Sentiment Lexicon Creation from Lexical Resources

Bas Heerschop, Alexander Hogenboom, and Flavius Frasincar

Erasmus University Rotterdam
PO Box 1738, NL-3000 DR, Rotterdam, The Netherlands
basheerschop@gmail.com,{hogenboom,frasincar}@ese.eur.nl

Abstract. Today's business information systems face the challenge of analyzing sentiment in massive data sets for supporting, e.g., reputation management. Many approaches rely on lexical resources containing words and their associated sentiment. We perform a corpus-based evaluation of several automated methods for creating such lexicons, exploiting vast lexical resources. We consider propagating the sentiment of a seed set of words through semantic relations or through PageRank-based similarities. We also consider a machine learning approach using an ensemble of classifiers. The latter approach turns out to outperform the others. However, PageRank-based propagation appears to yield a more robust sentiment classifier.

Key words: sentiment analysis, sentiment lexicon creation, sentiment propagation, page rank, machine learning

1 Introduction

Sentiment analysis, also referred to as opinion mining, encompasses a broad area of natural language processing, computational linguistics, and text mining. In general, the aim is to determine the attitude of the author with respect to the subject of the text, which is typically quantified in a polarity. Recent developments on the Web – enabling users to produce an ever-growing amount of virtual utterances of opinions or sentiment through, e.g., messages on Twitter, blogs, or on-line reviews – advocate an array of possibilities for business information systems. Mining sentiment in the vast amount of data on the Web has many interesting applications, such as in the analysis of on-line customer reviews, reputation management, or marketing. Proper tools for sentiment mining can enable businesses to monitor the public sentiment with respect to particular products or brands, which can yield invaluable input for their marketing strategies.

In recent work, we assessed the state-of-the-art in sentiment analysis [1]. We showed that many approaches essentially rely on a lexicon containing words or phrases and their associated sentiment scores. Such lexicons often need to be created first. Automated methods include supervised learning on a set of manually rated documents and learning through related word expansion – expanding a small, manually created set of words by exploiting word relationships such as synonyms, antonyms, and hypernyms. Several lexicon creation methods have been proposed, yet their performance has typically been evaluated by means of comparing generated lexicons with manually created golden lexicons. However, we argue that assessing lexicon creation methods in terms of their performance in the actual sentiment analysis process would be very insightful as well, as sentiment lexicons are typically developed for this process and should hence be evaluated as such.

Therefore, we propose to perform a corpus-based evaluation of sentiment lexicon creation methods. In this paper, we compare the performance of two commonly used variants of a sentiment propagating word expansion algorithm and a commonly used machine learning approach based on classifiers. Our focus here is on algorithms exploiting vast, readily available lexical resources like WordNet [2].

The remainder of this paper is organized as follows. First, Sect. 2 expands on WordNet and how this lexical resource can be exploited for sentiment lexicon creation. We then provide the specifics of our sentiment analysis framework as well as the considered sentiment lexicon creation approaches in Sect. 3. Subsequently, the evaluation of these methods is described in Sect. 4. Finally, we conclude in Sect. 5.

2 Related Work

When automatically creating a sentiment lexicon, a good starting point may be an existing lexical resource, the contents of which can subsequently be associated with a sentiment score. A widely used on-line (semantic) lexical resource is WordNet, the design of which is inspired by psycholinguistic theories of human lexical memory. WordNet is designed to be used under program control and enables the distinction between different word forms and word meanings. WordNet is organized into sets of synonyms – synsets – which can be differentiated based on their Part-of-Speech (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).

WordNet contains four main POS types: verbs, nouns, adjectives, and adverbs. The semantic network of English verbs in WordNet is typically considerably more complex than the network of nouns, which suggests that verb meanings are more flexible in usage than noun meanings [3]. Nouns in WordNet are grouped in a hierarchical way based on distinguishing features (i.e., modification, hyponymy, meronymy, and predication). This hierarchy seldomly exceeds more than a dozen levels. Adjectives in WordNet are divided into two classes: descriptive (e.g., "big" or "interesting") and relational (e.g, "presidential" or "nuclear") and may have several types of relations. Adverbs have the least complex structure of all POS types. Adverbs not derived from an adjective only have occasional antonym relations and derived adverbs are semantically related to their base adjectives.

The semantic relations expressed in WordNet can be exploited to generate a sentiment lexicon. A typical approach is to start with a seed set of words and their associated sentiment and to subsequently traverse the WordNet relations, while propagating the sentiment [4, 5, 6]. Rather than by traversing WordNet relations, sentiment can also be propagated to synsets that are similar to positive or negative synsets. One way of accomplishing this is by using an algorithm inspired by Google's PageRank algorithm [7], which uses the link structure of the Web to calculate a quality ranking for each Web page. Esuli and Sebastiani argue that this algorithm can also be employed to create a ranking of how closely synsets relate to positive or negative synsets by using the semantic structure of WordNet [8]. Their application uses eXtended WordNet[1], a publicly available version of WordNet in which each word occurring in a gloss of a synset is mapped to the synset to which it belongs.

Another way of creating a sentiment lexicon based on WordNet synsets and their associated sentiment is to iterate over WordNet synsets and assign sentiment scores to these synsets by means of a classifier which analyzes the glosses associated with the individual synsets. An example of a sentiment lexicon thus generated is SentiWordNet [9], where eight classifiers (trained using a semi-supervised method on different seed sets of glosses annotated with their associated synsets' sentiment) have been used to analyze the gloss of each WordNet synset σ in order to assign scores quantifying how objective $Obj(\sigma)$, positive $Pos(\sigma)$, and negative $Neg(\sigma)$ each synset is. Each score is determined by the (normalized) proportion of the eight classifiers that have assigned the corresponding label to it and the sum of the three scores is constrained to equal 1 for each synset.

3 Framework

In order to assess the performance of different sentiment lexicon creation approaches, we propose to test the performance of a simple sentiment analysis framework on a corpus, while using lexicons created with our considered approaches. The sentiment classification is further detailed in Sect. 3.1. Our considered sentiment lexicon creation methods are discussed in Sects. 3.2, 3.3, and 3.4.

3.1 Sentiment Classification

We propose a simple lexicon-based sentiment classifier for investigating the performance of sentiment lexicons created by means of our considered methods. This classifier focuses on adjectives, adverbs, verbs, and nouns. The sentiment score of a document d is computed by aggregating the scores for each sentence s, which in turn are computed by aggregating sentiment scores for each non-stopword w in the sentences. The score $\text{eval}\,(d)$ of a document d is thus computed as

[1] `http://xwn.hlt.utdallas.edu`

Algorithm 1: Document scoring.

```
   input  : A document $d$
   output: The sentiment score of document $d$
1  $docScore = 0$;
2  $docScoreSentenceCount = 0$;
3  foreach $sentence$ in $d$ do
4      $sentenceScore = 0$;
5      foreach $word$ in $sentence$ do
6          $pos$ = getPOS($word, sentence$);
7          $lemma$ = getLemma($word, pos$);
8          $sense$ = getWordSense($word, sentence, pos$);
9          $score$ = getWordScore($lemma, sense, pos$);
10         $sentenceScore = sentenceScore + score$;
11     end
12     $docScore = docScore + sentenceScore$;
13 end
14 return $docScore$;
```

$$\text{eval}\,(d) = \sum_{s \in d} \sum_{w \in s} \text{score}\,(w)\,, \tag{1}$$

after which the classification class (d) of a document d can be determined as

$$\text{class}\,(d) = \begin{cases} 1 & \text{if eval}\,(d) \geq 0, \\ -1 & \text{if eval}\,(d) < 0. \end{cases} \tag{2}$$

In this process, documents are first split into sentences, after which the words in each sentence are tagged by a POS tagger. In order to subsequently assign sentiment scores to the individual words, we need to first retrieve the lemma and then disambiguate the word sense before we can extract the associated sentiment from our lexicon, as detailed in Algorithm 1.

For the word sense disambiguation process, we propose to use a Lesk algorithm [10], as it has a freely available implementation for WordNet [11], which has proven to yield satisfactory results (50–70% accuracy). The algorithm, described in Algorithm 2, selects the word sense that is semantically most similar to the words in the context (i.e., the other words in the sentence). This similarity is measured in terms of the overlap of an ambiguous word's gloss and glosses of its context.

3.2 Traversing WordNet Relations

The sentiment classification process described in Sect. 3.1 requires a lexicon in order to find the sentiment scores associated with words. A typical approach to create such a lexicon is to start with a manually created seed set of words and their associated sentiment [4, 5, 6]. Such a seed set may for example contain the positive words “beautiful”, “proud”, “security”, “good”, and “success” and

Algorithm 2: Word Sense Disambiguation.

```
input  : The to be disambiguated word $w$ and the sentence $s$ that contains the
         word
output: The sense $sense$ of $w$ with the highest semantic similarity to the words
         in the context
$targetSenses = \emptyset$; // Senses of the target word $w$
$targetGlosses = \emptyset$; // Glosses of the word senses for $w$
$senseScores = \emptyset$; // Scores of the word senses for $w$
$bestSense = \emptyset$; // Best sense for $w$
$bestScore = -1$; // Score for best sense for $w$
$k = 8$; // Considered context around $w$
// Retrieve the sequence of words starting $k/2$ words to the left of
// $w$ and ending $k/2$ words to the right of $w$, excluding $w$
$context$ = getContext($s$, $w$, $k$);
// Look up and add all senses of POS noun and verb for $w$
$targetSenses$ = getSenses($w$);
foreach $sense$ in $targetSenses$ do
    // Retrieve the gloss of the sense and the glosses connected to
    // it through hypernym, hyponym, meronym, and troponym relations
    $targetGlosses$ = {$targetGlosses$, getRelGlosses($sense$)};
end
foreach $word$ in $context$ do
    // Look up and add all senses of POS noun and verb for $word$
    $senses$ = getSenses($word$);
    foreach $sense$ in $senses$ do
        // Retrieve the gloss of the sense and the glosses connected
        // to it through hypernymy, hyponymy, meronymy, and troponymy
        $glosses$ = getRelGlosses($sense$);
        foreach $gloss$ in $glosses$ do
            foreach $targetGloss$ in $targetGlosses$ do
                // Each overlap which contains $N$ consecutive words
                // contributes $N^2$ to the gloss sense combination score
                $senseScores[targetGloss]$ += overlap($gloss$, $targetGloss$);
            end
        end
    end
end
foreach $sense$ in $targetSenses$ do
    if $senseScores$[getGloss($sense$)] $> bestScore$ then
        $bestScore = senseScores$[getGloss($sense$)];
        $bestSense = sense$;
    end
end
return $bestSense$;
```

Algorithm 3: Propagating WordNet from a seed set.

```
  input : The WordNet files, a list $seedWords$ of the words to propagate, their
          associated scores $\xi$, an integer $K$ denoting the maximum number of
          iterations, and a double $limit$ which defines the score given to words in
          the last iteration
  output: A $sentLexicon$ containing all propagated words with their computed
          sentiment scores
1 $sentLexicon = \emptyset$;
2 $synsets$ = retrieveSynsets(); // Retrieve all synsets in WordNet
3 $\delta = limit^{\frac{1}{K}}$;
4 foreach $word$ in $seedWords$ do
5     $\xi$ = score($word$);
6     propWord($synsets, sentLexicon, word, \xi, \delta$, 1, $K$); // See Algorithm 4
7 end
8 return $sentLexicon$;
```

the negative words “unfortunate”, “distressed”, “sad”, “hate”, and “bad”. The scores of the seed words equal 1 for positive words and -1 for negative words.

For each word in the seed set, WordNet relations (hyponym, hypernym, and antonym relations) can then be traversed and each encountered word w can be stored with a computed word score based on the score of the seed word, a diminishing factor, and an iteration step (i.e., the number of relations between the word and the seed word). The word score must be multiplied by -1 when traversing an antonym relation. We thus define the word scoring function as

$$\begin{aligned} \text{score}\left(w, \xi, \tau, \delta, k\right) = \xi \tau \delta^{k}, \quad & \tau \in \{-1, 1\}, \quad k \in \{1, \ldots, K\}, \\ & -1 \leq \xi \leq 1, \quad 0 < \delta < 1, \end{aligned} \tag{3}$$

with ξ the score of the seed word, τ indicating whether to inverse (-1) the score or not (1), δ the diminishing factor, and k the iteration step with a constraint of a maximum number of iterations denoted as K. The word score function weights each word encountered with a confidence measure that represents how likely it is that the given word has the designated positive or negative sentiment of the seed word. We define an iteration as traversing a relation between two synsets. On every iteration of the algorithm, words in the graph that are closely related to a seed word will get a higher score than those that are less related to a seed word. Thus, a word that is not a seed word, but is a neighbor to at least one seed word, will obtain a sentiment score similar to that of its adjacent seed words. At each iteration this will then propagate to other words. If a word is reached through a different path, then the word is assigned the score obtained from the shortest path between the considered paths between a word and any of the seeds (i.e., the score of the lowest iteration step). This process, yielding a lexicon containing word, word sense, POS tag, and a computed sentiment score, is detailed in Algorithm 3. This algorithm in turn utilizes a recursive word propagation function detailed in Algorithm 4.

Algorithm 4: Propagating a single word in WordNet (propWord).

```
input: A set containing the parsed WordNet $synsets$, a $sentLxicon$ for storing
       the propagated words with their computed scores, a $word$ to propagate,
       the score $\xi$ of the $word$, a diminishing factor $\delta$, an integer $k$ denoting the
       current iteration step, and an integer $K$ denoting the maximum number
       of iterations
if $k \leq K$ then
    if $word.ReachedInIteration > k$ then
        // If this word has not been reached through another, shorter
        // path (default path length equals $\infty$), proceed propagation
        $synsetsWithWord$ = getSynsets($synsets$, $word$);
        foreach $synset$ in $synsetsWithWord$ do
            $pos$ = getPOS($synset$);
            foreach $syn$ in $synset.Synonyms$ do
                addToLexicon($syn$,$pos$,$\xi$);
            end
            foreach $relation$ in $synset.Relations$ do
                $\tau = 1$;
                if $relation.typeOf(antonym)$ then $\tau = -1$;
                foreach $syn$ in $relation.Synonyms$ do
                    propWord($synsets$, $sentLexicon$, $syn$, $\xi\tau\delta^k$, $\delta$, $k+1$, $K$);
                end
            end
        end
        $word.ReachedInIteration = k$;
    end
end
```

3.3 PageRank-Based Propagation

Rather than by traversing semantic relations in WordNet, sentiment can also be propagated to synsets that are similar to predefined positive and negative synsets. Google's PageRank algorithm [7] can be used to exploit the semantic structure of WordNet to create a ranking of how closely synsets relate to positive or negative synsets [8].

The input to PageRank is the parsed set of synsets, and its output is a vector $\mathbf{a} = (a_1, \ldots, a_N)$ with sentiment scores for all N (117,659) synsets in WordNet, where a_i represents the score for synset σ_i. PageRank iteratively computes vector $\mathbf{a}$ using

$$a_i^k = \alpha \sum_{j \in B(i)} \frac{a_j^{k-1}}{|F(j)|} + (1 - \alpha) e_i, \tag{4}$$

where a_i^k denotes the sentiment score of the i-th entry of $\mathbf{a}$ at the k-th iteration, B represents backward links, F represents forward links, e_i is an element of the constants vector $\mathbf{e} = (e_1, \ldots, e_N)$ with a constraint such that $\sum_{i=1}^{|N|} e_i = 1$, and α is a control parameter with a range of $[0, 1]$.

When creating a sentiment sentiment lexicon with PageRank, we first need to retrieve all N synsets (glosses) from eXtended WordNet and store their forward links. We subsequently loop over each synset and set its forward links as backward links of the synsets to which the synset points. After parsing eXtended WordNet, $\mathbf{e}$ is initialized, such that all elements other than the seed synsets are assigned a value of 0, while seed synsets are assigned proportional values such that the sum of elements in $\mathbf{e}$ equals 1. Alternatively, elements in $\mathbf{e}$ can be assigned values based on their scores in SentiWordNet, i.e., by dividing synsets' positivity (negativity) scores greater than 0 by the sum of positivity (negativity) scores and by assigning other synsets a score of 0. Sentiment scores in $\mathbf{a}$ are initialized at $\frac{1}{N}$. The PageRank algorithm iteratively updates the sentiment scores $\mathbf{a}$ and stops when the cosine of the angle between $\mathbf{a}^k$ and $\mathbf{a}^{k-1}$ exceeds χ. Following Esuli and Sebastiani [8], we use $\chi = 1 - 10^{-9}$ and $\alpha = 0.85$.

To create both a ranking for positivity and negativity, the PageRank algorithm must be run twice; one time where the elements of $\mathbf{e}$ are set for a positive seed of synsets and second time for negative ones. When the algorithm – described in Algorithm 5 – is completed, each extracted synset contains both a positive and a negative score. The sentiment associated with a synset is the sum of the positive and negative scores, which results in a real number in the interval $[-1, 1]$.

3.4 SentiWordNet

Besides creating sentiment lexicons by exploiting the WordNet relations or similarities, one could also create sentiment lexicons by iterating over WordNet synsets and their associated glosses and assign sentiment scores to these synsets by means of a classifier. SentiWordNet [9] has been created in such a way and would thus be a convenient resource to use in order to assess the performance of such a sentiment lexicon creation method.

In SentiWordNet, each WordNet synset σ has been assigned scores on objectivity $Obj(\sigma)$, positivity $Pos(\sigma)$, and negativity $Neg(\sigma)$. The sum of these scores always equals 1 for each WordNet synset. A score $Obj(\sigma) > 0$ means a less subjective word and thus weaker sentiment scores in terms of $Pos(\sigma)$ and $Neg(\sigma)$.

The objectivity, positivity, and negativity scores for all 117,659 WordNet synsets have been computed by an ensemble of eight ternary classifiers. Each classifier has classified a synset as either objective, positive, or negative, based on a vectorial representation of the associated gloss. The overall scores for a synset have then been determined by the (normalized) proportion of classifiers that have assigned the corresponding labels to the synset. The scores thus obtained have been evaluated on a set of 1,105 WordNet synsets which have been scored in a similar fashion by five human annotators.

Algorithm 5: PageRank-based propagation in WordNet.

```
input : The eXtendedWordNet files, a list $posSeedSynsets$ and $negSeedSets$
        of the positive and negative seed synsets and their sentiment scores
        (respectively), a double $\chi$ denoting the termination condition, and a
        double $\alpha$ which defines the control parameter
output: A $sentLexicon$ containing all ranked words with their computed scores
1  $sentLexicon = \emptyset$;
2  $synsets$ = retrieveSynsets(); // Retrieve all eXtendedWordNet synsets
3  foreach $synset$ in $synsets$ do setBackwardLinks($synset$);
4  foreach $seedSet$ in $\{posSeedSynsets, negSeedSynsets\}$ do
5      $\mathbf{e}$ = initializeE($seedSet$);
6      $\mathbf{a}^k$ = initializeA();
7      $\mathbf{a}^{k-1}$ = initializeA();
8      $\theta = 0$;
9      while $\theta < \chi$ do
10         foreach $a_i^k$ in $\boldsymbol{a}^k$ do $a_i^k$ = calculateAi($\boldsymbol{a}^{k-1}$);
11         $\theta$ = calculateCosAngle($\boldsymbol{a}^k$, $\boldsymbol{a}^{k-1}$);
12         $\mathbf{a}^{k-1} = \mathbf{a}^k$;
13     end
14     $synsets$ = assignScores($synsets$);
15 end
16 $sentLexicon$ = buildLexicon($synsets$);
```

The classifiers used by SentiWordNet differ from one another in their training data as well as in their implemented machine learning approaches. Training sets have been generated from a seed set of positive and negative synsets, which have been expanded by traversing WordNet relations such as see-also and antonymy. The considered number of expansion steps varies amongst the classifiers between 0, 2, 4, and 6. Neutral synsets in the training data have been determined as synsets which are neither positive nor negative in both the expanded seed sets and the General Inquirer lexicon [12]. The considered machine learning approaches are Support Vector Machines (SVMs) and Rocchio classifiers.

The sentiment scores generated by an ensemble of these classifiers can, combined with their associated synsets σ, easily be utilized as a sentiment lexicon. However, in our approach, we ignore the objectivity scores, as they implicitly influence the positive and negative scores. Instead, we define our own sentiment score for σ as a single real number computed by subtracting $Neg(\sigma)$ from $Pos(\sigma)$, which results in a real number in the interval $[-1, 1]$.

4 Evaluation

We have implemented the framework presented in Sect. 3 in order to be able to assess the performance of our considered sentiment lexicon approaches on a corpus. The implementation was done in C#.Net in combination with a Microsoft SQL Server database. For lemmatization and word sense disambiguation, we used functionalities provided by the open-source C# WordNet.Net WordNet API [2]. Our POS tagger – with an accuracy of 98.7% [13] – is based on SharpNLP[3] and is provided to us by Teezir[4].

The performance of our considered sentiment lexicon approaches was evaluated on a collection of 1,000 positive and 1,000 negative English movie reviews[5], which have been extracted from movie review web sites by Pang and Lee [14]. The review classifications have been derived from the accompanying numerical review scores. On this corpus, we have evaluated the performance of our simple sentiment analysis framework when using sentiment lexicons created by utilizing our discussed algorithm for traversing WordNet relations (WordNetRel), two PageRank-based propagation methods, and SentiWordNet. The considered PageRank-based methods differ in their values for **e**; in our first variant, we distribute the weights equally amongst the synsets that are part of our seed sets (PageRankSeed), whereas our second variant is bootstrapped based on the SentiWordNet scores of all synsets (PageRankSWN).

In our evaluation, several performance measures have been taken into account. For both the positive documents and the negative documents, we report precision, recall, and the F_1 measure. Precision is the percentage of the positively (negatively) classified documents which have an actual classification of positive (negative). Recall is the percentage of the actual positive (negative) documents which is also classified as such. The F_1 measure is a weighted average of precision and recall. We also report some statistics on our full corpus. We report the macro-level F_1 measure, which is the average of the F_1 scores of the two classifications, and the accuracy, which is the total percentage of correctly classified documents. Our results are reported in Table 1.

Creating a sentiment lexicon when propagating sentiment by exploiting WordNet relations (WordNetRel) yields an overall F_1 measure of 41.3%. This approach also classifies relatively more documents as positive than as negative and provides a correct classification in over 50% of the time. Conversely, both PageRank-based algorithms appear to misclassify more than half of the documents in our test corpus, albeit in different ways. PageRankSeed exhibits a high recall on positive documents, whereas PageRankSWN's recall on positive documents is relatively low. Conversely, PageRankSeed has a low recall on negative documents, whereas PageRankSWN exhibits a high recall on negative documents. This renders the performance of PageRankSWN more stable over all documents. Like most other considered approaches, sentiment lexicon creation based on machine learning techniques (SentiWordNet) turns out to exhibit a slightly biased performance on our corpus; relatively more documents appear to be classified as positive than as negative. Yet, the SentiWordNet approach yields a macro F_1 measure of 56.8% and moreover correctly classifies a similar

[2] `http://opensource.ebswift.com/WordNet.Net/`
[3] `http://sharpnlp.codeplex.com/`
[4] `http://www.teezir.com/`
[5] `http://www.cs.cornell.edu/People/pabo/movie-review-data/`

Table 1. Experimental Results.

	Positive			Negative			Overall	
Method	Precision	Recall	F_1	Precision	Recall	F_1	Accuracy	Macro F_1
WordNetRel	51.0%	**94.3%**	66.2%	62.3%	9.4%	16.3%	51.9%	41.3%
PageRankSeed	49.8%	86.8%	63.3%	48.6%	12.5%	19.9%	49.7%	41.6%
PageRankSWN	49.6%	43.0%	46.1%	49.7%	**56.3%**	**52.8%**	49.7%	49.4%
SentiWordNet	**56.3%**	84.3%	**67.5%**	**68.8%**	34.6%	46.0%	**59.5%**	**56.8%**

percentage of all documents in our corpus. The observation of many of the approaches typically being biased towards positive documents may be explained by people tending to avoid negative words when expressing negative opinions, thus rendering purely lexicon-based sentiment classification more difficult [15, 16, 17].

Our results show that in terms of accuracy, the SentiWordNet approach outperforms all other considered approaches. Conversely, the PageRank-based sentiment propagation method bootstrapped using SentiWordNet scores appears to be the most robust approach in that the difference between its F_1 measures for positive and negative documents is smaller than is the case for the other considered approaches.

5 Conclusions and Future Work

In order for today's businesses to, e.g., keep a close eye on how their brands or products are perceived by the market, recent developments in the field of sentiment analysis may prove to be crucial for business information systems facing the challenge of extracting relevant information from the massive amount of data available through the Web. Many existing sentiment analysis approaches rely on lexical resources containing words and their associated sentiment. Creating such resources may be a cumbersome task, yet several methods for automated sentiment lexicon creation have already been proposed.

In this paper, we have performed a corpus-based evaluation of a number of distinct automated sentiment lexicon creation methods exploiting vast, readily available lexical resources. We have considered an algorithm exploiting semantic relations in a lexical resource in order to propagate the sentiment of a seed set of words, as well as a PageRank-based algorithm propagating the sentiment of a seed set of words to related words. We have also considered a machine learning approach based on Support Vector Machines. The latter approach turns out to outperform the others in terms of accuracy and macro F_1 measure. However, creating a sentiment lexicon with a PageRank-based propagation algorithm appears to result in the most robust sentiment classifier.

In future work, we would like to consider in our comparisons different languages (e.g., Dutch, Romanian, etcetera). Other possible directions for future work include the development and analysis of novel sentiment lexicon creation methods, focusing not only on existing lexical resources, but on, e.g., texts annotated for sentiment as well.

References

1. Heerschop, B., van Iterson, P., Hogenboom, A., Frasincar, F., Kaymak, U.: Analyzing Sentiment in a Large Set of Web Data while Accounting for Negation. In: 7th Atlantic Web Intelligence Conference (AWIC 2011), Springer (2011) 195–205
2. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press (1998)
3. Fellbaum, C.: English Verbs as a Semantic Net. International Journal of Lexicography **3**(1) (1993) 259–280
4. Kim, S., Hovy, E.: Determining the Sentiment of Opinions. In: 20th International Conference on Computational Linguistics (COLING 2004), ACL (2004) 1367
5. Hu, M., Liu, B.: Mining and Summarizing Customer Reviews. In: 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004), ACM (2004) 168–177
6. Lerman, K., Blair-Goldensohn, S., McDonald, R.: Sentiment summarization: Evaluating and learning user preferences. In: 12th Conference of the European Chapter of the ACL (EACL 2009), ACL (2009) 514–522
7. Brin, S., Page, L.: The Anatomy of a Large-Scale Hypertextual Web Search Engine. In: 7th International World-Wide Web Conference (WWW 1998), Elsevier (1998) 107–117
8. Esuli, A., Sebastiani, F.: PageRanking WordNet Synsets: An Application to Opinion Mining. In: 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007), ACL (2007) 424–431
9. Esuli, A., Sebastiani, F.: SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining. In: 5th Conference on Language Resources and Evaluation (LREC 2006), European Language Resources Association (ELRA) (2006) 417–422
10. Lesk, M.: 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), ACM (1986) 24–26
11. Dao, T., Simpson, T.: Measuring Similarity between Sentences. Technical report, WordNet.Net (2005) Available online, `http://wordnetdotnet.googlecode.com/svn/trunk/Projects/Thanh/Paper/WordNetDotNet_Semantic_Similarity.pdf`.
12. Stone, P., Dunphy, D., Smith, M., Ogilvie, D.: The General Inquirer: A Computer Approach to Content Analysis. MIT Press (1966)
13. Buyko, E., Wermter, J., Poprat, M., Hahn, U.: Automatically adapting an NLP Core Engine to the Biology Domain. In: 9th Bio-Ontologies Meeting and the Joint Linking Literature Information and Knowledge for Biology (ISMB 2006), Oxford University Press (2006) 65–68
14. Pang, B., Lee, L.: A Sentimental Education: Sentiment Analysis using Subjectivity Summarization based on Minimum Cuts. In: 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004), ACL (2004) 271–280
15. Dave, K., Lawrence, S., Pennock, D.: Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. In: 12th International World Wide Web Conference (WWW 2003), ACM (2003) 519–528
16. Taboada, M., Voll, K.: Extracting Sentiment as a Function of Discourse Structure and Topicality. Technical Report 20, Simon Fraser University (2008)
17. Turney, P.: Thumbs up or Thumbs down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In: 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002), ACL (2002) 417–424