MultiSpot: Spotting sentiments with semantic aware multilevel cascaded analysis

Despoina Chatzakou, Nikolaos Passalis, and Athena Vakali

Informatics Department, Aristotle University of Thessaloniki, Greece
{deppych, npassalis, avakali} @csd.auth.gr

Abstract. Given a textual resource (e.g. post, review, comment), how can we spot the expressed sentiment? What will be the core information to be used for accurately capturing sentiment given a number of textual resources? Here, we introduce an approach for extracting and aggregating information from different text-levels, namely words and sentences, in an effort to improve the capturing of documents' sentiments in relation to the state of the art approaches. Our main contributions are: (a) the proposal of two semantic aware approaches for enhancing the *cascaded phase* of a sentiment analysis process; and (b) MultiSpot, a multilevel sentiment analysis approach which combines word and sentence level features. We present experiments on two real-world datasets containing movie reviews.

Keywords: Sentiment detection, Multilevel features

1 Introduction

Given a document, at which level can sentiment be captured? Typically, document-based sentiment analysis processes operate at a particular level, i.e. at the word or sentence level, for extracting a document's sentiment. Sentiment identification at the word level is a fine-grained approach, which emphasizes on sentimentally intense words, but its lack of awareness for words' dependencies limits its capability to accurately capture sentiments. The more coarse-grained sentence level sentiment identification permits the capturing of a document's semantic-related information since it involves ordering and words' similarity, but its higher text level processing prohibits the capturing of sentimental words' intensities. This work is motivated by the fact that sentiments' extraction out of separated sets of words or flat lined sentences leads to information loss. Here, the overall document's sentiment identification is considered as a multilevel process which aggregates both words and sentences characteristics (called features for the rest of the paper).

In machine learning, the most popular approach for sentiment analysis, the selection of appropriate features for representing a document is crucial. In sentiment identification at the word level different types of features have been introduced, which are either sentiment-based (e.g. words which express a specific sentiment), syntactic-based (e.g. part-of-speech and n-grams), or semantic-based (e.g. semantic word vector spaces which capture the meaning of each word). Having specified the features tailored for the problem at hand, a document classification methodology is applied. When dealing with sentence-level features,the cascaded sentiment analysis is a common approach which proposes the use of two different level classifiers for extracting the overall sentiment of a document. It initially utilizes a classifier for annotating each sentence based on the expressed sentiment (sentence-level classifier) and then an additional classifier for aggregating the output of the first classifier. Up to now, most of the proposed cascaded methods ignore semantically rich features (high-level features), as for instance, the semantic closeness of two sentences in a document. As semantics deal with the meaning of a document these high-level features which are relevant to words, phrases and sentences, are promising for a more accurate capturing of a document's sentiment [4].

In this paper, the aggregation of both word and sentence level features is proposed for more accurate documents' sentiments capturing. The proposed multilevel cascaded sentiment analysis method, MultiSpot, aims at an improved classification process due to its different text level features aggregation (i.e. word and sentence level) which are exploited in both the training and testing classification phases. The MultiSpot method introduces an aggregation phase which considers the *Sentiment Center* (Sentcenter), the *Semantic Center* (Semcenter), where for each sentence a sentiment and semantic vector is generated, respectively, and the so called *Centerbook* (has been inspired by the computer vision field, where CodeBook learning classification approaches are characterized by improved accuracies [1]), which builds on vectors that merge sentimentally and semantically similar sentences. Such center-based approaches are chosen since the overall sentiment expressed in a document is highly impacted by all of the available sentiment and semantic information.

The sentiment and semantic vectors are produced based on the RNTN (Recursive Neural Tensor Network) [16] model which is popular for its ability in capturing more effectively the meaning of a sentence. Based on the RNTN model each sentence is defined using the representation of its words' vectors (builds upon previous word vector representations [14],[15]) and a parse tree.

The contribution of this work is summarized as follows:

- **MultiSpot:** a two-level document-based classification approach which combines features extracted from different text-levels. Our motivation originates from the idea of exploiting both word and sentence level features to reach better accuracy in a cascaded sentiment analysis manner.
- **Center-based aggregation:** an accumulative process with sentiment and semantic relevant features' extraction, namely the *Sentcenter*, the *Semcenter* and the *Centerbook* for modelling the overall document's context. To best of the authors' knowledge, no earlier work has utilized such a *Centerbook* learning technique for texts' sentiment analysis.

2 Related work

Document-level sentiment analysis has been realized by a large number of methods which proceed with either word or sentence level information extraction.

Word-level feature extraction. The *Bag-of-Word (BoW)* model, where a document is represented as a set of its included words, has been heavily utilized. Several variations have been proposed, including the use of bigrams (e.g. [19]), the *term - frequency* (tf) and the *term - frequency inverse document - frequency* (tf-idf) (e.g. [9], [10]) weighting schemes. Despite their simplicity, such approaches can reach remarkable accuracy on large documents [19], but as their models mainly focus just on the words and not on the words' positioning and ordering, semantic features are missing (e.g. cannot effectively capture the negation). To maintain the semantic similarities among words, a *semantic word space* is typically used, where each word of a document is represented as a vector ([7], [14], [17]). The word vectors, for words with similar semantic and/or sentiment content, tend to be close (e.g. based on cosine similarity measure). However, these approaches fall behind as they fail to accurately capture the ordering of the words. Most recent efforts (such as in [6], [16]) have involved words' ordering and negation capturing leading to accuracy improvements in classification.

Sentence-level feature extraction. Cascaded sentiment analysis is a commonly utilized approach which exploits sentence-level information. The recognition of a document's overall sentiment involves an aggregation method, in which the mean sentiment score for a document is calculated by averaging the sentiment score of its sentences ([18], [21]). More advanced approaches assign different weights to sentences that may better convey information about a document's sentiment based on different aspects, such as their position, ordering, length and subjectivity. Finally, ([13]) involves mining initially sequential rules from the sentences' ordering, and then keeping only the top k rules as features.

MultiSpot accumulates both word and sentence level features, while it also manages to integrate sentiment with semantic information into a tailored aggregation phase, which contributes to further enhancing the overall accuracy.

3 Multilevel sentiment detection

3.1 Problem Definition

Most earlier approaches suffer from the absence of the semantic context and their deficiency in capturing sentiments' evolution across a document's length. Such information loss, is due to either their unawareness of words' dependencies (ordering, positioning) or their absence of knowledge on the sentiment's formation from the words to the sentences levels. Our intuition is that by exploiting features from both levels the accuracy of a document-based sentiment analysis approach will be improved.

Next, we initially define the problem of capturing sentiments from document-level textual resources and then outline the MultiSpot's proposed approach.

Problem 1 ***Document-based sentiment analysis***
Given*: a training dataset* $D = \{d_1, d_2, ..., d_n\}$*, and a sentiment label for each document,* $t_i \in \{pos, neg\}$*;*
Learn*: a model* G *to predict the sentiment label for any new document* $d_{test} \notin D$*.*

3.2 Proposed Approach

In this section we provide the pipeline of the MultiSpot and describe its processes and characteristics.

MultiSpot pipeline. MultiSpot comprises of three steps: (1) *Word-level feature extraction*, which extracts word-level features from the whole document, (2) *Sentence-level feature extraction*, where having initially split the document in sentences, we derive the corresponding sentence-level features under a cascaded phase (MultiSpot's cascaded phase), (3) *Features combination*, where features from both levels are used for training a document-based sentiment classifier.

```
Data: A set of documents $D = \{d_1, d_2, ..., d_n\}$
Result: The expressed sentiments for each document in $D$.
for document $d_i$ in $D$ do
    Extract word-level features;
    Split a document $d$ into a set of $S$ sentences;
    for sentence $s_j$ in $S$ do
        Extract sentence-level features;
    Aggregate word and sentence level features;
    Extract $docsent(d) \in \{pos, neg\}$;
return $docsent(d)\ \forall\ d_i \in D$
```

Algorithm 1: Pseudocode for MultiSpot

In our case we examine two models (as features) that use word-level information, namely the unigram bag-of-words (BoW)[1] and the NB (Naive Bayes) bigrams model[2] (as being presented in [19]). **Word-level feature extraction** considers two separate phases: (i) the word to vector mapping, and (ii) the document-level vectors aggregation via a weighting scheme. In the first phase, we model a word w as a vector $v \in \mathbb{R}^k$, where each element of this vector is related to a distinct word. All the elements of v are zero except for the one that corresponds to the word w. For example, the vector $x = [0, 1, 0]$ is related to the word "*of*" in the dictionary $\{bag, of, words\}$. Then, all the vectors of the words in a document are combined into one that describes the whole document. For instance, the tf weighting scheme adds these vectors together in order to get the term frequency vector of the document. Using the previously mentioned dictionary, the term frequency vector of the phrase "bag bag words" is (2,0,1). Consequently, the word-level feature extractor f maps each word w of a document d to a vector $f(w) \in \mathbb{R}^l$ (l is the dimensionality of the output features).

Sentence level information is related to higher-level sentiment and semantic information of a sentence. This information cannot be extracted without the aid of classifiers and/or dimensionality reduction methods (e.g. deep learning). For example, a **sentence-level feature extractor** might extract features related to the sentiment and/or the objectivity of each sentence. So, the sentence level feature extractor g maps each sentence s of a document to a vector $g(s) \in \mathbb{R}^k$ (k similar to l) using one or more sentence-level classifiers and/or dimensionality reduction methods to extract such high-level information from each sentence.

We can now define the **cascaded sentiment analysis** as a process where given a training dataset D, a sentence-level feature extractor, g, and a sentiment label for each document $t_i \in \{pos, neg\}$ learn a model, G, to predict the sentiment label for any new document d_{test}. The model G uses as input the multiset of sentence-level features of each document d, $\{g(s) | \forall s \in S_d\}$. In our case, we train a classifier to accomplish this task using the *sentiment* or the *sentiment/semantic center* of each document in D as features.

MultiSpot: Multilevel cascaded sentiment analysis exploits both word and sentence level information from a document. So, given a training dataset D, a word-level feature extractor $f(w)$, a sentence-level feature extractor $g(s)$, and a sentiment label for each document $t_i \in \{pos, neg\}$, learn a model G to predict the sentiment label for any new document $d_{test} \notin D$. If W_d is the multiset of

[1] BoW model: represents a document as a set of its words.
[2] NB bigrams: Naive Bayes log-count ratios of bigram features.

all words in a document d and S_d is the multiset of all sentences in d, then the model G uses as input the multiset of word-level features, $\{f(w)|\forall w \in W_d\}$, and the multiset of sentence-level features, $\{g(s)|\forall s \in S_d\}$, of each document d.

Here, we use the RNTN model as a sentence-level classifier and feature extractor. It was selected since it has been proven quite powerful and it was the first one to achieve 85.4% accuracy on binary (i.e. positive/negative) sentence-level classification. The RNTN model can classify individual sentences and produces a 5-value sentiment probability distribution vector that corresponds to the following sentiments: *1 - very negative, 2 - negative, 3 - neutral, 4 - positive, 5 - very positive*, identified by $sent_i$ (where $i = 1, ..., 5$). We define the positive sentiment of a sentence as $sent_{pos}(s) = sent_4(s) + sent_5(s)$ and the negative sentiment as $sent_{neg}(s) = sent_1(s) + sent_2(s)$ to map labels into our considered label set $\{pos, neg\}$. Moreover, during the training of the RNTN model s semantic vector is produced for each input phrase that captures its sentiment and semantic meaning. For now on we will refer to the sentiment distribution of a sentence s as $sent(s)$ and to the corresponding semantic vector as $vec(s)$.

Sentcenter: Sentiment Center estimation. Most of the proposed cascaded sentiment analysis methods use machine learning techniques to accomplish the task of capturing a document's sentiment. So, if a training set D and the corresponding labels T are available, then we can train a classifier to proceed with this task using the sentiment center (*Sentiment center vector*) of a document or the variance of such sentences around this center (*Sentiment variance vector*). Then we can use either of such vectors as features to represent a document.

Definition 1 *Sentiment center vector (Sentcenter):*

$$sent_{center}(d) = \sum_{s \in d} sent(s)/|d|$$

where $|d|$ is the number of sentences of document d.

Definition 2 *Sentiment variance vector (Sentcenter):*

$$sent_{var}(d) = \sum_{s \in d} (sent(s) - sent_{center}(d))^2/|d|$$

which contains the squared differences from the document's center (variance).

In the experimentation section we refer to the document representation based on $sent_{center}(d)$ as *SentC* and based on $sent_{var}(d)$ as *SentC(c).*

Semcenter: Semantic Center estimation. Here, we propose an approach that initially represents each sentence s with a semantic vector $vec(s)$. Then, we represent a document as a vector which contains the semantic center of all sentences' vectors, with either the *Semantic center vector* or the *Semantic variance vector*. Next, we define such two possible representations.

Definition 3 *Semantic center vector (Semcenter):*

$$vec_{center}(d) = \sum_{s \in d} vec(s)/|d|$$

where $|d|$ is the number of sentences in the document d.

Definition 4 *Semantic variance center*

$$vec_{var}(d) = \sum_{s \in d} (vec(s) - vec_{center}(d))^2 / |d|$$

which contains the squared differences from the document's center.

In the experimentation section we refer to the document representation based on $vec_{center}(d)$ as *SemC* and based on $vec_{var}(d)$ as *SemC(c)*.

Centerbook: Sentiment & Semantic Centered approach. The previous approaches summarize a document's sentences by using either their sentiment or semantic centers. However, as the sentences may vary greatly in terms of their sentiment and semantic content, here we propose the *Centerbook*, an approach that describes the documents in terms of a sentence-level semantic dictionary. This semantic dictionary will contain the basic sentiment and semantic concepts that appear in the sentences of our training set. We create such dictionary by clustering the set of all sentences appearing in D, where each cluster contains similar sentences in terms of their sentiment and semantic information. Such clusters are called *centercodes* since they actually provide an encoded reference to similar sentences. The resulting cendroid of a cluster captures the corresponding sentiment and semantic information. We can represent each sentence in a document by its nearest cluster. Thus, the overall document is then modelled by the set of all centercodes.

Based on the above, the objective is to learn a Centerbook with $cNum$ centercodes. The Centerbook is presented as a matrix $P \in \mathbb{R}^{m \times cNum}$, where each column of P corresponds to a distinct centercode ($P = [c_1\ c_2\ c_3\ ...\ c_{cNum}]$). Any vector $x_i \in \mathbb{R}^m$ can be reconstructed using a linear combination of these centercodes: $x_i = Psp_i + \epsilon$, where $sp_i \in \mathbb{R}^{cNum}$ is a sparse representation of x_i and ϵ is the reconstruction error (i.e. the distance between the original data and its estimate) vector.

The *Centerbook* aims to extract high level features as input to a classifier and so an encoding/decoding scheme is used. The *encoder* is responsible for mapping a sentence vector $x(s)$ to a lower dimensionality representation, while the *decoder* goes back from the high level feature representation to the original sentence vector. For going back to the original sentence there is loss of information, which is known as reconstruction error, ϵ. In Centerbook each vector of a sentence s is represented by its nearest center in the set of all the centercodes. So, the reconstruction error from a given vector is the distance between it and its representation. The encoder produces a sparse representation as each x_i is encoded by using only one center of the Centerbook. Based on the above, our objective is to compute P and each sp_i to minimize the reconstruction error:

Definition 5 *Minimization function of Centerbook:*

$$J_{Centerbook} = \sum_{i=1}^{n} ||Psp_i - x_i||_2^2$$
$$s.t. ||sp_i||_0 = 1$$

over a collection of n *vectors, where* $||sp||_0$ *is the number of non-zero elements of sp. Each* x_i *is encoded using one center in our Centerbook, i.e.* $||sp_i||_0 = 1$.

We use k-means clustering for extracting high-level representations from each sentence of a document. So, in our case, the minimization function is equivalent to minimizing the objective function of k-means [8]. Next, we initially present the k-mean's objective function, while then we prove that such objective function is equivalent with the *Centerbook*'s minimization of the reconstruction error.

Definition 6 *Minimization function of K-means:*

$$J_{kmeans} = \sum_{i=1}^{k} \sum_{j=1}^{|G_i|} ||c_i - x_i^{(j)}||_2^2$$

where G_i is the set of vectors of the i-th cluster, $x_i^{(j)}$ is the j-th element of G_i and c_i the centroid of the corresponding cluster.

Lemma 1. *The Centerbook's minimization function is equivalent to the minimization function of K-means.*

Proof.

$$\begin{aligned} min\ J_{Centerbook} &= \sum_{i=1}^{n} ||Psp_i - x_i||_2^2 \ s.t.\ ||sp_i||_0 = 1 \\ \text{equiv. to } min \sum_{i=1}^{n} &||c_{argmin(||c_j - x_i||_2^2)} - x_i||_2^2 \\ &= \sum_{i=1}^{cNum} \sum_{j=1}^{|G_i|} ||c_i - x_i^{(j)}||_2^2 = J_{kmean} \end{aligned}$$

where the Centerbook contains the centers of the clusters: $P = [c_1\ c_2\ c_3\ ...\ c_{cNum}]$.

The encoding function, $h(x)$, for extracting high level features from a sentence is defined as:

Definition 7 *Sentence encoding function:*

$$h_i(x) = \begin{cases} 1, & i == arg_j min(||c_j - x||_2^2) \\ 0, & otherwise \end{cases}$$

where $h_i(x)$ is the i-th element of $h(x)$ vector.

Here, we use the RNTN vectors ($vec(s)$) as input to the above method to create the Centerbook. Based on the sentence encoding, a document d is defined as a vector $code(d)$ that includes the sum of all the already encoded sentences (referred to as *C_Book(c)*):

Definition 8 *Document representation:*

$$code(d) = \sum_{s \in S_d} h(vec(s))$$

where s is each sentence of a document d.

Finally, a binary document modelling is also defined by replacing the (positive) non-zero elements of $code(d)$ with 1 (referred to as *C_Book(b)*):

Definition 9 *Binary document representation:*

$$code_{bin}(d) = sign(code(d))$$

where the $sign(x)$ *function is applied element-wise.*

The $code(d)$ or the $code_{bin}(d)$ vectors can be used as features for the classification process. Our intuition is that the first approach for modelling a document, namely $code(d)$, will result in a better classification accuracy as it provides information about the number of sentences existing in a cluster (centercode).

4 EXPERIMENTS AND RESULTS

Here, we evaluate the center-based aggregation approaches for document representation and the MultiSpot method using different proposed features in terms of their capability for accurate capturing the sentiments expressed on documents.

4.1 Datasets

We experimented on two datasets, the *Large Movie Review Dataset* (IMDB) [7] and the *Polarity Dataset v2.0* (RT-2k) [11], both of which contain movie reviews collected from the Internet Movie DataBase. These datasets were chosen due to their popularity and their suitability for comparisons in terms of the accuracy of the proposed and earlier sentiment analysis approaches. The *IMDB* dataset is composed of 25000 positive reviews, 25000 negative reviews and 50000 unlabeled reviews, while the *Polarity Dataset v2.0* is composed of 1000 positive and 1000 negative annotated reviews. Each movie review in the IMDB dataset has several sentences, while in the RT-2k dataset each review contains a single sentence.

4.2 MultiSpot Fundamental Processes

Next, we provide the specific choices for each step of the MultiSpot approach in order to carry out the experimentation under the chosen datasets.

Word-level features extraction. We use a simple BoW scheme and the NB bigrams features.

Sentence-level feature extraction. We relied to the RNTN model to extract sentence-level features. The model was trained using a Sentiment Treebank [16], which contains 11,855 sentences from the dataset introduced in [12].

Clustering. We run k-mean for 15 iterations (after that our method converged). Here, due to lack of space, we present and discuss experimentation rounds with the IMDB dataset having the number of output clusters to be 200 and with the RT-2k dataset having 100 clusters. The sizes of each Centerbook were empirically selected ([22]). Under our experimentation efforts we observed that around a certain number of clusters there is a little effect in the accuracy of the MultiSpot method (see experimentation results in Section 4.3).

Classification. We use a linear SVM (*liblinear* library [3]) to classify each document based on the word and/or sentence level features. We conduct ten-fold cross validation to select the best SVM model. We applied a ten-fold validation as we use the SVM training protocol of [11].

4.3 MultiSpot results and methods comparisons

Here, we initially evaluate the proposed center-based aggregated approaches and compare them with MultiSpot.

Center-based aggregated methods evaluation From Table 1, we observe that in the IMDB dataset the best accuracy is achieved through combining the proposed C_Book(c) and SentC methods with the Sentcenter approach, while in the RT-2k dataset the best accuracy is achieved with the Semcenter approach. Generally, we observe that the SemC and SemC(c) approaches which use semantic features, always provide better classification results than the semantic-free, Sentcenter (i.e. SentC), approach.

Table 1: Evaluation of the proposed cascaded sentiment analysis methods.

Features	**IMDB**			**RT-2k**		
	Accuracy	**Recall**	**F1**	**Accuracy**	**Recall**	**F1**
SentC	84.02	80.97	83.52	83.30	86.10	83.75
SentC(c)	84.01	80.96	83.51	83.05	86.30	83.58
SemC	84.90	83.45	84.68	**85.10**	86.80	**85.35**
SemC(c)	85.27	83.53	85.01	84.90	85.50	84.99
C_Book(c)	84.76	85.05	84.80	83.15	84.10	83.34
C_Book(b)	84.54	84.64	84.50	82.75	83.60	82.90
SemC(c) + SentC	85.27	83.53	85.01	85.05	85.40	85.10
SemC(c) + C_Book(c) + SentC	**85.35**	84.30	**85.20**	84.85	87.10	85.18

MultiSpot evaluation To combine word level with sentence level features initially we evaluate MultiSpot (see Table 2) with the BoW model (as word level features). For the IMDB dataset term frequency weighting scheme was utilized, while for the RT-2k the binary weighting scheme.

Table 2: Evaluation of the MultiSpot method using BoW features.

Features	**IMDB**			**RT-2k**		
	Accuracy	**Recall**	**F1**	**Accuracy**	**Recall**	**F1**
BoW	87.77	88.01	87.80	87.15	88.40	87.31
BoW + SentC	88.99	89.01	88.99	87.45	88.70	87.60
BoW + SemC(c)	89.36	89.18	89.34	88.20	89.60	88.36
BoW + C_Book(c)	89.29	89.26	89.29	88.85	90.80	89.06
BoW + SentC + SemC(c)	89.38	89.22	89.36	88.25	89.70	88.42
BoW + SentC + SemC(c) + C_Book(c)	**89.48**	89.19	**89.45**	**89.05**	91.50	**89.31**

By comparing Table 1 and Table 2 we observe that combining word and sentence level features always increases significantly the classification accuracy. More specifically, the combination of all the proposed sentence level features with the BoW word level features succeeds an 1.71% improvement for the IMDB dataset and 1.9% for the RT-2k dataset.

We have also experimented with the choice of the NB bigrams for examining how the "strength" of the word-level features affects the accuracy gain in a multilevel setup (see Table 3). We've used the same number of folds and parameters in the SVM classifier in the RT-2k dataset as in [19], to ensure the comparability of the results. We observe again that through combining word and sentence level features the overall classification accuracy is increased by 0.35% in the IMDB case and by 1.85% in the RT-2k.

The next experiments support our claim that the multilevel cascaded sentiment analysis approach can improve the accuracy of any document-based sentiment classifier which uses word-level information. Initially, we use *Friedman's*[3]

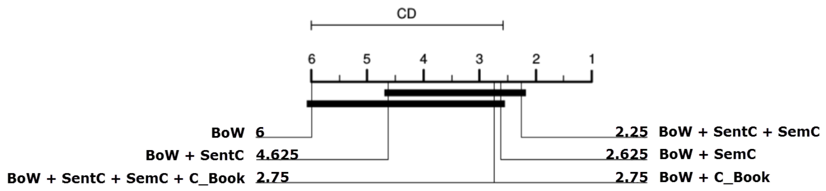
[3] It is used to test differences between different samples.

Table 3: Evaluation of the MultiSpot method using NB-bigram features.

Features	**IMDB**			**RT-2k**		
	Accuracy	**Recall**	**F1**	**Accuracy**	**Recall**	**F1**
NB bi	91.43	92.13	91.49	89.45	90.80	89.59
NB bi+ SentC	91.72	91.77	91.73	90.00	91.30	90.13
NB bi+ SemC(c)	91.76	91.49	91.73	90.90	91.90	90.99
NB bi+ C_Book(c)	91.72	91.90	91.73	**91.30**	93.50	**91.49**
NB bi+ SentC + SemC(c)	**91.78**	91.53	**91.76**	90.85	91.90	90.95
NB bi+ SentC + SemC(c) + C_Book(c)	91.60	91.33	91.58	90.65	93.10	90.87

test to confirm that the accuracy improvement succeeded with the MultiSpot is statistically significant. We use such test to detect statistical differences among the results of the proposed methods and the word-level methods. The tests were performed using both datasets. The Friedman test showed that the effect of MultiSpot for sentiment classification is statistically significant at a significance level of 5%. Thus, we can reject the null hypothesis: *Multilevel cascaded sentiment analysis does not increase the accuracy of the baseline classifier (BoW).*

Fig. 1: Nemenyi post hoc test



Based on this rejection, the *Nemenyi post hoc*[4] test was used to compare all classifiers with each other and spot the methods that are statistically better. Figure 1 shows that the combination of BoW with the Sentcenter (SentC) and Semcenter (SemC) features is statistically better than a simple BoW classifier. Thus, we can safely conclude that the accumulation of word and sentence level features improves the accuracy of a cascaded sentiment analysis approach.

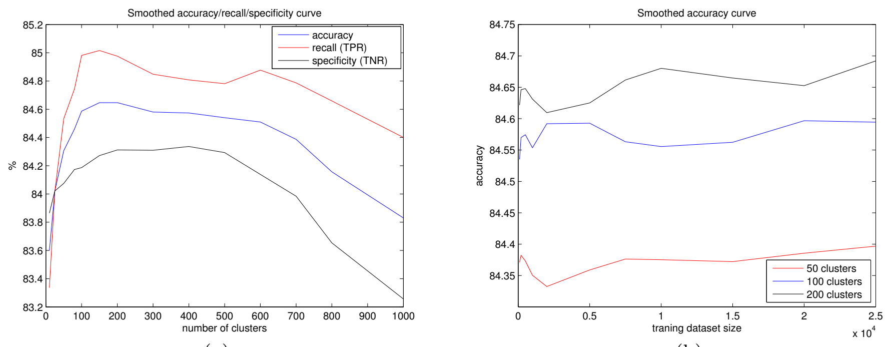
Table 4: Comparison of MultiSpot with the state-of-the-art approaches.

Method	**IMDB**	**RT-2k**
MultiSpot method		
BoW + SentC(c) + SemC(c)	89.38	88.25
BoW + SentC(c) + SemC(c) + C_Book(c)	89.48	89.05
NB bi + C_Book(c)	91.72	<u>**91.30**</u>
NB bi + SentC(c) + SemC(c)	**91.78**	90.85
State-of-the-art approaches		
Full + Unlabeled + BoW [7]	88.89	88.90
BoW SVM [11]	-	87.15
tfΔidf [9]	-	88.10
Appr. Taxonomy [20]	-	**90.20**
Word Repr. RBM + BoW [2]	89.23	-
NB SVM bigrams [19]	91.22	89.45
Paragraph Vector [6]	<u>**92.58**</u>	-

[4] Explores the groups of data that differ after a statistical test of multiple comparisons, e.g. the Friedman test.

Table 4 provides a comparison of the MultiSpot with the state-of-the-art approaches (we report only the best results of each approach). Our approach appears to outperform the state-of-the-art in the RT-2k dataset by 1.1%. In the IMDB dataset we achieved 91.78% accuracy, which even though it is 0.8% less than the most successful approach, it outperforms all the other methods. Here, due to space limits we did not have the opportunity to combine our method with the paragraph vector method. However, our intuition is that such a combination would lead to better classification accuracy. We also observe that MultiSpot's accuracy is in all cases improved when the NB binary features are used which demonstrates the importance of a careful selection of word-level features.

Centerbook analysis We examined how the number of clusters and the number of the available training data affect the quality of the Centerbook results. Due to lack of space, here we present the results obtained using the IMDB dataset. Figure 2a shows the effect of the number of clusters on the classification accuracy, with the best accuracy achieved at around 100 and 200 clusters. Figure 2b shows how the number of the available training data affect the quality of the Centerbook. We observe that the accuracy of the Centerbook method is not significantly affected (slowly increases as more training data is available) by the number of the training data. So, even with a small amount of training data the Centerbook approach will succeed to effectively capture a document's sentiment.



(a) (b)

Fig. 2: (2a) Effect of the number of clusters on Centerbook's quality (2b) Effect of different training dataset sizes on Centerbook's quality

5 Conclusions and Future Work

In this work, we elaborate the problem of efficiently capturing the sentiment expressed on textual resources. The contributions of this work are the following:

- we present MultiSpot, which exploits both word and sentence level features for acquiring a better sense of a document's sentimental content;
- two novel semantic related approaches (i.e. *Semcenter* and *Centerbook*) which further assist in better spotting a document's overall sentiment.

The conducted experiments have shown that the enhancement of a cascaded sentiment analysis approach with semantic aware features significantly increases its accuracy. Also, the initial hypothesis that the combination of word level with sentence level features provides better capturing of the sentiment expressed in a text is confirmed, as we observe that in all cases such a combination outperforms document-based word level approaches.

References

1. A. Coates et al. Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade*, 561–580. Springer, 2012.
2. G. E. Dahl, et al. Training restricted boltzmann machines on word observations. *arXiv preprint arXiv:1202.5695*, 2012.
3. R.-E. Fan, et al. LIBLINEAR: A Library for Large Linear Classification. *Machine Learning Research*, 9:1871–1874, 2008.
4. N. Godbole, et al. Largescale sentiment analysis for news and blogs. In *Proc. of the Conf. on Weblogs and Social Media (ICWSM).* Citeseer, 2007.
5. S. Lazebnik, et al. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition*, volume 2, 2169–2178. IEEE, 2006.
6. Q. V. Le et al. Distributed Representations of Sentences and Documents. *arXiv preprint arXiv:1405.4053*, 2014.
7. A. L. Maas, et al. Learning word vectors for sentiment analysis. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 142–150, 2011.
8. J. MacQueen. Some Methods for classification and Analysis of Multivariate Observations. In *Proc. of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 281–297, 1967.
9. J. Martineau et al. Delta TFIDF: An Improved Feature Space for Sentiment Analysis. In *ICWSM*, 2009.
10. G. Paltoglou et al. A study of information retrieval weighting schemes for sentiment analysis. In *Proc. of the 48th Annual Meeting of the Association for Computational Linguistics*, 1386–1395, 2010.
11. B. Pang et al. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proc. of the 42nd annual meeting on Association for Computational Linguistics*, 271, 2004.
12. B. Pang et al. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proc. of the 43rd Annual Meeting on Association for Computational Linguistics*, 115–124, 2005.
13. W. Shiyang et al. Emotion Classification in Microblog Texts Using Class Sequential Rules. In *28th AAAI Conf. on Artificial Intelligence*, 2014.
14. R. Socher, et al. Semantic compositionality through recursive matrix-vector spaces. In *Proc. of the 2012 Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 1201–1211, 2012.
15. R. Socher, et al. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proc. of the Conf. on EMNLP*, 151–161, 2011.
16. R. Socher, et al. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of the Conf. on EMNLP*, 1631–1642. Citeseer, 2013.
17. P. D. Turney, et al. From frequency to meaning: Vector space models of semantics. *Artificial intelligence research*, 37(1):141–188, 2010.
18. H. Wang, et al. Sentiment classification of online reviews: using sentence-based language model. *JETAI*, 26(1):13–31, 2014.
19. S. Wang et al. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proc. of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, 90–94, 2012.
20. C. Whitelaw, et al. Using appraisal groups for sentiment analysis. In *Proc. of ACM conf. on Information and knowledge management*, 625–631. ACM, 2005.
21. C. Zhang, et al. Sentiment analysis of Chinese documents: From sentence to document level. *JASIST*, 60(12):2474–2487, 2009.
22. W. Zhang, et al. Learning non-redundant codebooks for classifying complex objects. In *Proc. of the 25th conf. on Machine learning*, 1241–1248. ICML, 2009.