Dictionary-Based Problem Phrase Extraction from User Reviews

Valery Solovyev[1] and Vladimir Ivanov[1,2,3]

[1] Kazan Federal University, Kremlevskaya St., 18, Kazan, Russia
http://www.kpfu.ru
[2] National University of Science and Technology "MISIS", Leninskiy Pr., 4, Moscow, Russia
http://www.misis.ru
[3] Institute of Informatics, Tatarstan Academy of Sciences, Levoboulachnaya St., 36a, Kazan, Russia

Abstract. This paper describes a system for problem phrase extraction from texts that contain users' reviews of products. In contrast to recent works, this system is based on dictionaries and heuristics, not a machine learning algorithms. We explored two approaches to dictionary construction: manual and automatic. We evaluated the system on a dataset constructed using Amazon Mechanical Turk. Performance values are compared to a machine learning baseline.

Keywords: natural language processing, information extraction

1 Introduction

User reviews are an important element of feedback to the company from its customers. Most works in this area are devoted to analysis of sentiments expressed in reviews [8] and [6]. Problem detection and extraction of problem phrases from texts are less studied. Gupta in [4] and [3] studies extraction of problem phrases with AT&T products and services from English Twitter messages. In [1] authors study Japanese texts with arbitrary problems from the Web. Both efforts try to find (i) a problem phrase and (ii) an artifact the problem is related to. Both these works make use of machine learning models trained on annotated corpora. In contrast, our system is based on dictionaries and templates constructed by experts. Our system analyzes reviews about Hewlett-Packard products from the company's website (http://reviews.shopping.hp.com). The system pays more attention to recall than to precision, because it is much more important to find every single (and potentially serious) problem.

We will detect problem phrases at the sentence-level only. Consider the following sentences extracted from user reviews that represent three possible cases. Example 1 contains a problem phrase. Example 2 mentions a problem that is implicit. Example 3 does not contain problems.

Example 1. *My printer indicates that the cartridge has a problem.*

Example 2. *Edges around the laptop are sharp in some areas.*

Example 3. *Battery life is really good.*

In theory it is very hard to formally define why Example 2 contains a problem; however, it is almost obvious for a human expert. The system was evaluated on a dataset of

P. Sojka et al. (Eds.): TSD 2014, LNAI 8655, pp. 225–232, 2014.

1,669 sentences constructed using Amazon Mechanical Turk (http://www.mturk.com/). Performance results are slightly better than a simple machine learning baseline. The main research questions we study here are the following. How does quality of problem extraction change when switching domain and extraction methods (here we compare results with [4])? How does performance of our system depend on dictionaries' sizes and construction methods? How does performance depend on difficulties with understanding an input sentence?

The main feature of problem extraction is the lack of a clear definition for a «problem». An evaluation of well-known methods for text classification is the first step we made to solve this unstudied task. The novelty of the work is in adaptation of existing approaches.

In this paper, we study a lightweight dictionary-based approach and compare it to a weak machine learning baseline. The main rationale behind this decision is the following: we show that even a simple method may lead to good results, and we suppose that better results can be reached only by much more sofisticated approaches. In a closely related work [4], Gupta had also evaluated a rule-based approach, but he applied it to another domain (telecommunication) and Twitter messages. In our work, we adopt similar ideas and apply them to reviews of consumer goods (computers), so our results are comparable to the results shown in [4]. The second reason for evaluation of the rule-based method in this paper is a clarity of reasons for a classification decision to the end user. Most machine learning approaches deal with weights and/or parameters that are hard to interpret (if something goes wrong inside the system, one can only optimize the parameters for a given data set). Thus the reason for a system's solution usualy unclear to the end user. In our future work, we will focus on machine learning approaches (including generation of feature sets from the dictionaries).

The rest of the paper is organized as follows. In Section 2 we describe dictionaries and simple heuristics for problem extraction. Section 3 represents evaluation settings and system performance. Section 4 contains the conclusion and future work.

2 Dictionaries and Heuristics for Problem Extraction

A common approach to problem phrase extraction is to use problem indicators (i.e., words or multiword expressions) that indicate a problem in a sentence. In [4] and [3], a list of about 40 indicators were collected from a few hundred tweets. In [1] a dictionary was generated automatically. Starting with one (seed) word, "trouble", authors discover nouns that represent synonyms and hyponyms of the seed word using templates X "similar" Y, X "called" Y, X "like" Y, etc. After this step, they construct templates consisting of discovered nouns and a postposition word that means "because". Our system uses five different dictionaries: ProblemWord, Action, Contradiction, NegativeWord, and Negation. Three of them (Contradiction, NegativeWord, and Negation) play auxiliary roles. The Contradiction dictionary contains words like "but", "after", "when", "despite", etc. These words indicate a contrast that may be related to a problem. The Contradiction dictionary contains 43 words. The NegativeWord dictionary contains negative-sentiment words and has been obtained from a well-known opinion lexicon [5]. The dictionary contains about 4,800 words. Finally, the Negation dictionary contains common expres-

sions for negations (e.g., "not", "n't", "cannot"). The Action dictionary contains all verbs denoting some action, because a problem phrase may be represented as a negation of an action. All verbs found in the reviews have been included in this dictionary, except auxiliary verbs ("to have", "to do", "to be", etc.). This dictionary contains about 7,600 words.

The ProblemWord dictionary includes problem indicators. Initially the dictionary had very few problem indicators, such as "problem", "error", "failure", "malfunction", "defect", "damage", "deficiency", "weakness", "mistake", "fault". We explore two approaches to extend this dictionary: manual vs. automatic. In the manual approach, we collected synonyms for problem indicators and also added to the dictionary a few multiword expressions like "shut down", "have to replace", "need to exchange", "could be", "should have", etc. We found that WordNet is only a partially useful resource in collecting synonyms, because some problem indicators are neither synonyms nor fall under the same hierarchy of WordNet. The manually created dictionary (PWM) contains about 300 terms.

An automatic population of the ProblemWord dictionary makes use of Google Books NGram Viewer dataset (http://books.google.com/ngrams/) in the following manner. Starting with an initial set of seed words (pw_i), we collect all 4-grams of two kinds: "a pw_i or X" and "X or a pw_i". Then we extract words that fill the position of X, keeping nouns and verbs only. Finally, we remove terms that occur in the Action dictionary and include remaining terms into the ProblemWord dictionary. Thus, starting with ten initial indicators, we end up with 282 terms added to the dictionary (we call this dictionary PWA).

Both dictionaries, PWM and PWA, have only 47 terms in common, but as we show further, these (core) problem indicators cover most problem phrases in a test set. Core terms for problem extraction are provided in Table 1. Seed terms for PWA construction are presented in bold.

Table 1. Core terms for problem extraction.

defect	**deficiency**	injury	issue	worry	remove	breakdown
error	disappointment	**failure**	**fault**	**problem**	lack	accident
loss	**malfunction**	flaw	absence	break	confusion	complaint
insufficiency	crack	failed	problems	refusal	puzzle	shortcoming
obstacle	**damage**	emergency	fail	gap	lesion	misfortune
inability	miss	overload	rejection	risk	trouble	disability
setback			warning	collapse	inadequacy	struggle

To extract problem phrases, we use three heuristics: PW – problem indicator presence; AC – negation of some action (i.e., a negation followed by an action word, from the Action dictionary); CN – a contradiction word (from the Contradiction dictionary) followed by a negative word (from the NegativeWord dictionary). The following texts show how these heuristics work:

PW: *"Worked fine first time around. when i turn printer off, next time i turn it on, it tells me there is a **problem** with the new print head..."*

AC: *"While plugging in USB 3 from dock station, the battery power does **not switch** over to use USB 3 dock power supply ..."*

CN: *"I recently received a shipment of this presentation paper that was not the same presentation paper that I always receive from hp (same product number, **but** suddenly an entirely different, **poorer** quality paper)..."*

Further we briefly describe an algorithm for problem phrase extraction. To decide whether a sentence (s) contains a problem or not, the algorithm exploits the following dictionaries: Action, ProblemWord, Contradiction, NegativeWord, and Negation.

Step 1. If the sentence (s) mentions a problem word (pw) from the ProblemWord dictionary, then

check whether the sentence (s) contains a negation word (neg) from the Negation dictionary. If the negation is related to the mention of the problem word, go to Step 2 (there is no explicit problem in the sentence s).

if there is no negation word related to the pw, mark the sentence s as a problem sentence and extract from its parse tree[4] a node of type S, which contains a pw and does not contain any other node of type S.

Else go to Step 2.

Step 2. If the sentence (s) contains mention of an action word (a) from the Action dictionary along with a related negation word (neg) from the Negation dictionary (i.e., the sentence contains a couple Negation+Action), then

mark the sentence s as a problem sentence and extract from its parse tree a node of type S that contains both words and does not contain any other node of type S.

Else go to Step 3.

Step 3. If the sentence (s) contains mention of a contradiction word (c) from the Contradiction dictionary followed by a negative word (nw) from the NegativeWord dictionary (i.e., the sentence contains a couple Contradiction + NegativeWord), then the sentence (s) contains a problem.

3 Evaluation of Problem Extraction

To carry out evaluation of the system, we have created a corpus consisting of 1,669 sentences. Class labels for each sentence were acquired by using Amazon Mechanical Turk's service. MTurk workers were told to assign one of three labels to each sentence: "a problem is indicated in text", "a problem is implicit", and "no problem in text". The Fleiss' kappa [2] measured after four separate runs was 0.44 (we treat each run as a pseudoexpert). After merging two labels – "a problem is indicated in text" and "a

[4] In the parsing step we use the Stanford Parser.

problem is implicit" – into a single label, the Fleiss' kappa was 0.58, which may be treated as a fair inter expert agreement. However, there are still 161 sentences with two positive and two negative labels, which we excluded from the evaluation set. The final evaluation corpus contains 1,508 sentences. The distribution of labels is presented in Table 2.

Table 2. Distribution of positive and negative labels in the evaluation corpus after four MTurk runs.

ID	Count of positive marks	Class label	Sentence count
S0	0	-	643
S1	1	-	209
S2	2	?	161
S3	3	+	258
S4	4	+	398

3.1 A Machine Learning Baseline

We implemented a simple machine learning baseline. The baseline approach to problem phrase extraction is based on a Naïve Bayes classifier from WEKA machine learning toolkit (http://www.cs.waikato.ac.nz/ml/weka/). WEKA supports textual data via its transformation to the attribute-relation file format (arrf) using a text directory loader converter. Once data is transformed and loaded into WEKA, the StringToWordVector filter converts unstructured documents into feature vectors that can be used to train a number of WEKA's classifiers. We enabled stemming while converting to vectors what resulted in 1,780 features. We then ranked these features according to chi-square test and selected the best 500 features (this number was identified experimentally). In calculating performance metrics, we use the most common measures: precision (P), recall (R), F1 measure, and common approaches to averaging, represented as follows.

$$P_i = \frac{TP_i}{TP_i + FP_i}; \quad R_i = \frac{TP_i}{TP_i + FN_i}$$

$$P_{\text{micro}} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} TP_i + FP_i}; \quad R_{\text{micro}} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} TP_i + FN_i}$$

$$P_{\text{macro}} = \frac{1}{|C|} \sum_{i=1}^{|C|} \frac{TP_i}{TP_i + FP_i}; \quad R_{\text{macro}} = \frac{1}{|C|} \sum_{i=1}^{|C|} \frac{TP_i}{TP_i + FN_i}$$

The distribution of instances in our dataset was as follows: 852 instances belong to the "no problem" class and 656 instances to the "problem" class. The simplest classifier ZeroR, which determines the most common class and gives the lower-bound performance estimates, produces the following metrics: accuracy 56.5%, weighted

precision 31.9%, weighted recall 56.5%, and weighted F1 measure 40.8%. Naïve Bayes classifier considered by us as a baseline approach to problem phrase extraction using tenfold cross validation produces the following performance metrics: accuracy 69.2%, weighted precision 69.8%, weighted recall 69.2%, and weighted F1 measure 69.3%. Baseline performance metrics without weighting (for a problem sentences class): macro precision 69%, macro recall 69.4%, and macro F1 measure 69%.

3.2 Evaluation of Dictionary-Based Extraction

We evaluated different settings of the dictionary-based system with respect to the ProblemWord dictionary contents:

PWM – ProblemWord dictionary created manually

PWA – ProblemWord dictionary generated automatically

$PWA \cap PWM$ – ProblemWord dictionary is an intersection of manual and automatic versions

$PWA \cup PWM$ – ProblemWord dictionary is a union of manual and automatic versions

The performance metrics for these settings are provided in Table 3. All metrics are calculated on a set of 1,508 sentences (S0+S1+S3+S4). The metrics for the PWM setting for each subset are presented in Table 4.

Table 3. Evaluation of problem extraction with different dictionaries of problem indicators.

Dictionary	TP	FP	P	R	F1
$PWA \cap PWM$	440	205	.68	.67	.68
PWM	521	225	.70	.79	.74
PWA	486	366	.57	.74	.65
$PWA \cup PWM$	552	382	.59	.84	.69
Naïve Bayes	453	202	.69	.69	.69
ZeroR	374	790	.32	.57	.41

Table 4. Distribution of system (based on the PWM dictionary) decisions over the evaluation corpus.

ID	Sentences	Positive labels	Negative labels
S0	643	174	469
S1	209	51	158
S2	161	92	68
S3	258	186	72
S4	398	335	63

Table 5. Analysis of problem extraction heuristics.

Heuristic name	TP fraction	FP fraction
PW	258 (49.7%)	69 (30.7%)
AC	209 (39.9%)	120 (53.3%)
CN	54 (10.4%)	36 (16%)

Finally, we evaluated influence of each heuristic on "true positives" and on "false positives" respectively. These results are presented in Table 5.

3.3 Discussion

First of all, we can see that the system divided the subset (S2) into almost equal classes, but the problem class is a little bigger, which reflects a system's trade-off between precision and recall. This point is also illustrated by a comparison of positive labels in subsets S3 and S4 and negative labels from subsets S0 and S1 (Table 4). The latter case system shows a better performance. It is interesting, that more than a half of all problems in the test set were discovered using few (core) problem indicators (Table 3). Given this and the fact that core indicators are included in an automatically constructed dictionary, one could adapt a dictionary-based system to another domains using Google Books NGram Viewer dataset. An error analysis of false negatives revealed two additional sources of problem phrases:

1. The sentence may not contain a problem indicator but may contain indirect features, like "sent it back", "to return the tablet", "contacted hp product support", etc.
2. Some multiword expressions may indicate problem, even if they do not include problem indicators (e.g., "too small", "too many", "little low").

4 Conclusion and Future Work

It is hard to define the notion of a problem. We consider problem extraction as an information extraction task. Performance metrics comparison confirms this idea. The value of the F1 measure (about 75%) is less than the F1 measure for Named Entity Recognition (more than 90%), but it is slightly better than the best F1 values for event extraction, which is about 60% [7]. Thus, problem extraction holds an intermediate position. In order to improve performance, we will develop additional dictionaries, mentioned in the previous section. This will allow to find all sentences that may potentially contain problems (even if there is no problem indicator). We suppose that improving a recall higher than 85% will be difficult (using dictionaries and our current lightweight approach). Probably, the extraction of a problem sentence not found by the dictionary approach requires deeper semantic analysis (e.g., extending a bag-of-words model and using advanced machine learning).

The performance metrics of our system are very close to those that have been shown in [4] and [3] and close to the machine learning baseline. This may indicate some limit for state-of-the-art information extraction methods. Our future work will focus

on developing interactive algorithms for lexical feature extraction (including WordNet utilization) and adaptation of the system to new domains. We also plan to extend the corpus in order to carry out accurate evaluations. In particular, we will annotate sentences from full texts, not only single sentences separated from texts.

Acknowledgments. We are grateful to Sergey Serebryakov for their support of this research, useful discussions and help with our approaches. We are grateful to reviewers for their insightful and precise comments.

References

1. De Saeger, S., Torisawa, K., Kazama J.: Looking for Trouble. In Proceedings of the 22nd International Conference on Computational Linguistics, vol. 1, pp. 185–192. Association for Computational Linguistics (2008)
2. Fleiss, J.L.: Statistical Methods for Rates and Proportions. Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics. Wiley, New York (1981)
3. Gupta, N.: Extracting Descriptions of Problems with Products and Services from Twitter Data. In Proceedings of the Third Workshop on Social Web Search and Mining. Beijing (2011)
4. Gupta, N.: Extracting Phrases Describing Problems with Products and Services from Twitter Messages. Technical report, Conference on Intelligent Text Processing and Computational Linguistics (2013)
5. Hu, M., Liu B.: Mining and Summarizing Customer Reviews. In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’04, pp. 168–177. ACM, New York (2004a)
6. Hu, M., Liu, B.: Mining Opinion Features in Customer Reviews. In Proceedings of the 19th National Conference on Artificial Intelligence (2004b)
7. Indurkhya, N., Damerau, F.J.: Handbook of Natural Language Processing, vol 2. CRC Press (2010)
8. Liu, B., Hu, M., Cheng, J.: In WWW ’05. Proceedings of the 14th International Conference on World Wide Web, pp. 342–351. ACM, New York (2005)