Entity-Level Sentiment Analysis of Issue Comments

Jin Ding[*][†], Hailong Sun[*][†], Xu Wang[*][†], Xudong Liu[*][†]
SKLSDE Lab, School of Computer Science and Engineering, Beihang University[*]
Beijing Advanced Innovation Center for Big Data and Brain Computing[†]
Beijing, China
{dingjin,sunhl,wangxu,liuxd}@act.buaa.edu.cn

ABSTRACT

Emotions and sentiment of software developers can largely influence the software productivity and quality. However, existing work on emotion mining and sentiment analysis is still in the early stage in software engineering in terms of accuracy, the size of datasets used and the specificity of the analysis. In this work, we are concerned with conducting entity-level sentiment analysis. We first build a manually labeled dataset containing 3,000 issue comments selected from 231,732 issue comments collected from 10 open source projects in GitHub. Then we design and develop SentiSW, an entity-level sentiment analysis tool consisting of sentiment classification and entity recognition, which can classify issue comments into <*sentiment*, *entity*> tuples. We evaluate the sentiment classification using ten-fold cross validation, and it achieves 68.71% mean precision, 63.98% mean recall and 77.19% accuracy , which is significantly higher than existing tools. We evaluate the entity recognition by manually annotation and it achieves a 75.15% accuracy.

CCS CONCEPTS

• **Information systems → Sentiment analysis**; • **Software and its engineering → Open source model**;

KEYWORDS

entity-level sentiment analysis, open source software project, sentiment classification, entity recognition

ACM Reference Format:
Jin Ding[1][2], Hailong Sun[1][2], Xu Wang[1][2], Xudong Liu[1][2]. 2018. Entity-Level Sentiment Analysis of Issue Comments. In *SEmotion'18: SEmotion'18:IEEE/ACM 3rd International Workshop on Emotion Awareness in Software Engineering , June 2, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3194932.3194935

1 INTRODUCTION

Emotions are the results of complex mental activities and can greatly influence people's decision making and cooperation with others. And emotions generally imply people's sentiment that is usually classified into *positive*, *negative* and *neutral*. Such sentiment can affect task quality, productivity, creativity, group rapport, user focus, and job satisfaction [7] especially in the contexts of team work. As for software development, it relies heavily on human cooperation and collaboration, hence understanding the sentiment expressed in software development process and its key factors can help improve individual developers' performance at work and enhance teamwork as well. Essentially considering developers' sentiment will result in the improvement of overall software development productivity and software quality.

However, for software development involving large-scale developers, it is nearly impossible to learn each other's sentiment directly. For instance, open source software (OSS) communities are often confronted with such situation. OSS developers working in different locations commonly conduct development over Internet, and they usually do not know each other in real-world and have no physical contacts. In practice, OSS developers mainly communicate via texts through mailing lists, issue tracking systems, forums and source code management systems. Prior studies have found that developers express sentiment in texts generated in the process of various OSS activities, such as issue comments [4, 11, 22, 23, 25], forum posts [12, 24], emails [10, 31] and code review comments [11].

Sentiment analysis can be performed automatically through analyzing text information. Nonetheless, the texts in software development differ from customer reviews [13] or twitter comments [27] in terms of code snippets, terminologies, emojis. As a result, general tools generated from other domain like SentiStrength [32] and NLTK [3] have been proved to perform poorly in software engineering domain due to lack of domain specific information.

There have been some efforts [1, 5, 14] on sentiment analysis in software engineering. SentiCR [1], a sentiment analysis tool designed for code review comments in software engineering domain, is built through machine learning on the basis of a human labelled dataset. And SentiStrength-SE [14] adapts SentiStrength [32] to SE domain, but it is based on a fixed size of sentiment dictionary, thus its accuracy is largely limited by the quality and size of the dictionary. Senti4SD[5], trained on Stack Overflow posts, takes word embedding into account. In summary, most of existing work merely classify sentiment into two polars, i.e. being positive or negative. However, for a lot of circumstances, it is also important to know the targets toward which developers express their sentiment[9]. For instance, to know the exact developer whom developers show most gratitude toward may help understanding the role of that developer in a OSS project.

Thus in this work, we are concerned with developing SentiSW, an entity-level sentiment analysis tool for issue comments consisting of sentiment classification and entity recognition. SentiSW aims at classifying issue comments into three categories including *negative*, *positive* and *neutral* ones, and recognizing the entity of the subjective comments. In the end, a <*sentiment*, *entity*> tuple is generated for each issue comment, where entity is either 'Person' or 'Project'. To achieve the above goal, we have built a sentiment dataset by manually annotating 3,000 issue comments selected from 231,732 issue comments from 10 popular GitHub projects. To evaluate SentiSW, we use 10-fold cross validation technique for sentiment classification and receive a 68.71% average precision, 63.98% average recall and 77.19% accuracy, which is better than SentiStrength-SE [14]. To evaluate entity recognition, we manually labeled 660 subjective issue comments by 'Person' or 'Project' entity label, and it reaches an overall 76.58% precision, 88.73% recall and 75.15% accuracy.

In general, the major contributions of this work are as follows:

- We build the ever largest human labeled sentiment dataset for issue comments through a specifically designed labeling tool, which is shared publicly in GitHub [1] in the hope of supporting more research work on sentiment analysis in software engineering.
- We design an entity-level sentiment classification tool for issue comments with supervised machine learning algorithms, which not only performs better than the state-of-the-art methods but also can identify the targets which sentiment express toward.
- We have conducted an extensive set of experiments to evaluate the performance of our sentiment analysis tool and the results show the superiority in comparison with existing work.

The remainder of the paper is organized as follows. We present the details of our dataset and the design of our tool in Section 2. Section 3 shows the experimental results. We analyze the threats to validity of our work in Section 4. And Section 5 summarizes the related work and we conclude this paper in Section 6.

2 SYSTEM DESIGN

Due to the limitations of existing SE sentiment analysis tool, we develop Senti SW, an entity-level sentiment analysis tool for issue comments. Figure 1 shows the architecture of SentiSW. The input of SentiSW is the manually annotated issue comments gathered from GitHub, and the output is either 'neutral sentiment' when the text is objective or a <*sentiment*, *entity*> tuple when the text is subjective, and the entity we recognize here contains only 'Person' or 'Project'. SentiSW basically contains 4 modules: preprocessing, feature vectorization, classifier and entity recognition. The preprocessing module aims to remove the useless features and reduce the noise by words removal, words replacing and stemming techniques. The vectorization module is in an attempt to transfer bag of words into vectors by TF-IDF and Doc2vec. The classifier module classifies issue comments into positive, negative and neutral ones. The recognition module takes the subjective sentiment sentences generated from the classifier and recognizes the entity of the sentiment as 'Person' or 'Project'. The general workflow goes in the order of preprocessing, feature vectorization, classifying and at last entity recognition.

[1]https://github.com/Jasmine-DJ-420/SentiSW

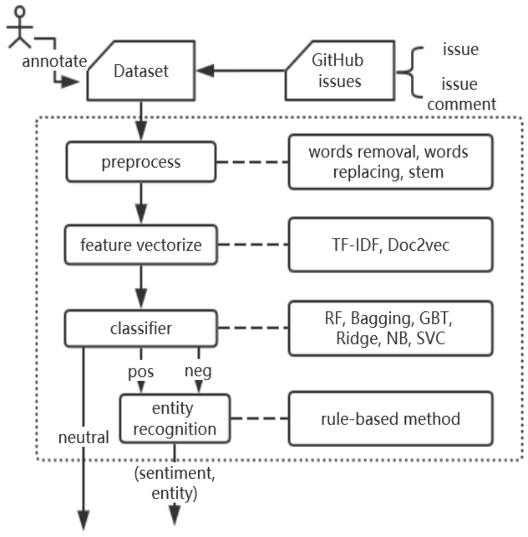
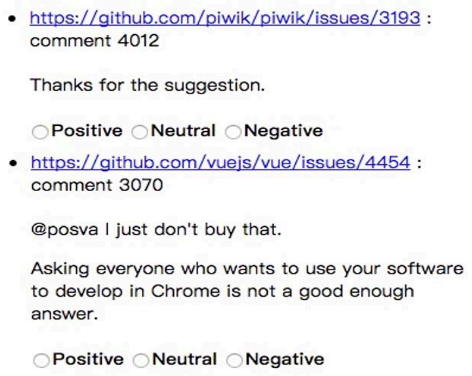


Figure 1: Architecture of SentiSW.



- https://github.com/piwik/piwik/issues/3193 : comment 4012

  Thanks for the suggestion.

  ○Positive ○Neutral ○Negative
- https://github.com/vuejs/vue/issues/4454 : comment 3070

  @posva I just don't buy that.

  Asking everyone who wants to use your software to develop in Chrome is not a good enough answer.

  ○Positive ○Neutral ○Negative

Figure 2: Annotation web service screencut.

2.1 Dataset Generation

GitHub is an open source software community that offers code hosting service and GitHub issue tracker. GitHub stores tens of millions of developers' information and collaboration information, thus we built an issue comments dataset from GitHub issue tracker through GitHub API. We mined 10 popular repositories over 10 different main languages from GitHub, meeting the following two conditions: **i)** each project contains at least 5,000 stars which reflects popularity; and **ii)** each project contains at least 5,000 issue comments which means that the project has an active GitHub issue tracker. We chose projects over different language to get a relatively wide range of vocabulary, in the end, we obtained a dataset containing a total number of 231,732 issue comments.

We randomly selected 300 issue comments from each project, and obtained a set of 3,000 comments for annotating. Each comment was annotated by at least 2 annotators independently, which means annotators do not discuss a criteria before labeling. For the

Table 1: Comparison between datasets.

Dataset	Words	Unique Words	Comments	Platform
SES	163929	8667	3000	GitHub
MSR2016 [26]	37768	3670	1992	JIRA

comments disagreed or mistaken by the annotators, a third annotator would step in and discuss with the others until reaching an agreement. Annotators were asked to classify the comments into 3 classes, 'negative', 'positive' or 'neutral' based on his/her cognition of the comments. We developed a web service storing annotators' basic information for annotation using Django framework (Figure 2), the code could be accessed on GitHub[2].

2,379 comments out of 3,000 comments are labeled without conflicts after the initial annotation, which reaches an agreement rate at about 79.2%. We use Cohen's Kappa to evaluate the agreement among the two raters. As a result, k value is 0.550, which indicates a moderate agreement [16]. Due to that sentiment analysis is a fairly subjective task, the agreement of the training set is acceptable.

After the second round of consensus, the final ratings are discussed to be determined. In our dataset, most comments are labeled 'neutral' (66.6%), 'positive' (19.9%) and 'negative' comments (13.5%) are less. Since the dataset is imbalanced, some sampling methods are used during classification task.

We name our dataset 'SES', which contains comments of various length. Figure 3 shows the Cumulative Distribution Function (CDF) for the distribution of the comment length. There are 86.2% comments containing less than 140 characters, cf. Twitter blog limitations. It reflects that the comments in SES are mainly short texts, but SES involves a number of long texts as well. Compared to the emotional dataset generated from JIRA issue comments [26], SES shows some advatanges. As shown in Table 1, MSR2016 dataset chooses comments of less words and less unique words, which means SES is more diverse and has more features. As for the number of comments, although MSR2016 dataset claims that they generate a gold standard dataset, their 4,000 sentence-level annotated corpus is verified to be insufficient [2] and inconsistant in human rating [14], so the dataset only contains 1,992 reliable comments, less than SES. The platform we choose is GitHub, as the most popular open source management community, GitHub offers more information than JIRA such as 'thumbs up reaction', which may indicate implicit sentiment. In general, we generate a dataset SES that contains more information than previous issue comments sentiment dataset.

2.2 Preprocessing

The issue comment texts are different from movie reviews or twitter microblogs in code snippets, terminologies and emojis, thus we need to preprocess the text so as to extract the most important information and reduce the noise. The specific steps are as follows.

Non-English Characters Deletion. We only consider English characters in SentiSW, so we ignore all the non-ascii characters in issue comments by replacing them with space.

Contraction Expansion. Contraction refer to a shortened version of a word or word group. To get the complete word, we replace

[2]https://github.com/Jasmine-DJ-420/annotationDjango

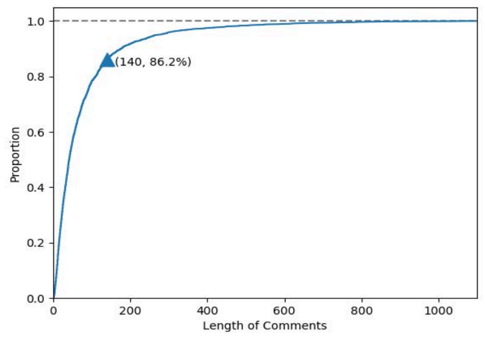


Figure 3: CDF of SES.

the abbreviation with the full version of the words. This step could reduce the number of unique words and make it easier to identify negative words.

Code Snippet Removal. GitHub Markdown is a way to style texts on the web, it applies punctuation marks to break down texts into different parts as code snippet, URL, image, quotation, etc. We discern code snippets and inline codes wrapped in quotation marks and remove them to produce pure texts without codes.

URL and Quotation Removal. Issue comments may contain URL (e.g., online reproduce process) or quotation to make issues better understood. Therefore, we remove links and quotations using GitHub Markdown.

Stop Words Removal. Stop words refer to those most regular used words in textual language, stop words removal could reduce the number of the features and thus reduce the noise. We take 'stopwords.txt' from StanfordNLP [20] into account and trim it including pronouns, auxiliary and preposition. We also create a stop word list for SE domain including common words that express no emotions (e.g., windows or chrome).

Emoticons and Punctuation Mark Processing. Emoticons are pictorial representation of facial expressions usually made of punctuation marks. In online communication platforms, it's clear that emoticons reflect sentiment to a large extent. Therefore, we generate a rule-based tool to recognize horizontal emoticons by identifying eyes, nose and mouth. We replace emoticons with 'POS_EMOTICON' or 'NEG_EMOTICON'. As for punctuation marks, we only consider emoticons, ellipsis, exclamation mark, question mark and underline, the rest punctuation marks are removed to reduce noise.

Negation Marking. Negation words including 'no', 'not' or 'none' may transform the sentimental meaning of the whole sentence. As discussed in other research before [13], negation words are identified and removed, and the other verbs and adjectives in the same sentence will be adding 'not_' in the front. For example, Sentence 'I don't agree with you' will be transferred into 'I do not not_agree with you'. This method have been proved to be effective in improving negative sentiment classification[1].

Word Tokenization and Stemming. Word tokenization delimit sentences into single words, and stemming reduces inflected words to their stems. NLTK 'word_tokenize' and 'Snowball Stemmer' are used to process.

2.3 Feature Vectorization

Feature vectorization aims to extract feature matrix from issue comments. We experiment on two feature extraction methods: TF-IDF (Term Frequency-Inverse Document Frequency) and Doc2vec (Document Vectors).

TF-IDF. TF-IDF is a statistic method to evaluate how important a word is to a document in a corpus, term frequency refers to the number of times that the term occurs while inverse document frequency refers to the number of documents containing the term.

Doc2vec. Word2vec [21], developed by Google, represents words with vectors and captures the contexts of words. Doc2vec [18] is an another version of Word2vec adapted to texts by adding a paragraph vector. We use 231,732 issue comments corpus gathered from GitHub to train a Doc2vec model, the preprocessing way is identical to Section 2.2. Using this model, we could transfer an issue comment to a 300-dimensional vector as a feature vector of our classifier. We set the hyper parameter *vector size* as 300, which is normally used in researches before [17].

2.4 Classifier

We use supervised machine learning method to do sentiment classification, because sentiment dictionary depends largely on human choice and is not easy to expand. We experiment on 6 supervised classification algorithms that shows great performance in the field of sentiment analysis of other domains before [1], such as Random Forest (RF), Bootstrap Aggregating (Bagging), Gradient Boosting Tree (GBT), Naïve Bayes (NB), Ridge Regression and Linear Support Vector Machine (SVC). We use interfaces and the default parameters implemented in Scikit-learn [29].

As mentioned before, our training set is imbalanced for having more than half neutral comments, which may lead to a large prediction offset, thus balancing the dataset is necessary. Like [1], we increase the number of positive and negative comments by applying an oversampling algorithm named Synthetic Minority Oversampling Technique (SMOTE) [6] to balance the dataset.

2.5 Entity Recognition

We classify entity of subjective issue comments from two aspects, 'Person' and 'Project'. We summarize a set of rules based on our dataset. According to our observation, developers mainly express positive sentiment in issue comments to show their kindness, agreement, gratitude toward other developers, or show love, hope to the project, while they mainly express negative sentiment to show disagreement, impatience toward other developers or disappointment, confusion to the project. It indicates that issue comments strengthen the relationship between developers and help developers express opinions on projects. Thus, to discover the developers' sentiment connection could reflect the relationships and and the law of interaction between developers and to acquire their sentiment toward a project can offer the project managers some suggestions for reference. Thus we recognize and classify sentiment entity into 'Person' and 'Project' so as to gain knowledge of developers' connection and their opinions on a project.

Table 2: Top 15 important entity labeled feature list.

Feature	Importance	Entity
thank	0.0931548	Person
sorri	0.0272497	Person
exclam	0.0269961	Vague
question_mark	0.0176884	Vague
pos_emoticon	0.0170072	Vague
but	0.0144705	Vague
describ	0.0116384	Vague
me	0.0110292	Vague
ellipsi	0.0108818	Vague
whi	0.0106325	Vague
good	0.0102902	Vague
regard	0.0097177	Vague
not_work	0.0095096	Project
weird	0.0090886	Vague
time	0.0088578	Vague

As discussed in Section 5.3, there are four main methods to extract entity information from texts. The most important feature in our case is that the entity is fairly clear as 'Person' and 'Project'. We do not have to extract the unknown topic list from the texts, and thus frequency-based method and unsupervised machine learning method is not suitable. As for syntax-based method and supervised machine learning method, supervised machine learning method requires huge and accurate annotated dataset, which is lack in Software Engineering domain, while syntax-based or known as rule-based method requires little preparation work but very depends on the rules discovered. For the reason that 'Person' entity contains a lot of rules itself because it could be found by pronoun or name of person, we apply rule-based technology to identify 'Person' entity. In former researches, NLTK [3] and StanfordNLP [8] has identified and classified named entity as 'Person', 'Organization', etc, we apply the named entity tagger tool in our rules to identity 'Person' names in issue comments. At the same time, the part of speech tag (POS) offers grammatical features of words such as 'noun', 'verb' and 'pronoun', etc, we apply POS as a part of our rules to increase precision of our method.

To recognize entity, we first generate a feature list learned from supervised classification tool discussed in Section 2.4, we annotate the top 15 features as shown in Table 2. Next, we recognize the named entity using the Stanford Named Entity Tagger [8] and the POS using NLTK. We thus generalize a word list with named entity tagger and POS tagger. At last, we apply a rule-based method to identify 'Person' entity. The rules is based on our cognition generated during the annotation process. The rules are listed below.

i) The sentences with top *m* important features are considered as the sentiment sentences, and only sentiment sentences are taken into account when identifying entity, we experiment different values of *m* to create a best performance. **ii)** The sentiment sentences containing '@' or 'PERSON' entity named tagger are considered as 'Person' entity. **iii)** As shown in Table 2. The sentiment sentences containing 'sorri' or 'thank' are considered as 'Person', the sentiment sentences containing 'not_work' are considered as 'Project'.

Table 3: Results on different feature vectorization and classifiers.

Algorithm		Positive			Negative			Neutral			Overall			
		Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure	Accuracy
TF-IDF	RF	70.71%	50.83%	58.91%	31.07%	27.99%	29.07%	78.48%	86.57%	82.28%	60.09%	55.13%	56.75%	71.49%
	Bagging	74.58%	61.00%	66.83%	30.12%	36.73%	32.73%	80.98%	81.87%	81.38%	61.89%	59.87%	60.31%	71.57%
	GBT	**83.26%**	63.84%	**72.11%**	**40.15%**	**39.20%**	**39.35%**	**82.73%**	**88.90%**	**85.67%**	**68.71%**	**63.98%**	**65.71%**	**77.19%**
	NB	68.83%	31.82%	43.45%	26.60%	37.92%	31.05%	75.57%	81.21%	78.24%	57.00%	50.31%	50.91%	65.53%
	Ridge	40.82%	55.86%	46.94%	31.27%	38.67%	34.41%	78.14%	66.04%	71.53%	50.07%	53.52%	50.96%	60.29%
	SVC	49.00%	**65.05%**	55.73%	31.86%	35.72%	33.45%	80.08%	70.25%	74.80%	53.64%	57.01%	54.66%	64.61%
Doc2Vec	RF	35.78%	11.92%	17.74%	22.66%	10.55%	14.25%	69.01%	90.14%	78.14%	42.48%	37.54%	36.71%	63.83%
	Bagging	39.26%	15.72%	22.25%	24.30%	13.50%	17.15%	69.96%	88.80%	78.22%	44.51%	39.34%	39.21%	64.07%
	GBT	63.22%	23.25%	33.68%	36.98%	8.08%	12.99%	70.88%	95.39%	81.30%	57.03%	42.24%	42.65%	69.23%
	NB	51.68%	22.40%	30.95%	22.60%	**40.65%**	**28.83%**	**73.99%**	74.55%	74.22%	49.43%	**45.87%**	**44.67%**	59.58%
	Ridge	**63.24%**	24.68%	35.30%	42.45%	4.24%	7.59%	70.85%	**96.65%**	81.73%	58.85%	41.86%	41.54%	69.84%
	SVC	61.40%	**27.75%**	**37.90%**	**43.84%**	6.76%	11.51%	71.71%	95.62%	**81.92%**	**58.98%**	43.38%	43.78%	**70.11%**

Table 4: SentiSW vs. SentiStrength vs. SentiStrength-SE.

Sentiment	Metric	SentiSW	SentiStrength	SentiStrength-SE
Positive	Precision	**83.26%**	39.55%	52.98%
	Recall	63.84%	64.21%	**70.90%**
	F-measure	**72.11%**	48.95%	60.53%
Negative	Precision	40.15%	24.78%	**40.45%**
	Recall	39.20%	**41.34%**	39.85%
	F-measure	39.35%	30.98%	**40.15%**
Neutral	Precision	82.73%	76.53%	**83.38%**
	Recall	**88.90%**	51.90%	75.08%
	F-measure	**85.67%**	61.86%	79.01%
Overall	Precision	**68.71%**	46.95%	58.88%
	Recall	**63.98%**	52.48%	61.94%
	F-measure	**65.71%**	47.26%	59.90%
	Accuracy	**77.19%**	52.93%	69.50%

iv) Personal pronouns describing person such as 'you' or 'him' near the sentiment features in 2 words are considered as 'Person' entity.

By detecting pronouns and 'PERSON' entity named tagger, we preliminary identify 'Person' entity and 'Project' entity.

3 EXPERIMENTAL EVALUATION

We evaluate SentiSW from sentiment classification and entity recognition. All the evaluations are done at comment level.

3.1 Evaluation of Sentiment Classification

We compared SentiSW to SentiStrength-SE [14] and SentiStrength [32]. We evaluated our method with precision, recall, f-measure and accuracy in the detection of positive, negative and neutral sentiment, thus we get 10 evaluation indicators shown in Table 3. 10-fold cross validation method was used to generalize training set as well as test set. For each experiment, 10-fold cross validation method was repeated for 10 times, which makes one hundred experiments in total, we then calculated the average value. We evaluated the performance of different feature vectorization methods, classifiers in Table 3 and compares SentiSW with SentiStrength, SentiStrength-SE in Table 4.

According to Table 3, for feature vectorization method, TF-IDF performs much better than Doc2vec, this is due to the lack of information in our training set containing only 3,000 comments. As for classifiers, GBT performs the best among others. This may is due to the boosting method that combines weak learners by paying attention to the nodes incorrectly classified.

According to Table 4, We compared SentiSW with previous sentiment analysis tool. SentiStrength, sentiment analysis tool generated from other domain, shows poorer effect than SentiSW and SentiStrength-SE, proving once again that domain specific sentiment analysis tool should be developed. SentiSW, on the other hand, significantly outperforms SentiStrength-SE in the detection of positive and neutral sentiment while it performs similarly of negative sentiment compared to SentiStrength-SE. Both SentiSW and SentiStrength-SE perform poorly on negative sentiment issue comments, which has always been a difficulty in sentiment analysis task. The various expressions of negative feelings including ironic comments could lead to the low performance of negative sentences. At the same time, the negative comments training set is rather small. Overall, SentiSW outperforms SentiStrength-SE on average precision, recall and f-measure and receives accuracy.

In general, the main reason why SentiSW outperforms SentiStrength is that it generates features directly from Software Engineering domain, and the main reason why SentiSW outperforms SentiStrength-SE is that SentiStrength-SE's dictionary generated from commit messages is not complete or totally accurate.

3.2 Evaluation of Entity Recognition

To validate the performance of our entity recognition method, we manually labeled 415 positive issue comments and 245 negative sentiment which are correctly classified by SentiSW, the distribution is shown in Table 5. We then used this dataset to validate our entity recognition tool, we experiment on different m value and find the tool performs the best when m equals 50. The overall precision, recall, f-measure and accuracy shown in Table 6 all reaches 75%.However, SentiSW shows low precision in detecting entity in negative comments, it indicates that some rules may be ignored.

We used our tool to analyze the sentiment and entity of the 231,732 issue comments we gathered in Section 2.1, and Table 7 represents the distribution of different sentiment and entities. Table 5 and Table 7 shows that a developer tends to be positive to

Table 5: Entity labeled distribution.

	Positive	Negative
Person	345	81
Project	70	164

Table 6: Validation results of entity recognition (*m*=50).

	Precision	Recall	F-measure	Accuracy
Positive	91.25%	90.72%	90.99%	85.06%
Negative	43.05%	80.25%	56.03%	58.37%
Overall	76.58%	88.73%	82.17%	75.15%

Table 7: Entity labeled distribution on whole dataset.

	Positive	Negative
Person	21068	9724
Project	3114	16241

other developers, and negative to project, which might be a general characteristic of developers.

4 THREATS TO VALIDITY

Our training set is generated from GitHub issue tracker, therefore it's possible that SentiSW is not general for all issue tracking systems such as JIRA or Bugzilla. At the same time, overfitting may also be a problem to consider. To validate these 2 problems, we use SentiSW to classify an annotated JIRA issue comment set [26], and receive an overall accuracy of 87.82%. This confirms the availability of SentiSW in other issue tracking systems and that SentiSW does not overfit the training set.

5 RELATED WORK

5.1 Issue Tracking System

Issues in software engineering often refer to bugs, new requests or enhancement suggestions toward a software system, and an issue tracking system (ITS) records, follows and reviews the issues until they are finally solved. Famous ITS include GitHub issue tracking system, Bugzilla and JIRA. ITS aims to catch issues and improve software quality. In ITS, developers discuss together, thus form a collaboration network containing valuable knowledge.

5.2 Sentiment Analysis

Sentiment analysis, also known as opinion mining, refers to the techniques mining quintuple opinion (e, a, s, h, t) from textual information [19], where 'e' stands for entity the sentiment expresses toward, 'a' for aspect of the entity, 's' for sentiment, 'h' for holder who comment, 't' for time. For instance, 'This book is very cheap' is a positive sentence, the entity is 'book' while the aspect is 'price'. Sentiment analysis could be classified by 3 level, the document level, the sentence level, and the entity or aspect level [28]. Document level and sentence level sentiment analysis require the holder express sentiment toward single entity, while entity level sentiment analysis recognizes a <*sentiment, entity*> tuple.

Sentiment analysis could be achieved with two major techniques, lexicon-based method and machine learning method. Lexicon-based method uses rule-based method to generate a sentiment dictionary [3, 32], and calculate the sentiment score of the whole paragraph by simply adding all words' sentiment score. This method does not take contexts into account and it's hard to generate a domain specific dictionary. On the other hand, a supervised machine learning method uses natural language process, labeled text and various classification algorithms to automatically develop a sentiment classification tool [13]. It could consider special purpose and contexts, but it is hard to require a gold-standard labeled dataset.

5.3 Entity Recognition

Entity recognition aims at detecting entity or aspect of the sentiment. Main methods of entity recognition include frequency-based method [13], syntax-based method [30], supervised machine learning method [15] and unsupervised machine learning method [33].

Frequency-based methods extract the most frequent nouns in a sentence as an entity, such methods may extract useless words as 'dollar' and may ignore implicit entities not appeared in a sentence. Syntax-based or rule-based methods apply a set of rules based on grammatical relations, which require little preparation work and only small amount of seeds are needed to work. Supervised machine learning method casts entity recognition problem into a labeling problem, which is solved by a linear chain Conditional Random Field (CRF). Unsupervised machine learning method basically uses Latent Dirichlet Allocation (LDA) to detect entity in a sentence, however, LDA depends largely on the dataset and may require a fixed set of global topics and a dynamic set of local topics.

6 CONCLUSION AND FUTURE WORK

In this study, we build a 3,000 labeled sentiment dataset, which is the largest software engineering dataset in sentiment analysis. Using this dataset, we develop SentiSW, an entity-level sentiment analysis tool specific for SE domain. SentiSW, consisting of sentiment classification and entity recognition, can identify the objectivity and the targets of issue comments, and return a <*sentiment, entity*> tuple when a comment is subjective. Our experimental results demonstrate the advantages of our method.

Our future work leads to expanding the scale and diversity of our dataset and exploring more advanced methods in the field of natural language processing and sentiment analysis of social media. We will compare our method to more methods developed from SE domain. We also plan to study the impact of sentiment on code quality and software productivity.

ACKNOWLEDGEMENT

This work was supported partly by National Key Research and Development Program of China under Grant No.2016YFB1000804, partly by National Natural Science Foundation under Grant No. 61421003, partly by SKLSDE Lab under Grant No. SKLSDE-2017ZX-14, and partly by National Natural Science Foundation of China (No. 61702024).

REFERENCES

[1] Toufique Ahmed, Amiangshu Bosu, Anindya Iqbal, and Shahram Rahimi. 2017. SentiCR: A customized sentiment analysis tool for code review interactions. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press, 106–111.

[2] Gabriele Bavota Bin Lin, Fiorella Zampetti, Michele Lanza Massimiliano Di Penta, and Rocco Oliveto. 2018. Sentiment Analysis for So ware Engineering: How Far Can We Go?. In *Ieee/acm International Conference on Software Engineering*.

[3] Steven Bird and Edward Loper. 2004. NLTK: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*. Association for Computational Linguistics, 31.

[4] Fabio Calefato and Filippo Lanubile. 2016. Affective trust as a predictor of successful collaboration in distributed software projects. In *Emotional Awareness in Software Engineering (SEmotion), IEEE/ACM International Workshop on*. IEEE, 3–5.

[5] Fabio Calefato, Filippo Lanubile, Federico Maiorano, and Nicole Novielli. 2017. Sentiment Polarity Detection for Software Development. *Empirical Software Engineering* 3 (2017), 1–31.

[6] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[7] Munmun De Choudhury and Scott Counts. 2013. Understanding affect in the workplace via social media. In *Proceedings of the 2013 conference on Computer supported cooperative work*. ACM, 303–316.

[8] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 363–370.

[9] Daviti Gachechiladze, Filippo Lanubile, Nicole Novielli, and Alexander Serebrenik. 2017. Anger and Its Direction in Collaborative Software Development. In *Ieee/acm International Conference on Software Engineering*. 11–14.

[10] David Garcia, Marcelo Serrano Zanetti, and Frank Schweitzer. 2013. The role of emotions in contributors activity: A case study on the Gentoo community. In *Cloud and green computing (CGC), 2013 third international conference on*. IEEE, 410–417.

[11] Emitza Guzman, David Azócar, and Yang Li. 2014. Sentiment analysis of commit comments in GitHub: an empirical study. In *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 352–355.

[12] Emitza Guzman and Bernd Bruegge. 2013. Towards emotional awareness in software development teams. In *Proceedings of the 2013 9th joint meeting on foundations of software engineering*. ACM, 671–674.

[13] Minqing Hu and Bing Liu. 2004. Mining opinion features in customer reviews. In *AAAI*, Vol. 4. 755–760.

[14] Md Rakibul Islam and Minhaz F Zibran. 2017. Leveraging automated sentiment analysis in software engineering. In *Proceedings of the 14th International Conference on Mining Software Repositories*. IEEE Press, 203–214.

[15] Niklas Jakob and Iryna Gurevych. 2010. Extracting opinion targets in a single-and cross-domain setting with conditional random fields. In *Proceedings of the 2010 conference on empirical methods in natural language processing*. Association for Computational Linguistics, 1035–1045.

[16] J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics* (1977), 159–174.

[17] Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368* (2016).

[18] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*. 1188–1196.

[19] Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies* 5, 1 (2012), 1–167.

[20] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 55–60.

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[22] Alessandro Murgia, Marco Ortu, Parastou Tourani, Bram Adams, and Serge Demeyer. 2017. An exploratory qualitative and quantitative analysis of emotions in issue report comments of open source systems. *Empirical Software Engineering* (2017), 1–44.

[23] Alessandro Murgia, Parastou Tourani, Bram Adams, and Marco Ortu. 2014. Do developers feel emotions? an exploratory analysis of emotions in software artifacts. In *Proceedings of the 11th working conference on mining software repositories*. ACM, 262–271.

[24] Nicole Novielli, Fabio Calefato, and Filippo Lanubile. 2014. Towards discovering the role of emotions in stack overflow. In *Proceedings of the 6th international workshop on social software engineering*. ACM, 33–36.

[25] Marco Ortu, Bram Adams, Giuseppe Destefanis, Parastou Tourani, Michele Marchesi, and Roberto Tonelli. 2015. Are bullies more productive? Empirical study of affectiveness vs. issue fixing time. In *Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on*. IEEE, 303–313.

[26] Marco Ortu, Alessandro Murgia, Giuseppe Destefanis, Parastou Tourani, Roberto Tonelli, Michele Marchesi, and Bram Adams. 2016. The emotional side of software developers in JIRA. In *Mining Software Repositories (MSR), 2016 IEEE/ACM 13th Working Conference on*. IEEE, 480–483.

[27] Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining.. In *LREc*, Vol. 10.

[28] Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval* 2, 1–2 (2008), 1–135.

[29] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12, Oct (2011), 2825–2830.

[30] Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2009. Expanding domain sentiment lexicon through double propagation.. In *IJCAI*, Vol. 9. 1199–1204.

[31] Peter C Rigby and Ahmed E Hassan. 2007. What can oss mailing lists tell us? a preliminary psychometric text analysis of the apache developer mailing list. In *Mining Software Repositories, 2007. ICSE Workshops MSR'07. Fourth International Workshop on*. IEEE, 23–23.

[32] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment strength detection in short informal text. *Journal of the Association for Information Science and Technology* 61, 12 (2010), 2544–2558.

[33] Hongning Wang, Yue Lu, and ChengXiang Zhai. 2011. Latent aspect rating analysis without aspect keyword supervision. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 618–626.