Query Quality Prediction on Source Code Base Dataset: A Comparative Study

Swathi B.P
Department of Information and Communication Technology
Manipal Institute of Technology
Manipal Academy of Higher Education
Manipal, India
swathi.bp@manipal.edu

Balachandra Muniyal
Department of Information and Communication Technology
Manipal Institute of Technology
Manipal Academy of Higher Education
Manipal, India
bala.chandra@manipal.edu

***Abstract*—Source code retrieval is a task under text retrieval which is performed by software developers regularly. The existing source code retrieval approaches are regular expression based and anticipate that the software developer querying the code base has an extensive acquaintance with the source code. Unlike keyword or regular expression based source code search which are difficult to remember, software developers should be able to query the code base in a sentential form. Although, performance of the search on text widely depends upon query quality, it succeeds when the quality of the textual query is high. Query quality prediction ahead of query execution on a source code retrieval system will save developers time and effort by notifying him/her when a query is unlikely to perform. This paper assesses the performance of prominent classification algorithms namely Support Vector Machine (SVM), Logistic Regression (LR), Gradient Boosted Tree (GBT) and Decision Tree (DT) to predict the query quality on a data set created from the documentation of the source code files. Experimental results using benchmark open source projects data set demonstrates that Gradient Boosted Tree performs better than others in comparison.**

Keywords—Data mining, Information retrieval, Text retrieval, Source code retrieval, Pre-retrieval metrics, Query quality prediction.

I. INTRODUCTION

Text retrieval is a branch under information retrieval where information retrieved is primarily in the form of text. Amid of legion approaches based on text retrieval, source code retrieval is one such approach which helps to locate source code files from a source code repository[1]. The existing source code retrieval approaches are keyword/regular expression based and look for software developer querying the code base to have an extensive familiarity with the source code. JSearch[2], Google Eclipse Search[3], Searchcode [4], InstaSearch[5], Strathcona[6], Catalog[7], Sando[8] are some of the tools in which source code is retrieved based on keyword search such as method name, class name, variable names. Nevertheless, such a proficiency in code base cannot be expected from a developer who is a newbie to the code base. At the time of change request during software development cycle software developers spend most of their time in searching code from large code repository in order to perform any modifications in the source code. This issue of unnecessary time being spent in searching for source code can be addressed by source code search by issuing a query to the code base in a sentential form. But, the common problem with text retrieval application is that the result of the retrieval mainly depends on the quality of the query. The retrieval result will be hindered when developer's vocabulary is different from that of code base. The query given by the user may or may not contain terms from code base. User query is said to be high quality when it has matching terms from the code base and low quality when query terms are not present in the code base[9].

When the query is not capable of retrieving relevant documents, a lot of time is unnecessarily spent in reading irrelevant documents to manually identify and locate the relevant source code. As a result, query should be tested for its quality in order to categorize as high or low quality. In this paper, the features to predict query quality incorporate techniques from natural language processing[10]. The various properties/features (e.g. specificity) of the query measured are called the pre-retrieval metrics of the query as the metrics are computed before a query runs on an information retrieval engine[11]. In this paper, the training data is created by collecting documentation (comments) from the source code files firstly and then computing the pre-retrieval measures of the collected documentation. The construction of the training data is followed by a comparative study of classification algorithms; Support Vector Machine, Logistic Regression, Gradient Boosted Tree and Decision Tree to predict query quality using the constructed training data. In this work, pre-retrieval metrics and algorithms are implemented using R language.

II. QUERY PROPERTIES

The pre-retrieval metrics which are required to analyze the query quality are specificity, coherency, similarity and term relatedness[11]. The models under comparison is trained and tested using the dataset constructed from the source code base. The dataset consists of 21 pre-retrieval metrics of natural language processing as features.

A. Specificity

Specificity metric is a property of the user query which focuses on the frequency of occurrence of query terms across the source code base. A user query has a scope to retrieve better result if the query terms appear more number of times in very few documents in the entire source code base. Appearance of same query term in most of the documents makes distinction of relevant and non-relevant documents difficult. The Eight specificity variants are listed in Table 1 where D is the set of documents in collection, D_t is the set of documents in the collection containing the term t, t is a query term,, q is the query term in the query, Q is the set of query terms, d is a document in the document collection D, $tf(t,D)$ is the frequency of term t in all docs, $tf(t,d)$ is the frequency of term t in d,

$$idf(t) = \log\left(\frac{|D|}{|D_t|}\right) \quad (1)$$

$$ictf(t) = \log\left(\frac{|D|}{tf(t,D)}\right) \quad (2)$$

$$entropy(t) = \sum_{d \in D_t} \frac{tf(t,d)}{tf(t,D)} . \log_{|D|} \frac{tf(t,d)}{tf(t,D)} \quad (3)$$

$$p_t(Q) = \frac{tf(t,Q)}{|Q|} \quad (4)$$

TABLE 1: Variants of Specificity metric

Metric	Description	Formula
AvgIDF	Average of the Inverse Document Frequency values over all query terms	$\frac{1}{\lvert Q\rvert}\sum_{q\in Q} idf(q)$
MaxIDF	Maximum of the Inverse Document Frequency values over all query terms	$max_{q\in Q}\, idf(q)$
DevIDF	The standard deviation of the Inverse Document Frequency values over all query terms	$\sqrt{\frac{1}{\lvert Q\rvert}\sum_{q\in Q}(idf(q)-avgIDF)}$
AvgICTF	Average Inverse Collection Term Frequency values over all query terms	$\frac{1}{\lvert Q\rvert}\sum_{q\in Q}(ict\, f(q))$
MaxICTF	Maximum Inverse Collection Term Frequency values over all query terms	$max_{q\in Q}(\, ict\, f(q))$
DevICTF	The standard deviation of the Inverse Collection Term Frequency (ictf) values over all query terms	$\sqrt{\frac{1}{\lvert Q\rvert}\sum_{q\in Q}(ict\, f(q)-avgICTF)}$
AvgEntropy	Average entropy values over all query terms	$\frac{1}{\lvert Q\rvert}\sum_{q\in Q} entropy(q)$
MedEntropy	Median entropy values over all query terms	$median_{q\in Q}\,(entropy(q))$
MaxEntropy	Maximum entropy values over all query terms	$max_{q\in Q}(entropy(q))$
DevEntropy	The standard deviation of the entropy values over all query terms	$\sqrt{\frac{1}{\lvert Q\rvert}\sum_{q\in Q}(entropy(q)-avgEntropy)}$
Query Scope (QS)	The percentage of documents in the collection containing at least one of the query terms	$\frac{\lvert\cup_{q\in Q} D_q\rvert}{\lvert D\rvert}$
Simplified Clarity Score (SCS)	The Kullback-Leiber divergence of the query language model from the collection language model	$\sum_{q\in Q} p_q(Q)\log\left(\frac{p_q(Q)}{p_q(D)}\right)$

B. Coherency

The second metric is coherency which is query property focusing on how concentrated the query is on a particular topic. In source code retrieval, the user query terms should focus on a particular feature (topic) so that set of all files implementing a particular feature can be retrieved. The four variants of coherency metric are listed in Table 2 where

$$VAR(t)=\sqrt{\frac{\sum_{d\in D_t}(w(t,d)-\bar{w}_t)^2}{df(t)}} \qquad (5)$$

$$w(t,d)=\frac{1}{\lvert D\rvert}\log(1+tf(t,d))idf(t) \qquad (6)$$

$$\bar{w}_t=\frac{1}{\lvert D_t\rvert}\sum_{d\in D_t} w(t,d) \qquad (7)$$

And $sim(d_i,d_j)$is the cosine similarity between the vector space representation of d_i and d_j.

TABLE 2: Variants of Coherency metric

Metric	Description	Formula
AvgVAR	Average of the variances of the query term weights over the documents containing the query term over all query terms	$\frac{1}{\lvert Q\rvert}\sum_{q\in Q} VAR(q)$
MaxVAR	Maximum of the variances of the query term weights over the documents containing the query term (VAR), over all query terms	$max_{q\in Q}\,(VAR(q))$
SumVAR	Sum of the variances of the query term weights over the documents containing the query term (VAR), over all query terms	$\sum_{q\in Q} VAR(q)$
Coherence Score (CS)	The average of the pairwise similarity between all pairs of documents containing one of the query terms among all documents in the corpus	$\frac{1}{\lvert Q\rvert}\sum_{q\in Q}\left(\frac{\sum d_i,d_j\in D_q sim(d_i,d_j)}{\lvert D_q\rvert.(\lvert D_q\rvert-1)}\right)$

TABLE 3: Variants of Similarity metric

Metric	Description	Formula
AvgSCQ	The average of the collection-query similarity (SCQ) over all query terms	$\frac{1}{\lvert Q\rvert}\sum_{q\in Q} SCQ(q)$
MaxSCQ	The maximum of the collection-query similarity (SCQ) over all query terms	$max_{q\in Q}\ (SCQ(q))$
SumSCQ	The sum of the collection-query similarity (SCQ) over all query terms	$\sum_{q\in Q} SCQ(q)$

C. Similarity

The third metric is similarity which focuses on entire query being similar to source code base. The user query can retrieve better result when all query term put together is similar to source code base. The variants of similarity metrics are listed in Table 3 where

$$SCQ(t) = (1 + \log(ictf(t, D)).idf(t) \qquad (8)$$

D. Term Relatedness

Term relatedness is the fourth query property which says that the user query can retrieve significant result when terms in the query co-occur in the source code base. The two measures from term relatedness are listed in Table 4 where

$$PMI(t_1, t_2) = \log\frac{p_{t_1,t_2}(D)}{p_{t_1}(D).p_{t_2}(D)} \qquad (9)$$

III. Classification Techniques

A. Support Vector Machine

Support vector machine is a decision machine used to classify both linear and non-linear data[12]. In SVM, the maximized margin is chosen to be the decision boundary. Statistical learning theory is used to learn the maximum margin solution. In this work, Radial Basis Function (RBF) kernel SVM is implemented with kernel gamma 1.

B. Logistic Regression

A logit model is used to apply on a binary dependent variable[13]. Estimating the parameters of logit model is logistic regression. Logistic regression is implemented using Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) solver.

C. Gradient Boosted Trees

Gradient boosting which is a machine learning technique for regression and classification problems[14]. GBT produces a prediction model by assembling weaker models typically decision trees. In this work, a tree count of 20 with depth of 5 yielded the best accuracy.

D. Decision Tree

Decision tree induction is the construction of decision trees from training sample which has class labels in it[15]. This implementation uses gain ratio as splitting criterion and tree depth of 10 for better accuracy.

IV. Methodology

Query quality prediction is composed of two main modules; the construction of dataset followed by query quality prediction. The input to the dataset construction module is a benchmark source code base which comprises of documentation in the form of comments. In this work, the comments from 3 open source projects such as VLC media player, Code Blocks and 7-zip has been retrieved. The retrieval of comments from the source code base is followed by query processing operations (lexical analysis, stemming, stop word removal) and computation of 21 pre-retrieval measures. The 21 metrics are computed for each comment in the source code base to form training data and are stored in database. An unstructured database called MongoDB is used during the implementation. Any missing value in the dataset is treated as average of domain values under the metric. In order to set up label for each sample in the training data, each comment from the source code base is run on the information retrieval engine. Lucene library is used for the implementation of information retrieval engine. If the document in which the comment is present is retrieved in the top few documents of the user target(say top 5 or 10 documents), then the label in the training data against that particular comment's retrieval measures is labelled as 1 otherwise it is labelled as 0. Fig 1 shows the model used to construct the training data set. The attribute 'Quality' is made as class label in the data set

Once the training set is prepared, the four models namely SVM, LR, GBT and DT are trained and tested on the data set. A configuration of 70% as training data and 30% as test data is maintained.

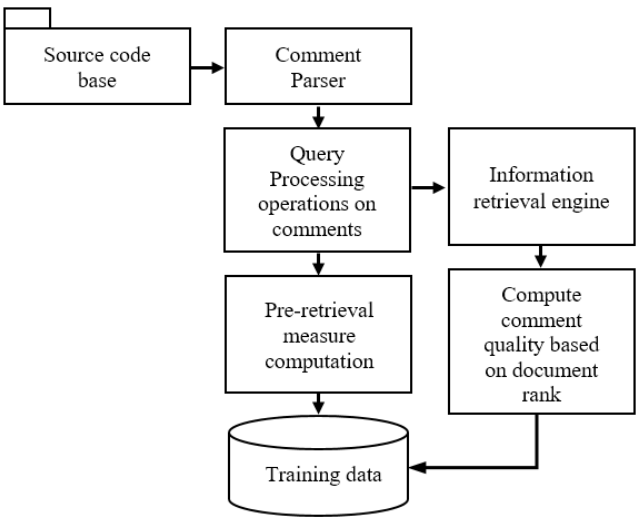


Fig 1. Model to construct training data set

Table 4: Variants of Term Relatedness metric

Metric	Description	Formula
AvgPMI	Average Pointwise Mutual Information over all pairs of terms in the query	$\frac{2(\lvert Q\rvert-1)!}{a}\sum_{q_1,q_2\in Q} PMI(q_1,q_2)$
MaxPMI	Maximum Pointwise Mutual Information (PMI) over all pairs of terms in the query	$max_{q\in Q}\ (PMI(q_1,q_2))$

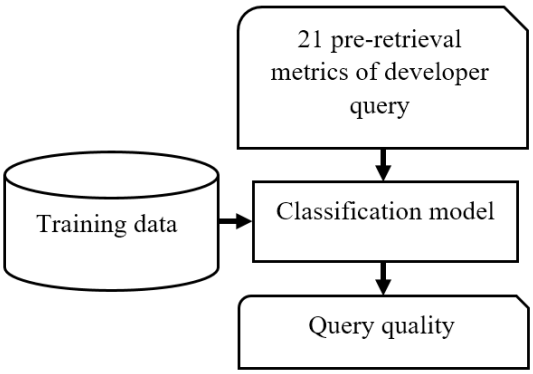


Fig 2. Framework to predict query quality

V. RESULTS

The dataset created from 7-zip, VLC media player and Code Blocks source code consists of 21 pre-retrieval measures of 319 comments, 1055 comments and 798 comments respectively. Table 5 and Table7 demonstrate how much accurate the classifiers are with respect to various datasets. The Kappa statistics of the classifiers are furnished in Table 6

TABLE 5. ACCURACIES OF CLASSIFIERS

	SVM	LR	GBT	DT
7-zip	92.2%	90%	94.8%	93.7%
VLC media player	89.4%	90.1%	91.3%	89.9%
Code Blocks	92.6%	87.6%	92.6%	91.6%

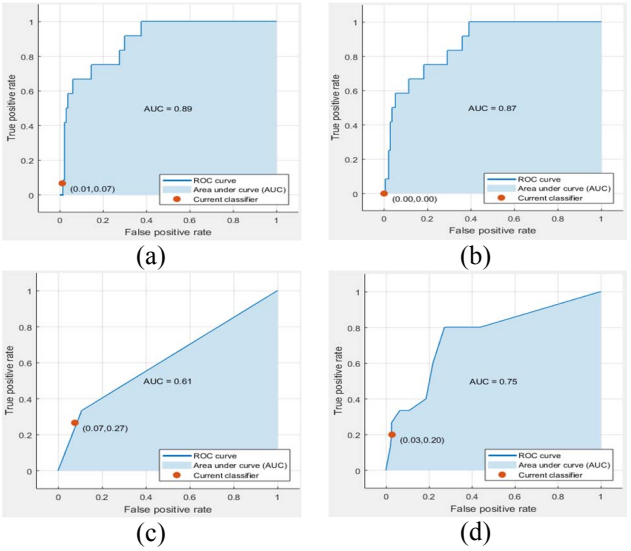
TABLE 6. KAPPA STATISTICS OF CLASSIFIERS

	SVM	LR	GBT	DT
7-zip	0.666	0.820	0.910	0.711
VLC media player	0.498	0.50	0.504	0.503
Code Blocks	0.368	0.521	0.564	0.506

TABLE 7. AUC SCORE OF CLASSIFIERS

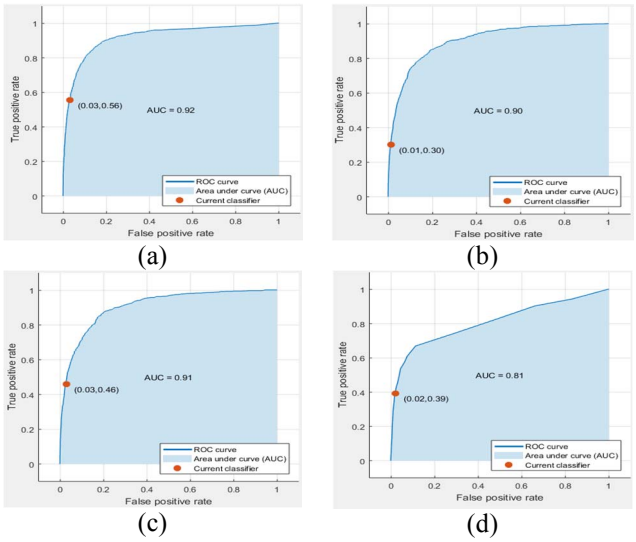
	SVM	LR	GBT	DT
7-zip	0.92	0.82	0.94	0.79
VLC media player	0.9	0.91	0.92	0.81
Code Blocks	0.87	0.61	0.92	0.75

Figures Fig 3, Fig 4 and Fig 5 show the ROC plot of the classifiers for various datasets. From the graphs, it is evident that the AUC score for GBT in all cases have higher value.



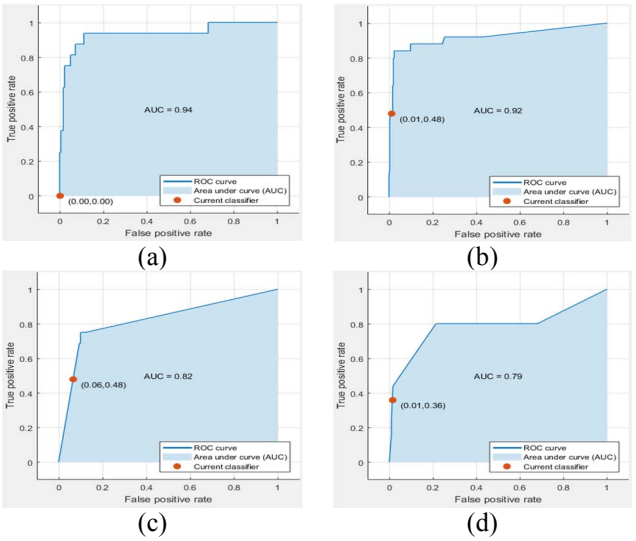
(a) (b) (c) (d)

Fig 3. ROC for CodeBlock dataset of (a)GBT (b)SVM (c) LR (d) Decision Tree



(a) (b) (c) (d)

Fig 4. ROC for VLC dataset of (a)GBT (b)SVM (c) LR (d) Decision Tree



(a) (b) (c) (d)

Fig 5. ROC for 7-ZIP dataset of (a)GBT (b)SVM (c) LR (d) Decision Tree

VI. Conclusion and Future work

The comparison performed among SVM, LR, GBT and DT classification algorithms using the data set constructed from standard open source projects to predict the query quality proved that GBT performs better than other models with accuracy of 94.8%,91.3%, 92.6% for the 7-zip, VLC media player, Code Blocks source codes respectively. Boosting algorithm has worked well that it builds model intelligently by giving more weight to the observation that are hard to classify. Therefore, GBT is one of the appropriate classification algorithm to predict the quality of the query run on a source code base to retrieve the classes/methods when a developer is altogether new to the code base.

In the future, a module can be developed which will reformulate the low quality developer query with terms from source code base to provide a better search result. The label "low quality" produced by GBT is the trigger to the query reformulation module. The reformulation has to be performed based on the concept of synonyms.

References

[1] B. Sisman, S.A. Akbar, and A.C Kak, "Exploiting spatial code proximity and order for improved source code retrieval for bug localization," Jounral of Software: Evolution and Process, vol. 29, No. 1, 2017.

[2] R. Sindhgatta, " Using an Information Retrieval System to Retrieve Source Code Samples", Proceedings of the 28th ACM international conference on Software engineerin, pp. 905-908, 2006

[3] D. Poshyvanyk, M. Petrenko, A. Marcus and X. Xie, and D. Liu, "Source code exploration with google", Proceedings of the 22nd International Conference on Software Maintenance, pp. 334-338, 2006

[4] searchcode.com, 'searchcode', 2018. [online]. Available: https://searchcode.com [Accessed: 31-March-2018]

[5] marketplace.eclipse.org, 'eclipse marketplace', 2018. [online]. Available: http://marketplace.eclipse.org [Accessed: 20 -March-2018]

[6] R. Holmes and G.C. Murphy , Marcus ,"Using structural context to recommend source code examples", proceedings of the 27th ACM International Conference on Software Engineering, pp. 117-125, 2005

[7] W.B Frakes and B.A Nejmeh, "Software reuse through information retrieval ",ACM SIGIR Forum, Vol. 21, No. 1-2, pp. 30-36, 1986

[8] X.G.D.S.K Damevski and, E. Murphy-Hill, "How developers use multi-recommender system in local code search", proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, pp. 69-76, 2014

[9] C. Mills, G. Bavota, S. Haiduc, R. Oliveto,, A. Marcus, and A. D Lucia, " Predicting query quality for applications of text retrieval to software engineering tasks", ACM Transactions on Software Engineering and Methodology, Vol. 26, No. 1, 2017

[10] M. Boughanem, B. Catherine, and M. Josiane, "Advances in Information Retrieval", Proceedings of the 31th European Conference on IR, Toulouse, France, 2009

[11] S. Haiduc, G. Bavota, R. Oliveto, A. D Lucia, and Marcus, "Automatic query performance assessment during the retrieval of software artifacts." Proceedings of the 27th IEEE/ACM international conference on Automated Software Engineering, pp. 90-99, 2012.

[12] C.M Bishop, "Pattern recognition and Machine learning", Springer-Verlag New York, Inc., Secaucus, NJ, 2006

[13] D.A. Freedman, " Statistical models: theory and practice", Cambridge University press, 2009.

[14] Friedman, H Jerome, "Greedy function approximation: a gradient boosting machine", Annals of statistics, JSTOR, pp. 1189—1232, 2001

[15] Han, Jiawei and Pei, Jian and Kamber, Micheline, "Data mining: concepts and techniques", Elsevier, 2011.