Linguistics and Literature Studies 5(1): 36-50, 2017
DOI: 10.13189/lls.2017.050103 http://www.hrpub.org

Subjective? Emotional? Emotive?: Language Combinatorics based Automatic Detection of Emotionally Loaded Sentences

Michal Ptaszynski[1,*], **Fumito Masui**[1], **Rafal Rzepka**[2], **Kenji Araki**[2]

[1]Department of Computer Science, Kitami Institute of Technology, Japan

[2]Graduate School of Information Science and Technology, Hokkaido University, Japan

Abstract In this paper presents our research in automatic detection of emotionally loaded, or *emotive* sentences. We define the problem from a linguistic point of view assuming that emotive sentences stand out both lexically and grammatically. To verify this assumption we prepare a text classification experiment. In the experiment we apply language combinatorics approach to automatically extract emotive patterns from training sentences. the applied approach allows automatic extraction of not only widely used unigrams (tokens), or n-grams, but also more sophisticated patterns with disjointed elements. The results of experiments are explained with the use of means such as standard Precision, Recall and balanced F-score. The algorithm also provides a refined list of most frequent sophisticated patterns typical for both emotive and non-emotive context. The method reached results comparable to the state of the art, while the fact that it is fully automatic makes it more efficient and language independent.

Keywords Affect Analysis, Sentiment Analysis, Pattern Extraction, Language Modeling, Emotive Expressions, Language Combinatorics

1 Introduction

Among recent developments in Natural Language Processing (NLP) research the one that has attracted increasing interest has been in the field of sentiment analysis (SA). The goal of SA research is to distinguish between sentences loaded with positive and negative attitudes. It has become popular to try different methods to distinguish between sentences loaded with positive and negative sentiments. However, a few research focused on a task more generic, namely, discriminating if a sentence is even loaded with emotional content or not. The difficulty of the task is indicated by three facts. Firstly, the task has not been widely undertaken. Secondly, in research which addresses the challenge, the definition of the task is usually based on subjective ad hoc assumptions. Thirdly, in research which do tackle the problem in a systematic way, the results are usually unsatisfactory, and satisfactory results can be obtained only with large workload.

To approach the problem from a systematic perspective we focused on emotive sentences which in linguistics are defined as fulfilling the emotive function of language. We also assumed that such sentences contain repetitive patterns representative for contents (sentences, utterances, etc.) produced with emotional attitude, in contrast to contents produced with emotionless attitude. Then, we used a novel algorithm based on the idea of language combinatorics to verify the above assumptions. The method proposed in this paper not only achieved F-score comparable to previous handcrafted state of the art while achieving much higher Recall rate, but also minimized human effort in constructing a list of such distinctive emotive patterns.

The outline of the paper is as follows. Firstly, we present the background for this research and define the problem in Section 2. We also present a general literature review discussing the emotive aspects of language, and describe particular previous research which try to deal with the problem of discriminating between emotive and non-emotive sentences. Section 3 describes the language combinatorics approach which we used to compare emotive and non-emotive sentences. In section 4 we describe our dataset and experiment settings. The results of the experiments as well as discussion are presented in Section 4.3. Finally the paper is concluded in Section 5 with several remarks regarding future work.

2 Background

In normal circumstances, for example, in during a natural face-to-face conversation, but also in modern means of communication, such as Internet chat, tweet exchange on Twitter, or in other online discussion environments, interlocutors are equipped with a number of tools to inform one another that they are in an emotional state. Such tools include both linguistic and paralinguistic means. Traditional linguistics distinguishes several means used particularly to express the emotional, or "emotive" meaning. These include such verbal and lexical means as exclamations [3, 14], hypocoristi-

cons (endearments) [8], vulgar language [5] or, for example in Japanese, mimetic expressions (*gitaigo*) [2]. A key role in expressing emotions is also played by the lexicon of words describing the states of emotions [12]. These might appear in sentences separately, or in combinations. However, when they appear the recipient (reader/listener) is immediately informed that the speaker/writer has produced their sentence in some kind of emotional state. What exactly the state was - might sometimes be ambiguous and context-dependent, but the fact that something emotionally loaded has been conveyed is unquestionable.

The analysis of elements of language such as intonation or tone of voice as well as nonverbal elements, like gestures or facial expressions, is also important in the task of detecting emotional load in a sentence. However, in our research we limit the realization of language (communication channel) to the transmission of lexical symbols, while all nonverbal information is represented by its textual manifestations, like exclamation marks, ellipsis, or emoticons.

The function of language gathering the knowledge about the above emotive elements is called the **emotive function of language**. It was first distinguished by Bühler in his *Sprachtheorie* [4] as one of three basic functions of language[1]. Bühler's theory was picked up further by Jakobson [7], who distinguished three other functions providing the basis for structural linguistics and communication studies. The realization of the emotive function in language enriches the uttered language with a feature called **emotiveness**. This feature was widely discussed by Stevenson (1963) [26], who defined it as a strong and persistent linguistic tendency used to inform interlocutors about the speaker's emotions and evoke corresponding emotions in those to whom the speaker's remarks are addressed. Also, Bahameed [27] argues after Shunnaq [28], that emotiveness is the speaker's emotive intention embedded in the text through specific language procedures or strategies, some of which convey neutral/objective meaning, whereas others convey emotive/subjective meaning.

To grasp the general view on how emotive meaning is realized within language, we performed a literature review on the general subject of studying emotions from the linguistic, socio-linguistic and cognitive linguistic perspective. The summary of this literature review is presented in the section 2.1.

2.1 Literature review

Research on emotions from a linguistic point of view, although still a young discipline, has already been done to some extent. For example, works of Wierzbicka [24] mark out a fresh track in research on cognitive linguistics of emotions among different cultures. Fussell [6] approached emotions from a wide cross-disciplinary perspective, trying to investigate the emotion phenomena form three broad areas: background theory of emotions, figurative language use, and social/cultural aspects of emotional communication. Weigand [23] tried to formulate a model of emotions in dialogic interactions proposing an attempt to explain emotions from the point of view of communication research. As for the Japanese language, which this research focuses on, Ptaszynski [16], made an attempt to explain both communicative and semiotic functions of emotive expressions, with a specific focus on expressions in Japanese.

Apart from research generalizing about emotions, there is also a wide range of study in the expressions of particular emotion types, or specific expressions of emotions. As for the former, a lifetime work in lexicography performed by Nakamura [12] resulted in the creation of a dictionary devoted particularly to the expressions describing states of emotions in Japanese. As for the latter, for example, Baba [2] studied Japanese mimetics in spoken discourse, Ono [14] studied emphatic functions of Japanese particle *-da*, and Sasai [20] examined *nanto*-type exclamatory sentences.

Unfortunately, there have been little linguistic research on more sophisticated emotive patterns in language. For example, a sentence "Oh, what a pleasant whether it is today, isn't it?" contains such emotive elements as interjection "oh", exclamatory sentence marker "what a", and emotive interrogative phrase "isn't it?". However, these emotive elements should rather be perceived as one pattern, like "oh, what a * isn't it?" (we discuss this pattern in more detail in section 2.3). In fact, this is one of the typical patterns of *wh*-type exclamative sentences [3]. However, although there are linguistic works investigating such emotive patterns, there has been no research experimentally confirming the existence of such patterns, or attempts to systematically and automatically extract them from larger text collections. The lack of such research is most likely caused by the limitation of typical linguistic approach in which the analysis is usually performed manually. A great help here could be offered by computer supported corpora analysis.

There has been a number of research in Computational Linguistics (CL) and Natural Language Processing (NLP) focusing on the task of recognizing whether a sentence is emotionally loaded or not. We describe them in section 2.2.

2.2 Previous research in Emotional Sentence Detection

Detecting whether sentences are loaded with emotional content has been undertaken by a number of researchers, most often as an additional task in either sentiment analysis (SA) or affect analysis (AA). SA, in great simplification, focuses on determining whether a language entity (sentence, document) was written with positive or negative attitude toward its topic. AA on the other hand focuses on specifying which exactly emotion type (joy, anger, etc.) has been conveyed. The fact, that the task was usually undertaken as a subtask, influences the way it was formulated. Below we present some of the most influential works on the topic, but formulating it in slightly different terms.

Emotional vs. Neutral: Discriminating whether a sentence is emotional or neutral is to answer the question of whether it can be interpreted as produced in an emotional state. This way the task was studied by Minato et al. (2006) [11], Aman and Szpakowicz (2007) [1] or Neviarouskaya et al. (2011) [13].

Subjective vs. Objective: Discriminating between subjective and objective sentences is to say whether the speaker presented the sentence contents from a first-person-centric perspective or from no specific perspective. The research formulating the problem this way include e.g, Wiebe et al.

[1] The other two functions being *descriptive* and *impressive*.

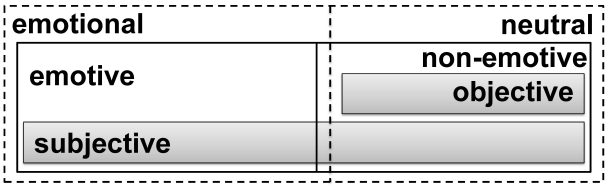

Figure 1. Comparison between different nomenclature used in sentiment analysis research.

(1999) [31], who classified subjectivity of sentences using naive Bayes classifier, or later improvement of this research by Wilson and Wiebe (2005) [25].

In other research Yu and Hatzivassiloglou (2003) [29] used supervised learning to detect subjectivity and Hatzivassiloglou and Wiebe (2000) [30] studied the effect of gradable adjectives on sentence subjectivity.

Emotive vs. Non-emotive: Saying that a sentence is emotive means to specify the linguistic features of language which where used to produce a sentence uttered with emphasis. Research that formulated and tackled the problem this way was done by, e.g., Ptaszynski et al. (2009) [17].

Each of the above nomenclature implies similar, though slightly different assumptions. For example, a sentence produced without any emotive characteristics (non-emotive) could still imply emotional state in some situations. Also Liu and Zhang (2012) [32] notice that "not all subjective sentences express opinions and those that do are a subgroup of opinionated sentences." A comparison of the scopes and overlaps of different nomenclature is represented in Figure 1. In this research we formulate the problem similarly to Ptaszynski et al. (2009) [17]. We also used their system to compare with our method.

2.3 Problem definition

The task of discriminating between emotive and non-emotive sentences could be considered as a kind of automated text classification task, which is a standard task in NLP. Some of the approaches to modeling language in text (or document) classification include Bag-of-Words (BOW) or n-gram. In the BOW model, a text or document is perceived as an unordered set of words. BOW thus disregards both grammar and word order. An approach in which word order is retained is called the n-gram approach. First proposed by Shannon [22] over half a century ago, this approach perceives a given sentence as a set of n-long ordered sub-sequences of words. This allows matching the words while retaining the sentence word order. However, the n-gram approach allows only for a simple sequence matching, while disregarding the structure of the sentence. Although instead of words one could represent a sentence with parts of speech (POS), or dependency structure, the n-gram approach still does not allow extraction or matching of more sophisticated patterns than the subsequent strings of elements. An example of such a pattern, more sophisticated than n-gram, can be explained as follows. A sentence in Japanese *Kyō wa nante kimochi ii hi nanda !* (What a pleasant day it is today!) contains a pattern *nante * nanda !*[2]. Similar cases can be easily found in other languages, for instance, in English or Spanish. An exclamative sentence "Oh, she is so pretty, isn't she?", contains a pattern "Oh * is so * isn't *?". In Columbian Spanish, sentences "*¡Qué majo está carro!*" (What a nice car!) and "*¡Que majó está chica!*" (What a nice girl!) contain a common pattern "*¡Que majó está * !*" (What a nice * !). With another sentence, like "*¡Qué porquería de película!*" (What a crappy movie!) we can obtain a higher level generalization of this pattern, namely "*¡Que * !*" (What a * !), which is a typical *wh*-exclamative sentence pattern [3, 15]. The existence of such patterns in language is common and well recognized. However, it is not possible to discover such subtle patterns using only n-gram approach.

In our research we aimed to contribute to dealing with the above problems. To do this we applied a method for language modeling and extracting from unrestricted text frequent sophisticated patterns. We also performed a text classification task with the use of such patterns. The method is based on language combinatorics (LC) idea developed by Ptaszynski et al. (2011) [19].

3 Pattern-based language modelling method

To deal with the sophisticated patterns mentioned in section 2.3 we applied a language modeling method based on the idea of language combinatorics [19]. This idea assumes that linguistic entities, such as sentences can be perceived as bundles of ordered non-repeated combinations of elements (words, punctuation marks, etc.). Furthermore, the most frequent combinations appearing in different sentences from one collection can be perceived as patterns specific to this collection.

This idea does not limit the meaning of a pattern to n-gram and assumes that sophisticated patterns with disjointed elements will provide better results than the usual bag-of-words or n-gram approach. Defining sentence patterns this way allows automatic extraction of such patterns by generating all ordered combinations of sentence elements and verifying their occurrences within a specified corpus.

Algorithms using combinatorial approach at first generate a massive number of combinations - potential answers to a given problem. This is the reason such algorithms are sometimes called brute-force search algorithms. Brute-force approach often faces the problem of exponential and rapid growth of function values during combinatorial manipulations. This phenomenon is known as combinatorial explosion [9]. Since this phenomenon often results in very long processing time, combinatorial approaches have often been disregarded. We assumed however, that combinatorial explosion can be contained on modern hardware to the extent needed in our research. Moreover, optimizing the combinatorial algorithm to the problem requirements should shorten the processing time making it advantageous in language processing task. Ptaszynski et al. [19] in their preliminary experiments verified the amount of generated patterns with comparison to n-grams, and evaluated their validity using a generic sentence pattern extraction architecture SPEC. According to the evaluation, in language processing tasks it is not necessary to generate patterns of all lengths, since the most useful ones usually appear in the group of 2 to 5 element-long patterns.

Based on the above assumptions we propose a method

[2]Equivalent of *wh*-exclamatives in English [3, 20]; asterisk "*" used as a marker of disjointed elements.

for automatic extraction of frequent sentence patterns distinguishable for a corpus, and perform a sentence classification experiment by training a classifier on the extracted patterns. Firstly, ordered non-repeated combinations are generated from all elements of a sentence. In every n-element sentence there is k-number of combination clusters, such as that $1 \leq k \leq n$, where k represents all k-element combinations being a subset of n. The number of combinations generated for one k-element group of combinations is equal to binomial coefficient, represented in equation 1. In this procedure the system creates all combinations for all values of k from the range of $\{1, ..., n\}$. Therefore the number of all combinations is equal to the sum of all combinations from all k-element combination groups, like in equation 2.

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{1}$$

$$\sum_{k=1}^{n} \binom{n}{k} = \frac{n!}{1!(n-1)!} + \frac{n!}{2!(n-2)!} + ... \\ ... + \frac{n!}{(n-1)!(n-(n-1))!} + \frac{n!}{n!(n-n)!} = 2^n - 1 \tag{2}$$

Next, all non-subsequent elements are separated with an asterisk ("*"). For all patterns generated this way their occurrences O are used to calculate their normalized weight w_j according to equation 3. In the task presented in this paper we apply the method to distinguish between sentences containing emotive patterns and sentences containing non-emotive patterns. Therefore the normalized weight w_j is calculated as a ratio of all occurrences from one corpus O_{pos} to the sum of all occurrences in both corpora $O_{pos} + O_{neg}$. The weight of each pattern is also normalized to fit in range from +1 (representing purely emotive patterns) to -1 (representing purely non-emotive patterns). The normalization is achieved by subtracting 0.5 from the initial score and multiplying this intermediate product by 2. The score of one sentence is calculated as a sum of weights of patterns found in the sentence, like in equation 4.

$$w_j = \left(\frac{O_{pos}}{O_{pos} + O_{neg}} - 0.5\right) * 2 \tag{3}$$

$$score = \sum w_j, (1 \geq w_j \geq -1) \tag{4}$$

The weight can be later modified in several ways. Two features are important in weight calculation. A pattern is the more representative for a corpus when, firstly, the longer it is (length k), and the more often it appears in the corpus (occurrence O). Thus the weight can be modified by

- awarding length,
- awarding length and occurrence.

The list of frequent patterns generated in the process of pattern generation and extraction can be also further modified. When two collections of sentences of opposite features (such as "positive vs. negative", or "emotive vs. non-emotive") are compared, a generated list will contain patterns that appear uniquely in only one of the sides (e.g. uniquely positive patterns and uniquely negative patterns) or in both (ambiguous patterns). Therefore the pattern list can be further modified by deleting

- all ambiguous patterns (which weight is not +1 or -1, but somewhere in between),
- only those ambiguous patterns which appear in the same number on both sides (later called "zero patterns", since their normalized weight is equal to 0).

Moreover, a list of patterns will contain both the sophisticated patterns (with disjointed elements) as well as more common n-grams. Therefore the experiments could be performed on either all patterns, or n-grams only.

Furthermore, if the initial collection of sentences was biased toward one of the sides (e.g., more sentences of one kind, or the sentences were longer, etc.), there will be more patterns of a certain sort. Thus to avoid bias in the results, instead of applying a rule of thumb, threshold is automatically optimized.

The above settings are automatically verified in the process of evaluation (10-fold cross validation) to choose the best model. The metrics used in evaluation are standard Precision (P), Recall (R) and balanced F-score (F). Finally, to deal with the combinatorial explosion mentioned on the beginning of this section we applied two heuristic rules. In the preliminary experiments Ptaszynski et al. [19] found out that the most valuable patterns in language usually contain no more than six elements. Therefore we limited the scope to $k \leq 6$. Thus the procedure of pattern generation will (1) generate up to 6-element patterns, or (2) terminate at the point where no more frequent patterns were found. A diagram of the whole system is represented on Figure 2.

4 Evaluation Experiment

4.1 Dataset preparation

In the experiments we used a dataset developed by Ptaszynski et al. (2009) [17] for the needs of evaluating their affect analysis system ML-Ask for Japanese language. The dataset contains 50 emotive and 41 non-emotive sentences. It was created in the following way.

Ptaszynski et al. performed an anonymous survey on thirty participants of different age and social groups. Each of them was to imagine or remember a conversation with any person or persons they know and write three sentences from that conversation: one free, one emotive, and one non-emotive. Additionally, the participants were asked to make the emotive and non-emotive sentences as close in content as possible, so the only perceivable difference was in whether a sentence was loaded with emotion or not. After that the participants also tagged the free utterances written by themselves whether or not they were emotive. Some examples from the dataset are represented in Table 1.

The system takes as an input sentences separated into elements (words, tokens, etc.). Therefore we needed to preprocess the dataset and make the sentences separable into elements. We did this in five ways to check how the preprocessing influences the results.

We used MeCab[3], a morphological analyzer for Japanese to preprocess the sentences from the dataset in the five following ways:

- **Tokenization:** All words, punctuation marks, etc. are separated by spaces.

[3] http://taku910.github.io/mecab/

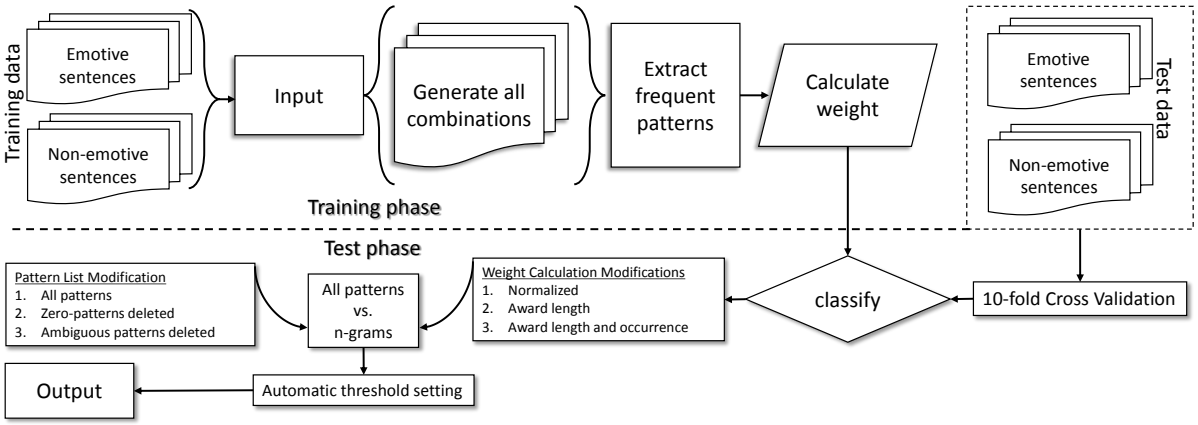

Figure 2. Diagram of the whole system.

Table 1. Some examples from the dataset representing emotive and non-emotive sentences close in content, but differing in emotional load expressed in the sentence (Romanized Japanese / Translation).

emotive	**non-emotive**
Takasugiru kara ne / 'Cause its just too expensive	*Kōgaku na tame desu.* / Due to high cost.
Un, umai, kangeki da. / Oh, so delicious, I'm impressed.	*Kono karē wa karai.* / This curry is hot.
Nanto ano hito, kekkon suru rashii yo! / / Have you heard? She's getting married!	*Ano hito ga kekkon suru rashii desu.* / / They say she is getting married.
Chō ha ga itee / Oh, how my tooth aches!	*Ha ga itai* / A tooth aches
Sugoku kirei na umi da naaa / Oh, what a beautiful sea!	*Kirei na umi desu* / This is a beautiful sea

Table 2. Five examples of preprocessing of a sentence in Japanese with and without POS tagging; N = noun, TOP = topic marker, ADV = adverbial particle, ADJ = adjective, COP = copula, INT = interjection, EXCL = exclamative mark.

Sentence example
Sentence: 今日はなんて気持ちいい日なんだ！ **Romanization:** *Kyōwanantekimochiiihinanda!* **Glosses:** Today TOP what pleasant day COP EXCL **Translation:** What a pleasant day it is today!

Preprocessing examples
1. Tokens: *Kyō wa nante kimochi ii hi nanda !* **2. POS:** `N TOP ADV N ADJ N COP EXCL` **3. Tokens+POS:** *Kyō*`[N]` *wa*`[TOP]` *nante*`[ADV]` *kimochi*`[N]` *ii*`[ADJ]` *hi*`[N]` *nanda*`[COP]` *!*`[EXCL]`

- **Lemmatization:** Like the above but the words are represented in their generic (dictionary) forms (lemmas). For example, "went" is normalized to "go", etc.
- **Parts of speech (POS):** Words are replaced with their representative parts of speech (nouns, verbs, adjectives, etc.).
- **Tokens with POS:** Both words and POS information is included in one element.
- **Lemmas with POS:** Like the above but with lemmas instead of words.

The examples of preprocessing are represented in Table 2. In theory, the more generalized a corpus is, the less unique patterns it will produce, but the produced patterns will be more frequent. This can be explained by comparing tokenized sentence with its POS representation. For example, in the sentence from Table 2 we can see that a simple phrase *kimochi ii* ("feeling good / pleasant") can be represented by a POS pattern `N ADJ`. We can easily assume that there will be more `N ADJ` patterns than *kimochi ii*, because many word combinations can be represented as `N ADJ`. Since there are more words in the dictionary than POS labels, the POS patterns will come in less variety but with higher occurrence frequency. By comparing the result of the classification using different preprocessing methods we can find out whether it is better to represent sentences as more generalized or as more specific.

4.2 Experiment setup

The preprocessed dataset provides five separate datasets for the experiment. The experiment was performed five times, once for each kind of preprocessing. For each version of the dataset a 10-fold cross validation was performed and the results were calculated using the metrics of Precision, Recall and balanced F-score, and Accuracy for the whole threshold span. There were two winning conditions. Firstly, we looked at which version of the algorithm achieves the top score within the threshold span. We checked that by looking at best F-score and Accuracy separately. However, theoretically, an algorithm could achieve its best score for only one certain threshold, while for others it could perform poorly. Therefore we also wanted to know which version of the algorithm achieves the highest score for the longest threshold span. We calculated this as a sum of scores for all thresholds. This shows whether algorithm is balanced within the threshold span. Additionally, we also checked which version obtained the highest Break-Even Point (BEP) of Precision and Recall. Finally, we checked the statistical significance of the results. We used paired Student's t-test because the classification results could represent only one of two results (emotive or non-emotive). To chose the best version of the algorithm we compared the results achieved by each group of modifications: "different pattern weight calculations", "pat-

tern list modifications" and "patterns vs n-grams". All classifier version abbreviations were listed in Appendix at the end of this paper. We also compared the performance to the state-of-the-art, namely the affect analysis system ML-Ask developed by Ptaszynski et al. (2009) [17].

4.3 Experiment results

One of the main questions when using the language combinatorics approach is whether it is even necessary to use the sophisticated patterns in classification. It could happen that it is equally effective to use the usual n-gram based approach. Moreover, if the n-gram based approach was sufficient, it would be not only equally good, but also advisable to reject the combinatorial approach, since the processing time needed to learn all patterns is incomparably longer.

Tokenized dataset

At first we checked the version of the algorithm using only tokenized sentences. The F-score results for tokenized sentences were not unequivocal. Usually for higher thresholds patterns achieved higher scores, while for lower thresholds the results were similar, or n-grams scored higher than patterns. However, in all situations where n-grams achieved visibly better results, the differences in results were not statistically significant. The highest score optimized for F-score was F = 0.754 with P = 0.605 and R = 1 for n-grams, and F = 0.739 with P = 0.599 and R = 0.963 for patterns. The algorithm usually reached its optimal F-score around 0.73–0.74. An example of F-score comparison between n-grams and patterns is represented in Figure 5. All F-scores for tokenized dataset were represented in Figure 3. The highest score optimized for Accuracy was achieved by patterns, with A = 0.649, F = 0.72, P = 0.663, an R = 0.787. All scores optimized for F-score and Accuracy were represented in Table 3.

When it comes to Precision, there always was at least one threshold for which n-grams achieved better Precision score than patterns. On the other hand, the Precision scores for patterns were more balanced, starting with a high score and slowly decreasing with the threshold span (from 1 to -1), while for n-grams, although they did achieve better results for several thresholds, they always started from a lower position and for lower thresholds more-less equaled their scores with patterns.

Recall scores were better for patterns within most of the threshold span with results equaling while the threshold decreases. However, the differences were not evident and rarely statistically significant.

Lemmatized dataset

Next, we tried a different preprocessing, namely, using sentences lemmatized (tokens converted to their original dictionary form, or *lemmas*). In theory this allows generation of a smaller number of patterns, but with higher occurrence frequency, since, e.g., different declensions of an adjective or conjugations of a verb become represented the same way.

By changing the preprocessing from tokens to lemmas the results became more straightforward. Patterns were usually better, especially for higher thresholds, while for lower thresholds the results were similar, with n-grams occasionally scoring slightly higher. The results were in most cases statistically significant ($p<0.05$), often very significant ($p<0.01$), or extremely significant ($p<0.001$). Similarly to using tokenized sentences the algorithm was optimized at around 0.73–0.74 of F-score. The highest score optimized for F-score was F = 0.744 with P = 0.666 and R = 0.843 or P = 0.646 and R = 0.877 for patterns. The highest score optimized for Accuracy was A = 0.661, with F = 7.44, P = 0.666, and R = 0.843. All scores optimized for F-score and Accuracy were represented in Table 4.

Table 3. Comparison of best F-scores and Accuracies for tokenized dataset within the threshold span for each version of the classifier. Best classifier version within each preprocessing kind - highlighted in bold type font.

Highest F-score within threshold				
	Pr	**Re**	**F1**	**Acc**
PAT-ALL	0.592	0.963	0.733	0.598
PAT-0P	0.592	0.963	0.733	0.598
PAT-AMB	0.586	0.983	0.735	0.591
PAT-LA-0P	0.592	0.963	0.733	0.598
PAT-LA	0.592	0.963	0.733	0.598
PAT-LA-AMB	0.599	0.963	0.739	0.610
NGR-ALL	0.584	1.000	0.737	0.590
NGR-0P	0.593	0.983	0.740	0.603
NGR-AMB	0.589	0.967	0.732	0.593
NGR-LA	0.603	0.963	0.742	0.612
NGR-LA-0P	**0.605**	**1.000**	**0.754**	**0.623**
NGR-LA-AMB	0.604	1.000	0.753	0.624
Highest Accuracy within threshold				
	Pr	**Re**	**F1**	**Acc**
PAT-ALL	0.633	0.820	0.715	0.627
PAT-0P	0.624	0.820	0.709	0.616
PAT-AMB	0.593	0.963	0.734	0.601
PAT-LA-0P	0.692	0.550	0.613	0.640
PAT-LA	0.642	0.820	0.720	0.637
PAT-LA-AMB	**0.663**	**0.787**	**0.720**	**0.649**
NGR-ALL	0.772	0.507	0.612	0.620
NGR-0P	0.618	0.823	0.706	0.611
NGR-AMB	0.589	0.967	0.732	0.593
NGR-LA	0.782	0.540	0.639	0.639
NGR-LA-0P	0.605	1.000	0.754	0.623
NGR-LA-AMB	0.604	1.000	0.753	0.624

An example of F-score comparison between n-grams and patterns for this dataset is represented in Figure 7.

In all versions of the algorithm for this preprocessing patterns achieved the highest Precision score in comparison with n-grams, however not on the whole threshold span. For many thresholds the results for patterns and n-grams crossed each other making the differences less significant. The highest overall Precision scores were P = 0.767 for patterns and P = 0.727 for n-grams.

When it comes to Recall, pattern-based approach achieved significantly higher Recall scores across the board for all compared cases, on nearly the whole threshold span. Thus since patterns catch much more of the data (higher Recall) while reaching highest Precision scores above n-grams, it can be said that using patterns is much more effective for this type of sentence preprocessing.

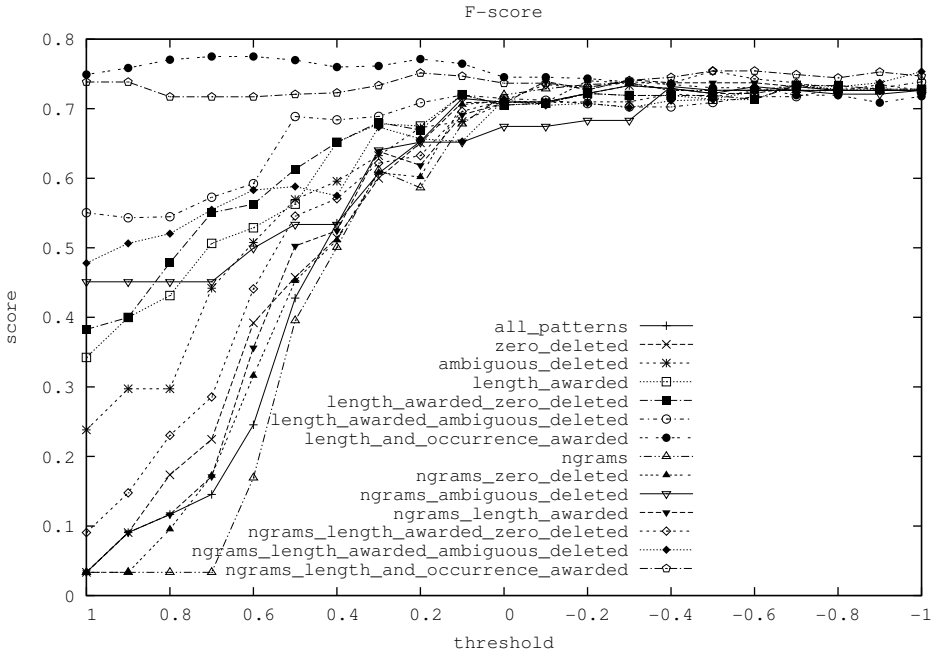

Figure 3. F-score comparison for all experiment settings for tokenized dataset.

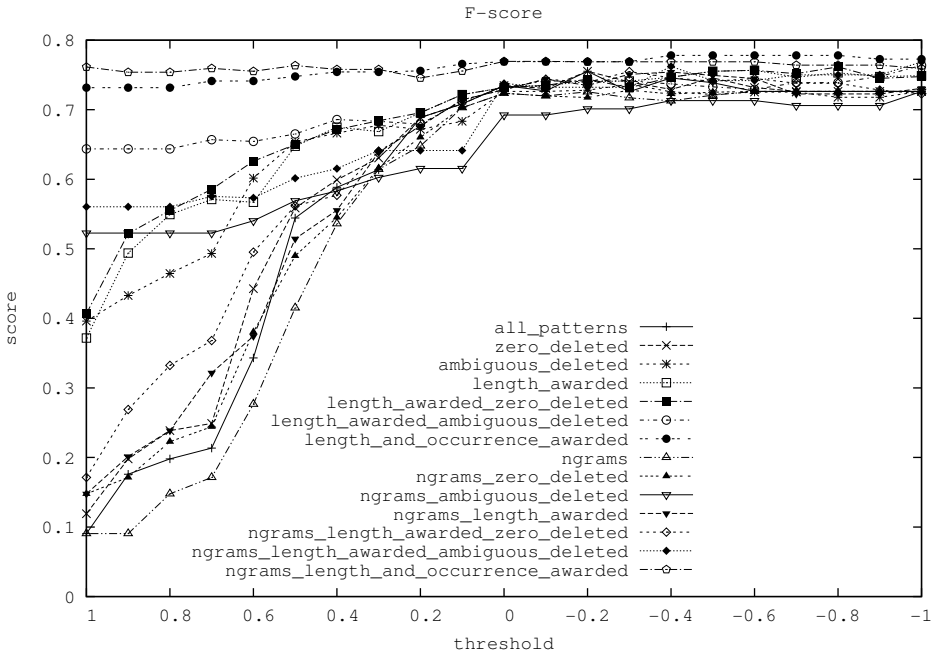

Figure 4. F-score comparison for all experiment settings for dataset with tokens and POS.

Parts-of-speech dataset

Next, we verified the performance using sentences preprocessed to represent Parts-of-speech (POS) information (nouns, verbs, etc.). In theory this type of preprocessing should provide more generalized patterns than tokens or lemmas, with smaller number of patterns but with higher occurrence frequencies.

Interestingly, F-scores for the algorithm with POS-preprocessed sentences revealed less constancy then for tokens. For most cases n-grams scored higher than patterns, but very only few results reached statistical significance. The highest F-scores were F = 0.774 with P = 0.704, and R = 0.86 for both n-grams and patterns. Similarly to tokens, the algorithm was usually optimized at F-score around 0.73–0.74. Slightly lower scores for patterns in this case could suggest that the algorithm itself works better with less abstracted, more specific preprocessing.

The highest score optimized for F-score was F = 0.774 with P = 0.704 and R = 0.86 for n-grams. This was also the highest score optimized for Accuracy. All scores optimized for F-score and Accuracy were represented in Table 5.

Results for Precision were ambiguous. For some versions of the algorithm (e.g., unmodified, zero pattern deleted) it was better for patterns, while for others (e.g., length awarded) n-grams scored higher. The highest achieved Precision for patterns was 0.723, while for n-grams 0.706.

Results for Recall confirm the results for tokens. Patterns achieved significantly higher Recall across the board.

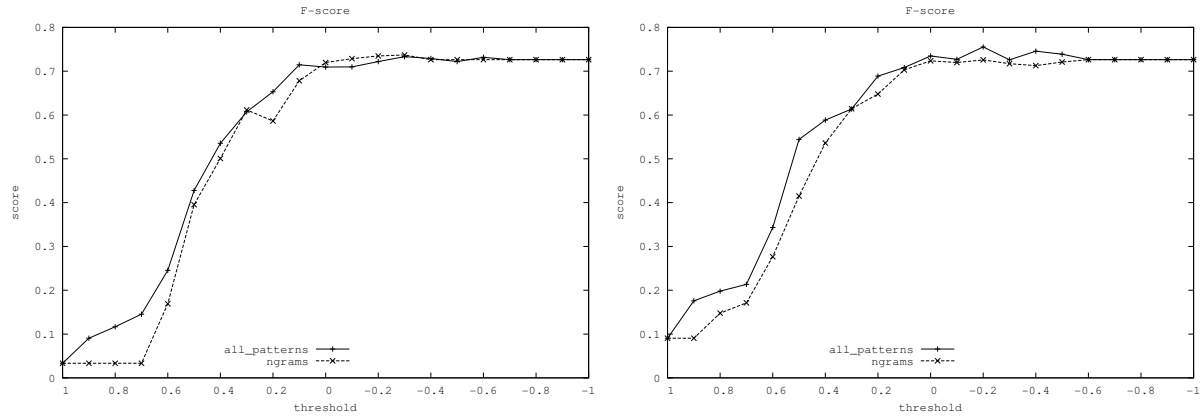

Figure 5. An example of F-score comparison between n-grams and patterns for two datasets (left: tokenized only and right: tokens with POS). Statistical significance (p-value) for both comparisons was p = 0.0209 for tokenized dataset, and p = 0.001 for dataset including tokens and POS.

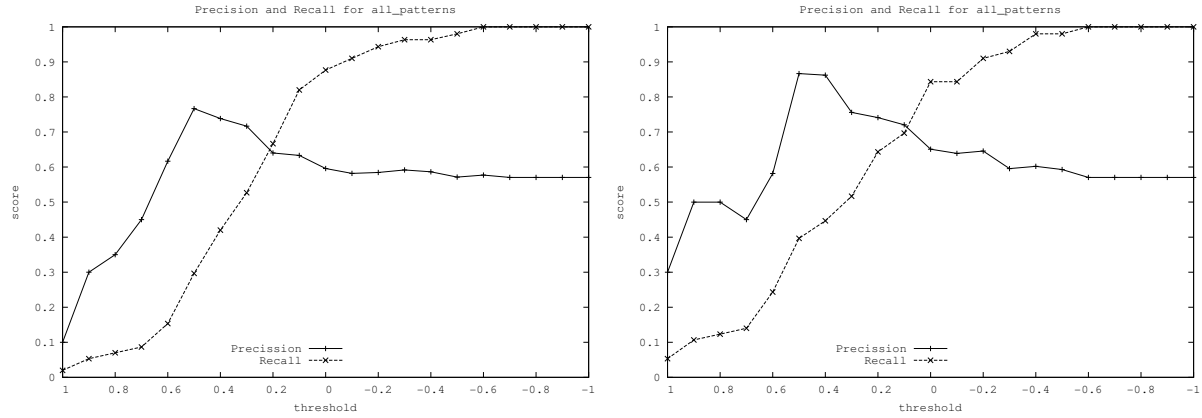

Figure 6. Precision and Recall with break-even points (BEP) for the F-score (all_patterns) for two datasets (left: tokenized and right: tokens with POS).

Tokens-POS dataset

Next we used sentences preprocessed so they included both tokens and POS information. While in the previous preprocessing the elements were more abstracted (POS), the token-POS preprocessing makes the elements more specific, thus allowing the extraction of a larger number, but less frequent patterns.

For most cases the pattern-based approach achieved significantly better results, with the difference between n-grams and patterns being in most cases very- or extremely significant (p-value <0.01 or <0.001, respectively).

The highest score optimized for F-score was F = 0.769 with P = 0.647 and R = 0.947 for n-grams and F = 0.764 with P = 0.656 and R = 0.913. The algorithm was usually reaching its optimal F-score values around 0.75–0.76. As for the scores optimized for Accuracy, the highest achieved Accuracy was 0.676 with F = 0.762, P = 0.674, and R = 0.877. All scores optimized for F-score and Accuracy were represented in Table 6.

An comparison of F-scores for all experiment settings for two datasets (tokenized and tokens with POS) are represented in Figures 3 and 4. An additional graph showing both Precision and Recall with the break-even point (BEP) for this F-score is represented in Figure 6.

The results for Precision were not as straightforward as for F-score. For many cases patterns scored higher, but not for the whole threshold span. However, the highest Precision was achieved by patterns with P = 0.867 and R = 0.397.

Recall was usually better for patterns with the scores getting closer as the threshold decreases.

Lemma-POS dataset

For version of the algorithm using sentences preprocessed so they would contain both lemmas and POS information, in almost all cases patterns reached better F-score results than n-grams for most thresholds (from 1 to -1). We noticed that patterns were especially better for higher thresholds, while for lower thresholds the results were similar, with n-grams occasionally scoring slightly higher. The results were nearly always statistically significant ($p<0.05$), often very significant ($p<0.01$), or extremely significant ($p<0.001$).

The highest score optimized for F-score was F = 0.746 with P = 0.595 and R = 1 for patterns. As for the scores optimized for Accuracy, the highest achieved Accuracy was 0.656 with F = 0.659, P = 0.745, and R = 0.59. All scores optimized for F-score and Accuracy were represented in Table 7. An example of F-score comparison between n-grams and patterns is represented in Figure 8.

When it comes to Precision alone, the results were not as straightforward as for F-score. Most often Precision for all patterns and n-grams only was similar, with n-grams often achieving their highest score slightly higher than patterns. Also for all cases of Precision comparison, pattern-based algorithm did not loose the Precision across the whole threshold span as it did for n-gram-based algorithm. This means that the pattern-based algorithm can be considered as more balanced. Furthermore, for n-grams, Precision in higher thresholds usually begins from a very low score and goes

Table 4. Comparison of best F-scores and Accuracies within the threshold span for lemmatized dataset for each version of the classifier. Best classifier version within each preprocessing kind - highlighted in bold type font.

Highest F-score within threshold	Pr	Re	F1	Acc
PAT-ALL	0.651	0.840	0.733	0.642
PAT-0P	0.651	0.840	0.733	0.642
PAT-AMB	0.579	0.983	0.729	0.581
PAT-LA-0P	0.646	0.877	0.744	0.650
PAT-LA	0.654	0.857	0.742	0.650
PAT-LA-AMB	**0.666**	**0.843**	**0.744**	**0.661**
NGR-ALL	0.622	0.893	0.734	0.621
NGR-0P	0.631	0.893	0.740	0.631
NGR-AMB	0.570	1.000	0.726	0.570
NGR-LA	0.631	0.893	0.740	0.631
NGR-LA-0P	0.589	1.000	0.742	0.601
NGR-LA-AMB	0.620	0.930	0.744	0.631
Highest Accuracy within threshold	**Pr**	**Re**	**F1**	**Acc**
PAT-ALL	0.651	0.840	0.733	0.642
PAT-0P	0.651	0.840	0.733	0.642
PAT-AMB	0.638	0.803	0.711	0.621
PAT-LA-0P	0.646	0.877	0.744	0.650
PAT-LA	0.654	0.857	0.742	0.650
PAT-LA-AMB	**0.666**	**0.843**	**0.744**	**0.661**
NGR-ALL	0.622	0.893	0.734	0.621
NGR-0P	0.631	0.893	0.740	0.631
NGR-AMB	0.632	0.673	0.652	0.571
NGR-LA	0.631	0.893	0.740	0.631
NGR-LA-0P	0.641	0.840	0.727	0.628
NGR-LA-AMB	0.620	0.930	0.744	0.631

Table 5. Comparison of best F-scores and Accuracies within the threshold span for POS-annotated dataset, for each version of the classifier. Best classifier version within each preprocessing kind - highlighted in bold type font.

Highest F-score within threshold	Pr	Re	F1	Acc
PAT-ALL	0.657	0.817	0.728	0.652
PAT-0P	0.570	1.000	0.726	0.570
PAT-AMB	0.585	0.983	0.734	0.591
PAT-LA-0P	0.570	1.000	0.726	0.570
PAT-LA	0.570	1.000	0.726	0.570
PAT-LA-AMB	0.589	1.000	0.742	0.601
NGR-ALL	0.649	0.877	0.746	0.628
NGR-0P	0.703	0.827	0.760	0.680
NGR-AMB	0.629	0.947	0.755	0.638
NGR-LA	0.683	0.877	0.768	0.669
NGR-LA-0P	0.664	0.877	0.756	0.649
NGR-LA-AMB	**0.704**	**0.860**	**0.774**	**0.686**
Highest Accuracy within threshold	**Pr**	**Re**	**F1**	**Acc**
PAT-ALL	0.657	0.817	0.728	0.652
PAT-0P	0.723	0.690	0.706	0.662
PAT-AMB	0.635	0.830	0.719	0.619
PAT-LA-0P	0.649	0.813	0.722	0.643
PAT-LA	0.681	0.760	0.718	0.653
PAT-LA-AMB	0.615	0.883	0.725	0.612
NGR-ALL	0.702	0.770	0.734	0.659
NGR-0P	0.703	0.827	0.760	0.680
NGR-AMB	0.629	0.947	0.755	0.638
NGR-LA	0.706	0.823	0.760	0.679
NGR-LA-0P	0.693	0.823	0.753	0.668
NGR-LA-AMB	**0.704**	**0.860**	**0.774**	**0.686**

gradually upward, while Precision score for patterns usually begins from a higher position.

While Precision was often comparable between patterns and n-grams, when it comes to Recall, pattern-based approach achieved significantly much higher Recall scores across the board. Thus since patterns catch much more of the data (higher Recall) while retaining the Precision, we can conclude that using patterns is much more effective. This is also reflected in the values of F-score.

4.4 Break-Even Point Analysis

The results so far indicated the following. While patterns usually achieve better scores in general, e.g., for the same classifier settings in the same threshold pattern-based classifier is usually much better. The improvement is the most visible in Recall, which influences the overall F-score. This can be especially noticed by comparing results in Figure 5, and Figures 8 and 7. However, in specific cases n-gram based classifier also scored higher. Therefore we performed an additional analysis of results, by comparing the Break-Even Points (BEP) for all classifier versions. BEP is a point where Precision and Recall cross, meaning, values of P, R as well as F-score are equal. In theory, higher BEP means the classifier is more balanced, extracts more relevant cases, and classifies them correctly. Comparison of all BEPs for all classifier versions and experiment settings is represented in Table 8.

The comparison indicated, similarly to previous analysis, that the dataset containing both tokens and POS almost for all cases performed best, achieving the highest BEP. This confirms the suggestion that the algorithm works better on more specific, less generalized features. The best BEP of all, with P=R=F=0.723 was achieved by n-gram based classifier awarding pattern length in weight calculation.

4.5 Analysis of Significance Test Results

We also compared statistical significance of differences between the results. We used Student's paired t-test, since the results could represent only one of two classes (*emotive* or *non-emotive*). All results of statistical significance tests were represented in Table 11 for F-score and Table 12 for Accuracy.

In result, especially for F-scores, differences between pattern based classifiers were almost always significant, meaning, that if an improvement appeared it was usually reliable. Differences among n-gram based classifiers were less often significant. When it comes to differences for the same parameter settings between n-gram and pattern based classifiers, the majority, but not all of the differences were statistically significant. The two classifier settings which achieved the highest BEP, namely, NGR-LA, and NGR-LA-0P for tokenPOS dataset, were significant when compared to most of remaining settings.

In general statistical significance was better for datasets with more specific features (tokePOS, lemmatized, lemma-

Table 6. Comparison of best F-scores and Accuracies within the threshold span for tokenized dataset with POS, for each version of the classifier. Best classifier version within each preprocessing kind - highlighted in bold type font.

	Pr	Re	F1	Acc
Highest F-score within threshold				
PAT-ALL	0.646	0.910	0.755	0.645
PAT-0P	0.646	0.910	0.755	0.645
PAT-AMB	0.635	0.877	0.737	0.626
PAT-LA-0P	0.636	0.950	0.762	0.647
PAT-LA	0.638	0.930	0.757	0.645
PAT-LA-AMB	0.656	0.913	0.764	0.648
NGR-ALL	0.570	1.000	0.726	0.570
NGR-0P	0.585	0.963	0.728	0.591
NGR-AMB	0.570	1.000	0.726	0.570
NGR-LA	0.620	0.963	0.754	0.634
NGR-LA-0P	0.628	0.947	0.755	0.643
NGR-LA-AMB	**0.647**	**0.947**	**0.769**	**0.658**
Highest Accuracy within threshold				
PAT-ALL	0.646	0.910	0.755	0.645
PAT-0P	0.646	0.910	0.755	0.645
PAT-AMB	0.638	0.843	0.726	0.627
PAT-LA-0P	0.723	0.720	0.722	0.656
PAT-LA	0.723	0.720	0.722	0.656
PAT-LA-AMB	0.656	0.913	0.764	0.648
NGR-ALL	0.670	0.787	0.724	0.637
NGR-0P	0.670	0.787	0.724	0.637
NGR-AMB	0.586	0.910	0.713	0.583
NGR-LA	0.666	0.843	0.744	0.657
NGR-LA-0P	0.666	0.843	0.744	0.657
NGR-LA-AMB	**0.674**	**0.877**	**0.762**	**0.676**

Table 7. Comparison of best F-scores and Accuracies within the threshold span for lemmatized dataset with POS, for each version of the classifier. Best classifier version within each preprocessing kind - highlighted in bold type font.

	Pr	Re	F1	Acc
Highest F-score within threshold				
PAT-ALL	0.592	0.963	0.733	0.598
PAT-0P	0.592	0.963	0.733	0.598
PAT-AMB	0.582	1.000	0.736	0.589
PAT-LA-0P	**0.595**	**1.000**	**0.746**	**0.610**
PAT-LA	0.589	1.000	0.742	0.601
PAT-LA-AMB	**0.595**	**1.000**	**0.746**	**0.610**
NGR-ALL	0.586	0.980	0.733	0.591
NGR-0P	0.592	0.980	0.738	0.601
NGR-AMB	0.570	1.000	0.726	0.570
NGR-LA	0.606	0.960	0.743	0.618
NGR-LA-0P	0.606	0.960	0.743	0.618
NGR-LA-AMB	0.632	0.903	0.744	0.648
Highest Accuracy within threshold				
PAT-ALL	0.843	0.487	0.617	0.641
PAT-0P	0.612	0.863	0.716	0.630
PAT-AMB	0.605	0.863	0.712	0.621
PAT-LA-0P	**0.745**	**0.590**	**0.659**	**0.656**
PAT-LA	0.628	0.830	0.715	0.640
PAT-LA-AMB	0.623	0.847	0.718	0.639
NGR-ALL	0.669	0.657	0.663	0.625
NGR-0P	0.797	0.453	0.578	0.628
NGR-AMB	0.666	0.540	0.596	0.576
NGR-LA	0.693	0.657	0.674	0.645
NGR-LA-0P	0.693	0.657	0.674	0.645
NGR-LA-AMB	0.632	0.903	0.744	0.648

POS), and worse for datasets with more general features (POS), which again confirmed that the method performs better when trained on a large number of specific features rather then on smaller number of generalized features.

4.6 Detailed analysis of learned patterns

Within some of the most frequently appearing emotive patterns there were for example: *!* (exclamation mark), *n*yo*, *cha* (emotive verb modification), *yo* (exclamative sentence ending particle), *ga*yo*, *n*!*, *n desu*, *naa* (interjection). Some examples of sentences containing those patterns are in the examples below (patterns underlined). Interestingly, most of those patterns appear in hand-crafted databases of ML-Ask developed by Ptaszynski et al. (2009) [17] (however in single word form). This suggests that it could be possible to improve ML-Ask performance by extracting additional patterns with SPEC.

Example 1. *Megane, soko ni atta nda yo.* (The glasses were over there!)

Example 2. *Uuun, butai ga mienai yo.* (Ohh, I cannot see the stage!)

Example 3. *Aaa, onaka ga suita yo.* (Ohh, I'm so hungry)

A major advantage of the proposed method over ML-Ask is the fact that it can mark both emotive and non-emotive elements in a sentence, while ML-Ask was designed to annotate only emotive elements. Some examples of extracted patterns distinguishable for non-emotive sentences were for example: *desu, wa*desu, mashi ta, masu, te*masu*. All of them are patterns described in linguistic literature as typically non-emotive, consisting in copulas (*desu*), verb endings (*masu*, and its past form *mashi ta*). Some examples of sentences containing those patterns are in the examples below (patterns underlined).

Example 4. *Kōgaku na tame desu.* (Due to high cost.)

Example 5. *Kirei na umi desu* (This is a beautiful sea)

Example 6. *Kono hon wa totemo kowai desu.* (This book is very scary.)

Table 8. Comparison of Break-Even Points (BEP) of Precision and Recall for all classifier versions and preprocessing types. Best within each preprocessing group in **bold type font**. Best within each classifier type underlined.

	tokens	POS	tokens +POS	lemmas	lemmas +POS
PAT-ALL	0.650	0.701	0.713	0.649	0.632
PAT-0P	0.637	**0.710**	0.713	0.627	0.653
PAT-AMB	0.624	0.560	0.702	**0.664**	0.637
PAT-LA-0P	0.676	–	0.722	0.626	0.659
PAT-LA	0.679	–	0.722	0.551	0.651
PAT-LA-AMB	**0.688**	–	0.716	–	–
NGR-ALL	0.594	0.695	0.712	0.629	0.665
NGR-0P	0.609	0.697	0.712	0.633	0.665
NGR-AMB	0.595	0.668	0.664	0.610	0.628
NGR-LA	0.620	0.680	**0.723**	0.635	**0.682**
NGR-LA-0P	0.633	0.680	**0.723**	0.645	**0.682**
NGR-LA-AMB	0.665	–	0.707	0.652	0.655

Example 7. *Kyo wa yuki ga futte imasu.* (It is snowing today)

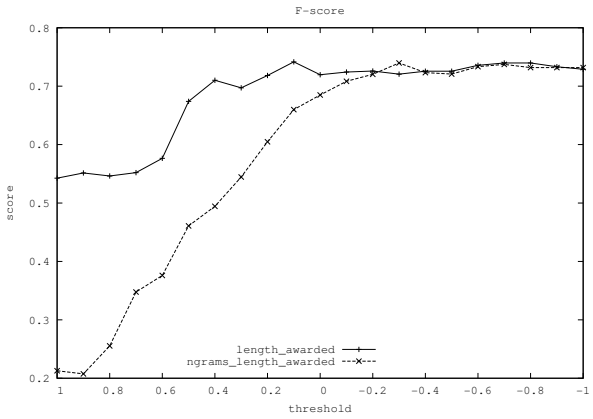

Figure 7. An example of F-score comparison between n-grams and patterns for lemmatized dataset. The version of algorithm using weight calculation modified by awarding length.

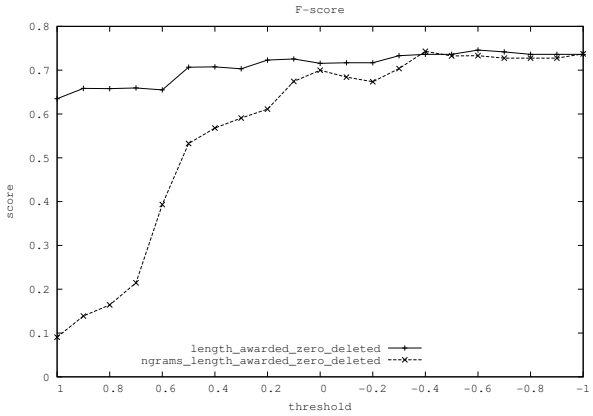

Figure 8. An example of F-score comparison between n-grams and patterns (p = 0.0021) for lemmatized dataset with parts of speech (lemmaPOS). The version of algorithm using weight calculation modified by awarding length and with zero-patterns deleted from pattern list.

4.7 Comparison to State-of-the-Art

The affect analysis system ML-Ask developed by Ptaszynski et al. [17] on the same dataset reached the following results. F-score = 0.79, Precision = 0.8 and Recall = 0.78. The results were generally comparable, however slightly higher for ML-Ask when it comes to general F-score and Precision. Recall was always better for the proposed method. However, ML-Ask is a system developed mostly manually for several years and is based specifically on linguistic knowledge concerning emotive function of language. On the other hand, the proposed method is fully automatic and does not need any particular preparations. Therefore, for example when performing similar task for other languages, rather than ML-Ask it would be more efficient to use our method, since it simply learns the patterns from data, while ML-Ask would require laborious preparation of appropriate databases.

We also compared the results of the proposed method to a standard text classification method, namely, a Support Vector Machine (SVM) classifier. The classifier was trained on Bag-of-Words language model. To verify multiple options, we checked the results for various weight settings in SVM, namely, occurrence (similar to O in the proposed method), tf (term frequency) and tf-idf (term frequency-inverse document frequency). Comparing to both the proposed method and ML-Ask system, all results of SVM classifier were much lower. The only setting for which SVM results were comparable (but still lower) to ours was for tf-idf weighting for all kinds of dataset preprocessing, and achieved 75% of F-score. The obtained results indicate that typical SVM classifier, trained on simple bag-of-words language model disregarding both grammar and word order, may not be suitable for classification of emotive language. Instead, more sophisticated patterns with disjointed elements are capable of finding subtle patterns expressed with such language.

Table 9. Best results for each version of the method compared with the ML-Ask system.

system:	ML-Ask	SPEC									
preprocessing:	–	tokenized		POS		token-POS		lemmatized		lemma-POS	
langauge model	–	n-grams	patterns	n-grams	patterns	n-grams	patterns	n-grams	patterns	n-grams	patterns
Precision	0.80	0.61	0.60	0.68	0.59	0.65	0.64	0.62	0.66	0.63	0.60
Recall	0.78	1.00	0.96	0.88	1.00	0.95	0.95	0.93	0.84	0.90	1.00
F-score	0.79	0.75	0.74	0.77	0.74	0.77	0.76	0.74	0.74	0.74	0.75

Table 10. Results of various versions of SVM classifier trained on bag-of-words language model.

preprocessing:	tokenized			POS			token-POS			lemmatized			lemma-POS		
weights:	occ.	tf	tf-idf	occ.	tf	tf-idf	occ.	tf	tf-idf	occ.	tf	tf-idf	occ.	tf	tf-idf
Precision	0.55	0.64	0.57	0.60	0.50	0.57	0.61	0.55	0.57	0.60	0.64	0.57	0.62	0.60	0.57
Recall	0.58	0.59	1.00	0.50	0.20	1.00	0.53	0.48	1.00	0.51	0.63	1.00	0.54	0.61	1.00
F-score	0.56	0.61	0.73	0.55	0.29	0.73	0.57	0.51	0.73	0.55	0.63	0.73	0.58	0.60	0.73

Table 11. T-test results (p-values) among all system results for F-score for all datasets.

		PAT-ALL	PAT-AMB	PAT-LA	PAT-LA-AMB	PAT-LA-0P	NGR	NGR-AMB	NGR-LA	NGR-LA-AMB	NGR-LA-0P	NGR-0P	PAT-0P
tokenized	PAT-ALL		.012*	.006**	.007**	.006**	.015*	.038*	.097	.014*	.007**	.869	.135
	PAT-AMB			.008**	.008**	.005**	.009**	.442	.021*	.037*	.066	.009**	.010**
	PAT-LA				.017*	.028*	.006**	.144	.011*	.237	.021*	.006**	.006**
	PAT-LA-AMB					.030*	.006**	.000***	.010*	.013*	.015*	.007**	.007**
	PAT-LA-0P						.006**	.020*	.010*	.703	.015*	.006**	.006**
	NGR							.023*	.009**	.011*	.005**	.050*	.032*
	NGR-AMB								.066	.001***	.169	.037*	.052
	NGR-LA									.022*	.007**	.006**	.862
	NGR-LA-AMB										.040*	.013*	.016*
	NGR-LA-0P											.002**	.001***
	NGR-0P												.040*
	PAT-0P												
POS	PAT-ALL		.014*	.018*	.016*	.018*	.051	.012*	.012*	.007**	.011*	.053	.301
	PAT-AMB			.048*	.037*	.047*	.005**	.357	.686	.006**	.295	.006**	.015*
	PAT-LA				.037*	.984	.006**	.048*	.070	.000***	.062	.006**	.019*
	PAT-LA-AMB					.019*	.005**	.035*	.056	.008**	.049*	.006**	.017*
	PAT-LA-0P						.006**	.048*	.068	.000***	.061	.006**	.019*
	NGR							.003**	.005**	.003**	.004**	.159	.053
	NGR-AMB								.543	.004**	.652	.003**	.013*
	NGR-LA									.009**	.405	.005**	.013*
	NGR-LA-AMB										.006**	.003**	.008**
	NGR-LA-0P											.004**	.012*
	NGR-0P												.054
	PAT-0P												
tokenPOS	PAT-ALL		.014*	.003**	.008**	.003**	.002**	.126	.080	.016*	.011*	.540	.019*
	PAT-AMB			.004**	.005**	.000***	.004**	.200	.017*	.102	.034*	.004**	.019*
	PAT-LA				.063	.038*	.001**	.004**	.003**	.981	.003**	.001***	.004**
	PAT-LA-AMB					.102	.003**	.000***	.008**	.003**	.011*	.003**	.010**
	PAT-LA-0P						.001**	.000***	.003**	.483	.003**	.001**	.004**
	NGR							.040*	.000***	.005**	.001**	.006**	.001**
	NGR-AMB								.173	.000***	.401	.072	.200
	NGR-LA									.017*	.013*	.001***	.902
	NGR-LA-AMB										.033*	.006**	.023*
	NGR-LA-0P											.000***	.020*
	NGR-0P												.003**
	PAT-0P												
lemmatized	PAT-ALL		.007**	.005**	.007**	.006**	.002**	.050	.472	.007**	.196	.003**	.014*
	PAT-AMB			.007**	.008**	.006**	.001***	.456	.001**	.022*	.002**	.001***	.007**
	PAT-LA				.017*	.015*	.001***	.000***	.001***	.802	.001**	.001***	.005**
	PAT-LA-AMB					.026*	.001**	.000***	.002**	.010**	.003**	.002**	.007**
	PAT-LA-0P						.001***	.000***	.001**	.014*	.002**	.001**	.006**
	NGR							.003**	.001**	.001***	.001***	.085	.001***
	NGR-AMB								.014*	.000***	.029*	.004**	.074
	NGR-LA									.001***	.026*	.001**	.552
	NGR-LA-AMB										.002**	.001***	.008**
	NGR-LA-0P											.001***	.891
	NGR-0P												.001***
	PAT-0P												
lemmaPOS	PAT-ALL		.028*	.022*	.019*	.020*	.006**	.802	.003**	.396	.005**	.006**	.160
	PAT-AMB			.018*	.014*	.016*	.001**	.022*	.001**	.312	.002**	.002**	.031*
	PAT-LA				.015*	.016*	.002**	.000***	.002**	.000***	.002**	.002**	.024*
	PAT-LA-AMB					.055	.002**	.000***	.002**	.000***	.003**	.003**	.020*
	PAT-LA-0P						.002**	.000***	.002**	.000***	.003**	.002**	.022*
	NGR							.059	.063	.021*	.022*	.029*	.004**
	NGR-AMB								.094	.000***	.144	.085	.987
	NGR-LA									.031*	.011*	.287	.002**
	NGR-LA-AMB										.049*	.031*	.503
	NGR-LA-0P											.043*	.002**
	NGR-0P												.004**
	PAT-0P												

$*p \leq 0.05, **p \leq 0.01, ***p \leq 0.001$

Table 12. T-test results (p-values) among all system results for Accuracy for all datasets.

		PAT-ALL	PAT-AMB	PAT-LA	PAT-LA-AMB	PAT-LA-0P	NGR	NGR-AMB	NGR-LA	NGR-LA-AMB	NGR-LA-0P	NGR-0P	PAT-0P
tokenized	PAT-ALL		.584	.001***	.001***	.002**	.026*	.562	.037*	.004**	.003**	.228	.680
	PAT-AMB			.000***	.000***	.001***	.061	.260	.300	.000***	.003**	.054	.711
	PAT-LA				.069	.256	.000***	.000***	.013*	.329	.036*	.000***	.000***
	PAT-LA-AMB					.244	.000***	.000***	.008**	.033*	.019*	.000***	.000***
	PAT-LA-0P						.001***	.000***	.014*	.160	.029*	.000***	.000***
	NGR							.713	.000***	.000***	.000***	.403	.059
	NGR-AMB								.186	.000***	.017*	.990	.384
	NGR-LA									.033*	.038*	.002**	.123
	NGR-LA-AMB										.107	.000***	.001***
	NGR-LA-0P											.000***	.000***
	NGR-0P												.019*
	PAT-0P												
POS	PAT-ALL		.022*	.076	.033*	.081	.258	.000***	.001**	.000***	.001***	.305	.792
	PAT-AMB			.243	.109	.252	.021*	.002**	.018*	.000***	.007**	.025*	.025*
	PAT-LA				.423	.723	.028*	.709	.873	.000***	.933	.035*	.075
	PAT-LA-AMB					.313	.013*	.822	.705	.000***	.759	.016*	.033*
	PAT-LA-0P						.030*	.684	.897	.000***	.958	.037*	.080
	NGR							.000***	.003**	.000***	.002**	.217	.289
	NGR-AMB								.358	.001***	.380	.000***	.000***
	NGR-LA									.000***	.587	.003**	.002**
	NGR-LA-AMB										.000***	.000***	.000***
	NGR-LA-0P											.002**	.001***
	NGR-0P												.340
	PAT-0P												
tokenPOS	PAT-ALL		.035*	.000***	.000***	.000***	.000***	.278	.010**	.000***	.000***	.079	.016*
	PAT-AMB			.000***	.000***	.000***	.000***	.004**	.937	.000***	.003**	.000***	.208
	PAT-LA				.878	.053	.000***	.000***	.000***	.299	.000***	.000***	.000***
	PAT-LA-AMB					.435	.000***	.000***	.001***	.131	.003**	.000***	.000***
	PAT-LA-0P						.000***	.000***	.000***	.098	.000***	.000***	.000***
	NGR							.386	.000***	.000***	.000***	.004**	.000***
	NGR-AMB								.035*	.000***	.001***	.819	.063
	NGR-LA									.002**	.008**	.001***	.191
	NGR-LA-AMB										.015*	.000***	.000***
	NGR-LA-0P											.000***	.000***
	NGR-0P												.002**
	PAT-0P												
lemmatized	PAT-ALL		.084	.000***	.002**	.000***	.007**	.136	.412	.000***	.080	.018*	.064
	PAT-AMB			.000***	.000***	.000***	.003**	.000***	.374	.001***	.688	.008**	.197
	PAT-LA				.052	.003**	.000***	.000***	.002**	.782	.010*	.000***	.000***
	PAT-LA-AMB					.267	.000***	.000***	.003**	.134	.008**	.001***	.002**
	PAT-LA-0P						.000***	.000***	.001***	.228	.004**	.000***	.000***
	NGR							.242	.000***	.000***	.000***	.135	.002**
	NGR-AMB								.005**	.000***	.005**	.463	.030*
	NGR-LA									.000***	.207	.000***	.864
	NGR-LA-AMB										.000***	.000***	.000***
	NGR-LA-0P											.000***	.345
	NGR-0P												.005**
	PAT-0P												
lemmaPOS	PAT-ALL		.753	.018*	.005**	.008**	.083	.069	.496	.015*	.867	.166	.811
	PAT-AMB			.004**	.001**	.002**	.032*	.010*	.366	.003**	.734	.089	.871
	PAT-LA				.031*	.002**	.003**	.000***	.037*	.718	.068	.006**	.017*
	PAT-LA-AMB					.280	.002**	.000***	.017*	.546	.033*	.004**	.005**
	PAT-LA-0P						.001**	.000***	.013*	.333	.024*	.003**	.007**
	NGR							.986	.018*	.000***	.001***	.062	.080
	NGR-AMB								.389	.000***	.198	.686	.056
	NGR-LA									.004**	.027*	.196	.462
	NGR-LA-AMB										.013*	.001***	.018*
	NGR-LA-0P											.007**	.806
	NGR-0P												.153
	PAT-0P												

$*p \leq 0.05, **p \leq 0.01, ***p \leq 0.001$

5 Conclusions and future work

We presented a method for automatic extraction of patterns from emotive sentences. We assumed emotive sentences stand out both lexically and grammatically and performed experiments to verify this assumption. In the experiments we used a set of emotive and non-emotive sentences.

The patterns extracted from those sentences were applied to recognize emotionally loaded and non-emotional sentences. We applied different preprocessing techniques (tokenization, POS, lemmatization, and combinations of those) to find the best version of the algorithm.

The algorithm reached its optimal F-score 0.75–0.76 for preprocessed sentences containing both tokens and POS information, with Precision equal to 0.64 and Recall 0.95. Precision for patterns, when compared to n-grams, was balanced, while for n-grams, although occasionally achieving high scores, the Precision was quickly decreasing. Recall scores were almost always better for patterns within most of the threshold span. By the fact that the results for sentences represented in POS were lower than the rest, we conclude that the algorithm works better with less abstracted, more specific elements.

The results of the proposed method were compared to state-of-the-art affect analysis system ML-Ask and Support Vector Machine classifier. The results for SVM were usually lower, while results of the proposed method and ML-Ask were comparable.

ML-Ask achieved better Precision, but lower Recall. However, since our method is fully automatic, it would be more efficient to use it for other languages. Moreover, many of the automatically extracted patterns appear in hand-crafted databases of ML-Ask, which suggests it could be possible to improve ML-Ask performance by extracting additional patterns automatically with our method. Moreover, the method is language independent while ML-Ask has been developed only for Japanese. In the near future we plan to perform experiments on datasets in other languages, as well as on larger datasets to analyze the scalability of the algorithm.

Acknowledgements

This research was supported by (JSPS) KAKENHI Grant-in-Aid for Encouragement of Young Scientists (B) (Project Number: 15K16044).

Appendix: List of Abbreviations

Descriptions of abbreviations used in this paper (in alphabetical order).

0P	patterns which appear with the same occurrence on both sides of data; called "zero patterns" because their weight is equal 0
ALL	all patterns or ngrams were used in classification
AMB	ambiguous patterns; refers to patterns which appear on both sides of data with different occurrence
NGR	ngrams; refers to only word ngrams extracted from sentences instead of all patterns
NGR-0P	zero-ngrams deleted
NGR-ALL	all ngrams used
NGR-AMB	ambiguous ngrams deleted
NGR-LA	ngram length awarded in weight calculation and all ngram used
NGR-LA-0P	length awarded and zero-ngrams deleted
NGR-LA-AMB	ngram length awarded and ambiguous ones deleted
PAT	patterns; refers to sophisticated patterns with disjointed elements
PAT-0P	zero-patterns deleted
PAT-ALL	all patterns used classification
PAT-AMB	ambiguous patterns deleted
PAT-LA	pattern length awarded in weight calculation and all patterns used
PAT-LA-0P	length awarded and zero-patterns deleted
PAT-LA-AMB	length awarded ambiguous patterns deleted
POS	parts-of-speech; nouns, verbs, particles, etc.

REFERENCES

[1] Saima Aman and Stan Szpakowicz. 2007. Identifying expressions of emotion in text. In *Proceedings of the 10th International Conference on Text, Speech, and Dialogue (TSD-2007)*, Lecture Notes in Computer Science (LNCS), Springer-Verlag.

[2] Junko Baba. 2003. Pragmatic function of Japanese mimetics in the spoken discourse of varying emotive intensity levels. *Journal of Pragmatics*, Vol. 35, No. 12, pp. 1861-1889, Elsevier.

[3] Fabian Beijer. 2002. The syntax and pragmatics of exclamations and other expressive/emotional utterances. *Working Papers in Linguistics 2*, The Dept. of English in Lund.

[4] Karl Bühler. 1990. *Theory of Language. Representational Function of Language*. John Benjamins Publ. (reprint from Karl Bühler. *Sprachtheorie. Die Darstellungsfunktion der Sprache*, Ullstein, Frankfurt a. M., Berlin, Wien, 1934.)

[5] David Crystal. 1989. *The Cambridge Encyclopedia of Language*. Cambridge University Press.

[6] Susan R. Fussell. 2002. *The Verbal Communication of Emotions: Interdisciplinary Perspectives*. Lawrence Erlbaum Associates.

[7] Roman Jakobson. 1960. Closing Statement: Linguistics and Poetics. *Style in Language*, pp.350-377, The MIT Press.

[8] Takashi Kamei, Rokuro Kouno and Eiichi Chino (eds.). 1996. *The Sanseido Encyclopedia of Linguistics*, Vol. VI, Sanseido.

[9] Klaus Krippendorff. 1986. Combinatorial Explosion, In: Web Dictionary of Cybernetics and Systems. Princia Cybernetica Web.

[10] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. 2002. Text classification using string kernels. *The Journal of Machine Learning Research*, Vol. 2, pp. 419-444.

[11] Junko Minato, David B. Bracewell, Fuji Ren and Shingo Kuroiwa. 2006. Statistical Analysis of a Japanese Emotion Corpus for Natural Language Processing. *LNCS* 4114, pp. 924-929.

[12] Akira Nakamura. 1993. *Kanjō hyōgen jiten* [Dictionary of Emotive Expressions] (in Japanese), Tokyodo Publishing.

[13] Alena Neviarouskaya, Helmut Prendinger and Mitsuru Ishizuka. 2011. Affect analysis model: novel rule-based approach to affect sensing from text. *Natural Language Engineering*, Vol. 17, No. 1 (2011), pp. 95-135.

[14] Hajime Ono. 2002. *An emphatic particle DA and exclamatory sentences in Japanese*. University of California, Irvine.

[15] Christopher Potts and Florian Schwarz. 2008. Exclamatives and heightened emotion: Extracting pragmatic generalizations from large corpora. Ms., UMass Amherst.

[16] Michal Ptaszynski. 2006. *Moeru gengo: Intānetto keijiban no ue no nihongo kaiwa ni okeru kanjōhyōgen no kōzō to kigōrontekikinō no bunseki – "2channeru" denshikeijiban o rei toshite –*, [Boisterous language. Analysis of structures and semiotic functions of emotive expressions in conversation on Japanese Internet bulletin board forum '2channel'] (in Japanese), M.A. Dissertation, UAM, Poznan.

[17] Michal Ptaszynski, Pawel Dybala, Rafal Rzepka and Kenji Araki. 2009. Affecting Corpora: Experiments with Automatic Affect Annotation System - A Case Study of the *2channel* Forum -, In *Proceedings of The Conference of the Pacific Association for Computational Linguistics (PACLING-09)*, pp. 223-228.

[18] M. Ptaszynski, P. Dybala, T. Matsuba, F. Masui, R. Rzepka, K. Araki and Y. Momouchi. 2010. In the Service of Online Order: Tackling Cyber-Bullying with Machine Learning and Affect Analysis. *International Journal of Computational Linguistics Research*, Vol. 1 , Issue 3, pp. 135-154.

[19] Michal Ptaszynski, Rafal Rzepka, Kenji Araki and Yoshio Momouchi. 2011. Language combinatorics: A sentence pattern extraction architecture based on combinatorial explosion. *International Journal of Computational Linguistics (IJCL)*, Vol. 2, Issue 1, pp. 24-36.

[20] Kaori Sasai. 2006. The Structure of Modern Japanese Exclamatory Sentences: On the Structure of the *Nanto*-Type Sentence. *Studies in the Japanese Language*, Vol, 2, No. 1, pp. 16-31.

[21] F. Sebastiani. 2002. Machine learning in automated text categorization. *ACM Comput. Surv.*. Vol. 34, No. 1, pp. 1-47.

[22] C. E. Shannon. 1948. A Mathematical Theory of Communication, *The Bell System Technical Journal*, Vol. 27, pp. 379-423 (623-656), 1948.

[23] Edda Weigand (ed.). 2002. *Emotion in Dialogic Interaction: Advances in the Complex*. John Benjamins.

[24] Anna Wierzbicka. 1999. *Emotions Across Languages and Cultures: Diversity and Universals*. Cambridge University Press.

[25] Theresa Wilson and Janyce Wiebe. 2005. Annotating Attributions and Private States. *Proceedings of the ACL Workshop on Frontiers in Corpus Annotation II*, pp. 53-60.

[26] Charles L. Stevenson. 1963. *Facts and Values-Studies in Ethical Analysis*. Yale University Press.

[27] Adel Salem Bahameed. 2008. Hindrances in Arabic-English Intercultural Translation. *Translation Journal*, Vol. 12, No. 1.

[28] Abdullah Shunnaq. 1993. Lexical Incongruence in Arabic-English Translation due to Emotiveness in Arabic. *Turjuman*. Vol. 2, No. 2, pp. 37-63.

[29] Hong Yu and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-2003)*, pp. 129-136, 2003.

[30] Vasileios Hatzivassiloglou and Janice Wiebe. Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings of International Conference on Computational Linguistics (COLING-2000)*, pp. 299-305, 2000.

[31] Janyce M. Wiebe, Rebecca F. Bruce and Thomas P. O'Hara. 1999. Development and use of a gold-standard data set for subjectivity classifications. In Proceedings of the Association for Computational Linguistics (ACL-1999), pp. 246-253, 1999.

[32] Bing Liu, Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. In *Mining Text Data*, pp. 415-463. Springer.