Journal of Engineering Sciences
Vol 10, Issue 9, Sept /2019
ISSN NO:0377-9254

Polarity Classification Through N Gram Machine Learning Approach

[1]J MEHANAZ BEGUM, [2]DR.G.VIJAY KUMAR

[1]M.Tech Student, [2]Associate Professor

DEPT OF CSE

G.PULLAREDDY ENGINEERING COLLEGE, KURNOOL

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.

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, assessing the polarity of app reviews, clarifying the impact of sentiment expressions to the issue resolution time, and so on. Because of the poor accuracy of existing sentiment analysis tools trained with general sentiment expressions, recent studies have tried to customize such tools with software engineering datasets. 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. 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.

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 and prepared a software-engineering specific sentiment dataset from a Stack Overflow dump. Despite their large amount of effort on fine-grained labeling to the tree structure of sentences, they reported negative results (low accuracy). In this system 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.

To address this problem, we utilize n-gram IDF, a theoretical extension of Inverse Document Frequency (IDF) proposed by Shirakawa et al. 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 hyper parameter 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). Using n-gram IDF and autosklearn tools, Wattanakriengkrai et al. outperformed the state-of-the-art self-admitted technical debt identification.

1.1 BACKGROUND

Machine learning is all pervasive today be it in business industry, science, government in solving the worlds important tasks. Machine learning is the enabler of AI (Artificial Intelligence) that it can learn from the experience and examples. Machine learning is the one which make computers program explicitly. The advantage in the ML is we need not write the code; just we can feed the data to the algorithms and build the logic on the given data.

These ML algorithms help's to perform the task based on the data or the examples. For example, one kind of algorithm is a classification algorithm. It can put data into different groups. The classification algorithm used to detect handwritten alphabets could also be used to classify emails into spam and not-spam.

A computer program is said to learn from experience E with some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.

II. LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then the next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

The major part of the project development sector considers and fully survey all the required needs for developing the project. For every project Literature survey is the most important sector in software development process. Before developing the tools and the associated designing it is necessary to determine and survey the time factor, resource requirement, man power, economy, and company strength. Once these things are satisfied and fully surveyed, then the next step is to determine about the software specifications in the respective system such as what type of operating system the project would require, and what are all the necessary software are needed to proceed with the next step such as developing the tools, and the associated operations.

Extracting problematic API features from forum discussions - Y. Zhang and D. Hou - 2013

Software engineering activities often produce large amounts of unstructured data. Useful information can be extracted from such data to facilitate software development activities, such as bug reports management and documentation provision. Online forums, in particular, contain extensive valuable information that can aid in software development. However, no work has been done to extract problematic API features from online forums. In this paper, we investigate ways to extract problematic API features that are discussed as a source of difficulty in each thread, using

natural language processing and sentiment analysis techniques. Based on a preliminary manual analysis of the content of a discussion thread and a categorization of the role of each sentence therein, we decide to focus on a negative sentiment sentence and its close neighbors as a unit for extracting API features. We evaluate a set of candidate solutions by comparing tool-extracted problematic API design features with manually produced golden test data. Our best solution yields a precision of 89%. We have also investigated three potential applications for our feature extraction solution: (i) highlighting the negative sentence and its neighbors to help illustrate the main API feature; (ii) searching helpful online information using the extracted API feature as a query; (iii) summarizing the problematic features to reveal the "hot topics" in a forum.

III. SYSTEM ANALYSIS

3.1 Existing System and proposed system

Existing System:

In past system, there are several difficulties available in sentiment classification, which are all the limitations present in classical model. Specifically, in a bag-of-words model or polarity shifting because of function words and constructions in sentences. 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 and prepared a software-engineering specific sentiment dataset from a Stack Overflow dump. Despite their large amount of effort on fine-grained labeling to the tree structure of sentences, they reported negative results (low accuracy).

Proposed System:

In this system, 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. To address this problem, we utilize n-gram IDF, a theoretical extension of Inverse Document Frequency (IDF) proposed by Shirakawa et al. 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 hyper-parameter 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 (onehot encoding, rescaling, etc.). Using n-gram IDF and auto-sklearn tools, Wattanakriengkrai et al. outperformed the state-of-the-art self-admitted technical debt identification.

(a) 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.

(b) 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 <= 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.

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

IV. SYSTEM IMPLEMENTATION

SYSTEM IMPLEMENTATION

5.1 System Architecture:

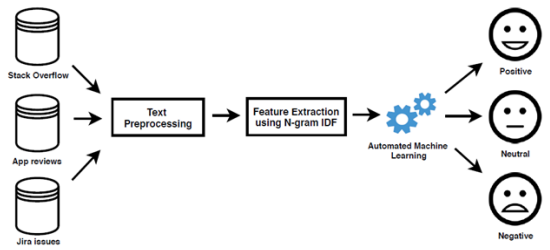
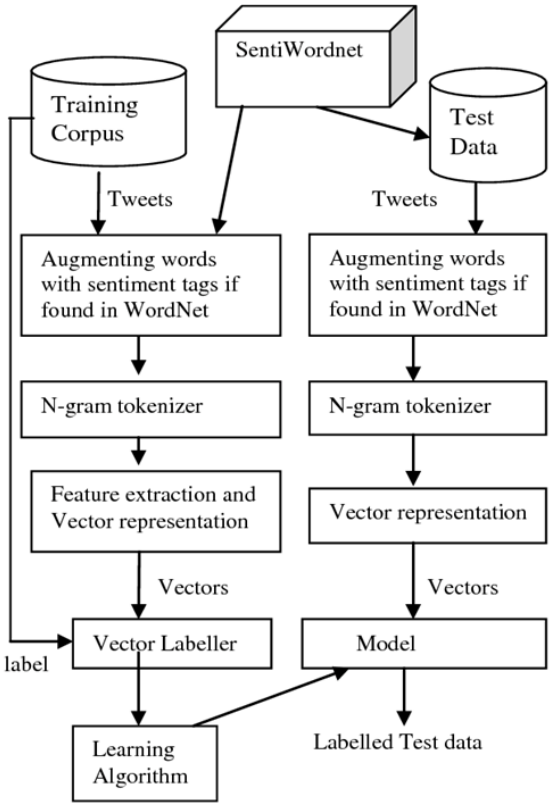


Fig: System Architecture

In the above architecture we can take input as Stack overflow, App reviews and Jira issues as input. We can text preprocess the given input and the data from the text preprocessing can process data in the Feature Extraction using N-gram IDF and the machine can convert the data into the emoji's like Positive, Neutral and Negative by using machine languages.

5.2 Flow chart of the algorithm selection:



In level-3 **SentiWordNet** is a lexical resource for opinion mining that assigns to each synset of **WordNet** three sentiment scores: positivity, negativity, and objectivity. Sentiwordnet can tweets directly to augmenting words with sentiment tags if found in wordnet (or) moved to Test data and can be tweet for augmenting words with sentiment tags if found in wordnet and then tweeted to N-gram tokenizer the data which is directly tweeted can perform Feature extraction and vector representation and the data which is moving from testdata can directly perform vector representation and then to vector labeller can move to learning algorithm and then to model. The data which can

tweeted from testdata from vector representation to model directly and the testdata can be labelled.

5.3 Dataset declaration and its attributes

Data set contains of the sentiment of the text which is positive or negative or neutral.

It contains a text attribute which has a data of 10,000 text test cases.

5.4 Algorithm Used

Naive Bayes Algorithm

Naive Bayes is a family of algorithms based on applying Bayes theorem with a strong(naive) assumption, that every feature is independent of the others, in order to predict the category of a given sample.

They are probabilistic classifiers, therefore will calculate the probability of each category using Bayes theorem, and the category with the highest probability will be output. Naive Bayes classifiers have been successfully applied to many domains, particularly Natural Language Processing(NLP).

We do have other alternatives when coping with NLP problems, such as Support Vector Machine (SVM) and neural networks. However, the simple design of Naive Bayes classifiers make them very attractive for such classifiers. Moreover, they have been demonstrated to be fast, reliable and accurate in a number of applications of NLP.

SKLearn:

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.

It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

- **NumPy**: Base n-dimensional array package
- **SciPy**: Fundamental library for scientific computing
- **Matplotlib**: Comprehensive 2D/3D plotting
- **IPython**: Enhanced interactive console
- **Sympy**: Symbolic mathematics
- **Pandas**: Data structures and analysis

Extensions or modules for SciPy care conventionally named SciKits. As such, the module provides learning algorithms and is named scikit-learn.

The vision for the library is a level of robustness and support required for use in production systems. This means a deep focus on concerns such as easy of use, code quality, collaboration, documentation and performance.

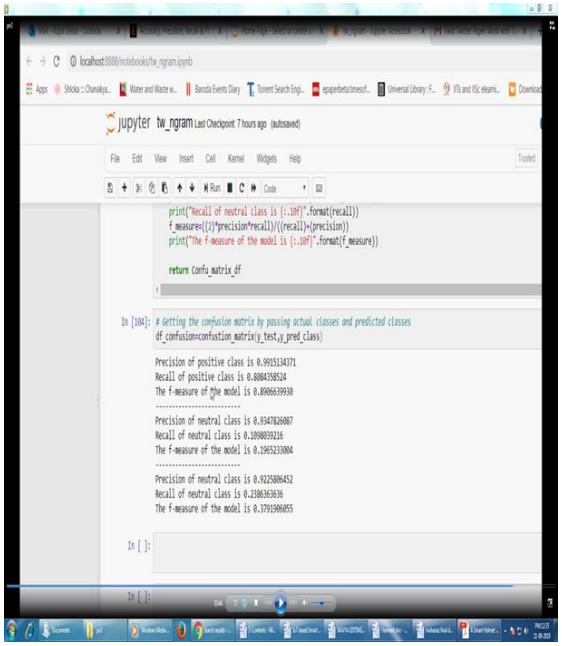
Although the interface is Python, c-libraries are leverage for performance such as numpy for arrays and matrix operations, LAPACK, LibSVM and the careful use of cython.

5.5 Results of the Algorithms

In this system, 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 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.

Reviews	No of Reviews	Extracted Reviews	Correct obtained	Precision(%)	Recall (%)	F-square (%)
Set 1	10000	2000	1535	76.75	15.35	25.58

V. SCREENSHOT:



VI. CONCLUSION

In this system, 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 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.

[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.