Biomedical Informatics Insights

ORIGINAL RESEARCH

Discovering Fine-grained Sentiment in Suicide Notes

Wenbo Wang, Lu Chen, Ming Tan, Shaojun Wang and Amit P. Sheth

Kno.e.sis Center, Wright State University, Dayton, USA. Corresponding author e-mail: wenbo@knoesis.org

Abstract: This paper presents our solution for the i2b2 sentiment classification challenge. Our hybrid system consists of machine learning and rule-based classifiers. For the machine learning classifier, we investigate a variety of lexical, syntactic and knowledge-based features, and show how much these features contribute to the performance of the classifier through experiments. For the rule-based classifier, we propose an algorithm to automatically extract effective syntactic and lexical patterns from training examples. The experimental results show that the rule-based classifier outperforms the baseline machine learning classifier using unigram features. By combining the machine learning classifier and the rule-based classifier, the hybrid system gains a better trade-off between precision and recall, and yields the highest micro-averaged F-measure (0.5038), which is better than the mean (0.4875) and median (0.5027) micro-average F-measures among all participating teams.

Keywords: sentiment analysis, emotion identification, suicide note

Biomedical Informatics Insights 2012:5 (Suppl. 1) 137–145

doi: 10.4137/BII.S8963

This article is available from http://www.la-press.com.

Introduction

Previous biomedical text mining research has mostly focused on dealing with the factual aspects in the text, such as identifying biomedical entities (e.g., gene and protein names), classifying biomedical articles based on whether the article discusses a given topic (e.g., proteinprotein interactions), and extracting relationships (e.g., gene regulatory relationships), etc. More recently, increasing attention has been paid to the analysis of sentiments of subjective biomedical text. Sentiment analysis of biomedical text (e.g., the text from patients with mental illnesses) is considered an important way to understand patients' thoughts, so as to facilitate the research and promote the treatment of the illness. The fifth i2b2 (Informatics for Integrating Biology and the Bedside) challenge announced such a task[1], which asked the participants to find fine-grained sentiments in suicide notes.

To be more specific, a collection of suicide notes was made available by the challenge organizers, and each note was manually annotated at the sentence level. The annotation schema consists of 15 categories, among which 13 categories are sentiment-related, including *abuse, anger, blame, fear, forgiveness, guilt, happiness-peacefulness, hopefulness, hopelessness, love, pride, sorrow*, and *thankfulness*, and the remaining two categories are *information* and *instructions*. Each sentence can have a single label, multiple labels, or not have any label. The participants need to classify the sentences in suicide notes according to the 15 predefined categories. Note that there are actually 16 categories, if we consider "no annotation" (ie, do not belong to any of the 15 categories) as one category.

This classification task has the following characteristics that separate it from many other similar tasks and make it more challenging: (1) the classes cover both factual (ie, information and instructions) and sentimental aspects. It separates this task from traditional topic classification that focuses on classifying text by objective topics (e.g., music vs. sports) and sentiment classification that engages in classifying text by subjective sentiment (e.g., positive vs. negative), (2) some sentences have more than one label, (3) sentences with similar content might have different labels, which suggests that it is important to capture the context of sentences for classification, and (4) the class distribution is highly imbalanced. For example, in the training data set, there are 820 sentences labeled as *instructions* and 296 sentences labeled as *love*, while only 25 sentences are in *fear* category and 9 sentences are in *abuse* category. Moreover, nearly half of the sentences belong to "no annotation" category.

Before giving an overview of our approach, we name a few relevant studies on suicide note analysis and fine-grained sentiment classification. Pestian et al.[2] utilized machine learning algorithms to differentiate genuine notes and elicited suicide notes. Pang et al.[3] classified movie reviews into multiple classes by modelling the relationships between different classes, e.g., *"one star"* is closer to *"two stars"* than to *"four stars"*. Tokuhisa et al.[4] automatically collected about 1.3 million sentences with 10 different emotions and showed that the two-step classification (positive/negative/neutral classification followed by more fine-grained emotion classification) achieved better performance than the single step classification. Similarly, Ghazi et al.[5] found that a three-level emotion classification was more effective than a single step classification given an imbalanced dataset. Yang et al.[6] automatically collected blogs with happy, joy, sad and angry emotions. They found CRF (Conditional Random Field) capable of capturing emotion transitions among sentences, thus it obtained better sentence level emotion prediction than SVM (Support Vector Machines). Wilson et al.[7] employed machine learning and a variety of features to classify phrases as positive, negative, both or neutral based on their contextual polarities. Refer to the previous paragraph for the differences between the tasks addressed in these studies and this i2b2 challenge.

In this paper, we create a hybrid system that combines both machine learning and rule-based classifiers. For the machine learning classifier, we investigate the effectiveness of different types of features for this specific classification task that covers both factual and sentimental categories. Knowledge-based and simple syntactic features that have been shown effective in many sentiment analysis studies are verified useful for this task. In addition, we find that sophisticated syntactic features (ie, sentence tense, subject, direct object, indirect object, etc.) can further improve the performance. For the rule-based classifier, we propose an algorithm for automatic construction of a pattern set with lexical and syntactic patterns extracted from training data set, and our experiments show that it

outperforms the baseline machine learning classifier using unigram features. Observing that the machine learning classifier achieves relatively high precision and low recall, in order to improve the performance, we combine it with the rule-based classifier to get a better trade-off between precision and recall in the hybrid system.

The rest of the paper is organized as follows. We first describe our approach and focus on the features used by the machine learning classifier and automatic construction of pattern set used by the rule-based classifier. Then we discuss the experiments and results for all three classifiers (ie, the machine learning classifier, the rule-based classifier and the hybrid classifier), before coming to the conclusion at last.

Approach

In this section, we first discuss the preprocessing of the input notes, then list different types of features for the machine learning classifier, and at last describe how we automatically extract patterns from the training data set and create the rule-based classifier.

The preprocessing serves two purposes: (1) to normalize the input text so that the language parser can achieve higher accuracy, and (2) to make generalization over raw text so that syntactically different but semantically similar signals can be aggregated. For (1), we correct misspellings (e.g., *buth* $\Rightarrow$ *but*), replace symbols with their formal expressions (e.g., + $\Rightarrow$ *and*) and normalize various forms of expressions into the unified form (e.g., *couldn't, couldnt* $\Rightarrow$ *could not*). For (2), we apply regular expressions to replace phrases of money (e.g., $1,000.00, $147.00), phone number (e.g., 513-636-4900, 6362051), name (e.g., John, Bill) and address (e.g., *burnet ave*) with symbols *$MONEY$*, 937-888-8888, *NAME* and *ADDRESS_SYMBOL* respectively. Take phone numbers for example, what matters is whether a phrase refers to a phone number, but not the specific digits in the number.

The machine learning classifier

SVM is an off-the-shelf supervised learning approach which has been shown to be highly effective for text classification. Its idea is to map input vectors into higher dimension space by a kernel function and then draw a separating hyperplane to maximize the margin between the plane and the nearest vectors. We use LIBSVM[8], an open source SVM implementation, which supports multi-class classification by applying a "one-against-one" approach. A variety of features used by the classifier can be divided into the following groups (also see Table 1):

- N-gram features: These are simple features based on unigrams, bigrams and trigrams. We apply MIT Java Wordnet Interface[a] for stemming. Observing that stop words might be useful for capturing the properties of some categories, we do not do any stop word removal. In addition, we require the minimum occurrence for each feature should be $\geq$3.
- Knowledge-based features: These are features based on prior knowledge about the subjectivity, sentiments or semantic categories of English words. Specifically, we use MPQA[7] and LIWC[b] (Linguistic Inquiry and Word Count). MPQA is a subjective lexicon, which provides sentiment polarities (positive/negative/neutral) and strength (strong-subj/weaksubj) of 8,211 words. For each input sentence, we count the numbers of positive, negative, neutral, strongsubj, and weaksubj words according to MPQA as features. LIWC is a text analysis program with an in-built dictionaries. For each piece of input text, it outputs a vector with 69 dimensions covering positive/negative sentiments, casual words, numbers, etc.
- Syntactic features: These are features based on syntactic information of the text, including dependency relation, POS (part-of-speech) tag and sentence tense. We first apply Stanford Parser[9] to parse each sentence and get corresponding collapsed dependencies. For each collapsed dependency *d*, we define the associated relation feature as (*d.name, d.gov, d.dep*), where *d.name* is the type of dependency, *d.gov* is the stemmed governor token of dependency *d*, and *d.dep* is the stemmed dependent token of dependency *d*. Take an artificial sentence *"Please pay them."* for example, we generate the following relation features: (*dep, please, pay*) and (*dobj, pay, them*). Moreover, considering that some types of words (e.g., adverbs, adjectives, etc) are likely to convey sentiments, we obtain POS features by counting the numbers of words with the

[a]http://projects.csail.mit.edu/jwi/.
[b]http://www.liwc.net/.

Table 1. Features used by the SVM classifier.

N-gram features	**Notation-N**
Unigram: count	N_u
Bigram: count	N_b
Trigram: count	N_t
Knowledge-based features	**Notation-K**
The numbers of strongsubj, weaksubj, positive, negative and neutral words regarding MPQA: count	K_m
feature vector generated by LIWC software	K_l
Syntactic features	**Notation-S**
Collapsed dependency relations by Stanford Parser: count	S_d
The numbers of adjectives, adverbs, nouns, pronouns, present verbs, past verbs and modals: count	S_p
The numbers of different verb tenses: count	S_t
Context features	**Notation-C**
K_m of the previous and the next sentences	C_m
S_p of the previous and the next sentences	C_p
Class-specific features (InFormation Features)	**Notation-F**
The numbers of two types of location phrases: count	F
Class-specific features (InstRuction Features)	**Notation-R**
Whether POSs of the first two words are VB/VBZ respectively: binary	R
The numbers of subjects that are the writer, other people and anything else respectively: count	
The numbers of direct objects that are the writer, other people and anything else respectively: count	
The numbers of indirect objects that are the writer, other people and anything else respectively: count	

following POS tags: adjective (JJ/JJR/JJS), adverb (RB/RBR/RBS), noun (NN/NNS/NNP/NNPS), pronoun (PRP), present verb (VB/VBG/VBP/VBZ), past verb (VBD/VBN) and modal (MD). To explore whether there are associations between different categories and sentence tenses, we use the counts of different verb tenses in each sentence as features.

- Context features: We hypothesize that the sentiments of the surrounding sentences may affect the sentiment of the current sentence. So we use MPQA feature K_m and POS count feature S_p of the previous and next sentences. If the previous or the next sentence is missing, then we set the corresponding features to be 0.

Besides the above generic features which don't focus on a specific class, we propose a few class-specific features targeting at *information* and *instruction* classes. Note that these features are sophisticated syntactic features.

- Information features: We observe that sentences indicating the location of property are more likely to be labeled as *information*. For example, *"my*/*PRP$ books*/*NNS* ***are***/*VBP up*/***RP*** *under*/***IN*** *the*/***DT*** *cash*/***NN***.*"* One feature is the frequency of the sequence that the word "is" or "are" is followed by zero or one particle (*RP*) or adverb (*RB*), a location preposition (e.g., *in*, *at*, *above*, *under*, etc.), zero or one determiner (*DT*), and a noun (*NN*/*NNS*/*NNP*/*NNPS*). Similarly, another feature is the frequency of the sequence that a noun is followed by a location preposition (*IN*), zero or one determiner (*DT*) and another noun. For example, *"$*/$* 100/*CD in*/*IN travelers*/*NNS checks*/*VBZ and*/*CC check*/*VBP book*/***NN*** *in*/***IN*** *glove*/***NN*** *compartment*/*NN ./."*
- Instruction features: We also observe that sentences that ask other people to do something or to give something to someone are usually labeled as *instructions*. To verify the observation, we sort the subject, direct object and indirect object of an action into three types: the writer himself/herself (e.g., *I, me, myself*), other people (e.g., *NNP, PRP, wife, brother*, etc.) and anything else, and count the frequency of each type as a feature. More specifically, we take the governor of nominal subject relation (*nsubj*) as the subject, the dependent of direct object relation (*dobj*) as the direct object, and the dependent of to-prepositional-modifier relation (*prep_to*) and indirect object relation (*iobj*) as the indirect object. For example, in sentence *"John J. Johnson please notify my wife at 3333 Burnet Ave. Tel."*, there are relations *nsubj(please, Johnson)* and *dobj (notify, wife)*, in which the subject of the

verb "please" is "Johnson (other people), and the object of the verb "notify" is "wife" (other people). In the sentence *"All my fortune will go to Pat Johnson."*, there exists the to-preposition-modifier relation *prep_to(go, Johnson)*, and the indirect object is "Johnson" (other people).

The rule-based classifier

Rule-based approaches are widely employed for sentiment classification. One example is the usage of sentiment bearing words, e.g., the word "excellent" suggests positive sentiment, while "nasty" indicates negative sentiment in the text. The text is classified as positive or negative depending on the sentiment bearing words it contains.

Following a similar idea, we developed a rule-based classifier which leverages lexical and syntactic patterns to classify sentences in suicide notes. Manually constructing such a set of lexical and syntactic patterns in different categories can be laborious and time-consuming, especially in this case where the patterns should be collected for 15 categories. Therefore, we propose an algorithm to automatically extract patterns from the training data set.

Let $P = \{p_1, p_2, \ldots p_n\}$ be the set of patterns, which will be used by the rule-based classifier, and $C = \{c_1, c_2, \ldots, c_{15}\}$ be the set of 15 categories. We define the *g-measure* of pattern p_i with respect to category c_j as $g(p_i, c_j)$ $(0 \leq g(p_i, c_j) \leq 1)$. Intuitively, a higher value of $g(p_i, c_j)$ indicates that the sentence containing p_i is more likely to belong to c_j. Our algorithm takes the training data set as input and outputs the pattern set P and corresponding values of *g-measure* for pattern-category pairs.

The algorithm starts with extracting candidate patterns, which include (1) n-grams up to length four (e.g., *leave my, i can not face*), (2) n consecutive POS tags up to length five (e.g., *JJ NNP NN, VBP TO VB VBN*), (3) dependency relations (e.g., *cop(cause,be)*, *nsubj(burden,i)*), and (4) generalized dependency relations (e.g., *prep_in(put,place)* → *prep_in(put,*)*, *prep_in(*,place)*). In total, there are more than 50 K candidate patterns extracted from training data. Mention that we intentionally differentiate the patterns used by the rule-based classifier and the SVM classifier. The reason is that the performance of applying all the features is worse than that of applying a carefully-selected subset of features for SVM.

After collecting all candidate patterns, the χ^2 test is applied to reduce pattern search space. The χ^2 test is a commonly used method for feature selection in machine learning. More specifically, for each category in C, we calculate χ^2 scores[10] for each candidate pattern p' as following:

$$\chi^2(p', c) = \frac{(N_a + N_b + N_c + N_d) \times (N_a N_d - N_b N_c)^2}{(N_a + N_c) \times (N_b + N_d) \times (N_a + N_b) \times (N_c + N_d)} \quad (1)$$

where c is a category ($c \in C$), N_a is the number of sentences that belong to category c and contain p', N_b is the number of sentences that contain p' but do not belong to category c, N_c is the number of sentences that belong to category c but do not contain p', and N_d is the number of sentences that do not belong to category c or contain p'. For each category, the candidate patterns are sorted in descending order according to their χ^2 scores. Overall we get 15 sorted lists (one list for each of 15 categories). We only keep the top M patterns in each list, and put these top patterns into pattern set P. Since one pattern might be included in different lists, the number of patterns in P is less than or equal to 15 * M. In this paper, we set $M = 1000$.

In the following step, the algorithm estimates the value of $g(p_i, c_j)$. As discussed before, the higher $g(p_i, c_j)$ suggests the sentence containing the pattern p_i more likely to belong to category c_j. Following this intuition, we take the conditional probability $p(c_j|p_i)$ as the *g-measure*.

$$g(p', c) = p(c \mid p') = \frac{N_a}{N_a + N_b} \quad (2)$$

We remove patterns with *g-measure* values equal to 1.0 from P. According to our observation, such patterns usually, by chance, co-occur with the same category in a few sentences, and are not strong indicators of that category. Note that we do not use χ^2 score as the *g-measure*, because it does not match our requirement of the *g-measure*. Unlike the *g-measure*, χ^2 score evaluates the usefulness of a pattern for classification. Intuitively, a pattern p gets a high χ^2 score with respect to a given category c in two cases: (a) its presence is associated with the presence of the category (ie, high N_a in Equation 1) or (b) its absence

is associated with the absence of the category (ie, high N_d in Equation 1). In our case, the value of the *g-measure* is not related to N_d.

Based on the pattern set P and the *g-measure* values, the rule-based classifier is created for the multi-class classification of the sentences in suicide notes. The general idea is that a sentence s is assigned to category c if there is a pattern p present in s and $g(p, c)$ is the highest among the values of all patterns in s with any categories. We use a threshold τ to tune the performance of the classifier. The sentence s is labeled as category c only if $g(p, c) > \tau$. Otherwise, s is not classified into any category. We investigate the effect of varying values of τ through experiments.

Experiments and Discussions

In this section, we present and discuss the experimental results. The challenge provided a total of 900 suicide notes, 600 of which were released as the training set, and the other 300 notes were used for testing. Each note had been annotated at the sentence level. As the challenge required, the classification results were evaluated using micro-averaged F-measure. We first conducted experiments using the SVM classifier and the rule-based classifier separately, and then examined the performance of the hybrid classifier created by combining both SVM and rule-based classifiers. We first trained all classifiers on the training dataset and then applied them to the testing dataset. All the results below are obtained from 300 testing suicide notes.

Evaluation of the machine learning classifier

We applied LIBSVM to do multi-class classification. We chose Radius Basis Function (RBF) as the kernel function, and we applied grid search script in LIBSVM package to find the optimal values for parameters C and γ. The main idea is to list different value combinations of C and γ, and choose the combination with highest performance. The original evaluation metrics for performance in LIBSVM had been changed from accuracy to micro averaged F-measure. Moreover, all the experiments were done using 5-fold cross validation. Our baseline method is a SVM classifier using unigrams only.

Table 2 gives the results of the SVM classifier using different feature combinations. Since there are many different features, we applied a greedy fashion approach to find an optimal feature combinations. We started with combining features in n-gram category and found the optimal n-gram feature combination. Then based on this optimal feature combination, we incorporated features from the next category, and searched for a new feature combination with a better result. We repeated the above procedures until all the feature categories had been explored. For each

Table 2: Performance of the SVM classifier with different feature combinations on the testing data.

	Feature set	Micro-averaged F-measure	Precision	Recall
N-gram feature	N_u	0.4492	0.5971	0.3601
	$N_u + N_b$	**0.4707**	0.6505	0.3687
	$N_u + N_b + N_t$	0.4542	0.6128	0.3609
Knowledge-based features	$N_u + N_b + K_m$	0.4623	0.5946	0.3781
	$N_u + N_b + K_l$	0.4650	0.6161	0.3734
	$N_u + N_b + K_m + K_l$	**0.4750**	0.6525	0.3734
Syntactic features	$N_u + N_b + K_m + K_l + S_d$	0.4781	0.6667	0.3726
	$N_u + N_b + K_m + K_l + S_p$	0.4783	0.6553	0.3766
	$N_u + N_b + K_m + K_l + S_t$	0.4798	0.6584	0.3774
	$N_u + N_b + K_m + K_l + S_t + S_p$	**0.4818**	0.6612	0.3789
	$N_u + N_b + K_m + K_l + S_t + S_p + S_d$	0.4804	0.6657	0.3758
Context features	$N_u + N_b + K_m + K_l + S_t + S_p + C_m$	0.4697	0.6218	0.3774
	$N_u + N_b + K_m + K_l + S_t + S_p + C_p$	0.4758	0.6508	0.3750
	$N_u + N_b + K_m + K_l + S_t + S_p + C_m + C_p$	0.4787	0.6593	0.3758
Class-specific features	$N_u + N_b + K_m + K_l + S_t + S_p + R$	**0.4883**	0.6667	**0.3852**
	$N_u + N_b + K_m + K_l + S_t + S_p + R + F$	**0.4878**	**0.6694**	0.3837
All	All features	0.4720	0.6279	0.3781

feature category in Table 2, we highlight the best feature combination if its performance is better than the best performance in previous feature category.

Applying selected features from n-grams, knowledge-based, syntactic and class-specific feature categories, we got the best micro averaged F-measure, the best recall and the second best precision. And the best F-measure, 0.4883, is 3.9% higher than the F-measure of the baseline and is slightly higher than the mean F-measure among all participating teams (0.4875). More specifically, we want to analyze the utility of different features. For n-gram features, the combination of unigrams and bigrams gets an F-measure of 0.4707, while adding trigrams decreases the F-measure to 0.4542. For knowledge-based features, it is interesting to see that MPQA or LIWC features alone decrease the performance, but applying both of them increases the performance by 0.43%. Among individual syntactic features, adding sentence tense features increases F-measure by 0.48%, which verifies that sentence tense features are useful for differentiating different categories. It's surprising that adding context features does not improve the result, which may suggest that it is not sufficient to capture context with only the previous and next sentences. Applying class-specific features for *instructions* improves the F-measure by 0.65%, which shows that sophisticated syntactic features like different types of subjects, direct objects and indirect objects can be effective.

Evaluation of the rule-based classifier

We studied the effect of varying values of threshold τ ($0 \leq \tau \leq 1$) to the performance of the rule-based classifier. The classifier finds a pattern p and a category c for an input sentence using the algorithm described earlier, and assigns the label of c to the sentence only if $g(p, c) > \tau$. Figures 1 and 2 show the precision-recall curve and the micro-averaged F-measure of the results, respectively.

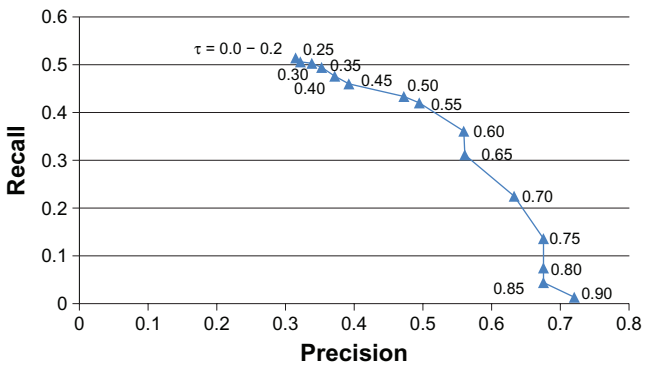

Figure 1. Precision-recall curve of the rule-based classifier with varying threshold τ on the testing data.

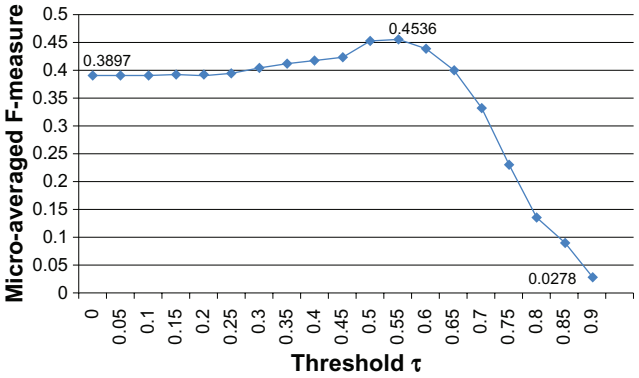

Figure 2. F-measure of the rule-based classifier with varying threshold τ on the testing data.

According to Figure 1, the precision increases and the recall decreases with the threshold τ increasing. It is because that the increased threshold leads to less patterns with higher quality being used for classification, and as a result, this raises the precision while brings down the recall. Figure 2 shows that the F-measure improves as the threshold τ increasing from 0 to 0.55, and the best F-measure 0.4536 is achieved at $\tau = 0.55$. Note that it outperforms the machine learning baseline (0.4492). However, when we keep increasing the threshold, the F-measure goes down. It can be explained by the precision-recall curve in Figure 1, from which we can see that the precision rises faster than recall falls until the threshold reaches 0.55, and after that recall decreases faster than precision increases. Note that when τ is decreased to 0, the classifier still achieves the F-measure as 0.3897, which verifies that χ^2 test is effective to select patterns.

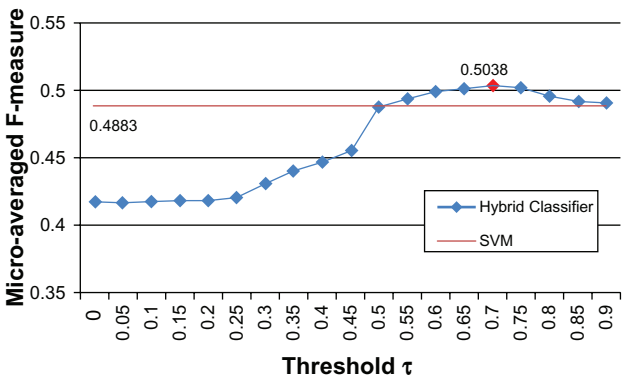

Figure 3. F-measure of the combined classifier on the test data.

Evaluation of the hybrid classifier

A hybrid classifier was created by combining the SVM classifier and the rule-based classifier. Since SVM classifier exhibited the property of relatively high precision and low recall, we considered using the rule-based classifier to improve the recall. Following this idea, we applied a simple combination algorithm.

Each sentence was fed to both SVM classifier and rule-based classifier to get the judgements respectively. If a sentence is assigned the label of any of the 15 categories by the SVM classifier, we keep the label, otherwise, we accept the label given by the rule-based classifier. For example, a sentence s_1 is labeled as "love" by the SVM classifier, as a result, no matter what the label is given by the rule-based classifier, s_1 is classified into category "love". A sentence s_2 is not classified into any of the 15 categories according to the SVM classifier, but it is labeled as "guilt" by the rule-based classifier, consequently, the final label of s_2 is "guilt".

We combined the SVM classifier that got the best result in the previous experiments with different rule-based classifiers tuned by the threshold τ. Figure 3 shows the results in terms of the micro-averaged F-measure. Observing the figure, we can see that the hybrid classifier outperforms the SVM classifier, as with the rule-based classifier with $\tau \geq 0.55$. The best F-measure achieved by the hybrid classifier is 0.5038 at the point $\tau = 0.7$, which is 1.55% higher than the best F-measure achieved by the SVM classifier and is slightly higher than the median F-measure among all the participating teams (0.5027). Our best F-measure is less than the best F-measure among all the teams (0.6139) and part of the reason is that we assigned maximumly one label to every sentence, while more than 10% of the labeled sentences are supposed to have more than one label.

Conclusion

In this paper, we presented our approach for the i2b2 Challenge of fine-grained sentiment classification of suicide note. We developed a hybrid system by combining both a machine learning classifier and a rule-based classifier for this task. For the machine learning classifier, we focused on examining the effectiveness of different types of features. Our experiments showed how much the various features contributed to the performance of the classifier. For the rule-based classifier, we proposed a method for creating the pattern set automatically, and the performance of the classifier could be tuned up by a threshold τ. The hybrid classifier was built by combining the machine learning classifier and the rule-based classifier in the way that it could get better trade-off between precision and recall. The experiments demonstrated that the hybrid classifier could achieve improved performance when we set appropriate values of the threshold τ of the rule-based classifier. We assigned maximumly one label to each sentence, but more than 10% of all the labeled sentences should have 2 or more labels. We achieved a best F-measure of 0.5038 by the hybrid classifier and it is higher than both the mean (0.4875) and median (0.5027) F-measures among all the participating teams. As the next step, we plan to strengthen the classifiers to allow them assigning multiple labels to one sentence and we believe it will improve the performance. In addition, we will work on the more efficient learning strategy for dealing with the imbalanced data sets.

Disclosures

Author(s) have provided signed confirmations to the publisher of their compliance with all applicable legal and ethical obligations in respect to declaration of conflicts of interest, funding, authorship and contributorship, and compliance with ethical requirements in respect to treatment of human and animal test subjects. If this article contains identifiable human subject(s) author(s) were required to supply signed patient consent prior to publication. Author(s) have confirmed that the published article is unique and not under consideration nor published by any other publication and that they have consent to reproduce any copyrighted material. The peer reviewers declared no conflicts of interest.

Acknowledgements

This work was supported by NSF IIS-1111182: SoCS: Collaborative Research: Social Media Enhanced Organizational Sensemaking in Emergency Response.

References

1. Pestian JP, Matykiewicz P, Linn-Gust M, Wiebe J, Bretonnel Cohen K, Brew C, et al. Sentiment analysis of suicide notes: A shared task. *Biomedical Informatics Insights*. 2012;5 (Suppl. 1):3–16.
2. Pestian John, Nasrallah Henry, Matykiewicz Pawel, Bennett Aurora, Leenaars Antoon. Suicide Note Classification Using Natural Language Processing: A Content Analysis. *Biomedical Informatics Insights*, August 2010, (3):19–28. doi: 10.4137/BII.S4706.

3. Pang Bo, Lee Lillian. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the* 43*rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 115–124, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.
4. Tokuhisa R, Inui K, Matsumoto Y. Emotion classification using massive examples extracted from the web. In *Proceedings of the* 22*nd International Conference on Computational Linguistics-Volume 1*, pages 881–888. Association for Computational Linguistics, 2008.
5. Ghazi Diman, Inkpen Diana, Szpakowicz Stan. Hierarchical versus flat classification of emotions in text. In: *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, CAAGET '10, pages 140–146, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1860631.1860648.
6. Yang Changhua, Lin Kevin Hsin-Yih, Chen Hsin-Hsi. Emotion classification using web blog corpora. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, WI '07, pages 275–278, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-3026-5. doi: http://dx.doi.org/10.1109/WI.2007.50. URL http://dx.doi.org/10.1109/WI.2007.50.
7. Wilson Theresa, Wiebe Janyce, Hoffmann Paul. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 347–354, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.
8. Chang Chih-Chung, Lin Chih-Jen. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.
9. Klein Dan, Manning Christopher D. Accurate unlexicalized parsing. In *Proceedings of the* 41*st Annual Meeting on Association for Computational Linguistics—Volume* 1, ACL '03, pages 423–430, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.
10. Yang Yiming, Pedersen Jan O. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN 1-55860-486-3.