Noname manuscript No.
(will be inserted by the editor)

Label prediction on issue tracking systems using text mining

Jesús M. Alonso-Abad · Carlos López-Nozal · Jesús M. Maudes-Raedo · Raúl Marticorena-Sánchez

Received: date / Accepted: date

Abstract Issue tracking systems are overall change-management tools in software development. The issue-solving life cycle is a complex socio-technical activity that requires team discussion and knowledge sharing between members. In that process, issue classification facilitates an understanding of issues and their analysis. Issue tracking systems permit the tagging of issues with default labels (e.g., bug, enhancement) or with customized team labels (e.g., test-failures, performance). However, a current problem is that many issues in open-source projects remain unlabeled. The aim of this paper is to improve maintenance tasks in development teams, evaluating models that can suggest a label for an issue using its text comments. We analyze data on issues from several GitHub trending projects, first by extracting issue information and then by applying text mining classifiers, (i.e.,

Jesús M. Alonso-Abad
Universidad de Burgos, EPS Edificio A1, Avd. Cantabria s/n 09006, Burgos, España
Tel.: +34-947-258989
Fax: +34-947-258910
E-mail: jesusaa@ubu.es
ORCID: 0000-0001-5235-137X

Carlos López-Nozal
Universidad de Burgos, EPS Edificio A1, Avd. Cantabria s/n 09006, Burgos, España
Tel.: +34-947-258989
Fax: +34-947-258910
E-mail: clopezno@ubu.es
ORCID: 0000-0001-8462-212X

Jesús M. Maudes-Raedo
Universidad de Burgos, EPS Edificio A1, Avd. Cantabria s/n 09006, Burgos, España
Tel.: +34-947-258989
Fax: +34-947-258910
E-mail: jmaudes@ubu.es
ORCID: 0000-0001-8808-412X

Raúl Marticorena-Sánchez
Universidad de Burgos, EPS Edificio A1, Avd. Cantabria s/n 09006, Burgos, España
Tel.: +34-947-258989
Fax: +34-947-258910
E-mail: rmartico@ubu.es
ORCID: 0000-0002-2633-635X

Support Vector Machine and Naive Bayes Multinomial). The results suggest that very suitable classifiers may be obtained to label the issues or, at least, to suggest the most suitable candidate labels.

Keywords Text Classifier · Experimentation in Software Engineering · Issue Tracker System · Text Mining · Label Prediction

1 Introduction

Social software for an enterprise must have a mandatory functionality that allows users to tag/label content. In these information systems, a tag/label is a non-hierarchical keyword or term assigned to a piece of information (e.g. text comments, images, videos, bibliography). Labels are composed of three parts: a namespace, a predicate and a value. Metadata of this kind helps to describe an item and allows it to be found again by browsing or searching. A label is generally the informed choice of the system designer, viewers, or other collaborators, in accordance with each system. In this way, free form classification allows system designer, viewers and collaborators to manage what would otherwise be unwieldy amounts of information.

In software engineering, issue tracking systems are social systems where issues are the items. An issue must necessarily have: a title, a description and a state (open/closed). However, issues can also have optional data: assigned team member, milestone, label and team members' comments. Management of issues in the work flow is an essential part of the software development process. The development team reports an issue/task and sometimes to improve communication, a label with a type software task (e.g., implement, design, document, test, etc.) For example, *"label:bug=fix wrong reference check in FlattenIterable #4165"*. After the issue is opened the team discusses how to solve it, and finally the issue is closed.

Nowadays GitHub [1] is the largest open source community in the world (over 57 million repositories). Every project repository comes with the same powerful tools: version control system, integrated issue tracking, collaborative code review, team management within organizations, etc. Integrated issue tracking is set up with the following default set of labels: bug, duplicated, enhancement, help wanted, question, invalid and wontfix. The team can also customize labels. For instance: performance, test-failures, documentation, etc. Senior software engineers label issues to improve understandability in maintenance tasks. When projects reach a state of maturity in their development process, labeling an issue is invariably good practice, even if many issues are not labeled. Fig.1 shows a concrete case.

On the left side of the Fig.1 all issues are counted (325 open + 5258 closed), and on the right side, all non labeled issues (177 open + 1785 closed). In this case only 65% of the issues are labeled.

The aim of this paper is to improve the maintenance tasks for development teams, by providing a model that suggests a label for issues using their text comments. To do so, we analyze issue data on several GitHub trending projects. First, we extract issue information (title, description, labels, comments), using our own extraction programs (based on Node.js). Then, all the issue text information is concatenated into a single field or document.

[1] `https://github.com/`

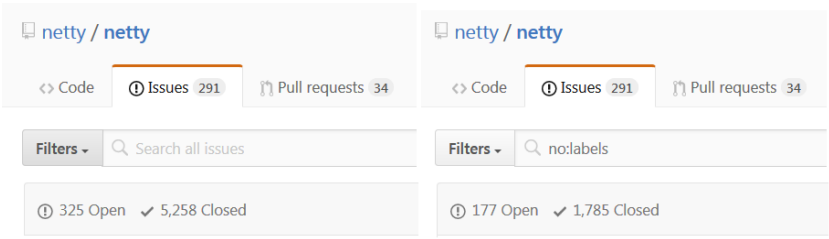


Fig. 1 Problem description: issues vs. not labeled issues on Github netty project

A text classifier is an algorithm that can predict the class c_i to which a document d belongs. The classifier predicts the class using a set of features or input values $\{x_1, ..., x_n\}$ for this document. In this case the class to be predicted is an issue label. For this purpose, we transform original datasets having several issues by using the binary relevance method [64]. In this way, a binary dataset (i.e., label vs. no label) is generated for each issue label. Then, state-of-the-art text mining classifiers, Support Vector Machine [14, 54] and Naive Bayes Multinomial [40], are applied on these binary datasets and the results are analyzed based on Area under ROC curve (AUROC) and F_1-score metrics.

Following the above-mentioned procedure, we wish to answer the following research question: Is it possible to tag an issue using text information with a high level of performance?

The structure of the rest of the article is presented as follows: Section 2 presents a review of related work. Section 3 describes the experiment design: goals, planning and some operational aspects, such as objects and data collection of the case study. Section 4 describes how the operation is prepared. Section 5 presents an analysis based on text classifiers of the data that have been collected. Section 6 shows the results and the final Section 7 presents the conclusions and future work.

2 Related work

2.1 Labels in social software projects

Treude and Storey [52] explored ways in which tagging, a lightweight social computing mechanism, is used to communicate matters of concern in the management of development tasks. They presented the results from two empirical studies of over 36 and 12 months, respectively, on how tagging has been adopted and what role it plays in the development processes of several professional development projects with over 1,000 developers. Their research showed that the tagging mechanism was eagerly adopted by the teams, and that it has become a significant part of many informal processes. Different kinds of tags are used by various stakeholders to categorize work items and to organize them. The tags are used to support task searching, articulation work, and information exchange. Implicit and explicit mechanisms have evolved to manage tag vocabulary. Their findings indicate that this lightweight informal tool, prevalent in the social computing domain, may play an important role in improving team-based software development practices. Some of their ideas give purpose to the automatic labeling of issues.

In a similar way, Xia et al. [63] proposed TagCombine, an automatic tag recommendation method, which analyzes objects in software information sites. TagCombine has three different components: the first is a multilabel ranking component that considers tag recommendation as a multi-label learning problem; the second is a similarity-based ranking component that recommends tags from similar objects; the third is a tag-term-based ranking component that considers the relations between different terms and tags, and recommends tags after analyzing the terms in the objects. The same research group in [61], used multi-label learning combining genetic algorithms, to classify crash reports and failures. These works cover a general spectrum, using multilabeling in software information sites and crash reports, although that it is outside the scope of the current work.

Cabot et al. in [9] explored the use of labels to categorize issues in Open-Source Software projects. In this paper they analyzed a population of more than three million GitHub projects and provided some insight into how those projects use labels. Their preliminary results revealed that, even if the label mechanism is scarcely used, using labels favors the resolution of issues. Their analysis also suggests that not all projects use labels in the same way (e.g., for some, labels are only a way to prioritize the project, while others use them to mark a temporal evolution as the development workflow advances). Further research is needed to characterize these label "families" precisely and to improve the ideal application scenarios for each of them. As the use of labels has a positive impact on the projects, there is a real need for automatic labeling. The previous research team presented GiLA [27]to complete the previous task; a tool that generates a set of visualizations to facilitate the analysis of issues in a GitHub project, depending on their label-based categorization. The visualizations are useful to see the most popular labels (and their relationships) in a project, and to identify the most active community members for those labels. It also permits comparisons between the evolution of label issues, for each category.

2.2 Text mining and bug classification problems

In [28] text mining is defined as a knowledge discovery process used to extract interesting and non-trivial patterns from natural language. The technique comprises multiple fields, such as information retrieval (IR), text analysis, natural language processing (NLP), information classification and database technology. Text mining techniques become more complex compared to data mining due to the unstructured and diffuse nature of natural language text.

In [30] a review of machine learning algorithms for text-documents classification is presented. The text-documents can be classified by three ways, unsupervised, supervised and semi supervised methods. From last few years, the task of automatic text classification have been extensively studied and rapid progress seems in this field, including the machine learning approaches such as Bayesian classifier, Decision Tree, K-nearest neighbor(KNN), Support Vector Machines (SVMs), Neural Networks, Latent Semantic Analysis, Rocchio's Algorithm, Fuzzy Correlation and Genetic Algorithms.

Bug classification problems is the broadest research field where text mining can be found, with extensive literature on bugs labeling in Bugs Tracking Systems.

Although the problem is outside the area of our work, we can establish some relationships.

For example, in [50], the authors wished to solve the problem of triaging duplicate bug reports. The techniques they used were Information Retrieval (IR) and Support Vector Machine (SVM).We can see how the use of machine learning, against a manual classification, could be a good solution to a well-known problem in software development.

In [62], bug resolution refers to the activity that developers perform to diagnose, fix, test, and document bugs during software development and maintenance. They would like to recommend the set of bug resolvers that could potentially contribute their knowledge to fix it, providing a solution to the problem of developer recommendation.

A comparison of the different machine learning techniques, which are used with bug reports and applied to natural language classification, can be also found in [47]. In general, in this problem, a first step to take into account is that we are facing a text-mining problem, where some of these techniques should be applied at an initial stage. In this work, we found a thorough study on the different approaches used in these problems.

In this line of work in [53], the authors used a prediction model based on decision trees, with the aim of characterizing and predicting blocking bugs in open source projects. Again, using textual description, but in this case, with the location and information (relationships) on the people who were involved. Their final results suggested a time improvement using these kinds of models. The use of additional information, such as the people involved, should perhaps be considered in later works.

In [58], the researchers attempted to solve the problem of classifying bug reports with the CoLUAtool. Using textual information (Chi-Square) and machine learning techniques (Naive Bayes, decision tree and logistic regression). They labeled the bug reports to predict the root causes of configuration bugs.

Finally, [24] presents a review and research discussion on emerging topics in software repositories mining over the past decade. They also identify applied machine learning strategies, current working topics, and future challenges for the improvement of enterprise decision-making systems. Sophisticated data selection and processing techniques require further discussion, due to the increasing variety and volume of information associated to issues and commits. They have observed that text mining techniques, such as supervised and unsupervised text classification, are emerging machine learning strategies.

We can conclude that natural language processing, machine learning and text mining could be the correct answer to the wider problem of issues labeling, as previous experience with more specific problems, such as bug classification and prediction, have provided good results.

2.3 Imbalanced classification

Problems in class imbalance appear when there are many more instances of some classes than others [12]. Imbalanced data sets are very common in certain fields such as bioinformatics (in DNA sequence recognition [22], gene recognition [6]),

medicine [17,33], engineering [36], finance [1], fraud detection [42], software development and quality predictions [16, 44, 51], etc. A more extended enumeration of real-life applications in this field is detailed in [33].

Specific methods are needed for the problem of imbalanced classes. First, standard classifiers focus on accuracy so the minority class could be ignored. Second, standard classification methods work under the assumption that the data sample is representative of the population, and this is not always true for this type of problem. Finally, classification methods for these problems must allow different costs, for errors coming from different classes. Galar et al. [21] classify recent techniques into four categories:

1. Algorithms. This category includes variants of learning algorithms of existing classifiers biased towards the more accurate learning on the minority class. Some examples are decision tree algorithm, insensitive to class sizes, such as Hellinger Distance Decision Tree (HDDT) [13], Class Confidence Proportion Decision Tree (CCPDT) [38] and an SVM classifier with different penalty constants for different classes [55].
2. Data. The main idea in this category is to pre-process the data to transform the imbalanced problem into a balanced one by manipulating the distribution of the classes. These algorithms are often used in combination with sets of classifiers. This category can be subdivided into methods that increase the number of examples of minority classes: Oversampling [5], SMOTE [10], Borderline-SMOTE [26] and Safelevel-SMOTE [8] among others; and methods that reduce the size of the majority class, such as random sub-sampling, this approach has been used with and without replacement [3]. These techniques can be applied together to increase the size of the minority class and at the same time, decrease the majority class.
3. Cost-sensitive learning. Traditional algorithms aim to increase accuracy by giving equal weight to examples of any kind, but cost-sensitive methods (cost-sensitive decision trees [37] or cost-sensitive neural networks [34]), assign a different cost to each class. The best-known methods in this category are cost-sensitive versions of AdaBoost: AdaCost [19, 29], AdaC1, AdaC2 and AdaC3 [49].
4. Ensemble learning. Classifier ensembles have often offered solutions to complex problems where standard methods have been insufficient. An approach with ensembles for imbalanced datasets is based on the use of data-level approaches: each base classifier is trained with a preprocessed dataset. As the data level approaches use random values, the preprocessed data sets and corresponding classifiers will be different. Another strategy is based on combining conventional ensemble methods (i.e., not imbalance-specific) with data-level approaches. Examples of this strategy are SMOTEBagging [57], SMOTEBoost [11] and RUS-Boost [45]. It is also possible to have ensembles combining classifiers obtained with different methods [32].

3 Experiment design

This experiment has been completed following the recommendations given in [60].

The main goal of the study presented here, as a Goal Question Metric (GQM) [4], is stated as follows:

- *Analyze* issues from open source GitHub projects:
  - https://github.com/ReactiveX/RxJava,
  - https://github.com/netty/netty,
  - https://github.com/google/guava,
  - https://github.com/iluwatar/java-design-patterns,
  - https://github.com/pushtorefresh/storio,
  - https://github.com/chartjs and
  - https://github.com/jquerytools/jquerytools,
- *in order to* assist tagging issues, study their information knowledge (text description, comments associated and ispullrequest)
- *with regard* to the performance indicators of the classification area under ROC curve (AUROC) and F_1-score,
- *from the point of view* of the researchers,
- *in the context* of open source software development.

The independent variables of the study are the text information (title, description, comments) related to the issues. Another piece of information is a boolean pull request field, which indicates whether the issue is automatically generated to request an integration, called pullrequest (true) in GitHub. The dependent variables are labels, which can take a binary value: "true" if tagged or "false" if otherwise (binary relevance strategy).

The following sections illustrate the experiment in order to achieve the research goal.

3.1 Data collection

The data of the study are issues from Open Source projects, obtained through the open source software repository GitHub (`https://github.com/`), particularly:

- Rxjava (https://github.com/ReactiveX/RxJava) is a library for composing asynchronous and event-based programs using observable sequences for the Java VM.
- Netty (https://github.com/netty/netty) is an asynchronous event-driven network application framework for rapid development of maintainable high-performance protocol servers and clients.
- The Guava project (https://github.com/google/guava) contains several of Google's core libraries that we rely on in our Java-based projects: collections, caching, primitives support, concurrency libraries, common annotations, I/O, string processing, and so forth.
- java-design-patterns (https://github.com/iluwatar/java-design-patterns) is a repository documentation of a catalog of design patterns. The readme Github file is used to define template patterns and GitHub code funcionality to show Java code examples.
- Storio (https://github.com/pushtorefresh/storio) is an API for SQLiteDatabase and ContentResolver for Android developers. An alternative approach to ORM, DAO or custom solutions. With storio's "blend" of good concepts (put/get/delete operation,thread safe, observer pattern, readable code...), ideas from different libraries and the user's experience in SoftwareDevelopment, they really think and hope that many Android Developers will use it to replace standard SQLiteDatabase and ContentResolver APIs.

- ChartJS (https://github.com/chartjs) is a community maintained open-source library that helps you easily visualize data using JavaScript. It supports 8 different chart types (including bars, lines, & pies), and they're all responsive.
- JQueryTools (https://github.com/jquerytools/jquerytools) is a collection of the most important user interface components for the web using JavaScript based on JQuery. These are tabs, accordions, tooltips, overlays, exposing effects and scrollables. They mainly focus on presenting information and visual appeal.

GitHub provides easily accessible data issues for experimentation. The collection of these massive amounts of data has been facilitated by GHTorrent [23], a repository that collects datasets to make them available to the public and the scientific community. The project workflow, undertaken by our research team at the University of Burgos included the following steps (see Fig.2):

1. Obtain data collected by GHTorrent from the GitHub repository and load them into MongoDB databases.
2. Filter the data according to needs and expand the data where possible. Some pre-processing of the data using JavaScript was completed during the database querying step and a number of Node.js scripts were used in several operations afterwards.
3. Pack the expanded data into a data table that will be inputted into a data mining tool to be used for a range of different techniques including text mining classification.

3.2 Analysis procedure: Text Mining Techniques

A text classifier is an algorithm that can predict the class c_i to which a document d belongs. The classifier predicts the class using a set of features or input values $\{x_1, ..., x_n\}$ for this document. The so called *Bag of Words* transformation is the most extended approach to represent documents in text classification. In this representation every word in the training corpus is considered as a feature x_j. For example, these features can be boolean. In that case x_j would take a value of 1, if the word w_j is present in the document, or 0 if otherwise. There are other non-boolean representations, as shown later in this section. The bag of words scheme leads to a sparse representation of documents with a very high number of features. In this paper all documents were preprocessed using the *Bag of Words* transformation.

To answer the research question two state-of-the-art text mining classifiers were used:

- SVM (Support Vector Machine) [14, 54]: this classifier works by building a hyperplane in the multidimensional space of the 'Bag of Words' (i.e., there is one dimension for each word in the documents collection). It classifies the instances (i.e., documents) among two given classes (SVM is a binary classifier). Documents are classified as belonging to one class or another depending on which side of the hyperplane they are located. Therefore, in our problem, the documents placed on one side of the hyperplane will be classified as having one of the issue labels under study, whereas the documents on the other side

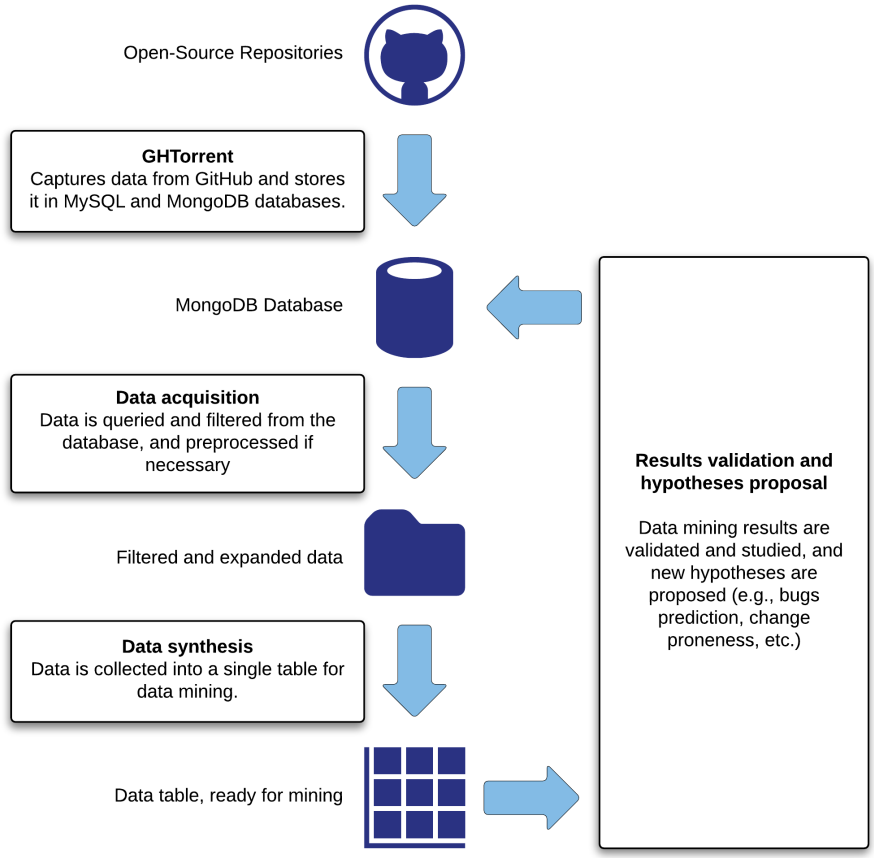


Fig. 2 Data collection procedure

will be classified as not having the aforementioned label. So, an SVM has to be built for each issue label. The support vectors are the instances closest to the decision boundary. It is said that the SVM hyperplane optimizes the margin. Intuitively, margin optimization means that the hyperplane distance to the decision surface enclosed by the support vectors from one of the classes is the same as the distance to the decision surface enclosed by the support vectors from the other class. Sometimes there are training instances or documents on the wrong side of that hyperplane (i.e., the training set is not linearly separable). A complexity parameter C is then used to compute an SVM that can correctly classify some of these instances by narrowing the margin.

– NBM (Naive Bayes Multinomial) [40]: this classifier is based on the Bayes rule. NBM assumes that the distribution of each feature given a class (i.e., issue), is a multinomial distribution. As in the 'Bag of Words' approach each feature represents the presence or absence of a word, these distributions are estimated from the ratio of documents that contain that word in that class.

NBM therefore classifies a document by computing the conditional probabilities of each class given the words in that document. Then the document is classified as belonging to the highest probability class.

3.3 Data set transformation

SVM and NBM with several different 'Bag of Words' transformations were tested for both classifiers:

- Boolean: this is the simplest transformation. If a word w_j is present in the document, the feature x_j the value 1, if not, it takes the value 0 as explained above.
- Raw TF (Term Frequency): in this transformation, the feature x_j takes the number of occurrences of word, w_j, in the document d_i (i.e., $f_{i,j}$) as its value. Although it seems to be a more sophisticated representation, it has the disadvantage that very frequent meaningless words often have the highest figures.
- Logarithmically scaled TF: this TF version is used to flatten the RawTF representation. There are a few standard logarithmic TF transformations. The one used in the present work is computed as $log(1 + f_{i,j})$ [59] for each word w_j in the document d_i. In this paper, 'Logarithmically TF' is represented as *log-TF*.
- IDF (Inverse Document Frequency): IDF gives the value $f_{i,j} log(N/N_j)$, where N is the total number of documents in the training dataset, and N_j is the number of documents containing the word w_j. Using this transformation, a word has a higher weight if infrequent in the document corpus.
- TF-IDF: both TF and IDF transformations are multiplied. Therefore it increases the feature values when a word is frequent in the document, but infrequent in the document corpus.

Orthogonally to the above transformations, words can be stemmed to reduce them to their morphological root [2]. Thus, words sharing the same root are represented by the same feature. In the present work two popular stemming algorithms were tested: Porter [43] and Lovins [39]. For the Lovins algorithm a version that tries to stem words until no further changes (i.e., *Iterated Lovins* [25]) was also used.

A list of stop-words can also be used, to get rid of meaningless words. In this work we used the stop list in the *Bow* library [41].

Term frequencies can be biased by the length of documents (i.e., a word is more likely to appear more times in a larger document). Therefore, when there is high variance in the length of documents, document normalization can help to achieve better results. Document normalization is done after the application of the corresponding Bag of Words transformation. Document normalization used in the present work is taken from [59]. It firstly computes the $l^2 - norm$ of each document (i.e., $\|d_i\|$) as well as the average $l^2 - norm$ from all documents represented by $\bar{d}$. Every value $x_{i,j}$ of feature f_j in document d_i is then multiplied by the quotient $\bar{d}/\|d_i\|$. So, if a document norm is close to the average, their features remain almost the same, but if otherwise, the $x_{i,j}$ values would increase or decrease, depending on whether $\|d_i\|$ is respectively smaller or larger than the average norm.

3.4 Evaluation of validity

For each GitHub project considered in the study, all documents with no labeled issues were discarded. Then, a binarization process was performed with the remaining documents (i.e., Binary Relevance method [64]). Binarization creates a dataset, D_i, for each issue label, l_i, used in the project. There are two classes in such datasets: documents with the label l_i and documents that do not use label l_i.

A dataset where the number of instances from one class is larger than the number of instances from the other class is known as an *imbalanced dataset*. The Binary Relevance transformation applied in the data under analysis produces unbalanced datasets, where only a few of the instances are labeled with the corresponding issue.

In binary imbalanced datasets, the minority class is denoted as positive and the majority class is denoted as negative. The True Positives (TP) are the positive instances correctly classified, the False Positives (FP) are the negative instances incorrectly classified, the True Negatives (TN) are the negative instances correctly classified, and the False Negatives (FN) are the positive instances incorrectly classified.

This comforms the well-known confusion matrix showed in Table. 1.

Table 1 Confusion Matrix for a Two-Class Problem

	Positive Prediction	Negative Prediction
Positive Class	True Positive (TP)	False Negative (FN)
Negative Class	False Positive (FP)	True Negative (TN)

Accuracy percentage is normally used to report the performance of a classifier (1).

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{1}$$

However, this measure is meaningless for imbalanced datasets, as a naive classifier that always predicts the majority class regardless the feature values, would reach an accuracy as high as the percentage of instances from the majority class. Therefore, other performance metrics are used for imbalanced classification.

Four ratios can be computed, according these previous four measures:

- The true positive rate (TPR), also called *recall*: $TPR = TP/(TP + FN)$
- The false negative rate (FNR): $FNR = FN/(TP + FN)$
- The true negative rate (TNR): $TNR = TN/(TN + FP)$
- The false positive rate (FPR): $FPR = FP/(TN + FP)$

Note that $TPR + FNR = 1$ and $TNR + FPR = 1$.

Some classifiers can output a probability estimation of the class to which the instance belongs. They usually predict that an instance is of a given class, if that probability is bigger than 0.5. However, it is possible to select another threshold, t, other than 0.5 (e.g., a t close to zero for the minority class will bias predictions toward positive class, while at the other extreme, a t close to one will mean that

the classifier is biased toward negative predictions). It is therefore possible to make a more or less conservative classifier, by varying t.

The NBM classifier used in the experiments in a straightforward way provides probability estimations. The SVM classifier can also output calibrated probabilities applying a logistic regression to the SVM output.

The ROC (Receiver - Operator Characteristics) curve plots FPR (in the X axis) against TPR (in the Y axis) with a threshold, t, ranging from 0 to 1. The area under ROC curve (i.e., AUROC) is a very popular metric used in Information Retrieval and Imbalanced Classification, because it is insensitive to changes in class distribution [20]. It is defined as (2):

$$AUROC = \frac{1 + TPR - FPR}{2} \tag{2}$$

The AUROC values range from 0.5 to 1 (0.5 represents random classification, 1 represents perfect classification). AUROC is one of two metrics used in this work to assess the performance of issues classifiers.

The other metric used in the experiments is the F_1-score. F_1-score is the harmonic mean of *precision* (3) and *recall* (4) (i.e., TPR) as is shown in (5). The precision is defined as the ratio of true positives among all test instances predicted as positives. F_1 ranges in $[0, 1]$. An F_1 value close to one means a good precision (i.e., when the classifier predicting an instance as positive tends to be right) and a good recall (i.e., the number of positive instances not predicted as positives is very small).

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{5}$$

One important difference between AUROC and F_1 is that AUROC can be computed at varying values of t. So, AUROC is insensitive to t. However, it is possible to find better F_1 values by properly tuning t.

The 10×10 stratified cross-validation [31] is performed over all dataset versions for the two classifiers (SVM, NBM) to obtain the AUROC and F_1-score results that will be analyzed.

When evaluating performance at predicting all the issues from an open source project, an average of one of these two metrics for all the issues used in the project is calculated. There are two ways of computing that average: either by $Macro - average$ (6) and $Micro - average$ (7) [46]. $Macro - average$ is simply the average of the metric value for each class (i.e., each issue label). Assuming that a metric is computed using the number of true positives, true negatives, false positives and false negatives as inputs, then $Micro - average$ is computed by applying the metric on the sums of the number of true positives, true negatives, false positives and false negatives for all classes.

$$macro - F_1 = \frac{1}{q} \sum_{\lambda=1}^{q} F_1(TP_\lambda, TN_\lambda, FP_\lambda, FN_\lambda) \tag{6}$$

$$micro - F_1 = F_1(\sum_{\lambda=1}^{q} TP_\lambda, \sum_{\lambda=1}^{q} TN_\lambda, \sum_{\lambda=1}^{q} FP_\lambda, \sum_{\lambda=1}^{q} FN_\lambda) \quad (7)$$

where λ is a label and $L = \{\lambda_j : j = 1..q\}$ is the set of all labels.

So, in $Micro-average$ each issue label has the same weight. However, $Micro-average$ favors bigger classes [48](i.e., most frequent issue labels). In this paper both $macro - F_1$ and $micro - F_1$ values are reported. For AUROC, only $macro - average$ is provided, as there is no accepted standard ROC for multi-class problems [35].

4 Execution

The data were extracted from the GitHub repositories. The set of labels assigned to each issue and specific to each project was obtained for training purposes. The data of each project were binarized, so that there is one dataset for each project-label pair.

Table 2 details the issue labels and their distribution in each project.

The experiments to find responses to the research question of this paper were designed using the WEKA data-mining tool [59]. As explained in section 3.4, all the documents in each project with no labeled issues were discarded and Binary Relevance transformation was then applied. So, a binary dataset D_i was generated for each issue label, l_i, used in the project. D_i will contain two classes (i.e., one with documents labeled as l_i, and another with unlabeled documents).

For each dataset D_i:

1. All the text features (i.e., *title, body, commentBodies*) are merged as a unique text feature.
2. The boolean feature *isPullRequest* is added to the text feature shown above, by adding the artificial keyword *isPullRequest* when it is equal to *True* and *isNotPullRequest* if otherwise.
3. All the text in the unique text feature is converted to lowercase.
4. The *Bag of Words* transformation is applied. Several configurations are tested, by picking one option from each item on the following list:
   - Either apply the Bow stop list [41] or apply no stop list.
   - 4 possible stemming approaches: apply the Porter algorithm [43], Lovins [39], an enhanced version of Lovins called *Iterated Lovins* implemented within WEKA (see Sec. 3), or apply no stemming algorithm at all.
   - Use one of these 5 representations: Boolean, raw-TF, log-TF, IDF or TF-IDF.
   - Use of document normalization or otherwise.

Therefore, $2 \times 4 \times 5 \times 2 = 80$ dataset versions are built for each issue label in each project that is analyzed. The two classifiers introduced in Sec. 3.2 were tested on these data versions. The WEKA implementation was used for NBM, while the *LIBLINEAR* implementation was used [18] for SVM, because it runs faster than WEKA's SVM native implementation and experiments in [18] have demonstrated that LIBLINEAR is very efficient on large sparse datasets.

Table 2 Projects with their labels and the number of issues using these labels. Labeled Instances = Number of instances of the dataset having at least one label. Num. = Number of issues that use the label in the row. Imbalance Ratio = IR = Num. minority Class instances/(Labeled Instances - Num. minority class instances).

Project	Labeled Instances	Issue Label	Num.	IR
dessign-pattern	53	bug	3	6.00%
		enhancement	3	6.00%
		help wanted	18	51.43%
		needs splitting	3	6.00%
		new pattern	16	43.24%
		question	4	8.16%
		technical	8	17.78%
guava	2028	cla no	11	0.55%
		cla yes	20	1%
		filler	216	11.92%
		package annotations	4	0.2%
		package base	187	10.16%
		package cache	92	4.75%
		package collect	396	24.26%
		package concurrent	113	5.9%
		package escape	7	0.35%
		package eventbus	47	2.37%
		package general	57	2.89%
		package hash	52	2.63%
		package io	90	4.64%
		package math	7	0.35%
		package net	44	2.22%
		package primitives	27	1.35%
		package reflect	33	1.65%
		package testing	21	1.05%
		status accepted	83	4.27%
		status duplicated	165	8.86%
		status fixed	445	28.11%
		status invalid	96	4.97%
		status obsolete	23	1.15%
		status research	193	10.52%
		status started	5	0.25%
		status won't fix	286	16.42%
		status working as intended	217	11.98%
		type addition	431	26.99%
		type api docs	84	4.32%
		type defect	297	17.16%
		type dev	21	1.05%
		type documentation	23	1.15%
		type enhancement	401	24.65%
		type other	36	1.81%
		type patch	2	0.1%
		type performance	56	2.84%
		type task	20	1%
		type tests	5	0.25%
rx-java	116	bug	16	16%
		enhancement	96	20.83%
		question	5	4.5%
storio	414	bug	35	9.23%
		enhancement feature	382	8.38%
		question	22	5.61%
		won't fix	6	1.47%

Continued on next page.

Continued from previous page.

Project	Labeled Instances	Issue Label	Num.	IR
netty	1098	android	1	0.09%
		cleanup	66	6.4%
		defect	524	91.29%
		discussion	3	0.27%
		documentation	14	1.29%
		duplicate	14	1.29%
		feature	215	24.35%
		improvement	205	22.96%
		needs info	33	3.1%
		not a bug	91	9.04%
		security	2	0.18%
		won't fix	59	5.68%
jquerytools	508	all tools	21	4.31%
		bug	89	21.24%
		confirmed	89	21.24%
		dateinput	60	13.39%
		deferred to 1.3	3	0.59%
		feature request	52	11.40%
		fix posted	51	11.16%
		high priority	23	4.74%
		incomplete	35	7.40%
		low priority	11	2.21%
		medium priority	10	2.01%
		need more info	37	7.86%
		no fix planned	9	1.80%
		not reproducible	26	5.39%
		overlay	52	11.40%
		question	37	7.86%
		rangeinput	16	3.25%
		scrollable	70	15.98%
		tabs	71	16.25%
		tests for tools 1.2.8	14	2.83%
		toolbox	28	5.83%
		tooltip	56	12.39%
		validator	79	18.41%
		website and docs	48	10.43%
chartjs	2691	api change: breaking	11	0.41%
		api change: compatible	20	0.75%
		category: bug	812	43.21%
		category: documentation	141	5.53%
		category: enhancement	558	26.16%
		category: infrastructure	109	4.22%
		category: new chart types	47	1.78%
		category: question	631	30.63%
		category: support	274	11.34%
		help wanted	424	18.70%
		implement as plug-in	55	2.09%
		inactive: duplicate	164	6.49%
		inactive: invalid	8	0.30%
		inactive: wontfix	68	2.59%
		needs investigation	145	5.70%
		needs test case	203	8.16%
		performance	11	0.41%
		priority: p0	35	1.32%
		priority: p1	327	13.83%
		priority: p2	210	8.46%
		time scale	42	1.59%
		version: 1.x	387	16.80%

For LIBLINEAR the *L2-regularized logistic regression primal* was set to enable the SVM to estimate class probabilities. As explained in Sec. 3 these estimates can be used to compute AUROC and optimize F_1-score. Feature normalization is also set in LIBLINEAR (as recommended in [18]). The complexity parameter was not tuned because LIBLINEAR is not very sensitive to this parameter, according to [18], and there are a lot of SVMs to compute in the experiments. The remaining parameters are kept to their default values.

4.1 Validation procedure

As explained in Section 3.4, the F_1-score is sensitive to the threshold t, used to classify a document according to the probabilities estimated by the classifier. Hence, a t-threshold was computed for each classifier and dataset version, in order to optimize the F_1-score. For optimization, the training data was divided into 3 folds, with 2 folds used for training and 1 for testing. From the point of view of the variants for addressing the problem of imbalanced datasets, this approach does no use any specific imbalance methods described in Section 2.3. On the other hand, to achieve our research question (i.e.; Is it possible to tag an issue using text information with a high level of performance?) state of art text mining algorithms are used. The results of the experiments in Section 6 show that is not necessary to use methods for unbalanced datasets, as without them, good results are already obtained. However, the imbalanced nature of the problem is taken into account by tuning the threshold for F_1-score maximization [65]. This is performed by using internal cross-validation in each trained classifier.

Finally, a 10×10 stratified cross-validation was performed over all the dataset versions for the 2 classifiers to obtain the AUROC and the F_1-score results that will be analyzed in the following subsections.

Datasets corresponding to issue labels used less than 10 times in a project were discarded. Their removal was motivated because a 10 stratified cross validation needs at least 10 different instances from a positive class (i.e., 10 labeled documents), to ensure each fold has at least one positive instance, unlike the positive instances in the other 9 folds. The discarded labels were:

- *question* from *rxjava*
- *android, discussion* and *security* from *netty*
- *package annotations, package math, package escape, status started, type patch* and *type tests* from *guava* project
- *bug, enhancement, need splitting, question* and *technical* from *design-patterns* project
- *wont-fix* from *storio*
- *inactive: invalid* from *chartjs*
- *no fix planned* and *deferred to 1.3* from *jquerytools*

In addition, the *filler* label from the *guava* project was also discarded, as there were configurations that can correctly predict 100% of the tested instances. The inclusion of this label would be unfair as all *filler* tags were automatically generated during the issue tracker migration process from GoogleCode project repository. All these issues have the same date of creation, author and title and description, (i.e., *"Filler for non-existent Google Code issue 1356. This issue only exists to ensure*

that GitHub issues have the same IDs they had on Google Code. Please ignore it.").

5 Analysis

As explained in Sec. 3.4, the data for each project are binarized, producing imbalanced datasets. So, there is one dataset for each project-label pair. The class of each one of these datasets can take one of two possible values (i.e., the document was or was not labeled with the corresponding issue), and typically, there will be many more unlabeled documents in all these datasets than there will be documents labeled with a given issue. Table 2 summarizes the data and shows several descriptive statistics: number of labeled issues, number of issues tagged with a specific label and an imbalance ratio for each project-label pair.

In Sec. 3.3 Bag of Words transformations were presented: Boolean, Raw Term Frequency (Raw TF), Logarithmic Term Frequency (TF), and Inverse Document Frequency (IDF). Other text pre-processing techniques were also presented such as stemming algorithms, list of meaningless words, and document normalization. When we applied these transformations and techniques, we generated several versions for each dataset corresponding to each project-label pair. We tested the experiment with the AUROC and F_1 obtained in cross validation.

Tables 4 and 5 summarize the text classifiers and configurations that predict the best AUROC or F_1, for each label, and are ordered by AUROC and by F_1, respectively.

6 Results

Two strategies are used in order to answer the research question: Is it possible to tag an issue using text information with a high level of performance? One based on AUROC optimization, and another based on F_1-macro optimization:

1. For each binary problem the best average AUROC configuration from the 10×10 cross validation is selected (see Table 4 in Appendix). Then, for each project a classifier consisting of these best configurations is built. Table 3 shows the values of the metrics described in Section 3.4 for the classifiers in the three columns under the headings *From best AUROCs*.
2. The same procedure is repeated, but using the best average F_1-score for each issue label (see Table 5 in Appendix). The three last columns of the table show the metric values for this second strategy.

Three of the seven projects under analysis achieved an average AUROC greater than 0.9 for both strategies, which is a very high value. The other four projects achieved values of around 0.8. The average AUROC from the seven projects was 0.9002 when selecting the best AUROC configurations, and 0.8534 when selecting the best F_1 configurations. Therefore, with regard to AUROC values, the answer to research question is that very accurate issue-label classifiers can be obtained by taking into account the textual data in the issues.

F_1-macro tends to obtain slightly more pessimistic values than F_1-micro, because F_1-macro gives the same weight to all labels. So, minority labels, which are

typically more difficult to classify, make F_1-macro values to decrease compared to F_1-micro. F_1-score values are also very interesting in the three first projects in the table and the two last. In particular *rxjava*, *charjs* and *jquerytools* show values greater than 0.8 with both strategies. On the other hand, the *storio* and the *netty* projects showed poor F_1-scores. Therefore, there was more variance in F1-score values than in the AUROC values. However, the average F_1-macro from the seven projects was 0.6797 when selecting the best AUROCs, and 0.7080 when selecting the best F_1s. So, in general, it is possible to get very suitable classifiers to label the issues, or at least to suggest the most suitable candidate labels to the user.

Table 3 AUROC, F_1-Macro and F_1-Micro of each project. Results in the 3 first columns are computed from the best AUROC configuration for each issue. Results in the 3 last columns come from the best F_1 configurations for each issue. Last line reports the average values.

	From best AUROCs			From best F_1s		
Project	AUROC	F_1-macro	F_1-micro	AUROC	F_1-macro	F_1-micro
design-patterns	0.9280	0.6837	0.7165	0.9198	0.7308	0.7638
guava	0.9584	0.6224	0.6843	0.9317	0.6595	0.7193
rxjava	0.9956	0.8263	0.8668	0.9343	0.8585	0.8729
storio	0.8319	0.3233	0.3228	0.8103	0.3532	0.3572
netty	0.8218	0.3843	0.5504	0.7910	0.4200	0.5812
chartjs	0.8853	0.9572	0.9597	0.7850	0.9632	0.9654
jquerytools	0.8802	0.9605	0.9617	0.8018	0.9711	0.9719
average	0.9002	0.6797	0.7232	0.8534	0.7080	0.7474

7 Conclusions and future works

Issue tracking systems are overall change-management tools used as part of the software development process. Issue classification using labels is an important maintenance task that development software teams have to complete. It facilitates an understanding of the issue and its analysis. When we analyzed open source projects in GitHub, we observed many issues with no associated label.

In this study, we have selected seven trending projects in GitHub and built classification models based on issue information, text description and comments. Text information was preprocessed with text data-mining techniques and information retrieval. Then we evaluated the performance of classifiers with AUROC and F_1 metrics. Our results have led us to conclude that very suitable classifiers may be obtained to label the issues or at least to suggest the most suitable candidate labels.

We have identified one limitation in the results of this experiment. Open source software repositories such as GitHub provide a rich and varied source of issues to mine. Their open source policy welcomes contributors with very different skill sets and experience levels and the absence or low levels of standardized workflows (as opposed to the more structured workflow patterns experienced when using, for instance, a branch-per-task branching policy [7]).

From the point of view of functionality, an issue can be tagged with multiple labels. In this work, we have solved this problem with a binary relevance approach, which means that one classifier is built for each label. This approach does not use

possible correlations existing between labels. As future work, we suggest the application of multi-label classifiers [64] that take advantage of such correlations. The dataset of this work infrequently assigns multiple labels to an issue. We think that it would be interesting to repeat the experiment with other projects, to validate multi-label classifiers, including both open source and private projects. Furthermore, we have considered defining time windows to characterize different project maturity levels, to repeat the experiment, and to study the feasibility and the evolution of label predictions.

Acknowledgements We would like to thank the Ministerio de Economía y Competitividad of the Spanish Government for financing the project TIN2015-67534-P (MINECO/FEDER, UE) and the Junta de Castilla y León for financing the project BU085P17 (JCyL/FEDER, UE) both co-financed from European Union European Regional Development Fund (ERDF/FEDER) funds. We gratefully acknowledge the support of NVIDIA Corporation for the donation of TITAN Xp GPUs used for this research.

References

1. Anil Kumar, R. and Ravi, V. Predicting credit card customer churn in banks using data mining. *International Journal of Data Analysis Techniques and Strategies*, 1(1):4–28, 2008.
2. Anjali, M., Jivani, G.: A comparative study of stemming algorithms. Int. J. Comp. Tech. Appl. **2**(6), 1930–1938 (2011)
3. Barandela, R., Valdovinos, RM., and Sánchez, JS. New applications of ensembles of classifiers. *Pattern Analysis & Applications*, 6(3):245–256, 2003.
4. Basili, V., Caldiera, G., Rombach, D.H.: The goal question metric approach. In: J. Marciniak (ed.) Encyclopedia of Software Engineering. Wiley (1994). URL `https://docweb.lrz-muenchen.de/cgi-bin/doc/nph-webdoc.cgi/000110A/http/scholar.google.de/scholar=3fhl=3dde&lr=3d&cluster=3d4068380033007143449`
5. Batista, G.E., Prati, R.C. and Monard, M.C. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1):20–29, 2004.
6. Batuwita, R. and Palade, V. micropred: effective classification of pre-mirnas for human mirna gene prediction. *Bioinformatics*, 25(8):989–995, 2009.
7. Berczuk, S., Appleton, B.: Software Configuration Management Patterns: Effective Teamwork, Practical Integration, edition: 01 edn. ADDISON WESLEY LONGMAN INC DIV PEARSON SUITE 300, Boston (2002)
8. Bunkhumpornpat, C., Sinapiromsaran, K. and Lursinsap, C. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining(PAKDD09)*, volume 5476 of *Lecture Notes on Computer Science*, pages 475–482. Springer-Verlag, 2009.
9. Cabot, J., Izquierdo, J.L.C., Cosentino, V., Rolandi, B.: Exploring the use of labels to categorize issues in Open-Source Software projects. In: 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER), pp. 550–554 (2015). DOI 10.1109/SANER.2015.7081875
10. Chawla, N.V., Bowyer,K.W., Hall, L.O. and Kegelmeyer, W.P. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.
11. Chawla, N.V., Lazarevic, A., Hall, L.O. and Bowyer, K.W. Smoteboost: Improving prediction of the minority class in boosting. In *7th European Conference on Principles and Practice of Knowledge Discovery in Databases(PKDD 2003)*, pages 107–119, 2003.
12. Chawla, N,V., Japkowicz, N., and Kotcz, A. Editorial: special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations Newsletter*, 6(1):1–6, 2004.
13. Cieslak, David A. and Chawla, Nitesh V. Learning decision trees for unbalanced data. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*, ECML PKDD '08, pages 241–256, Berlin, Heidelberg, 2008. Springer-Verlag.

14. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995). DOI 10.1023/A:1022627411411. URL http://dx.doi.org/10.1023/A:1022627411411
15. Díez-Pastor, José F., Rodríguez, Juan J.,García-Osorio, César. and Kuncheva, Ludmila I. Random balance: Ensembles of variable priors classifiers for imbalanced data. *Knowledge-Based Systems*, 85:96 – 111, 2015.
16. Drown, D. J., Khoshgoftaar, T. M. and Seliya, N. Evolutionary sampling and software quality modeling of high-assurance systems. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 39(5):1097–1107, Sep. 2009.
17. Eskildsen, Simon Fristed., Coupé, Pierrick., Fonov, Vladimir. and Collins, D. Louis. Detecting alzheimer's disease by morphological mri using hippocampal grading and cortical thickness. In Esther Bron, Marion Smits, John van Swieten, Wiro Niessen, and Stefan Klein, editors, *Challenge on Computer-Aided Diagnosis of Dementia based on structural MRI data*, pages 38–47, 9 2014.
18. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research **9**, 1871–1874 (2008)
19. Fan, Wei., Salvatore J. Stolfo, Junxin Zhang, and Philip K. Chan. Adacost: Misclassification cost-sensitive boosting. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 97–105, San Francisco, CA, USA, 1999.Morgan Kaufmann Publishers Inc.
20. Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters **27**(8), 861–874 (2006). DOI 10.1016/j.patrec.2005.10.010. URL http://dx.doi.org/10.1016/j.patrec.2005.10.010
21. Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *Systems, Man, and Cybernetics, Part C: Applications and Reviews,IEEE Transactions on*, 42(4):463 –484, july 2012.
22. García-Pedrajas, Nicolás., Pérez-Rodríguez, Javier., García-Pedrajas, María D., Ortiz-Boyer, Domingo. and Fyfe, Colin. Class imbalance methods for translation initiation site recognition in dna sequences. *Knowl.-Based Syst.*, 25(1):22–34, 2012.
23. Gousios, G.: The ghtorrent dataset and tool suite. In: Proceedings of the 10th Working Conference on Mining Software Repositories, MSR '13, pp. 233–236. IEEE Press, Piscataway, NJ, USA (2013). URL http://dl.acm.org/citation.cfm?id=2487085.2487132
24. Güemes-Peña, D., López-Nozal, C., Marticorena-Sánchez, R., Maudes-Raedo, J.: Emerging topics in mining software repositories. Progress in Artificial Intelligence **7**(3), 237–247 (2018). DOI 10.1007/s13748-018-0147-7. URL https://doi.org/10.1007/s13748-018-0147-7
25. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: An update. SIGKDD Explor. Newsl. **11**(1), 10–18 (2009). DOI 10.1145/1656274.1656278. URL http://doi.acm.org/10.1145/1656274.1656278
26. Han, H., Wang, W.Y. and Mao, B.H. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *2005 International Conference on Intelligent Computing (ICIC05)*, volume 3644 of *Lecture Notes on Computer Science*, pages 878–887. Springer-Verlag, 2005.
27. Izquierdo, J.L.C., Cosentino, V., Rolandi, B., Bergel, A., Cabot, J.: Gila: Github label analyzer. In: 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER), pp. 479–483 (2015). DOI 10.1109/SANER.2015.7081860
28. Irfan, R., King, C., Grages, D., Ewen, S., Khan, S., Madani, S., Kolodziej, J., Wang, L., Chen, D., Rayes, A., Tziritas, N., Xu, C.-Z., Zomaya, A., Alzahrani, A., and Li, H. : A survey on text mining in social networks. In: Knowledge Engineering Review 30, 2 (2015), 157–170.
29. Joshi, M.V., Kumar, V, and Agarwal, R.C. Evaluating boosting algorithms to classify rare classes: Comparison and improvements. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 257–264. IEEE, 2001.
30. Khan, A., Baharudin, B., Lee, L. H., Khan, K., and Tronoh, U. T. P.: A review of machine learning algorithms for text-documents classification. In: Journal of Advances In Information Technology Vol 1, 1 (2010), 4–18.
31. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'95, pp. 1137–1143. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1995). URL http://dl.acm.org/citation.cfm?id=1643031.1643047

32. Kotsiantis, S. B. and Pintelas, P. E. Mixture of expert agents for handling imbalanced data sets. *Annals of Mathematics, Computing & Teleinformatics*, 1(1):46–55, 2003.
33. Krawczyk, Bartosz., Galar, Mikel., Jeleń, Łukasz. and Herrera, Francisco. Evolutionary undersampling boosting for imbalanced classification of breast cancer malignancy. *Applied Soft Computing*, 38:714 – 726, 2016.
34. Kukar, M. and Kononenko, I. Cost-sensitive learning with neural networks. In *Proceedings of the 13th European conference on artificial intelligence (ECAI-98)*, pages 445–449. Citeseer, 1998.
35. Lachiche, N., Flach, P.A.: Improving accuracy and cost of two-class and multi-class probabilistic classifiers using roc curves. In: ICML (2003)
36. Liao, T.W. Classification of weld flaws with imbalanced class data. *Expert Systems with Applications*, 35(3):1041 – 1052, 2008.
37. Ling, C.X., Sheng, V.S. and Yang, Q. Test strategies for cost-sensitive decision trees. *Knowledge and Data Engineering, IEEE Transactions on*, 18(8):1055–1067, 2006.
38. Liu, Wei., Chawla, Sanja., Cieslak, David A. and Chawla, Nitesh V. A robust decision tree algorithm for imbalanced data sets. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2010*, pages 766–777, 2010.
39. Lovins, J.B.: Development of a stemming algorithm. Mechanical Translation and Computational Linguistics **11**, 22–31 (1968)
40. McCallum, A., Nigam, K.: A comparison of event models for naive bayes text classification. In: AAAI-98 workshop on learning for text categorization, vol. 752, pp. 41–48 (1998)
41. McCallum, A.K.: Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering (1996). Http://www.cs.cmu.edu/ mccallum/bow
42. Phua, C.,Alahakoon, D., and Lee, V. Minority report in fraud detection: classification of skewed data. *ACM SIGKDD Explorations Newsletter*, 6(1):50–59, 2004.
43. Porter, M.: An algorithm for suffix stripping. Program **14**(3), 130–137 (1980)
44. Seiffert,C., Khoshgoftaar, T. M., and Van Hulse, J. Improving software-quality predictions with data sampling and boosting. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 39(6):1283–1294, Nov 2009.
45. Seiffert, C., Khoshgoftaar, T.M.,Van Hulse, J. and Napolitano, A. Rusboost: A hybrid approach to alleviating class imbalance. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 40(1):185–197, 2010.
46. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. **34**(1), 1–47 (2002). DOI 10.1145/505282.505283. URL `http://doi.acm.org/10.1145/505282.505283`
47. Sohrawardi, S.J., Azam, I., Hosain, S.: A comparative study of text classification algorithms on user submitted bug reports. In: Digital Information Management (ICDIM), 2014 Ninth International Conference on, pp. 242–247. IEEE (2014)
48. Sokolova, M., Lapalme, G.: A systematic analysis of performance measures for classification tasks. Inf. Process. Manage. **45**(4), 427–437 (2009). DOI 10.1016/j.ipm.2009.03.002. URL `http://dx.doi.org/10.1016/j.ipm.2009.03.002`
49. Sun ,Y., Kamel, M., Wong, A., and Wang, Y. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40:3358–3378, 2007.
50. Sun, C., Lo, D., Wang, X., Jiang, J., Khoo, S.C.: A discriminative model approach for accurate duplicate bug report retrieval. In: Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE '10, pp. 45–54. ACM, New York, NY, USA (2010). DOI 10.1145/1806799.1806811. URL `http://doi.acm.org/10.1145/1806799.1806811`
51. Sun,Z., Song, Q. and Zhu, X. Using coding-based ensemble learning to improve software defect prediction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1806–1817, Nov 2012.
52. Treude, C., Storey, M.A.: Work item tagging: Communicating concerns in collaborative software development. IEEE Transactions on Software Engineering **38**(1), 19–34 (2012). DOI 10.1109/TSE.2010.91
53. Valdivia Garcia, H., Shihab, E.: Characterizing and predicting blocking bugs in open source projects. In: Proceedings of the 11th Working Conference on Mining Software Repositories, MSR 2014, pp. 72–81. ACM, New York, NY, USA (2014). DOI 10.1145/2597073.2597099. URL `http://doi.acm.org/10.1145/2597073.2597099`
54. Vapnik, V.N.: The Nature of Statistical Learning Theory (Information Science and Statistics). Springer (1999)

55. Veropoulos, K., Campbel, C. and Cristianini, N. Controlling the sensitivity of support vector machines. In *Proceedings of the International Joint Conference on AI*, pages 55–60, 1999.
56. Visa, S. and Ralescu, A. Issues in mining imbalanced data sets - a review paper. In *Proceedings of the Sixteen Midwest Artificial Intelligence and Cognitive Science Conference*, pages 67–73, 2005.
57. Wang, S and Yao, X. Diversity analysis on imbalanced data sets by using ensemble models. In *IEEE Symposium Series on Computational Intelligence and Data Mining(IEEE CIDM 2009)*, pages 324–331, 2009.
58. Wen, W., Yu, T., Hayes, J.H., undefined, undefined, undefined, undefined: Colua: Automatically predicting configuration bug reports and extracting configuration options. 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE) **00**(undefined), 150–161 (2016). DOI doi.ieeecomputersociety.org/10.1109/ISSRE.2016.29
59. Witten, I.H., Frank, E., A.Hall, M.: Data Mining: Practical Machine Learning Tools and Techniques, 3nd edn. Morgan Kaufmann, San Francisco (2011)
60. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in software engineering. Springer Science & Business Media (2012)
61. Xia, X., Feng, Y., Lo, D., Chen, Z., Wang, X.: Towards more accurate multi-label software behavior learning. 2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE) **00**(undefined), 134–143 (2014). DOI doi.ieeecomputersociety.org/10.1109/CSMR-WCRE.2014.6747163
62. Xia, X., Lo, D., Wang, X., Zhou, B.: Accurate developer recommendation for bug resolution. In: Proceedings of the 20th Working Conference Reverse Engineering (2013)
63. Xia, X., Lo, D., Wang, X., Zhou, B.: Tag recommendation in software information sites. In: Proceedings of the 10th Working Conference on Mining Software Repositories, MSR '13, pp. 287–296. IEEE Press, Piscataway, NJ, USA (2013). URL `http://dl.acm.org/citation.cfm?id=2487085.2487140`
64. Zhang, M., Zhou, Z.: A review on multi-label learning algorithms. IEEE Trans. Knowl. Data Eng. **26**(8), 1819–1837 (2014). DOI 10.1109/TKDE.2013.39. URL `https://doi.org/10.1109/TKDE.2013.39`
65. Zou, Q., Xie, S., Lin, Z., Wu, M. and Ju, Y. Finding the best classification threshold in imbalanced classification. *Big Data Research*, 5:2 – 8, 2016. Big data analytics and applications.

A Appendix

Table 4 Methods and configurations that predict the best AUROC for each label. The table is ordered by AUROC. stop-w = stop word list is used, norm= document normalization is applied, process=word transformation (*bool* if Term Frequency is not used, *RawTF* if raw Term Frecuency is applied, *logTF* means logarithmic Term Frecuency, *IDF* means Inverse Document Frequency and *TF-IDF* means IDF applied over logarithmic TF).

dataset	label	method	stop-w	stemmer	norm	process	AUROC
rxjava	bug	SVM	yes	no	yes	bool	1.0000
guava	package-eventbus	NBM	yes	iterlovins	no	TF-IDF	0.9989
guava	cla-no	SVM	no	no	no	IDF	0.9989
guava	cla-yes	SVM	yes	no	no	logTF	0.9979
guava	package-reflect	NBM	no	no	yes	TF-IDF	0.9978
guava	package-hash	SVM	no	iterlovins	yes	logTF	0.9958
guava	package-cache	SVM	yes	no	yes	TF-IDF	0.9957
guava	package-testing	NBM	yes	no	yes	IDF	0.9927
guava	status-invalid	SVM	yes	no	yes	bool	0.9918
rxjava	enhancement	SVM	yes	no	yes	bool	0.9912
guava	package-io	SVM	yes	iterlovins	no	logTF	0.9899
chartjs	performance	NBM	yes	porter	yes	TF-IDF	0.9885
guava	status-duplicate	SVM	no	no	yes	bool	0.9866
guava	package-net	NBM	no	lovins	yes	IDF	0.9859
chartjs	time-scale	SVM	yes	porter	no	logTF	0.9854
jquerytools	validator	SVM	no	porter	no	logTF	0.9852
guava	status-fixed	SVM	yes	no	yes	bool	0.9821
guava	status-wont-fix	SVM	yes	no	yes	bool	0.9796
guava	package-concurrent	NBM	yes	no	yes	TF-IDF	0.9770
guava	type-task	SVM	no	no	yes	bool	0.9724
guava	status-work-as-intended	SVM	yes	no	yes	bool	0.9722
design-p	new-pattern	SVM	yes	no	yes	RawTF	0.9704
guava	package-primitives	NBM	yes	no	yes	IDF	0.9696
jquerytools	overlay	SVM	no	iterlovins	no	logTF	0.9676
jquerytools	dateinput	SVM	yes	no	no	logTF	0.9670
jquerytools	tooltip	SVM	yes	iterlovins	yes	logTF	0.9601
guava	type-api-docs	SVM	yes	iterlovins	yes	bool	0.9546
guava	package-collect	SVM	yes	no	yes	logTF	0.9530
guava	type-dev	SVM	no	iterlovins	yes	bool	0.9502
jquerytools	tabs	SVM	no	lovins	yes	logTF	0.9497
guava	type-documentation	SVM	yes	lovins	yes	logTF	0.9476
chartjs	category-infrastructure	SVM	yes	no	yes	RawTF	0.9456
netty	documentation	SVM	yes	no	yes	logTF	0.9437
jquerytools	rangeinput	SVM	yes	porter	yes	RawTF	0.9423
jquerytools	scrollable	SVM	yes	no	no	logTF	0.9423
guava	package-base	SVM	yes	no	yes	TF-IDF	0.9389
jquerytools	feature-request	SVM	no	porter	yes	logTF	0.9367
chartjs	implement-as-plug-in	SVM	no	no	yes	bool	0.9360
jquerytools	tests-for-tools-1.2.8	SVM	yes	porter	no	IDF	0.9356
guava	status-obsolete	SVM	no	porter	yes	bool	0.9301
guava	type-defect	SVM	yes	iterlovins	yes	logTF	0.9274
guava	package-general	SVM	no	no	yes	bool	0.9271
guava	type-addition	SVM	no	no	no	logTF	0.9263
chartjs	api-change-breaking	SVM	yes	porter	yes	RawTF	0.9253
chartjs	category-new-chart-types	SVM	yes	porter	yes	RawTF	0.9246
chartjs	inactive-wontfix	SVM	yes	porter	yes	RawTF	0.9238
chartjs	category-documentation	SVM	yes	porter	yes	RawTF	0.9222
guava	type-performance	SVM	no	no	yes	bool	0.9213
jquerytools	website-and-docs	SVM	yes	no	yes	RawTF	0.9194
guava	status-research	SVM	no	iterlovins	yes	logTF	0.9137
chartjs	inactive-duplicate	SVM	yes	no	no	logTF	0.9067
chartjs	needs-test-case	SVM	yes	no	yes	logTF	0.9020
guava	type-other	NBM	yes	porter	yes	logTF	0.9009
chartjs	version-1.x	SVM	yes	iterlovins	yes	logTF	0.8955
storio	question	SVM	no	lovins	no	logTF	0.8893
design-p	help-wanted	NBM	no	iterlovins	yes	TF-IDF	0.8856
jquerytools	not-reproducible	SVM	no	lovins	yes	bool	0.8813
chartjs	category-question	SVM	no	lovins	yes	logTF	0.8778
chartjs	help-wanted	SVM	no	no	yes	logTF	0.8739
guava	type-enhancement	SVM	yes	no	yes	bool	0.8718
chartjs	needs-investigation	SVM	no	no	yes	logTF	0.8671
guava	status-accepted	SVM	yes	no	yes	bool	0.8639
netty	cleanup	SVM	yes	iterlovins	yes	logTF	0.8638

Continued on next page.

Continued from previous page.

dataset	label	method	stop-w	stemmer	norm	process	AUROC
jquerytools	toolbox	SVM	yes	lovins	yes	RawTF	0.8539
jquerytools	all-tools	SVM	yes	no	yes	RawTF	0.8539
netty	feature	NBM	no	iterlovins	yes	bool	0.8518
chartjs	priority-p1	SVM	no	no	yes	RawTF	0.8516
chartjs	category-bug	SVM	no	porter	yes	logTF	0.8483
jquerytools	confirmed	SVM	yes	iterlovins	yes	RawTF	0.8481
jquerytools	bug	SVM	yes	no	yes	RawTF	0.8478
chartjs	category-support	SVM	no	porter	yes	logTF	0.8367
jquerytools	incomplete	SVM	no	iterlovins	yes	bool	0.8326
netty	defect	SVM	no	porter	yes	logTF	0.8323
jquerytools	need-more-info	SVM	yes	lovins	no	logTF	0.8293
chartjs	priority-p2	SVM	no	lovins	yes	logTF	0.8288
jquerytools	question	SVM	no	porter	yes	bool	0.8278
netty	not-a-bug	SVM	no	no	no	logTF	0.8226
chartjs	category-enhancement	SVM	no	no	yes	RawTF	0.8225
jquerytools	medium-priority	SVM	no	lovins	yes	bool	0.8132
netty	needs-info	SVM	no	lovins	yes	logTF	0.8099
storio	bug	SVM	no	lovins	yes	bool	0.8081
netty	wont-fix	SVM	no	lovins	yes	bool	0.8055
netty	improvement	NBM	no	no	yes	bool	0.8015
jquerytools	high-priority	SVM	yes	no	no	IDF	0.8007
storio	enhancement-feature	SVM	yes	lovins	yes	TF-IDF	0.7982
chartjs	priority-p0	SVM	no	porter	no	IDF	0.7971
jquerytools	fix-posted	SVM	yes	iterlovins	no	IDF	0.7448
chartjs	api-change-compatible	SVM	yes	no	no	IDF	0.7320
jquerytools	low-priority	NBM	no	lovins	yes	RawTF	0.7244
netty	duplicate	SVM	yes	no	no	IDF	0.6652

Table 5 Methods and configurations that predict the best optimized F_1 score for each label. The table is ordered by F_1. stop-w = stop word list is used, norm = document normalization is applied, process=word transformation (*bool* if Term Frequency is not used, *RawTF* if raw Term Frecuency is applied, *logTF* means logarithmic Term Frecuency, *IDF* means Inverse Document Frequency and *TF-IDF* means IDF applied over logarithmic TF).

dataset	label	method	stop	stemmer	normDoc	process	F-1
chartjs	performance	NBM	no	no	no	logTF	0.9987
chartjs	api-change-breaking	NBM	no	porter	no	bool	0.9984
chartjs	inactive-wontfix	SVM	yes	lovins	no	IDF	0.9972
chartjs	api-change-compatible	SVM	no	iterlovins	yes	TF-IDF	0.9970
jquerytools	tests-for-tools-1.2.8	SVM	no	no	no	logTF	0.9964
chartjs	priority-p0	NBM	yes	iterlovins	no	logTF	0.9960
jquerytools	medium-priority	NBM	yes	porter	no	logTF	0.9959
jquerytools	low-priority	NBM	yes	lovins	no	bool	0.9950
chartjs	category-new-chart-types	SVM	yes	porter	no	bool	0.9945
chartjs	time-scale	NBM	yes	lovins	no	logTF	0.9933
chartjs	implement-as-plug-in	SVM	yes	no	no	bool	0.9920
jquerytools	rangeinput	SVM	yes	iterlovins	no	logTF	0.9915
jquerytools	website-and-docs	SVM	yes	iterlovins	yes	logTF	0.9858
chartjs	category-infrastructure	SVM	no	porter	no	bool	0.9824
jquerytools	validator	SVM	no	porter	no	logTF	0.9800
jquerytools	all-tools	SVM	no	no	no	logTF	0.9800
chartjs	category-documentation	SVM	no	porter	no	logTF	0.9792
jquerytools	toolbox	NBM	yes	lovins	no	TF-IDF	0.9782
jquerytools	not-reproducible	SVM	no	no	no	logTF	0.9774
jquerytools	tooltip	SVM	yes	no	no	bool	0.9766
chartjs	needs-test-case	SVM	yes	no	no	bool	0.9766
jquerytools	dateinput	SVM	yes	iterlovins	no	logTF	0.9761
jquerytools	high-priority	NBM	no	no	yes	IDF	0.9751
jquerytools	overlay	SVM	yes	no	no	logTF	0.9738
chartjs	inactive-duplicate	SVM	yes	no	no	logTF	0.9727
jquerytools	feature-request	SVM	yes	porter	no	bool	0.9709
jquerytools	need-more-info	SVM	yes	lovins	no	bool	0.9693
jquerytools	incomplete	SVM	yes	no	no	bool	0.9665
chartjs	needs-investigation	NBM	no	no	no	IDF	0.9622
jquerytools	tabs	SVM	no	lovins	no	logTF	0.9602
jquerytools	question	SVM	no	iterlovins	no	logTF	0.9590
chartjs	version-1.x	SVM	yes	no	no	logTF	0.9573
jquerytools	scrollable	SVM	yes	porter	no	logTF	0.9561
chartjs	category-support	NBM	yes	iterlovins	no	IDF	0.9458
chartjs	priority-p2	SVM	no	lovins	no	logTF	0.9368
jquerytools	fix-posted	SVM	yes	iterlovins	no	logTF	0.9345
jquerytools	bug	SVM	yes	no	no	logTF	0.9344
jquerytools	confirmed	SVM	no	no	yes	IDF	0.9322
chartjs	help-wanted	SVM	yes	porter	yes	logTF	0.9241
guava	status-work-as-intended	SVM	no	no	no	IDF	0.9241
chartjs	priority-p1	SVM	yes	iterlovins	yes	logTF	0.9236
guava	package-eventbus	NBM	no	no	yes	TF-IDF	0.9085
chartjs	category-enhancement	SVM	no	iterlovins	no	TF-IDF	0.9082
guava	status-duplicate	SVM	yes	no	no	bool	0.9081
chartjs	category-question	SVM	no	lovins	yes	RawTF	0.9060
guava	status-fixed	SVM	yes	no	no	bool	0.8937
chartjs	category-bug	SVM	no	porter	no	TF-IDF	0.8841
guava	status-wont-fix	SVM	yes	iterlovins	no	IDF	0.8794
guava	status-invalid	SVM	yes	no	no	bool	0.8787
rxjava	enhancement	NBM	no	porter	no	bool	0.8590
rxjava	bug	NBM	yes	lovins	no	IDF	0.8580
guava	package-cache	SVM	no	porter	no	logTF	0.8504
guava	package-reflect	NBM	no	no	yes	TF-IDF	0.8371
guava	package-io	SVM	yes	porter	no	logTF	0.8144
guava	package-hash	NBM	no	no	yes	TF-IDF	0.8119
guava	cla-yes	SVM	yes	porter	yes	IDF	0.7952
guava	cla-no	NBM	no	no	no	bool	0.7740
guava	package-collect	SVM	yes	no	yes	logTF	0.7719
design-p	new-pattern	SVM	yes	no	no	bool	0.7703
guava	package-concurrent	NBM	yes	no	yes	TF-IDF	0.7667
netty	defect	SVM	no	porter	yes	logTF	0.7545
guava	type-addition	SVM	yes	no	no	logTF	0.7095
design-p	help-wanted	NBM	no	lovins	yes	TF-IDF	0.6913
guava	package-net	NBM	yes	no	yes	IDF	0.6800
guava	type-defect	SVM	yes	iterlovins	no	logTF	0.6746
guava	package-testing	NBM	yes	no	yes	IDF	0.6330
guava	type-enhancement	SVM	yes	no	yes	bool	0.6258
guava	package-base	SVM	yes	no	yes	logTF	0.6181
guava	status-research	SVM	yes	porter	no	IDF	0.5877
netty	feature	SVM	no	lovins	no	logTF	0.5817
guava	type-api-docs	SVM	yes	porter	no	bool	0.5597
guava	status-obsolete	SVM	no	no	no	IDF	0.5365
guava	type-performance	SVM	no	no	no	logTF	0.5261

Continued on next page.

Continued from previous page.

dataset	label	method	stop	stemmer	normDoc	process	F-1
guava	package-primitives	NBM	yes	no	no	IDF	0.5096
netty	improvement	SVM	no	no	no	logTF	0.5014
netty	cleanup	NBM	no	iterlovins	yes	TF-IDF	0.4711
guava	package-general	SVM	no	no	yes	bool	0.4468
netty	not-a-bug	SVM	no	iterlovins	no	logTF	0.4135
storio	bug	SVM	no	no	no	logTF	0.3826
guava	type-task	NBM	no	iterlovins	no	IDF	0.3790
netty	documentation	SVM	yes	no	no	bool	0.3739
storio	question	SVM	no	iterlovins	no	logTF	0.3645
guava	status-accepted	SVM	yes	no	no	logTF	0.3536
guava	type-documentation	NBM	yes	iterlovins	no	RawTF	0.3369
netty	needs-info	SVM	yes	porter	no	bool	0.3225
storio	enhancement-feature	SVM	yes	porter	no	IDF	0.3124
netty	wont-fix	NBM	no	no	no	RawTF	0.2955
guava	type-dev	NBM	yes	iterlovins	no	RawTF	0.2461
guava	type-other	NBM	yes	iterlovins	no	RawTF	0.2074
netty	duplicate	SVM	yes	porter	no	bool	0.0659