Automatic Extraction of Emotive and Non-emotive Sentence Patterns

Michal Ptaszynski †　Fumito Masui †　Rafal Rzepka ‡　Kenji Araki ‡

† Department of Computer Science, Kitami Institute of Technology
{ptaszynski,f-masui}@cs.kitami-it.ac.jp
‡ Graduate School of Information Science and Technology, Hokkaido University
{kabura,araki}@media.eng.hokudai.ac.jp

Abstract

In this research we focus on automatic extraction of patterns from emotive (emotionally loaded) sentences. We assume emotive sentences stand out both lexically and grammatically and verify this assumption experimentally by comparing two sets of such sentences. We use a novel pattern extraction method based on the idea of language combinatorics. Extracted patterns are applied in a text classification task of discriminating between emotive and non-emotive sentences. The method reached balanced F-score of 76% with Precision equal to 64% and Recall 93%.

1 Introduction

Recently the field of sentiment analysis has been attracting great interest. 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 whether a sentence is even loaded with emotional content or not. In this research we decided to tackle the problem in a standardized and systematic way. We defined emotionally loaded sentences as those which in linguistics are described as fulfilling the emotive function of language. We assumed that there are repetitive patterns which appear uniquely in emotive sentences. We performed experiments using a novel unsupervised clustering algorithm based on the idea of language combinatorics. By using this method we were also able to minimize human effort and achieve F-score comparable to the state of the art while achieving much higher Recall rate.

The outline of the paper is as follows. Firstly, we define the problem as such in section 2. 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, present the overall results of the classification experiments as well as discuss the particular patterns extracted during the experiment. Finally the paper is concluded in Section 5.

2 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 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 grammar and word order. An approach in which word order is retained is called the n-gram approach, proposed by Shannon over half a century ago [4]. This approach perceives a given sentence as a set of n-long ordered sub-sequences of words. This allows for matching the words while retaining the sentence word order. However, the n-gram approach allows only for a simple sequence matching, while disregarding the grammar 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 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 !*[1]. 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 to extract such patterns we used SPEC [2] a system for extracting from unrestricted text frequent patterns more sophisticated than n-grams and preserving the word order.

3 Language Combinatorics

SPEC, or **S**entence **P**attern **E**xtraction ar**C**hitecturte is a system that automatically extracts frequent sentence patterns distinguishable for a corpus (a collection of sentences). Firstly, the system generates ordered non-repeated combinations from all elements of a sentence. In every n-element sentence there is k-number of combination groups, such as that $1 \geq k \geq n$, 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 combinations from all k-element groups of combinations, like in equation 2.

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

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

Next, the system places a wildcard ("*") between all non-subsequent elements. SPEC uses all original patterns generated in the above procedure to extract frequent

[1] equivalent of *wh*-exclamatives in English [1, 5]; asterisk "*" used as a marker of disjoint elements

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.
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 gatting 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

patterns appearing in a given corpus and calculates their weight. The weight can be calculated in several ways. Two features are important in weight calculation. A pattern is the more representative for a corpus when, firstly, the longer the pattern is (length k), and the more often it appears in the corpus (occurrence O). Therefore the weight can be calculated by

- awarding length,
- awarding length and occurrence,
- awarding none (normalized weight).

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

- using all patterns,
- erasing all ambiguous patterns,
- erasing only those ambiguous patterns which appear in the same number in both sides (later called zero patterns).

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

- all patterns,
- only n-grams.

Finally, if the initial collection of sentences was biased toward one of the sides (e.g., more emotive sentences, or the sentences were longer, etc.), there will be more patterns of a certain sort. Thus agreeing to a rule of thumb in classification (fixed threshold above which a new sentence is classified as either emotive or non-emotive) might be harmful. Therefore assessing the threshold is another way of optimizing the classifier. All of the above mentioned modifications are automatically verified in the process of evaluation to choose the best model. The metrics used in evaluation are standard Precision, Recall and balanced F-score.

4 Evaluation Experiment

4.1 Dataset Preparation

To evaluate the method we used the dataset developed by Ptaszynski et al. [3] for the needs of evaluating their affect analysis system ML-Ask. The dataset contains 50 emotive and 41 non-emotive sentences. It was created in the following way.

The sentences were gathered through an anonymous survey in which participated thirty people of different age and social groups. Each of the participants wrote two-three sentences: one emotive, one non-emotive and one free (optional). The participants were asked to make the emotive and non-emotive sentences as close in content as possible, so the only difference was in whether a sentence was loaded with emotion or not. Some examples from the dataset are represented in Table 1.

Table 2: Three examples of sentence preprocessing.

Sentence: 今日はなんて気持ちいい日なんだ！
Translation: What a pleasant day it is today!
Preprocessing examples
1. Words: *Kyō wa nante kimochi ii hi nanda !*
2. POS: `N TOP ADV N ADJ N COP EXCL`
3.Words+POS: *Kyō*`[N]` *wa*`[TOP]` *nante*`[ADV]` *kimochi*`[N]` *ii*`[ADJ]` *hi*`[N]` *nanda*`[COP]` *!*`[EXCL]`

In our research the dataset was further preprocessed in order to verify which kind of sentence representation would give the best results. We used MeCab[2] to preprocess the sentences in the three following ways:

- **Tokenization:** All words, punctuation marks, etc. are separated by spaces.
- **Parts of speech (POS):** Words are replaced with their representative parts of speech.
- **Tokens with POS:** Both words and POS information is included in one sentence element.

The preprocessing examples are represented in Table 2.

4.2 Experiment Setting

The preprocessing provided three separate datasets for the experiment. The experiment was performed three times, one time for each kind of preprocessing to choose the winner. Each time the dataset was randomly separated into ten parts and a 10-fold cross validation was performed. The results were calculated using the metrics of Precision, Recall and balanced F-score for the whole threshold span (1, ..., -1). There were two winning conditions. Firstly, we looked at which modification of the algorithm achieves the top score within the threshold span. However, an algorithm could achieve the best score for one certain threshold, while for others it could perform poorly. Therefore we also wanted to know which version achieves the highest score for the longest threshold span. We calculated this as a sum of scores for all thresholds. This shows whether an algorithm is balanced thorough the whole threshold span. Finally, we checked the statistical significance of the results. We used paired t-test because the classification results could represent only one of two classes (emotive or non-emotive). To chose the best version of the algorithm we compared the results achieved by each modification. We also compared the performance to the state-of-the-art affect analysis system ML-Ask developed by Ptaszynski et al. [3].

[2]https://code.google.com/p/mecab/

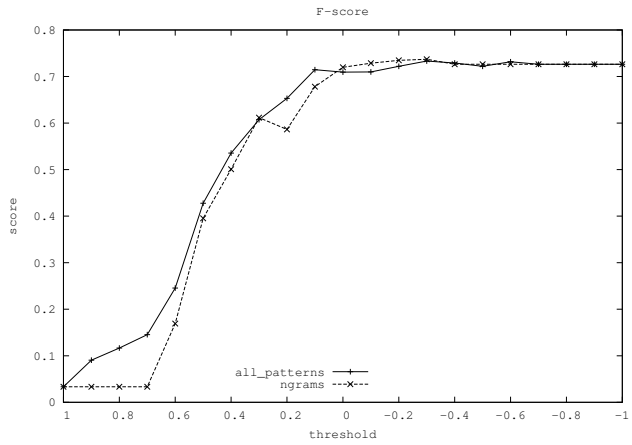

Figure 1: F-score comparison between n-grams and patterns (p = 0.0209) for tokenized dataset.

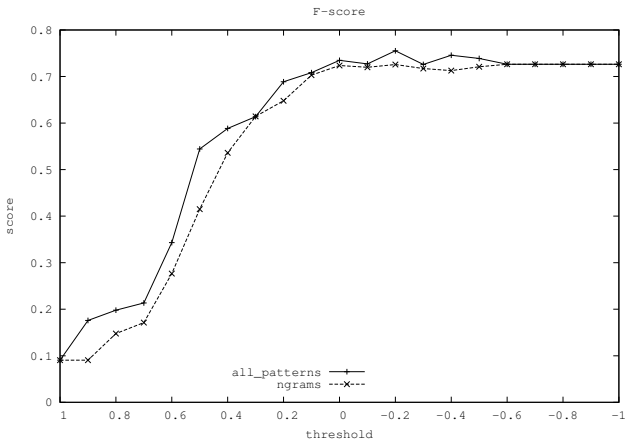

Figure 2: F-score comparison between n-grams and patterns (p = 0.001) for dataset with POS and tokens.

Table 3: Add caption

	ML-Ask	SPEC					
		tokenized		POS		token-POS	
		n-grams	patterns	n-grams	patterns	n-grams	patterns
Precision	0.80	0.61	0.6	0.68	0.59	0.65	0.64
Recall	0.78	1.00	0.96	0.88	1.00	0.95	0.95
F-score	0.79	0.75	0.74	0.77	0.74	0.77	0.76

4.3 Classification Results and Discussion

We evaluated the version of the algorithm using tokenized sentences, without any other modifications. The F-score results were not unequivocal. For higher thresholds patterns achieved higher scores, while for lower thresholds the results were similar, or n-grams scored higher than patterns. Interestingly, in all situations where n-grams achieved visibly better results, the differences in results were not statistically significant. The scores, when significant, were significant on 5% level ($p<0.05$). The highest score was F = 0.75 with P = 0.61 and R = 1 for n-grams, and F = 0.74 with P = 0.6 and R = 0.96 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 1. When it comes to Precision, there always was at least one threshold for which n-grams achieved much better Precision score than patterns. On the other hand, the Precision scores for patterns were quite 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 one or two 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 lowers. However, the differences were not evident and rarely statistically significant.

Next, we verified the performance using sentences preprocessed to represent POS information (nouns, verbs, etc.). In theory this type of preprocessing should provide more generalized patterns than tokens, with smaller number of patterns but with high occurrence frequency. However, F-scores for the algorithm with POS-preprocessed

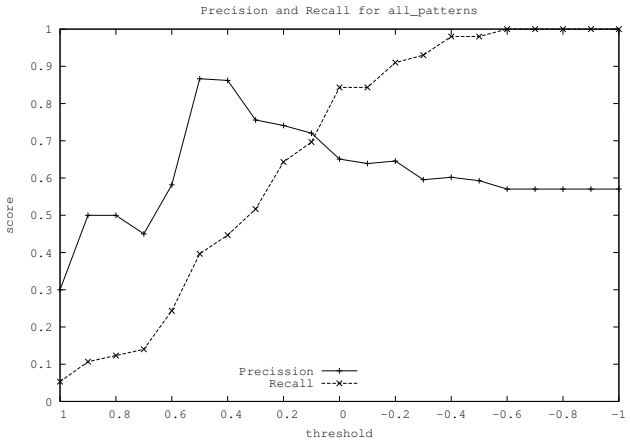

Figure 3: Both Precision and Recall with break-even point (BEP) for the F-score (all_patterns) from Figure 2.

sentences revealed less constancy then with previous tokenized sentences. For most cases n-grams scored higher than patterns, but almost none of the results reached statistical significance. The highest F-scores were F = 0.77 with P = 0.68, and R = 0.88 for n-grams, and F = 0.74 with P = 0.59 and R = 1 for patterns. Similarly to previous type of preprocessing, the algorithm was usually optimized at F-score around 0.73–0.74. Slightly lower scores for patterns in this case suggest that the algorithm itself works better with less abstracted, but more specific preprocessing. Results for Precision were also not consistent. For some versions of the algorithm (e.g., unmodified, zero pattern deletion) Precision was better for patterns, while for others (e.g., length awarded) n-grams scored higher. The highest achieved Precision for patterns was P = 0.72, while for n-grams P = 0.71. Results for Recall confirm the results for tokenized sentences, namely, patterns achieve significantly higher Recall across the board.

Next we used sentenced preprocessed so they included both tokens and POS information. While in previous types of preprocessing the elements were more abstracted (POS-only), the token-POS preprocessing makes the elements more specific, thus this preprocessing method al-

lows extracting a larger number, but less frequent patterns. For almost all cases the pattern-based approach achieved significantly better results, with the difference between n-grams and patterns being usually very- or extremely significant (p-value <0.01 or <0.001, respectively). The highest results for F-score were F = 0.76, with P = 0.64 and R = 0.95. The algorithm was usually reaching its optimal values around 0.75–0.76. An example of F-score comparison between n-grams and patterns is represented in Figure 2. An additional graph showing both Precision and Recall with the break-even point (BEP) for this F-score is represented in Figure 3. 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 = 0.87 for R = 0.50. Recall was usually better for patterns with the scores getting closer a the threshold lowers.

The affect analysis system ML-Ask developed by Ptaszynski et al. [3] 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 SPEC. However, ML-Ask is a system developed mostly by hand for several years and is based specifically on linguistic knowledge concerning emotive function of language. On the other hand, SPEC is fully automatic and does not need any particular preparations. Therefore, for example when performing similar task for other languages, it would be more efficient to use SPEC rather than ML-Ask, since SPEC simply learns the patterns from data, while ML-Ask would require laborious preparation of appropriate databases.

4.4 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 handcrafted databases of ML-Ask (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)

Another advantage of SPEC over ML-Ask is the fact that the former can mark both emotive and non-emotive elements in sentence, while ML-Ask is 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.)

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

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, token-POS) to find the best version of the algorithm. The algorithm usually reached its optimal F-score around 0.73–0.74 for tokenized sentences and 0.75–0.76 for tokens with POS information. The best results were achieved by patterns with both tokens and POS and reached balanced F-score of 76% with Precision equal to 64% and Recall 95%. Precision for patterns, when compared to n-grams, was balanced, while for n-grams, although sometimes achieving top scores higher than patterns, 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, and more specific elements. The results of SPEC and the affect analysis system ML-Ask were comparable. ML-Ask achieved better Precision, but lower Recall. However, since SPEC is a fully automatic method, it would be more efficient to use it for other languages. Moreover, many of the automatically extracted patterns appear in handcrafted databases of ML-Ask, which suggests it could be possible to improve ML-Ask performance by extracting additional patterns with SPEC.

References

[1] 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.

[2] 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.

[3] 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.

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

[5] C. Potts and F. Schwarz. 2008. Exclamatives and heightened emotion: Extracting pragmatic generalizations from large corpora. Ms., UMass Amherst.