. Quinlan. Induction of Decision Trees. *Machine Learning*, 5(1):71–100, 1986.

[28] R. Soricut and D. Marcu. Sentence Level Discourse Parsing using Syntactic and Lexical Information. In *Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL 2003)*, pages 149–156. Association for Computational Linguistics, 2003.

[29] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede. Lexicon-Based Methods for Sentiment Analysis. *Computational Linguistics*, 37(2):267–307, 2011.

[30] M. Taboada, K. Voll, and J. Brooke. Extracting Sentiment as a Function of Discourse Structure and Topicality. Technical Report 20, Simon Fraser University, 2008. Available online, `http://www.cs.sfu.ca/research/publications/techreports/#2008`.

[31] P. Turney. Thumbs Up or Thumbs Down?: Semantic Orientation Applied to Unsupervised Classification of Reviews. In *40th Annual Meeting on Association for Computational Linguistics (ACL 2002)*, pages 417–424. Association for Computational Linguistics, 2002.

[32] C. Whitelaw, N. Garg, and S. Argamon. Using Appraisal Groups for Sentiment Analysis. In *14th ACM International Conference on Information and Knowledge Management (CIKM 2005)*, pages 625–631. Association for Computing Machinery, 2005.