ISSN 2007-9737

Automatic Opinion Extraction from Short Hebrew Texts Using Machine Learning Techniques

Dror Mughaz[1], Tzeviya Fuchs[1], Dan Bouhnik[2]

[1] Bar-Ilan University, Department of Computer Science, Ramat-Gan, Israel

[2] Jerusalem College of Technology, Department of Computer Science, Jerusalem, Israel

myghaz@gmail.com, fuchstz@cs.biu.ac.il, bouhnik@jct.ac.il

Abstract. Sentiment analysis deals with classifying written texts according to their polarity. Previous research in this topic has been conducted mostly for Latin languages, and no research has been done for Hebrew. This is important because it turns out that the task of text classification is extremely language-dependent. Furthermore, the work on sentiment analysis for English texts was mostly performed on relatively long documents. In this work, we focus specifically on classifying Modern Hebrew sentences according to their polarity. We compare various Machine Learning algorithms and techniques of classification. We added optimizations and methods that have not previously been used, and adjusted commonly used techniques so they would suit a Hebrew corpus. We elaborate on the differences in classifying short texts versus long ones and about the uniqueness of working specifically with Hebrew. Finally, our model achieved nearly 93% accuracy, which is higher than accuracies achieved previously in this field.

Keyword. Automatic classification, machine learning, sentiment analysis, short Hebrew texts.

1 Introduction

Sentiment analysis, also known as opinion mining, is a type of Natural Language Processing (NLP) used to determine the attitude of a writer towards a certain topic, that is, to automatically classify the given text as belonging to one of the categories *positive* or *negative*.

Sentiment analysis started emerging in the late 1990's, but became a more prominent field from 2000 onwards.

This is due to the rise of social media which has caused interest in sentiment analysis. With the rapid growth of online data, automatic sentiment classification is important for marketing research, and is used mainly on data generated by internet users.

Intuitively, it seems that sentiment analysis is very similar to the traditional topic-based classification, where in this case the topics are *positive* and *negative*. Surprisingly, it turns out that sentiment categorization is more difficult and challenging than topic-based classification, as algorithms used for topic-based classification do not perform as well on sentiment [1]. For example, while topics can usually be identified with only keywords, sentiment can be conveyed subtly, without using explicit positive or negative words.

There are several more challenges in sentiment analysis [2]. For example, the polarity of an opinion could at times heavily depend on context or on writing style. Overall, sentiment analysis is a very domain-specific problem, and it is hard to create a domain independent classifier.

There has been some research conducted specifically on sentiment analysis, but most of it concerns identifying the general polarity of entire documents (text classification), rather than on short text segments. Within the sentence classification field, a portion of it concerns the classification of different types of texts (e.g. distinguishing between questions and responses, etc.), thus not addressing sentiment specifically. This is surprising, due to the rising need to classify short data segments such as talkbacks and tweets according to their polarities.

In this work, we focus specifically on classifying Modern Hebrew sentences according to their polarity. We divide the classification process into three phases, and in each phase we test several methods and compare between them, in order to find the optimal model to classify our type of corpus. We compare various Machine Learning algorithms and techniques of classification, and analyze the differences of our results compared to others'. In addition to the techniques commonly used for English, some of which had to be adjusted specifically for the special limitations of Hebrew, we added a few optimizations and methods that have not previously been used. In the model we finally chose, we achieved nearly 93% accuracy, which is higher than accuracies achieved previously in this field.

2 Related Work

There are two main techniques for sentiment classification: symbolic techniques and machine learning (ML) techniques [3]. The ML approaches to sentiment analysis tend to have better results than those of the symbolic approaches [4], but their results depend on the features selected. Among the ML approaches, it is widely agreed that the Support Vector Machine (SVM) algorithm performs best.

Annett & Kondrak in [4] compare between classification of blogs with lexical methods and ML methods. The lexical approach used a dictionary of pre-tagged words (using WordNet). ML methods, including SVM, Naive bayes (NB) and Alternating Decision Tree (ADTree), achieved 65.4%-77.5% accuracy. The relatively high accuracy of the ML methods proves the superiority of the approach. The authors point out that in ML, the types of features chosen have a strong influence on classification accuracy, whereas in lexical approaches, there is an upper bound of accuracy that they could have (it is difficult to get beyond 65%).

Regarding supervised ML methods, [5] and [1] compare SVM's performances on texts with other ML algorithms. [5] uses it for topic-based categorization, and compares it with NB, Rocchio, C4.5 and k-NN. SVM consistently achieves good performances of approximately 87%, outperforming the other methods significantly. The author further elaborates on the advantages of the SVM algorithm, whose ability to learn is independent of the dimensionality of the feature space, which makes it suitable for text classification.

On the other hand, [1] apply the ML algorithms specifically on sentiment classification, which they point out to be more difficult than topic-based categorization. They compare SVM with NB, MaxEnt, and a human-produced baseline, on a domain of 1400 document length movie reviews. The ML techniques obviously outperformed the human-produced baseline, and among the ML techniques, NB was the worst, at 77-81%, and SVM was the best, at 82%. For the ML techniques, they tested several representation options of the text vectors.

Boiy et al. in [3] did a similar comparison testing several Machine Learning algorithms on documents, this time between SVM to Naive Bayes Multinomial and MaxEnt, achieving 75.85%-87.4% accuracy. A better approach, which hasn't been described, achieved 90.25% accuracy.

An alternative approach was proposed by [6]. In their work, they use a Recursive Neural Tensor Network to classify sentence-length reviews as being positive or negative, achieving an accuracy of 85.4%.

Khoo et al. in [7] work on sentence classification is closer to ours in the sense that it applied ML algorithms specifically on sentences (as opposed to longer texts), but it differs by the fact that it deals with topic based categorization. In their work, the authors check different methods of text representation in a feature vector, and compare feature selection methods. Their experiments involved using different combinations of stop-words removal, tokenization and lemmatization on the text, and testing the methods with NB, DT and SVM. SVM outperformed the rest with micro-F1 averaging of 0.853, and with a specific representation technique they reached 0.883. They then applied feature selection and concluded that it did not change performances much for SVM and DT.

To sum up previous achievements in sentiment analysis; supervised methods on the document level (i.e., blogs, reviews, etc.) achieved accuracies of approx. 80%, and approx. 87% on the sentence level. When performing topic-based categorization rather than sentiment analysis, the accuracies were somewhat higher.

The aforementioned research and results show clearly that one of the most important steps of text classification is the extraction of features for the feature vector. Some research has been conducted to find the optimal feature vector representation. In Section V we provide a detailed overview of previous work done on representational issues.

3 The Hebrew Language

One of the difficulties of sentiment analysis is creating a language-independent classifier. The reason for this is the grammatical and morphological differences between languages. Therefore, a good classifier is usually one that has been constructed only for a specific language. Translating the text to English, or using language-independent classifiers would cause a great loss of important information.

Languages that have been studied most in this context are English and Chinese. At the present, there is very few research on sentiment classification for other languages. We have found no research on sentiment in the Hebrew language.

In this context, Semitic languages, specifically Hebrew, are much tougher languages than others. In Hebrew, a root word can take numerous forms depending on its context, tense, gender, and surrounding words. Thus, the number of different meanings and combinations a root word could have is much larger than those in English and Latin languages. Furthermore, Hebrew is considered to be a highly ambiguous language (i.e., every phrase could be interpreted in multiple different ways), with only 40-45% of its words being unambiguous.

The causes for ambiguity are mainly the acronyms and abbreviations used in Hebrew, and the fact that it is an unvocalized language (all of its letters are consonants), so words carry much information that does not appear in the script. Finally, the result of Modern Hebrew usage is the need to deal with Hebrew slang, foreign terms and many linguistic errors.

Some related work has been done in different fields of NLP in Hebrew [8]. Works that are related to text classification and refer to the challenges of Hebrew include the classification of Hebrew-Aramaic texts according to style [9]; authorship verification, including dealing with forgers and pseudonyms [10]; and classification of documents according to their historical period and ethnic origin [11].

4 The Corpus

Our corpus consists of cellphone reviews in Modern Hebrew written by users on a price comparison website called *Zap*[1]. The reviews are mostly short ones, typically containing approximately 2-5 lines. The reviews were divided and tagged manually into 3223 sentences: 1710 positives and 1512 negatives (so a random baseline, like Weka's [12] ZeroR, would have an accuracy of 53%). Various users wrote lists of attributes they liked/disliked about their device, and being lists more than sentences, were therefore not included in the corpus.

The sentences in the corpus, being user generated, are not necessarily syntactically and grammatically correct or well formed. In fact, many of them lack punctuation marks, have misspellings, and foreign terms are spelled in various different forms. These issues are sure to have a negative influence on classification. In many cases, when no period appeared to mark the end of a sentence, it was determined by the end of line. Sentence length could vary from (rarely) 2-3 words to long and elaborate ones.

5 Methodology

In order to find an optimal categorization technique, we divided the classification process into three separate phases, and studied each of them thoroughly. As mentioned previously, good classifiers are usually constructed for specific languages; therefore, we put emphasis on

[1] http://www.zap.co.il/

representational issues of feature extraction, it being language-dependent, and thus of great importance. We compare our methods with previously used methods, and suggest upon some of the characteristics of Hebrew sentiment classification, with an emphasis on what makes it unique and different from other classification tasks.

ML text classification could be roughly divided into three steps: *choosing the input vector*; *applying Feature Selection (FS)*; and *running the ML algorithm*.

Details on these three steps are as follows:

a) Choosing the Input Vector

This deals with the representational issues of extracting features from the text. Finding the best way to represent a given text as an input vector is probably one of the most important questions in text classification. In fact, apart from choosing which algorithm to use, most of the research in this field deals with finding the optimal features for the input vector.

1. Tokenization

Tokenization is the act of separating words from symbols. Without it, each word-symbol combination would be treated as a distinct feature; this would add unnecessary features with lower weights, and consequently reduce their power of classification. In our experiments, we have used the default tokenizer provided by a Hebrew POS tagger [13].

2. Word representation-unit

A single word could appear in various forms in a text. Therefore, it is reasonable to try to find a method that would map all the variations of a word to one unique representation. Notice that Hebrew has many more inflected forms for every word, making the problem more prominent.

In Hebrew, every word could be represented in one of the following manners: by its root, i.e., a sequence of three consonants, from which all Hebrew verbs and most of its nouns are derived; by its lemma, i.e. the base form of the word; by its stem, i.e. the word after having its affixes removed; or simply by using the raw terms as the features.

Thus, when all of the forms of a word are mapped to a single unique form, the power/ranking of that word could increase if it appears in different forms in a specific class. Consequently, it would also reduce the size of the feature vector [7]. Indeed, some research report a slight increase of accuracy when using stems/lemmas [4, 5].

However, many researchers [14, 7] claim that stemming/lemmatizing causes some decrease in accuracy. A unique representation of a word in the methods proposed here could be ambiguous, and using it would come with loss of detail in the language, and thus causing overgeneralization.

A different approach regarding word representation, is to add every bigram in the text as a feature (instead of using the unigrams). Hopefully, bigrams could capture some of the context of the words in the sentence. [1] show that using bigrams instead of and in addition to unigrams doesn't yield higher results. In fact, using bigrams alone caused some decline in the results.

3. Negation tags

A sentence, particularly a review, typically contains negation words that could potentially reverse the meaning of a sentence. Unfortunately, ML Bag of Words (BoW) techniques fail to preserve word order, and therefore are oblivious to the fact that certain words in a sentence are negated. An attempt to solve this problem is to add negation tags (a.k.a. sentiment shifters); this approach involves tagging every word after the negation word until the first punctuation mark.

Finding a list of Hebrew negation words is more complicated than in other languages. This is due to their many inflected forms, and due to the fact that in Hebrew, the negation word could be joined with other words in the sentence, meaning that the negation word is not necessarily implicit. Therefore, in addition to identifying negative words by a predefined negation word list, we also had to check the polarity of given words using a POS tagger.

Adding negation tags was performed by [1], and they report a very slight positive effect on performance. [14] on the other hand, state that it only hurt performances in their work.

4. Parts of Speech (POS) tagging

Adding POS tags is another way to help distinguish a word's meaning from similar words, thus performing some type of sense disambiguation, as words with a different POS tagging may be treated differently. However, experiments on English corpora show that POS tags don't affect performances very much, and sometimes degrade them [1].

The POS tagging could have another purpose; theoretically, it seems natural that sentiment would be expressed mainly by adjectives. Therefore, it would seem plausible to represent sentences with their adjectives only. [1] tried this technique, yielding relatively poor results. [3] reason this with the fact that adjectives only represent approximately 7.5% of the text in a document.

5. Stop words removal

Stop words are the most common words that appear in a text, and are there for grammatical reasons only, e.g., a, the, is, etc. They do not contribute to the sentences' meaning, so they are usually removed. Not all stop-lists are suitable for all applications, because words that could be interpreted as stop-words for one application could turn out to be vital for sentiment.

One way to construct a stop list is to find the most frequent words in the corpus. This must be done carefully because this list is sure to consist of words important for classification. In our corpus, for example, words like recommend, strong, works, satisfied, etc. appear very frequently, but they are certainly not stop words.

[15] compared the performance of sentiment classification on tweets with and without stop words removal. They conclude that stop-words removal harms classification. [7] also report stop words removal to be harmful for classification; in fact, words in their stop-lists were in the top of the list of words produced by FS methods. However, stop-words removal is still widely used in many text classification experiments.

b) Feature Selection

Applying Feature Selection is the act of filtering out irrelevant or low ranking features from the feature vector. It must be first noted that FS should be applied gently, as even the lowest ranking features hold some relevance for classification - classifying using them alone provides performances that are better than random. Consequently, a good classifier would probably have a high dimensional feature vector, and aggressive FS could yield diminished results [4].

Table 1. POS Patterns from the Corpus

POS Pattern	Example (translated)
Negation Adverb	*'**not simple** to use'*
Interrogative Negation	*'OK for **whoever isn't** [...]'*
Negation Adjective	*'**Uncomfortable**'*
Existential Noun	*'Too bad **there isn't** access [...]'*
Noun Adjective	*'...with **high color** precision'*

In fact, it could be argued that SVM, with its suitability to high dimensional feature spaces, performs sufficiently without applying FS [5].

As the feature space in text classification could be very large, FS could be used to reduce high computational load. In practice, it is used to improve classification performance [7].

Feature selection could be applied by removing the least frequent words from the feature vector, or by removing the words with the least weight assigned by a linear SVM model.

Previous research report that the first method does not significantly affect classification [1], while the second method yields good performances.

c) Running the ML Algorithm

For this part, we chose three algorithms: Bayesian Logistic Regression (BLR), Voted Perceptron (VP), and Support Vector Machines (SVMs). For BLR and VP we used the Weka package with default parameters. For the SVM algorithm, we used the LIBSVM package [16] for training and testing, using a linear kernel: $K(x_i, x_j) = x_i^T x_j$. After performing grid-search, the penalty parameter C is set to 0.5. The results presented are 10-fold cross validation accuracy. A similar package, LibShortText [17] (which also implements SVM), was tested as well; despite its suitability for short texts, it yielded similar results.

6 Results

a) Baseline

As a baseline experiment, we tested the classification performance of our corpus, when every sentence was classified according to the number of positive and negative words appearing in it. This seamed plausible due to the corpus' nature as persuasive quality reviews. In this experiment, a word was considered to be positive if it appeared to be a synonym of one of the Hebrew words *good* or *excellent*, taken from two Hebrew thesauri [18] [19] and was considered to be negative if it appeared to be a synonym of one of the Hebrew words *bad* or *very bad*. These lists of synonyms contained 204 and 154 phrases respectively. The texts were tokenized before classification.

The accuracy was 24.73%, with a very high percentage of ties (64.78%). We repeated the experiment, this time after lemmatizing the corpus. This was done so conjugated polarity words would be counted as well. Indeed, the accuracy increased to 34.75%. However, lemmatization also caused some over-generalization, for not only did the percentage of correct classification increase, but also the percentage of the sentences incorrectly classified. The percentage of ties decreased, but was still high, at 52.4%. This comes to show that in many cases, explicit polarity terms may not be enough for classification.

b) Machine Learning Techniques

The main part of this research deals with the optimal way to represent Hebrew text. We have experimented with various combinations of representation techniques, and they are fully presented in Table 4. After investigating the different techniques in section Methodology, we shall now elaborate on our approach and results.

The following results have been evaluated using SVM. The results of BLR were generally similar, but lower on average. VP yielded much lower results.

Firstly, all of our experiments were implemented on the corpus after it has been tokenized (with a Hebrew POS tagger's default tokenizer). We begin with the most basic representation, involving a binary feature vector of raw unigrams. Its performance, extremely close to 80%, as represented in line 4 of Table 4, shows that this basic representation yields reasonable results, with SVM generally outperforming the other classifiers.

We now proceed with experiments of the various techniques presented in Section 5.

1. Word representation unit

The Hebrew root being too vague and the stem being extremely inaccurate, we experiment with representing every word in the text with its lemma (Table 4, line 5) rather than its raw form, yielding accuracy rates higher in more than 2%. We used a Hebrew POS-tagger to find the lemma of every word.

Upon examining the influence that lemmatization has on classification, we found that, as expected, it reduces the number of features in the feature vector, provides a unique representation to groups of related words - and of various spellings of words - and thus gives words more accurate rankings. For example, without lemmatizing, the word *the cellphone* (which is a single word in Hebrew) and *a cellphone* (again, a single word) are treated as different words when in fact they are not.

A more critical example is found with the words *no*, *and not* and *that isn't*, (which in Hebrew are the same base word with different prefixes) which are treated as different words, and the high negative ranking the word *no* received when using lemmatization is distributed between the different variants of the word when no lemmatization is applied, thus reducing the influence of the word. Lemmatization also clustered different spelling forms of specific words, by converting them all to the *ktiv maleh*[2] form.

Bigrams. When using bigrams, we distinguished between two cases; representing the text with bigrams of words or with bigrams of lemmas. When using bigrams + words, performances declined, probably due to their being too many un-generalized features. However, when using bigrams + lemmas, performances remained the same. Still, upon examining the weight vector, it

[2] *Ktiv maleh* - rules of spelling-without-*Niqqud*; it involves adding certain consonants (such as Waw and Yod) to words, to be used as vowels.

seems that the bigrams served their purpose: they 'caught' the meaning of phrases. Thus, for example, the word *small* (when using unigrams) was ranked negatively, whereas *small and comfortable* was ranked positively.

2. Negation tags

Surprisingly, negation tags proved to be effective (Table 4, line 2); increasing performances in approximately 2%, it seems to be significant for classification. In fact, the word *no* is one of the most frequently used words in the corpus. This comes in contrast to previous results in English stating that negation tags have a slight effect on performance.

Adding negation tags, however, is not a perfect solution due to the use of implicit negation terms, or alternatively the use of the negation term after the description of the problem, instead of before it. Nevertheless, adding negation tags turned out to be very useful.

3. Thwarted expectations

One of the common problems in sentiment analysis is the presence of *thwarted expectations* sentences, that is, sentences that build up a certain impression and then conclude with a contradicting phrase. An example from our corpus is the following negative sentence: *'At first the phone looks very comfortable, but after some time flaws start to appear.'*

Using the BoW feature, a learning algorithm has no way to understand that the words after the *but* are more important than those before it. Therefore, we tried a different, simple approach that, upon detecting the word *but* (or one of its synonyms), removes the words that appear before it, assuming that they are irrelevant since the words after the *but* contradict them. Indeed, there was a slight increase in performance (Table 4, line 3).

4. Parts of speech

Surprisingly, adding POS tags did not have a crucial impact on classification. Comparing Table 4's lines 1 and 6, there is less than a 1% difference. With all of the ambiguity problems that exist in Hebrew, it would have seemed that adding POS tags is a vital. However, the light impact it had could be explained by the corpus being domain specific, thus somewhat reducing ambiguity concerns. Another explanation is the inaccuracy of the POS tagger.

Further expanding the usage of POS tags, we experimented on finding whether any specific POS patterns that could indicate on sentiment repeated themselves throughout the corpus. Technically, we extracted bigrams and their POS tags, discarding the words to which the tags belonged, for the feature vector. A similar method has been used by [20] to classify between subjective and objective sentences, and yielded precision of 70%-80%. In our case, accuracy rates are slightly lower, at 68%, and at 69% when applying FS. However, the classifier did in fact find some interesting POS patterns; several of the features ranking highest are shown in Table 1. Apart from helping classification, these attributes come to show that it does not only matter what is said in the sentence, but also *how* it is said. For example, various writers wrote their reviews in the form of *'The <product> is OK for whoever needs it only for...'* etc., which brought up the *Interrogative Negation* feature.

5. Adjectives

Adding adjectives alone to the feature vector yielded surprisingly low results (74%), showing that sentiment, even on products, isn't necessarily expressed using adjectives (Table 4, line 7). As we have mentioned before, the word *no*, for example, which obviously isn't an adjective, turned out to be an important negative feature.

6. Stop words

Removing stop words is commonly applied before classification. As shown in Table 4 lines 11 and 12, removing stop words has a negligible, but slightly harmful effect on predictions. Out of a list of around 60 stop words, more than half turned out to be ranked between 0.1 and 0.6, meaning that they could, in fact, be quite significant. Among the highest ranking stop words are *and therefore*, *if* and *but*, which puts into question the liability of a manually constructed stop-list.

It should be noted that stop-word removal in Hebrew is slightly different and perhaps less necessary than it is in English, because typical English stop words such as *a*, *and* or *the* either do not exist in Hebrew or are represented by prefixes (and are not removed with stop-lists).

7. Feature selection

We now first apply FS according to the weight of the words, removing features that ranked less than 0.1. The results were drastic: performance jumped from 79.9% without FS (Table 4, line 4) to 86.5% with it (Table 4, line 10). This contradicts previous works that show that SVM is not very sensitive to FS [7, 5]. This could be due to the amount of noise that exists in the corpus. To check this type of FS further, we tried removing features with heavier weights; that is, removing features that ranked less than 0.2, 0.3, etc., with jumps of 0.1. It turns out that once features of higher rankings are removed, the classification accuracy starts to decline.

Another form of FS, which is in fact quite common, is the removal of infrequent words from the feature vector. We tried this, removing words that appeared less than 4 times in the corpus. It turned out to be slightly harmful. We then experimented with a new method that essentially applies some form of feature selection. The method we propose attempts to find an important part of the sentence, and use it alone for classification.

Initially, the corpus was tagged manually as follows: a unique symbol was placed around the object of the sentence - that is, around the phrase (usually a single word) that the writer was referring to. As some figures of speech allow the writer to omit the object of the sentence, not all sentences suited our needs and we used a subset of 3046 sentences, out of 3223 in the original corpus.

Once the object of the sentence is determined, a window size is chosen; thus, the only part of the sentence that the classifier uses is the window size surrounding both sides of the object.

Assuming that the words surrounding the object hold most importance, this method should yield reasonable results.

We'll take the following sentence as an example: *'The sound quality is excellent, the best I've heard on mobile devices, especially when you hear music with headphones, because there is an option for Dolby Surround.'*

Placing a unique symbol around the object:

Table 2. Feature Selection using various windows sizes

Window Size	Sentence Remaining Content
0	*'<obj> sound quality </obj>'*
1	*'The <obj> sound quality </obj> is'*
2	*'The <obj> sound quality </obj> is excellent'*

Table 3. Accuracies according to window size

Window Size	Accuracy
0	61.88%
1	72.88%
2	77.97%
3	80.56%
4	83.39%
5	86.01%
(no window)	88.49%

'The <obj> sound quality </obj> is excellent, the best I've heard on mobile devices, especially when you hear music with headphones, because there is an option for Dolby Surround.'

Examples of various window sizes are in Table 2. Results vary according to the chosen window size, and are shown in Table 3.

As we assumed, most results perform well - and increase in performance as the window size expands. Surprisingly, a window size of zero performed better than random; indeed, even the object of a sentence implies its sentiment, as the advantages and disadvantages of cellphones are usually common.

Using the whole sentence yielded 88.49% accuracy (Table 4, line 1). Using other portions of the sentences for classification did not perform as well as when using the window.

This method obviously did not improve classification; however, it proved that the words surrounding the object are significant for classification. This fact could be used when there are concerns for the computational load of the corpus, or when sentences tend to be exceptionally long.

Table 4. Results

	Unigram/ Bigram	Word/ lemma	Neg. tags	'buts' tags	FS[a]	SVM ACC[a]	BLR ACC[a]	VP ACC[a]
1.	Unigram	word	True	False	True	88.49 %	87.10%	81.67%
2.	Unigram	word	True	False	False	81.91 %	81.81%	79.27%
3.	Unigram	word	True	True	False	82.35 %	80.93%	78.69%
4.	Unigram	word	False	False	False	79.93 %	78.17%	76.22%
5.	Unigram	lemma	True	False	True	84.46 %	82.90%	80.45%
6.	Unigram + POS	word	True	False	True	89.30 %	87.79%	81.64%
7.	Unigram (adjectives only)	word	True	False	True	74.19 %	73.91%	72.26%
8.	Unigram + Bigram	lemma	True	False	True	91.9%	92.37%	86.13%
9.	Unigram + Bigram + POS	lemma	True	False	True	91.19 %	92.58%	86.32%
10.	Unigram	word	False	False	True	86.47 %	83.97%	79.15%
11.	Unigram (with sw[a])	word	True	False	True	88.89 %	88.03%	81.80%
12.	Unigram (with sw[a])	lemma	True	False	True	88.95 %	86.70%	82.61%

Comparison of results using three machine learning algorithms and various vector representations.
aList of abbreviations for this table: Acc - accuracy; FS - feature selection (specifically referring to the removal of weights under 0.1); sw - stop words.

c) Best results

Combining both unigrams and bigrams seems to be a natural choice, as spoken language is made of both individual words and phrases (phrases that are longer than 2 words are usually abbreviated when written).

Indeed, the best feature representations involved the combination of bigrams and unigrams with lemmas, with POS tags (Table 4, line 9) and without (Table 4, line 8) (binary vector, weight FS), yielding 91% and nearly 92% accuracy (respectively). It should be noted that in these two cases, Bayesian Logistic Regression gave slightly higher results than the other algorithms; 92.6% and 92.4% (respectively) which are exceptionally high.

7 Conclusion

We have presented a set of experiments on sentence classification by sentiment, on reviews written in Hebrew. Sentiment analysis has been widely performed on texts, but far less on sentences, and none has been performed on the Hebrew language. Here we showed unique characteristics of classifying Hebrew texts by polarity, with an emphasis on the fact that the texts are considered to be short.

It is generally believed that when classifying sentences, one should be very careful not to lose any piece of their precious and scarce information.

Here, on the other hand, the results implied that the feature vectors had much noise in them; lemmatization and FS, which independently reduce the feature vector size, caused improvements in classification, when previous works show that they do not influence classification much (at least when using SVMs). This could in fact be a side effect of working with Modern Hebrew - the inflected words, the various spelling forms of them, etc. could all be causes of unnecessary features.

Another result that stood out was the fact that despite previous work claiming it is unnecessary, and despite its inaccuracy in our corpus - adding negation tags (in a method we designed specifically for Hebrew) improved classification. In fact, negation expressions were very dominant among the words of the corpus (which of course isn't skewed). This, too, could be a result of working with a Hebrew corpus; it shows the different language structure of Hebrew, and perhaps shows a cultural difference between Hebrew writers to the writers of Latin languages.

Regarding additional results that seem to be language specific, stop words removal might not be as necessary as in other languages, as it seems that many words that would be considered to be stop words in Latin languages either do not exist in Hebrew, or exist as prefixes (and are therefore not removed with stop-lists).

Finally, it should be noted that while SVMs are considered to be the best classification methods, and indeed they yielded the highest results, Bayesian Logistic Regression yielded results almost as high (on average slightly lower). The highest results were achieved by Bayesian Logistic Regression, at nearly 93%, when previous somewhat similar works achieved a maximum of 85%-88%.

In addition, we introduced two new methods: adding tags that attempted to solve the problem of *thwarted expectations* in a sentence, which improved classification, and adding a window size which determined the part of the sentence to classify; this method yielded reasonable results and significantly reduced computational load. Furthermore, it provided us an insight on the structure of the sentences classified, which could be useful when classifying exceptionally long sentences.

References

1. **Bo, P., Lee, L., & Vaithyanathan, S. (2002).** Thumbs up?: Sentiment classification using machine learning techniques. *Proceedings of the ACL-02 conference of Empirical methods in natural language,* Association for Computational Linguistics, Vol. 10. DOI: 0.3115/1118693. 11118704.
2. **Vinodhini, G. & Chandrasekaran, R. M. (2012).** Sentiment analysis and opinion mining: a survey. *International Journal of Advanced Reseach in Computer Science and Software Engineering.*
3. **Boiy, E., Hens, P., Deschacht, K., & Moens, M. F. (2007)**. *Automatic Sentiment Analysis in On-line Text*. in ELPUB.
4. **Annett, M. & Grzegorz, K. (2008).** A comparison of sentiment analysis techniques: Polarizing movie blogs. *Conference of the Canadian Society for Computational Studies of Intelligence*. DOI: 10.1007/978-3-540-68825-9_3.
5. **Joachims, T. (1998).** *Text categorization with support vector machines: Learning with many relevant features*. Springer Berlin Heidelberg. DOI: 10.1007/BFb0026683.
6. **Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013).** Recursive deep models for semantic compositionality over a sentiment treebank. *Proceedings of the conference on empirical methods in natural language processing (EMNLP).*
7. **Khoo, A., Marom, Y., & Albrecht, D. (2006).** Experiments with sentence classification. *Proceedings of the Australasian language technology workshop.*
8. **Wintner, S. (2004).** Hebrew computational linguistics: Past and future. *Artificial Intelligence Review*, Vol. 21, No. 2, pp. 113–138, DOI: 10.1023/B:AIRE.0000020865.73561.bc.
9. **Mughaz, D. (2003).** *Classification of Hebrew texts according to style.* M.Sc. thesis (in Hebrew), Ramat-Gan, Israel: Bar-Ilan University.
10. **Koppel, M., Mughaz, D., & Akiva, N. (2003).** CHAT: A System for Stylistic Classification of Hebrew-Aramaic Texts. *Third KDD Workshop on Operational Text Categorization*.
11. **HaCohen-Kerner, Y., Beck, H., Yehudai, E., & Mughaz, D. (2010).** Stylistic Feature Sets as Classifiers of Documents According to their Historical Period and Ethnic Origin. *Applied Artificial Intelligence*, Vol. 24, No. 9. DOI: 10.1080/08839514.2010.514197.

12. **Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutermann, P., & Witten, I. H. (2009).** The WEKA Data Mining Software. *An Update SIGKDD Explorations*, Vol. 11, No. 1. DOI: 10.1145/ 1656274.1656278.
13. **Adler, M. & Elhadad, M. (2006).** An unsupervised morphemebased hmm for hebrew morphological disambiguation. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics.* DOI: 10.3115/1220175.1220259.
14. **Dave, K., Lawrence, S., & Pennock, D. M. (2003).** Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. *Proceedings of the 12th international conference on World Wide Web*. DOI: 10.1145/775152.775226.
15. **Saif, H., He Y., & Alani, H. (2012).** Semantic sentiment analysis of twitter. *The Semantic Web-ISWC´12.*
16. **Chang, C. C. & Lin, C. J. (2011).** LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, Vol. 2, No. 3, DOI: 10.1145/1961 189.1961199.
17. **Yu, H. F., Ho, C. H., Juan, Y. C., & Lin, C. J. (2012).** *LibShortText: A Library for Short-text Classification and Analysis*.
18. **Eitan, A. (2000)** *Word-by-Word: A Thesaurus* (Hebrew), Itab Ltd.
19. **Milog (2014).** Available: http://milog.co.il/.
20. **Rilofff, E. & Wiebe, J. (2003)** *Learning extraction patterns for subjective expressions*. DOI: 10.3115/1119355.1119369.

Article received on 19/12/2017; accepted on 15/02/2018.
Corresponding author is Dror Mughaz.