A Methodology for Detecting Programming Languages in Stack Overflow Questions

Aman Swaraj and Sandeep Kumar
Indian Institute of Technology, Roorkee, India

Keywords: Software Engineering, Programming Language Detection, Tagging, Stack Overflow, Spacy.

Abstract: Stack Overflow (SO) is the pre-eminent source for knowledge sharing among developers. The Question-Answer (Q-A) site witnesses a heavy traffic with around 5000 questions being posted every day. Given this scenario, it has now become mandatory for users to provide at least one tag for their questions for better discussion. However, novice developers often incorrectly tag their questions which leads to down voting of the post and eventual loss of information. An automatic tag generation mechanism is therefore needed for associating the posts with their respective programming languages. In this work, we have presented a rule based approach for detecting programming languages in question titles. The rules are used to match specific patterns in question titles and generate programming language tags. We then compare the tags generated by our proposed model with the pre-existing tags provided by stack overflow in the dataset. Our model is able to predict languages with an accuracy of 87%. Additionally, our model can detect multiple programming languages in a post and also identify different versions of a language such Python 2.7, Python 3 etc. We further record interesting observations with respect to existing approaches.

1 INTRODUCTION AND RELATED WORK

Stack Overflow (SO) is one of the foremost resource in software development. In recent years, it has seen a massive rise in number of queries and by the end of 2019, it was reported to have more than 18 million questions (Kavuk and Tosun, 2020). Apart from the huge size of the repository, the diversity of SO posts is also quite vast and therefore it heavily relies on tags of the posts for finding potential users to address those queries.

Today, many neophyte programmers depend on SO for their technical queries. Often due to inexperience, negligence, or bias, they tag their posts incorrectly (Kavuk and Tosum, 2020). This often leads to down-voting of posts by the moderators, despite the fact that these questions might contain relevant information that could add value to the community (Alrashedy et al., 2020).

Subsequently, tagging of questions has been recommended as a solution for proper segregation of posts (Jain and Lodhavia, 2020; Saini and Tripathi, 2018; Wang et al., 2018). Further, it has been reported that questions with relevant tags stand greater chance to be answered (Saha et al., 2013). Therefore, it is in the interest of the questioner and the community at large to ensure that the questions be assigned relevant tags.

In general, a post on SO comprises of two parts, i.e. the title and the body. However, in case of a post concerning programming languages, the body can further have some snippet of code attached to it (figure 1) and many authors make use of the snippets for generating tags of posts (Alrashedy et al., 2020; Baquero et al., 2017).

Machine Learning (ML) and Natural Language Processing (NLP) based methods have performed quite well in identifying programming languages in source code files (Dam and Zaytsev, 2016; Gilda, 2017; Khasnabish et al., 2014). However, working on smaller snippets of code is much more difficult and subsequently, researchers have tried to combine features from the body and title of the questions along with the snippets for more accurate results.

In this connection, (Alrashedy et al., 2020) tried to predict programming languages in questions and snippets of SO posts. They achieved an accuracy of 88.9% in classifying programming languages when the data comprised of a combination of snippets, title and body of the question. However, their accuracy

Swaraj, A. and Kumar, S.
A Methodology for Detecting Programming Languages in Stack Overflow Questions.
DOI: 10.5220/0011310400003266
In *Proceedings of the 17th International Conference on Software Technologies (ICSOFT 2022)*, pages 478-483
ISBN: 978-989-758-588-3; ISSN: 2184-2833

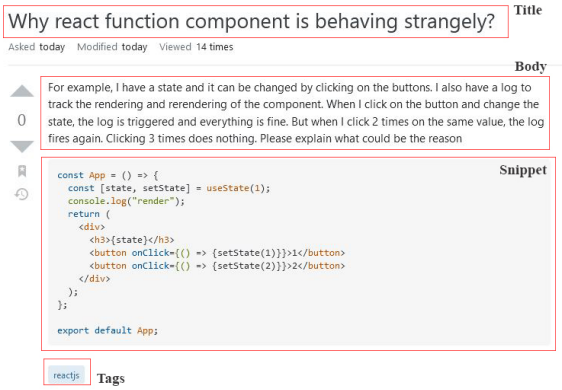

Figure 1: Example of a typical stack overflow post.

dropped to 78.9% when they tried to predict the language based on title and body alone. This goes to show that prediction of programming language of a SO post without considering the snippet is also a non-trivial task. Similarly in their earlier work also on source code classification, (Alrashedy et al., 2018) observed relatively low performance on post title and body as compared to all three combined.

Further, in most of the studies, researchers have approached this task as a classification problem, where the model is trained on labelled data (Jain and Lodhavia, 2020; Kavuk and Tosun, 2020; Saha et al., 2013). However, this approach has few drawbacks, since widely used languages like Java or python have usually more questions as compared to lesser known languages such as 'Go', and therefore, even though the overall performance of the model might be decent, it won't perform as better for the newer languages. We discuss it in more detail in later sections when we introduce our rule based approach.

Besides that, many snippets contain only two to three lines of code which is quite difficult to classify and many posts don't have snippets attached to them in the first place.

Therefore detection of programming language in a SO post based on question title alone has also become a pertinent task for the research community. With this motivation, we propose a model that can detect programming languages from the title of questions without considering the body or snippet attached to it.

Furthermore, many questions include multiple languages and it has been reported that queries having more than one tag gets more views and accepted answers (Saha et al., 2013). Our model is capable of identifying multiple languages in a question and it can also detect different versions of a programming language such as Python 2.7, Python 3 etc.

Rest of the paper is structured as follows: In section 2, we describe the methodology carried out in the study along with the description of the dataset. Section 3 analyses the performance of the model and section 4 depicts the conclusion. Finally, in section 5, we outline potential threats to validity.

2 PROPOSED METHODOLOGY

A brief overview of our proposed framework is depicted in figure 2. It can be broadly classified into the following four stages i.e., Dataset collection, NLP rule formation, Tag generation, and performance analysis.

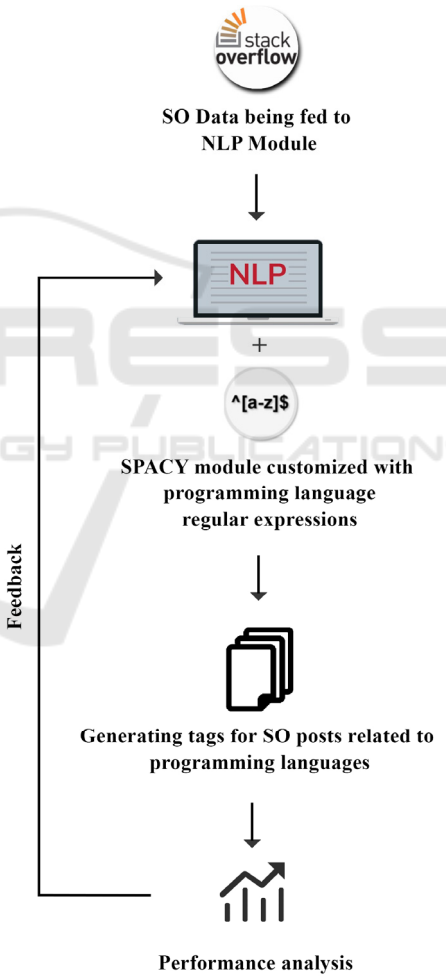

Figure 2: Overview of our proposed approach.

Spacy is an open-source library in python used for advanced NLP. It has a flexible architecture which allows one to add custom components for specific language processing tasks.

The Language class, the Vocab and the Doc object comprises the central data structures of Spacy where the language class is used for processing text and coordinating the components of the pipeline.

Complete sequence of tokens and their annotations are taken care by the doc object. Vocab helps avoid redundancy and save memory by centralizing strings, word vectors and lexical attributes, thereby it maintains a single source of truth.

The default pipeline comprises of multiple components that are called on the Doc in sequence. They can contain a statistical model and trained weights, or only make rule-based modifications to the Doc (Vasiliev, 2020).

For faster processing, we disabled some of the components which was not relevant for our work (figure 3).

Spacy supports matcher components that can extract information from Doc objects based on patterns that a user is searching for. We apply the matcher in our context of detecting programming languages.

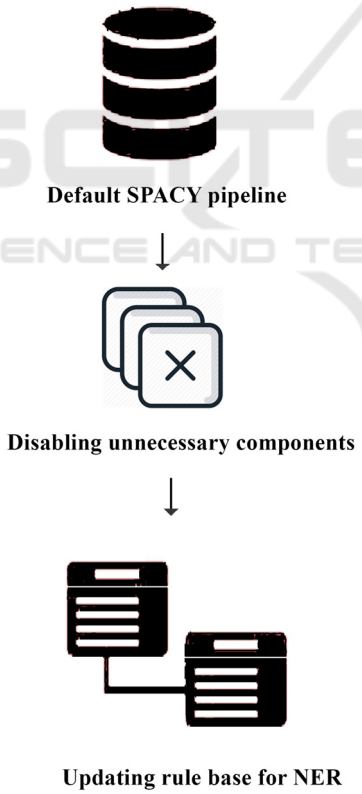

Figure 3: Making changes in the existing pipeline of the module.

First, the raw text is processed and sequence of tokens is generated. Specific pattern rules similar to regular expressions are then set up for matching the generated sequence analogous to respective programming languages. Some of the rules are described in table1.

Table 1: Some of the languages along with some of their rules.

Languages	Set of Rules
C#	[{'LOWER': 'c'}, {'LOWER': '#'}], [{'LOWER': 'c'}, {'LOWER': 'sharp'}], [{'LOWER': 'c#'}],
Go	[{'LOWER': 'go', 'POS': {'NOT_IN': ['VERB']}}], [{'LOWER': 'golang'}], [{'LOWER': 'goprogramminglang'}],
For detecting different versions of a programming language	[{'LOWER': name}], [{'LOWER': {'REGEX': f'({name}\d+\.?\d*.?\d*)'}}], [{'LOWER': name}, {'TEXT': {'REGEX': '(\d+\.?\d*.?\d*)'}}],
Objective-C	[{'LOWER': 'objective'}, {'IS_PUNCT': True, 'OP': '?'}, {'LOWER': 'c'}], [{'LOWER': 'objectivec'}],

In context of our work, in case of a positive match, the matched language is saved as the tag for the post.

Queries that are not directly related with programming language generate null tags.

Further regular expressions are set up for matching different versions of a programming language.

The whole process is repeated several times to ensure all languages are covered.

3 RESULT AND COMPARITIVE ANALYSIS

3.1 Dataset

The total length of data at SO is huge and therefore, in this study, we have used publicly available dataset from kaggle.com which consists of 10% of stack overflow posts.[1] The dataset consists of three separate files containing questions, answers and tags respectively having a mutually inclusive column ‘id’.

The data comprises of questions from all domains of computer science including programming based questions. However, pertinent to our task are only

[1] https://www.kaggle.com/datasets/stackoverflow/stack sample/code

those rows which are related with programming languages and therefore while generating tags we focus only on those rows.

'ISO-8859-1' encoding scheme served as a pre-requisite for the dataset to work efficiently.

3.2 System Settings

All the experiments were carried out on Intel i5, 2.20 GHz processor with 8 GB RAM. We use Jupyter notebook for implementing the spacy library and making needful changes in it.

3.3 Experimentation

In this section, we discuss the experimentation details of the proposed methodology.

After generating the named entity recognition model based on pattern rules, the model was run iteratively over the dataset for thousand random rows and resulting tags were studied.

In the first few iterations, the model was not able to recognize some of the languages as they were either not included in the rule set or were mentioned in unusual ways. For example, in the question titled "Learning Objective-C. Using Xcode 3.2.1. What is error: Program received signal: EXC_ARITHMETIC", the use of 'Objective-C involves the hyphen in between Objective and C which was not matched by the regex. Subsequently, more rules were added based on the feedback and the model was rerun over specific examples to ensure the tags were generated properly.

Additionally, some of the languages such as 'Go' that also have a general usage as a verb were detected out of context. Some of the instances where our model detected the language incorrectly are mentioned in table 2.

Table 2: Specific case of the programming language 'Go'.

Queries	Whether 'Go' is used as a programming language?
"Removing all event handlers in one go"	No
"How do I disable multiple listboxes in one go using jQuery?"	No
"Where does Console.WriteLine go in ASP.NET?"	No
"making generic algorithms in go"	Yes
"How can I run multiple inserts with NHibernate in one go?"	No
"How do I allocate memory for an array in the go programming language?"	Yes
"Embedding instead of Inheritance in Go"	Yes

We selectively kept rules that made sure 'Go' was not used as a verb, but as a noun or had a position dependency on a noun. For instance, dependency graph (Figure 4) reveals one of the instances where 'Go' has been correctly identified as a programming language.

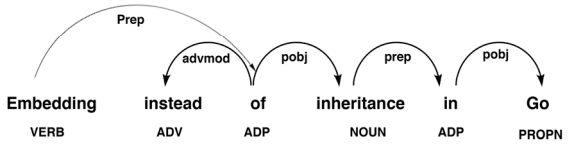

Figure 4: Dependency graph depicting the usage of 'Go' in case of successful detection.

Again, the patterns were modified accordingly to prevent incorrect tag generation.

This is one of the shortcomings of the ML based classification approach where it fails to include or exclude specific usage of certain words as it is trained on a pre-existing dataset, it can only generalise the existing tags and can't deal with unknown or exceptional cases.

The whole iterative process of rule addition was repeated several times to ensure that model was equipped with all the necessary patterns.

After a certain saturation, the tags generated by the model were saved into a separate file along with the unique id of the post. Later on, these tags were compared with the pre-existing tags provided in the dataset as a natural join with unique id of the post acting as the common column.

Queries not related with programming languages generate 'null' tags, and for a fair comparison, we removed all such rows so that the intersection of two files would only include the unique ids of programming related posts.

Furthermore, the questions in the dataset not just included tags for programming language being referred, but also the context of the questions. To generate tags concerning the context of the question is a separate task and out of scope of the current work. Therefore, while matching the tags, we focus only on the programming language tag and consider it a success if our model is able to match the language of the specific post.

3.4 Results and Comparative Analysis

Table 3 gives overview of results for 5000 random rows.

Except for (Jain and Lodhavia 2020; Saha et al., 2013) who have previously worked on SO question titles, most of the researchers have considered

Table 3: Performance of our methodology.

Time	0.18s
Words/sec	59203
Named Entity Recognition Precision	85.62%
NER Recall	94.11%
NER F1-Score	89.29%
Accuracy	87.49%

working on a combination of question title, body and snippet associated with the question. Nevertheless, an overall comparison can be made on the basis of the end goal that is to identify programming language in a given SO post. Table 4 gives a comparative analysis of the same.

To the best of our knowledge, (Alreshedy et al., 2018) have achieved the highest accuracy of 91% in programming language classification for SO posts. However, along with post title, they also combined features from the body and snippet associated with the post. But their accuracy dropped when they excluded the snippets.

Table 4: Comparative analysis with earlier works

Reference	Dataset type	Accuracy, Precision, Recall, and F1
(Kuo, 2011)	Title + Body	47%
(Saha et al., 2013)	Title	68%
(Stanley et al., 2013)	Title + Body	65%
(Baquero et al., 2017)	Title + Body	60.8%, 68%, 60%, and 60%
(Alreshedy et al., 2018)	Title + Body Title + Body + Snippet	81%, 83% 81%, and 0.81%; 91.1%, 91%, 91% and 91%
(Alreshedy et al., 2020)	Title + Body Title + Body + Snippet	78.9%, 81%, 79% and 79% 88.9%, 88%, 88% and 88%
(Saini and Tripathi, 2018)	Title + Body + Snippet	65%, 59%, 36%, and 42%
(Jain and Lodhavia, 2020)	Title	75%, F1 Score-81%
(Kavuk and Tosum, 2020)	Title + Body	75%, 62%, 55% and 39%
(Yang et al., 2021)	Title + Body + Snippet	87.2%, 87%, 87%, 87%
Our work	**Title**	**87%, 85%, 94% and 87%**

Our work is also different in the way that it is not a classification problem as presented in some of the earlier works. Rather we assigned tags to a post based on specific keywords present in the post title itself.

Additionally, our model could detect multiple languages in a post title and also identify different versions of a programming language such as python 2.7 and python 3. From Table-4, it can be observed that our proposed model achieved significantly higher accuracy compared to not only the best matching existing works, (Saha et al., 2013; Jain and Lodhavia, 2020) where only title has been used for language detection, but also performed better than the works (Stanley et al., 2013; Alreshedy et al., 2018; Alreshedy et al., 2020; Kavuk and Tosum, 2020) considering title and body. Our proposed model has shown comparable results to existing works which have used title, body as well as snippet.

4 CONCLUSIONS AND FUTURE WORK

In this work, we propose a model for detecting programming languages in SO questions based on the title of the post. We adapt the existing pipeline structure of the Spacy model to accommodate rules for matching patterns that indicate the presence of a programming language in the post. We achieve an accuracy of 87% on a public dataset. Our model is also capable of detecting different versions of a programming language.

In future, we plan to extend the model on body of SO posts and synchronize them with other ML based techniques. We also plan to validate our model on social media sites such as Reddit and Research Gate for sentiment analysis of programmers towards specific programming languages.

5 THREATS TO VALIDITY

For comparison, we excluded the rows with null tags, however this does not ensure that the null tags were generated accurately. However, since the comparison was made with respect to the unique ids of the post, so for all those posts, which our model could detect, the comparison was fair.

Consider a question with title “I am having error in the following code, code attached”, in such a case our model will not predict the language as it is absent in the question title. For this end, we plan to include analysis of the body and snippets in our future work.

REFERENCES

Alreshedy, K., Dharmaretnam, D., German, D. M., Srinivasan, V., & Gulliver, T. A. (2018). Predicting the Programming Language of Questions and Snippets of StackOverflow Using Natural Language Processing. arXiv preprint arXiv:1809.07954.

Alrashedy, K., Dharmaretnam, D., German, D. M., Srinivasan, V., & Gulliver, T. A. (2020). Scc++: Predicting the programming language of questions and snippets of stack overflow. Journal of Systems and Software, 162, 110505.

Baquero, J. F., Camargo, J. E., Restrepo-Calle, F., Aponte, J. H., & González, F. A. (2017, September). Predicting the programming language: Extracting knowledge from stack overflow posts. In Colombian Conference on Computing (pp. 199-210). Springer, Cham.

Gilda, S. (2017, July). Source code classification using Neural Networks. In 2017 14th international joint conference on computer science and software engineering (JCSSE) (pp. 1-6). IEEE.

Jain, V., & Lodhavia, J. (2020, June). Automatic Question Tagging using k-Nearest Neighbors and Random Forest. In 2020 International Conference on Intelligent Systems and Computer Vision (ISCV) (pp. 1-4). IEEE.

Kavuk, E. M., & Tosun, A. (2020, June). Predicting Stack Overflow question tags: a multi-class, multi-label classification. In Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (pp. 489-493).

Khasnabish, J. N., Sodhi, M., Deshmukh, J., & Srinivasaraghavan, G. (2014, July). Detecting programming language from source code using bayesian learning techniques. In International Workshop on Machine Learning and Data Mining in Pattern Recognition (pp. 513-522). Springer, Cham.

Kuo, D. (2011). On word prediction methods. Technical report, Technical report, EECS Department.

Saha, A. K., Saha, R. K., & Schneider, K. A. (2013, May). A discriminative model approach for suggesting tags automatically for stack overflow questions. In 2013 10th Working Conference on Mining Software Repositories (MSR) (pp. 73-76). IEEE.

Saini, T., & Tripathi, S. (2018, March). Predicting tags for stack overflow questions using different classifiers. In 2018 4th International Conference on Recent Advances in Information Technology (RAIT) (pp. 1-5). IEEE.

Stanley, C., & Byrne, M. D. (2013, July). Predicting tags for stackoverflow posts. In Proceedings of ICCM (Vol. 2013).

Van Dam, J. K., & Zaytsev, V. (2016, March). Software language identification with natural language classifiers. In 2016 IEEE 23rd international conference on software analysis, evolution, and reengineering (SANER) (Vol. 1, pp. 624-628). IEEE.

Vasiliev, Y. (2020). Natural Language Processing with Python and SpaCy: A Practical Introduction. No Starch Press.

Wang, S., Lo, D., Vasilescu, B., & Serebrenik, A. (2018). EnTagRec++: An enhanced tag recommendation system for software information sites. Empirical Software Engineering, 23(2), 800-832.

Yang, G., Zhou, Y., Yu, C., & Chen, X. (2021). DeepSCC: Source Code Classification Based on Fine-Tuned RoBERTa. arXiv preprint arXiv:2110.00914.