Title	Sentiment Classification Using N-Gram Inverse Document Frequency and Automated Machine Learning
Author(s)	Rungroj Maipradit Hideaki Hata Kenichi Matsumoto
Citation	IEEE Software （ Volume: 36 , Issue: 5 , Sept.-Oct. 2019 ）
Issue Date	2019/5/29
Resource Version	author
Rights	© 2019IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
DOI	info:doi/10.1109/MS.2019.2919573
URL	http://hdl.handle.net/10061/13411

Sentiment Classification Using N-gram IDF and Automated Machine Learning

Rungroj Maipradit*, Hideki Hata*, Kenichi Matsumoto*
*Nara Institute of Science and Technology, Japan
{maipradit.rungroj.mm6, hata, matumoto}@is.naist.jp

***Abstract*—We propose a sentiment classification method with a general machine learning framework. For feature representation, n-gram IDF is used to extract software-engineering-related, dataset-specific, positive, neutral, and negative n-gram expressions. For classifiers, an automated machine learning tool is used. In the comparison using publicly available datasets, our method achieved the highest F1 values in positive and negative sentences on all datasets.**

keywords **Automated machine learning; N-gram IDF; Sentiment classification**

I. Introduction

As software development is a human activity, identifying affective states in messages has become an important challenge to extract meaningful information. Sentiment analysis has been used to several practical purposes, such as identifying problematic API features [1], assessing the polarity of app reviews [2], clarifying the impact of sentiment expressions to the issue resolution time [3], and so on.

Because of the poor accuracy of existing sentiment analysis tools trained with general sentiment expressions [4], recent studies have tried to customize such tools with software engineering datasets [5]. However, it is reported that no tool is ready to accurately classify sentences to negative, neutral, or positive, even if tools are specifically customized for certain software engineering tasks [5].

One of the difficulties in sentiment classification is the limitation in a bag-of-words model or polarity shifting because of function words and constructions in sentences [6]. Even if there are positive single words, the whole sentence can be negative because of negation, for example. For this challenge, Lin et al. adopted `Stanford CoreNLP`, a recursive neural network based approach that can take into account the composition of words [7], and prepared a software-engineering-specific sentiment dataset from a Stack Overflow dump [5]. Despite their large amount of effort on fine-grained labeling to the tree structure of sentences, they reported negative results (low accuracy) [5].

In this paper we propose a machine-learning based approach using n-gram features and an automated machine learning tool for sentiment classification. Although n-gram phrases are considered to be informative and useful compared to single words, using all n-gram phrases is not a good idea because of the large volume of data and many useless features [8]. To address this problem, we utilize n-gram IDF, a theoretical extension of Inverse Document Frequency (IDF) proposed by Shirakawa et al. [9]. IDF measures how much information the word provides; but it cannot handle multiple words. N-gram IDF is capable of handling n-gram phrases; therefore, we can extract useful n-gram phrases.

Automated machine learning is an emerging research area targeting the progressive automation of machine learning. Two important problems are known in machine learning: no single machine learning technique give the best result on all datasets, and hyperparameter optimization is needed. Automated machine learning addresses these problems by running multiple classifiers and tries different parameters to optimize the performance. In this study, we use auto-sklearn, which contains 15 classification algorithms (random forest, kernel SVM, etc.), 14 feature pre-processing solutions (PCA, nystroem sampler, etc.), and 4 data pre-process solutions (one-hot encoding, rescaling, etc.) [10]. Using n-gram IDF and auto-sklearn tools, Wattanakriengkrai et al. outperformed the state-of-the-art self-admitted technical debt identification [11].

II. Method

Figure 1 shows an overview of our method with the following three components.

Text Preprocessing. Messages in software document sometimes contain special characters. We remove characters that are neither English characters nor numbers. Stop words are also removed by using spaCy library. spaCy tokenizes text and finds part of speech and tag of each token and also checks whether the token appears in the stop word list.

Feature extraction using N-gram IDF. N-gram IDF is a theoretical extension of IDF for handling words and phrases of any length by bridging the gap between term weighting and multiword expression extraction. N-gram IDF can identify dominant n-grams among overlapping ones [9]. In this study, we use N-gram Weighting Scheme tool [9]. The result after applying this tool is a dictionary of n-gram phrases ($n \leq 10$ as a default setting) with their frequencies. N-gram phrases appear only one time in the whole document (frequency equal one) are removed, since they are not useful for training.

Automated machine learning. To classify sentences into positive, neutral, and negative, we use auto-sklearn, an automated machine learning tool. Automated machine learning tries running multiple classifiers and applying different parameters to derive better performances. Auto-sklearn composes two steps: meta-learning and automated ensemble construction [10]. We ran auto-sklearn with 64 gigabytes of memories, set 90 minutes limitation for each round, and configure it to

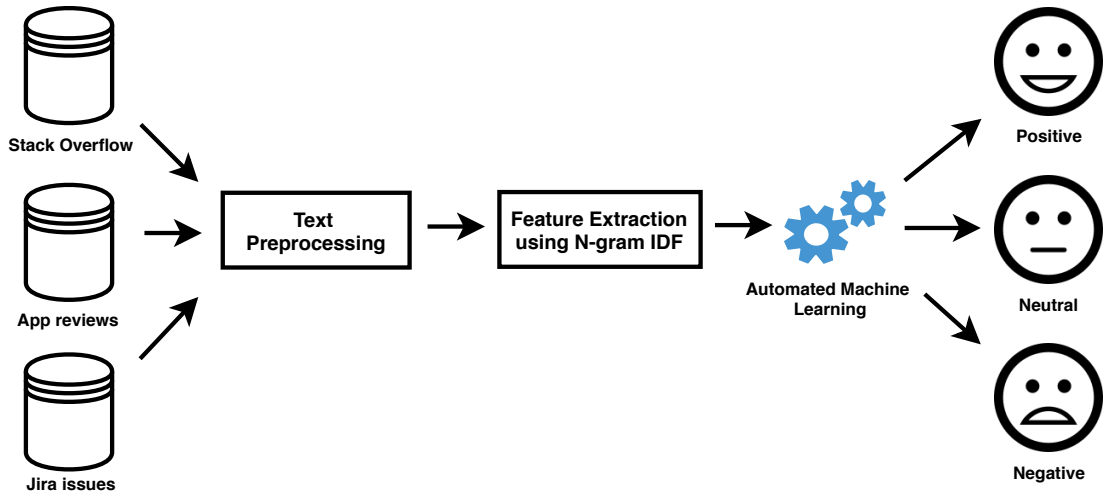

Fig. 1. An overview of our sentiment classification approach

optimize for a weighted F1 value, an average F1 value for three classes weighted by the number of true instance of each class.

III. Evaluation

A. Datasets and Settings

We employ a dataset provided by Lin et al.'s work [5]. There are three types of document in the dataset; sentences in questions and answers on Stack Overflow, reviews of mobile applications, and comments on Jira issue trackers. Each dataset has texts and labels of positive, neutral and negative.

Since our method requires just labeled (positive, neutral, or negative) sentences, we can use dataset-specific data for training. For training and testing, we apply 10-fold cross-validation for each dataset, that is, we split sentences in a dataset into ten subsets with maintaining the ratio from the oracles by using the function StratifiedShuffleSplit in scikit-learn.

B. Sentiment Classification Tools

To assess the performance of our method, we compare our method with tools presented in the previous work [5].

SentiStrength estimates the strength of positive and negative scores based on the sentiment word strength list prepared from MySpace comments [12].

NLTK is a natural language toolkit and is able to do sentiment analysis based on lexicon and rule-based VADER (Valence Aware Dictionary and sEntiment Reasoner), which is specifically tuned for sentiments expressed in social media [13].

Stanford CoreNLP adopts a deep learning model to compute the sentiment based on how words compose the meaning of the sentence [7]. The model has been trained on movie reviews.

SentiStrength-SE is a tool build on top of SentiStrength and has been trained on JIRA issue comments [14].

Stanford CoreNLP SO prepared Stack Overflow discussions to train a model of Standford CoreNLP [5].

C. Result

Table I shows the number of correct predictions, precision, recall, and F1 values with all tools including our method (`n-gram auto-sklearn`). These values were presented in [5] (F1 values are calculated by us). Precision, recall, and F1 values are derived as the average from the 10 rounds of our 10-fold cross-validation, since same data can appear in different rounds with StratifiedShuffleSplit.

We can see that the number of correct predictions are higher with our method in all three datasets, and our method achieved the highest F1 values for all three positive, all three negative, and one neutral. Although the values are low for the neutral class in App reviews, this is because the amount of neutral sentences is small in this dataset. In summary, our method using n-gram IDF and automated machine learning (auto-sklearn) largely outperformed existing sentiment analysis tools. Since our method relies on n-gram phrases, it cannot properly classify text without known n-gram phrases. Although a negative sentence "They all fail with the following error" was correctly classified with SentiStrength, NLTK, and Stanford CoreNLP, our method classified as neutral. Preparing more data is preferable to improve the performance.

Note that only our method trains within-dataset for all three cases. Although within-dataset training can improve the performances of other tools, preparing training data for those sentiment analysis tools require considerable manual effort [5]. Since our method can automatically learn dataset-specific text features, learning within-dataset is practically feasible.

The following are classifiers achieved the top three performances in `auto-sklearn` for each dataset.

- **Stack Overflow**: Linear Discriminant Analysis, LibSVM Support Vector Classification, and Liblinear Support Vector Classification
- **App reviews**: Random forest, LibSVM Support Vector Classification, and Naive Bayes classifier for multinomial models
- **Jira issues**: Naive Bayes classifier for multinomial models, Adaptive Boosting, and Linear Discriminant Analysis

TABLE I
THE COMPARISON RESULT OF THE NUMBER OF CORRECTED PREDICTION, PRECISION, RECALL, AND F1-SCORE

dataset	tool	# correct prediction	positive precision	positive recall	positive F1	neutral precision	neutral recall	neutral F1	negative precision	negative recall	negative F1
Stack Overflow	SentiStrength	1043	0.200	**0.359**	0.257	0.858	0.772	0.813	0.397	0.433	0.414
positive: 178	NLTK	1168	0.317	0.244	0.276	0.815	**0.941**	0.873	**0.625**	0.084	0.148
neutral: 1,191	Standford CoreNLP	604	0.231	0.344	0.276	**0.884**	0.344	0.495	0.177	**0.837**	0.292
negative: 131	SentiStrength-SE	1170	0.312	0.221	0.259	0.826	0.930	0.875	0.500	0.185	0.270
sum: 1,500	Stanford CoreNLP SO	1139	0.317	0.145	0.199	0.836	0.886	0.860	0.365	0.365	0.365
	N-gram auto-sklearn	**1317**	**0.667**	0.316	**0.418**	0.871	0.939	**0.904**	0.600	0.472	**0.514**
	N-gram auto-sklearn with SMOTE†	-	0.680	0.005	0.009	0.344	0.930	0.499	0.657	0.160	0.251
App reviews	SentiStrength	213	0.745	0.866	0.801	0.113	0.320	0.167	0.815	0.338	0.478
positive: 186	NLTK	184	0.751	0.812	0.780	0.093	**0.440**	0.154	**1.000**	0.169	0.289
neutral: 25	Standford CoreNLP	237	0.831	0.715	0.769	**0.176**	0.240	**0.203**	0.667	0.754	0.708
negative: 130	SentiStrength-SE	201	0.741	0.817	0.777	0.106	0.400	0.168	0.929	0.300	0.454
sum: 341	Stanford CoreNLP SO	142	0.770	0.253	0.381	0.084	0.320	0.133	0.470	0.669	0.552
	N-gram auto-sklearn	**293**	**0.822**	**0.894**	**0.853**	0.083	0.066	0.073	0.823	**0.808**	**0.807**
	N-gram auto-sklearn with SMOTE†	-	0.520	0.885	0.641	0.100	0.058	0.073	0.648	0.622	0.607
Jira issues	SentiStrength	714	0.850	**0.921**	0.884	-	-	-	0.993	0.703	0.823
positive: 290	NLTK	276	0.840	0.362	0.506	-	-	-	**1.000**	0.269	0.424
neutral: 0	Standford CoreNLP	626	0.726	0.621	0.669	-	-	-	0.945	0.701	0.805
negative: 636	SentiStrength-SE	704	0.948	0.883	0.914	-	-	-	0.996	0.704	0.825
sum: 926	Stanford CoreNLP SO	333	0.635	0.252	0.361	-	-	-	0.724	0.409	0.523
	N-gram auto-sklearn	**884**	**0.960**	0.839	**0.893**	-	-	-	0.932	**0.982**	**0.956**
	N-gram auto-sklearn with SMOTE†	-	0.986	0.704	0.809	-	-	-	0.781	0.988	0.872

† Applying SMOTE, a oversampling technique, for our method.

If we have a new unlabeled dataset, we can first try one of the common classifiers. By manually annotating labels, we can try auto-sklearn to find the best classifier for the dataset.

IV. DISCUSSIONS

A. Threats to Validity

Competitive sentiment classification tools are trained only with specific dataset. Since our method is based on a general text classification approach, we could conduct within-dataset training. However, because of a considerable amount of manual effort for training sentiment classification tools, they had been trained only with specific datasets. Although we think this is an advantage of our method, the comparison is not with the same condition.

Imbalanced data. In this multi-class classification, some datasets are not balanced; neutral class is the majority for Stack Overflow and no neural data for Jira issues. Applying some balancing techniques may improve the overall performances.

Our study might not be generalize to other datasets. Our approach is applied to comments, reviews, and questions and answers. Other types of document related to software engineering may derive different results.

B. Obtained N-gram Phrases

Why our method achieved high accuracy performance in sentiment classification? Table II shows selected n-gram phrases, which were useful for classifying positive, neutral, and negative statements, obtained in each dataset. For negative, we see 'bug', a software-engineering-specific negative word, and many negation expressions. We can also see reasonable n-gram phrases for positive cases, such as 'useful', 'really like this app', 'awesome work', and so on. We can think that because of these dataset-specific positive, neutral, and negative patterns, n-gram IDF worked well for resolving the limitation in a bag-of-words model and our method have resulted in good performance.

V. CONCLUSION

In this paper, we proposed a sentiment classification method using n-gram IDF and automated machine learning. We apply this method on three datasets including question and answer from Stack Overflow, reviews of mobile applications, and comments on Jira issue trackers.

Our good classification performance is not based only on an advanced automated machine learning. N-gram IDF also worked well to capture dataset-specific, software-engineering-related positive, neutral, and negative expressions. Because of the capability of extracting useful sentiment expressions with n-gram IDF, our method can be applicable to various software engineering datasets.

REFERENCES

[1] Y. Zhang and D. Hou, "Extracting problematic API features from forum discussions," in *Proceedings of 21st International Conference on Program Comprehension (ICPC)*, 2013, pp. 142–151.

[2] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can i improve my app? classifying user reviews for software maintenance and evolution," in *Proceedings of 31st IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2015, pp. 281–290.

[3] M. Ortu, B. Adams, G. Destefanis, P. Tourani, M. Marchesi, and R. Tonelli, "Are bullies more productive?: Empirical study of affectiveness vs. issue fixing time," in *Proceedings of 12th Working Conference on Mining Software Repositories (MSR)*, 2015, pp. 303–313.

TABLE II
OBTAINED N-GRAM PHRASES (SELECTED)

dataset	positive	neutral	negative
Stack Overflow	'useful' 'the', 'simplest', 'solution' 'more', 'efficient', 'to' 'helpful' 'a', 'good', 'example'	'suggest', 'using' 'everyone' 'limitation' 'appointment' 'technical'	'wrong' 'bug' 'i', 'do', 'n', 't', 'understand' 'i', 'do', 'n', 't', 'know' 'does', 'n', 't', 'work'
App reviews	'thanks', 'for' 'really', 'like', 'this', 'app' 'well', 'done' 'awesome' 'easy', 'to', 'use', 'and'	'let', 'me', 'know' 'gets', 'the', 'job', 'done' 'android', 'device' 'allow' 'suggestions'	'impossible', 'to', 'use' 'game', 'constantly', 'freezes' 'uninstalled' 'disappointing' 'lack', 'of', 'features'
Jira issues	'thank', 'you', 'very', 'much' 'looks', 'good', 'we' 'awesome', 'work' 'thanks', 'for', 'your', 'help' 'awesome', 'stuff'	- - - - -	'problems' 'is', 'bad' 'i', 'disagree' 'really', 'sucks' 'this', 'bug'

[4] R. Jongeling, P. Sarkar, S. Datta, and A. Serebrenik, "On negative results when using sentiment analysis tools for software engineering research," *Empirical Software Engineering*, vol. 22, no. 5, pp. 2543–2584, Oct. 2017.

[5] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, M. Lanza, and R. Oliveto, "Sentiment analysis for software engineering: How far can we go?" in *Proceedings of 40th International Conference on Software Engineering (ICSE)*, 2018, pp. 94–104.

[6] S. Li, S. Y. M. Lee, Y. Chen, C.-R. Huang, and G. Zhou, "Sentiment classification and polarity shifting," in *Proceedings of 23rd International Conference on Computational Linguistics (COLING)*, 2010, pp. 635–643.

[7] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013, pp. 1631–1642.

[8] D. Bespalov, B. Bai, Y. Qi, and A. Shokoufandeh, "Sentiment classification based on supervised latent n-gram analysis," in *Proceedings of 20th ACM International Conference on Information and Knowledge Management (CIKM)*, 2011, pp. 375–382.

[9] M. Shirakawa, T. Hara, and S. Nishio, "N-gram IDF: A global term weighting scheme based on information distance," in *Proceedings of 24th International Conference on World Wide Web (WWW)*, 2015, pp. 960–970.

[10] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Proceedings of 28th International Conference on Neural Information Processing Systems (NIPS)*, 2015, pp. 2962–2970.

[11] S. Wattanakriengkrai, R. Maipradit, H. Hata, M. Choetkiertikul, T. Sunetnanta, and K. Matsumoto, "Identifying design and requirement self-admitted technical debt using n-gram IDF," in *Proceedings of 9th IEEE International Workshop on Empirical Software Engineering in Practice (IWESEP)*, 2018, pp. 7–12.

[12] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas, "Sentiment strength detection in short informal text," *Journal of the American Society for Information Science and Technology*, vol. 61, no. 12, pp. 2544–2558, Dec. 2010.

[13] C. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Proceedings of 8th International AAAI Conference on Weblogs and Social Media*, 2014, pp. 216–225.

[14] M. R. Islam and M. F. Zibran, "Leveraging automated sentiment analysis in software engineering," in *Proceedings of 14th International Conference on Mining Software Repositories (MSR)*, 2017, pp. 203–214.