2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)

Man vs Machine – a Study into Language Identification of Stack Overflow Code Snippets

1st Jens Dietrich
Victoria University of Wellington
Wellington, New Zealand
jens.dietrich@vuw.ac.nz

2nd Markus Luczak-Roesch
Victoria University of Wellington
Wellington, New Zealand
markus.luczak-roesch@vuw.ac.nz

3rd Elroy Dalefield
Victoria University of Wellington
Wellington, New Zealand
elroy@dalefield.com

***Abstract*—Software engineers produce large amounts of publicly accessible data that enables researchers to mine knowledge, fostering a better understanding of the field. Knowledge extraction often relies on meta data. This meta data can either be harvested from user-provided tags, or inferred by algorithms from the respective data. The question arises to which extent either type of meta data can be trusted and relied upon.**

We study this problem in the context of language identification of code snippets posted on Stack Overflow. We analyse the consistency between user-provided tags and the classification obtained with GitHub linguist, an industry-strength automated language recognition tool. We find that the results obtained by both approaches are often not consistent. This indicates that both have to be used with great care. Our results also suggest that developers may not follow the evolutionary path of programming languages beyond one step when seeking or providing answers to software engineering challenges encountered.

I. Introduction

The mining of software repositories has become an important source of knowledge for the software engineering research community. Examples of widely studied repositories include the Stack Overflow Q&A forum, the GitHub code and the Maven and NPM package repositories. This is facilitated by the fact that many repositories expose content through APIs, in some cases snapshots are provisioned and curated by third parties that facilitate the use of traditional database technologies to access data [16], [2], [6]. Data analysis is further facilitated by the fact that repository data is usually structured, and often enriched by meta data provided by users. Meta data provides valuable guidance as to what the data means, and how it should be interpreted. While sophisticated meta data standards with formal semantics have been proposed [18], [7], the most successful scheme to date is simple tagging [26]. I.e., data is associated with short strings (usually simple nouns) that derive their meaning from how they are used by a wider community, as opposed to taxonomies curated by a central authority.

The question arises how accurate those tags are to describe data. This is not trivial to answer and at first surprising, as the user who tags provides both the data and the meta data and therefore establishes some sort of ground truth one would expect to be accurate. However, in practice things are not that simple.

Consider for instance *code snippets* shared on Stack Overflow. These snippets provide a rich source to study how language features are used [6]. Snippets are embedded in *posts*, which are either questions, or answers to questions. Users asking questions are encouraged to tag their questions in order to improve visibility and find a suitable person to answer the question [3]. Programming language names are popular tags, and it is therefore reasonable to assume that if the name of a programming language is used as a tag then it refers to the language of an embedded code snippet. However, we cannot assume that this provides accurate information for several reasons: (1) posts may not be associated with any tag, or tags used do not correspond to any programming language name, (2) posts may use multiple tags, e.g. when asking about a cross-language integration issue, or if the question is comparative, (3) users may post a question tagged with a certain language, but the code is not correct (in a way that it cannot be parsed), so technically this makes the tag incorrect as it describes not the snippet, but the intention of the user.

If assigning a language to a code snippet is seen as a classification, this means that there are both missing and incorrect classifications. The question arises whether alternatives exist. Firstly, users may provide further hints that can be seen as implicit tags, such as rendering (highlighting) instructions in code blocks. In particular the markdown syntax provides such a feature. To the best of our knowledge, the repository snapshot we use, SOTorrent, does not extract and represent this information, and if it did, it would still have to deal with issues (1) and (3) from the list above. In general, while tagging provides a good source of information, it does not provide the ground truth.

If tags are considered as explicit (i.e., *asserted*) meta data, an alternative is to use some classification algorithm to *infer* meta data. An example for such a tool is GitHub's linguist library [1]. It uses a combination of heuristics and a Bayesian classifier to detect the language of artefacts in GitHub projects. It supports a large number of languages. [1]

Inferring the language has similar problems to solve as in human tag assertion, such as unsupported languages, and code matching multiple languages. The latter is particularly common. For instance, a basic loop instruction in Java would be valid code in many similar C-style languages.

This leads to an interesting research question: Which

[1] 498 on 24 Jan 19, counted using https://github.com/github/linguist/blob/master/lib/linguist/languages.yml, accessed 25 January 19

2574-3864/19/$31.00 ©2019 IEEE
DOI 10.1109/MSR.2019.00041

method is better, and how do they compare? Answering this question requires some sort of ground truth, and the most obvious way to obtain this would be by sampling and manually labelling data, and then compare the classifications against it. Here we suggest another approach: we study the consistency between both approaches. This is particularly interesting for studies that combine different classifications, e.g. to use tags when available, and otherwise fall back to use linguist, or any other classifier employing machine learning, in order to understand the shortcoming and inconsistencies of the respective approaches. And it addresses another problem: If we had a ground truth, we could identify incorrect classifications by any two tagging methods. However, this would not provide any inside into the extent to which snippets misclassified by either method overlap. In this sense, comparing two methods directly leads to better insights into their consistency.

II. Methodology

A. Data Set Construction: Snippet and Tag Selection

We study the code snippets in the SOTorrent data set [6], version *2018-12-09*. Snippets are extracted from the `content` column of the `PostBlockVersion`. We only consider the latest version (i.e., using `MAX(PostHistoryId)`) assuming that the editing process will improve the data (quality of snippets and tags), and to avoid double-counting. Note that posts may have multiple snippets and multiple tags, so the association of snippets and tags is through posts. This causes an issue: if a post has for instance two snippets written in Java and Python, and assigns the respective tags `#java` and `#python` correctly, then each snippet will be associated with one correct and one incorrect tag. We take this situation into account when defining the metrics in section II-B, by looking for the intersection of tag sets, i.e. whether there is any match.

We find $48,807,762$ code snippets with at least one tag associated with them. Tags do not only represent programming languages, but may also refer to operating systems (like `#ios`), generic technologies (like `#database`), etc. We therefore focus on a fixed set of tags representing programming languages, which we derive from the widely used 2018 TIOBE top 20 index [2] covering the most popular languages, and then remove all snippets that are not associated with any of these. Since we want to cross-reference programming language tags with tags inferred by GitHub linguist, we remove (tags associated with) three languages that are not supported by the Bayesian classifier used by linguist [3]. We also normalise tags [4]. This is to deal with situations where a user tags a post with *#java-8* or *#python-3*, in this case we normalise tags by removing the version information to *#java* and *#python*, respectively. The process resulted in the following 17 tags we study: `#java`, `#c`, `#python`, `#c++`, `#javascript`, `#c#`, `#php`, `#sql`, `#objective-c`, `#matlab`,`#r`, `#perl`, `#assembly`, `#swift`, `#go`, `#ruby` and `#plsql`. We refer to those tags as *PL-tags*.

This process resulted in $29,920,851$ snippets associated with at least one PL-tag. All of these snippets can be analysed with linguist. We also note that within this data set, there are only $1,361,441$ (i.e., ca 4.5%) snippets with more that one PL-tag, i.e. the problem discussed above (of snippets being associated with incorrect PL-tags due to the indirect association with tags through posts) does not have a major impact on our results.

B. Metrics

We set out to define a notion of consistency between two tagging schemes. The tag-based scheme associates a code snippet $s \in S$ with a non-empty set of *PL-tags* T: $tag := S \rightarrow 2^T$. On the other hand, the linguist scheme associates each snippet with a *list* of PL-tags where the position in the list indicates confidence in the classification. It is often practical to only consider the first k results, therefore defining a family of classification functions: $ling_k := S \rightarrow 2^T$. We can then define a simple metric representing agreement or overlap α_k between the two tagging schemes, scaled to a value between 0 and 1.

$$\alpha(k) := \frac{|\{s \in S | ling_k(s) \cap tag(s) \neq \emptyset\}|}{|S|} \tag{1}$$

This metric measures whether there is some overlap between the extracted tags by either method, it is therefore a weak notion of consistency. It handles the situation discussed above (see section II-A) that one post contains two correctly tagged snippets written in different languages well. Assuming linguist correctly classifies each snippet, the intersection between $tag(s)$ and $ling_k(s)$ would be non-empty in both cases.

C. Measuring Co-Occurrence

In order to better understand how tags are associated by snippets in either scheme, we also analyse co-occurrence of tags. For this purpose, we collect pairs of tags (tag_1, tag_2) whenever we encounter those two tags being associated with the same snippet. We hypothesise that co-occurrence will be linked stronger to the semantics of code in user-provided tags (e.g., `#c` and `#assembly` are used together as both relate to embedded programming and are often used together), whereas in the linguist classification co-occurrence would stronger relate to syntactical similarities (e.g., this would reflect the fact that `#c#` and `#java` have very similar syntax).

From the extracted user tags as well as computed tags assigned to each code snippet by the linguist library we constructed two co-occurrence matrices, where values represent how often two particular tags co-occur in the same snippet[5]. In order to determine which languages group well together based on co-occurrence, we apply the t-SNE algorithm (dimensions=2; theta=0.1; perplexity=5) on the co-occurrence

[2]https://www.tiobe.com/tiobe-index/, accessed 20 December 2018

[3]*VisualBasic .NET*, *Delphi/Object Pascal* and *Visual Basic*, note that linguist may still support the recognition of those languages through other heuristics, like file extensions, but we use only the Bayesian classifier functionality of linguist as embedded snippets do not have file extensions

[4]Stack Overflow itself already normalises tags through the synonyms and renames mechanism (https://StackOverflow.com/tags/synonyms, accessed 20 December 18)

[5]Note that we only use the top 3 results from the linguist library in this case, and not the full list that ranks all 17 languages in order of probability.

matrices for dimensionality reduction [19], followed by a traditional k-means clustering on the resulting two dimensional space.

D. Experiment Setup and Notes on Scalability

The experiments were conducted using the following pipeline: (1) the SOTorrent dataset was loaded into a local MySQL database; (2) we developed a set of Ruby scripts to create two intermediate tables represented as CSV files (a. snippet and post ids with associated user tags, extracted from the respective SOTorrent tables; b. snippet and post ids with associated linguist tags); (3) we then processed the intermediate data with scripts written in Java and R, producing the result data presented in Section III.

Step 2 proved to be resource-intensive, due to the execution of the linguist script for all snippets. Running on all cores using a 8 core Xeon CPU E5-2620 v4 @ 2.10GHz, this took about 2 days to complete. Step 3 was much faster and all tasks (computation of α and co-occurrence metrics) finished in less than 5 minutes on a 2018 MacBook Pro, utilising 12 threads on a 2.2GHz 6-core i7. The respective scripts and intermediate data sets are public [13].

E. Limitations and Threats to Validity

The tag normalisation mechanism we currently use only relies on the (implicit use of the) Stack Overflow synonym (renames) mechanism, and the removal of version suffixes. This is rather coarse, and could be improved by adding rules that handle for instance frameworks associated with certain programming languages, such as `#jquery` and `#rubyonrails`. However, we do not expect this to have a major impact on the consistency observed (i.e., α) as we expect that posts tagged with a framework tag are either also tagged with the respective PL-tag (in which case we capture the tag correctly), or not tagged with any PL-tag (in which case we ignore the snippets associated with this post).

Linguist uses a list of candidate languages as input. By restricting these languages to the 17 most popular languages we considered, there will be cases where, if other languages were present, this would have led to them being ranked higher in the list of inferred tags. Therefore, the alpha values we compute are upper bounds of the values we would observe with a more comprehensive language list. In this sense, our results on consistency are *optimistic*. Since we selected a comprehensive list of the most popular languages, we expect the impact of this decision to be small, i.e., the upper bound to be tight.

III. Results

A. Classification Consistency

The results of the classification consistency analysis are summarised in Figure 1. The figure shows the $\alpha(k)$ values for all k from 1 to 17. The curve converges against 1 at $k = 17$ – this reflects the fact that linguist takes the list of (17) languages we consider as input, and produces a list of ranked matches against these languages. This means that all those languages will occur eventually in this list, and since we only consider snippets associated with some PL-tags, the tag and the linguist classifications will become trivially consistent at k=17. What is more meaningful is to consider the $alpha$ values for small k. If we only consider the top linguist match, then we find that this only matches any of the user tags in less than half (45%) of the cases. As expected, the curve flattens out quickly, although not as quickly as we had expected. This indicates **a significant level of inconsistency between the two classification schemes**. This suggests that great care must be taken when relying on either scheme, and in particular on a combination of various schemes.

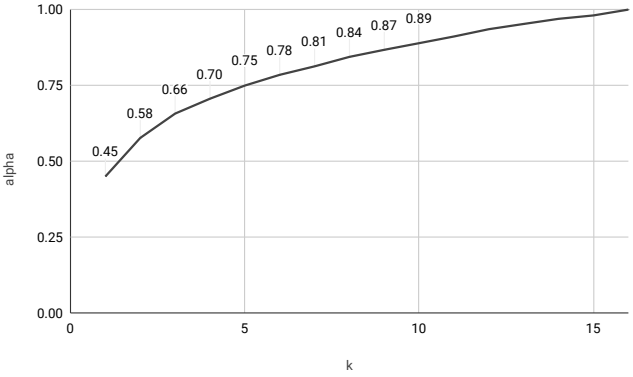

Fig. 1. Alpha values by k

B. Tag Co-occurrence

The analysis of tag co-occurrence provides additional insight into how each classification scheme works (i.e. user-generated compared to computed classification scheme), and helps to explain the level of inconsistency.

1) User-generated Tag Co-occurrence: The results of the user-generated tag co-occurrence analysis are shown in Figure 2. Values (i.e., number of tags with a pair of co-occurring tags) range from $408,428$ to 2. The most popular pair by far is `#php` and `#javascript` $(408,428)$, followed by `#php` and `#sql` $(162,596)$ and `#c` / `#c++` $(104,182)$. This indicates that **humans tag pairs of languages that are used in the same context**: for integrated web development (with php as serverside and javascript as client-side language), database development (with php backed by a relational database, e.g. using the popular LAMP stack), or using the same compiler and tool infrastructure in case of C and C++. The relatively small values are caused by the fact that only a small percentage of snippets has more than one pl tag.

An interesting case is objective-c and swift, both languages are used for developing iOS applications. While they do frequently co-occur these languages do not cluster together, because objective-c features co-occurrences that are not present for swift. Our interpretation is that more practices from C programming are relevant for objective-c (which was influenced by those languages), whereas swift developers only rely on objective-c practices. This is an anecdotal yet

interesting observation, asking for a deeper investigation into **whether programming practices are actually transitive along the path of how languages evolved and whether this observation is true for all families of languages**.

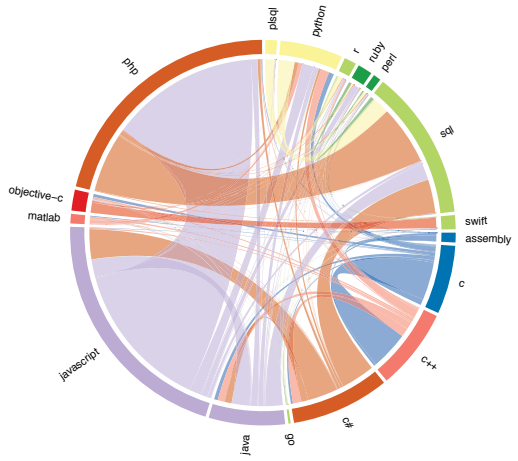

Fig. 2. Co-occurrence of tags added by users.

2) Linguist Tags: With respect to tag co-occurrence among the top 3 results of the automatic snippet classification (cf. Figure 3) we can see most prominently how the Bayesian classifier differs from human classification. We particularly highlight the family of "c languages" detected by linguist as well as the relationship between Java and C#, which emphasises that **syntax and therefore also evolutionary dependence among programming languages play a larger role for automatic classifiers**, while user generated tagging shows more inter-cluster co-occurrences most likely for the reasons described above.

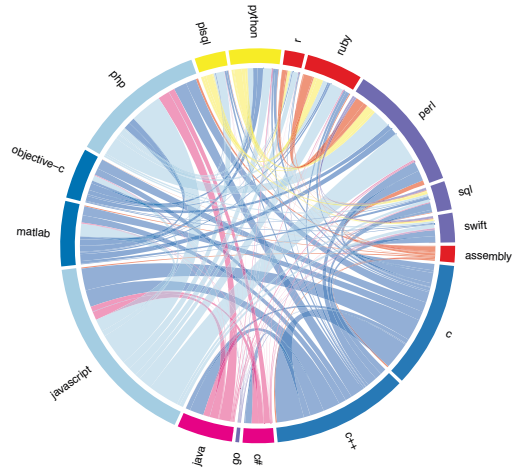

Fig. 3. Co-occurrence of tags computed by linguist.

IV. Related Work

Research on Stack Overflow happens across multiple research communities (e.g. software engineering, peer-production systems and online communities, web science and social informatics), because of the multitude of purposes this question answering site fulfils. It is an online community where expert software engineers as well as more casual developers get answers to questions [21], a platform to build a professional reputation [10], a platform for crowdsourced documentation [25], and a recruitment platform [22].

There have been numerous studies exploring the role Stack Overflow plays in the software industry. Various authors studied factors influencing the likelihood of a question to get answered [23], [4], [5]. Among those features tags have been found to matter most [9], and tag popularity matters more than quality. We argue that this highlights the importance of work like the one we present here, because systems that promote content based on popularity may create certain biases (e.g. popularity bias [28]) as users are incentivised to decrease tag variety and optimise towards ranking and visibility. This has important implications for the increasing amount of work that looks into intelligent systems based on emergent semantics [17], [11], [20] for use cases such as automatic questions answering over knowledge graphs [12].

Several authors have looked into how to infer tags and/or tag synonyms [27], [8], [15], in order to overcome human inconsistencies and inaccuracies. This includes a suggestion to incorporate programming language detection algorithms into tag recommendation [24]. However, research so far only uses small samples of data from Stack Overflow, and falls short to fully evaluate the relevance of the various features used for this task, in particular the accuracy of the automatic programming language detection. This is the focus of the work presented here. Our work compares human tagging with computational tagging mechanisms. This is related to effort in other domains investigating whether hybrid human-machine computation can provide a viable alternative to pure crowdsourcing or pure computation to solve information extraction and classification tasks [14].

V. Conclusion

We have studied how code snippets found on Stack Overflow can be associated with a programming language. In particular, we looked into two different methods to extract the language classification – from user-provided meta data (tags), and by employing GitHub linguist, an industry-strength library to automatically classify code. Both schemes are suitable to work with large and diverse code repositories, in particular, both proved to be sufficiently scalable. We find however that the results obtained with the two approaches are often not consistent. This indicates that both have to be used with great care, and that a hybrid approach that combines the strength of human and machine classification should be investigated as a potential way forward. We also looked into co-occurrence of languages in tags, and found that developers seem not to follow the evolutionary path of programming languages beyond one step when seeking or providing answers to problems encountered.

REFERENCES

[1] Language savant. https://github.com/github/linguist, accessed 24 Jan 19.

[2] Libraries.io. https://libraries.io/, accessed 24 Jan 19.

[3] Stackoverflow helpcenter: What are tags, and how should i use them? https://stackoverflow.com/help/tagging, accessed 24 Jan 19.

[4] A. Baltadzhieva and G. Chrupała. Predicting the quality of questions on stackoverflow. In *Proceedings of the international conference recent advances in natural language processing*, pages 32–40, 2015.

[5] A. Baltadzhieva and G. Chrupała. Question quality in community question answering forums: a survey. *Acm Sigkdd Explorations Newsletter*, 17(1):8–13, 2015.

[6] S. Baltes, C. Treude, and S. Diehl. Sotorrent: Studying the origin, evolution, and usage of stack overflow code snippets. In *Proceedings MSR'19*. IEEE, 2019.

[7] S. Bechhofer, F. Van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, L. A. Stein, et al. Owl web ontology language reference. *W3C recommendation*, 2004.

[8] S. Beyer and M. Pinzger. Synonym suggestion for tags on stack overflow. In *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension*, pages 94–103. IEEE Press, 2015.

[9] V. Bhat, A. Gokhale, R. Jadhav, J. Pudipeddi, and L. Akoglu. Min (e) d your tags: Analysis of question response time in stackoverflow. In *Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 328–335. IEEE Press, 2014.

[10] A. Bosu, C. S. Corley, D. Heaton, D. Chatterji, J. C. Carver, and N. A. Kraft. Building reputation in stackoverflow: an empirical investigation. In *Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on*, pages 89–92. IEEE, 2013.

[11] C. Cattuto, D. Benz, A. Hotho, and G. Stumme. Semantic grounding of tag relatedness in social bookmarking systems. In *International semantic web conference*, pages 615–631. Springer, 2008.

[12] P. Cimiano, V. Lopez, C. Unger, E. Cabrio, A.-C. N. Ngomo, and S. Walter. Multilingual question answering over linked data (qald-3): Lab overview. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 321–332. Springer, 2013.

[13] J. Dietrich, M. Luczak-Roesch, and E. Dalefield. Man vs Machine – a Study into Language Identification of Stack Overflow Code Snippets (Artifact), Mar. 2019. https://doi.org/10.5281/zenodo.2596715.

[14] O. Feyisetan, M. Luczak-Roesch, E. Simperl, R. Tinati, and N. Shadbolt. Towards hybrid ner: a study of content and crowdsourcing-related performance factors. In *European Semantic Web Conference*, pages 525–540. Springer, 2015.

[15] J. R. C. González, J. J. F. Romero, M. G. Guerrero, and F. Calderón. Multi-class multi-tag classifier system for stackoverflow questions. In *Power, Electronics and Computing (ROPEC), 2015 IEEE International Autumn Meeting on*, pages 1–6. IEEE, 2015.

[16] G. Gousios and D. Spinellis. Ghtorrent: Github's data from a firehose. In *Proceedings MSR'12*. IEEE, 2012.

[17] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Emergent semantics in bibsonomy. *GI Jahrestagung (2)*, 94(305-312):38, 2006.

[18] O. Lassila, R. R. Swick, et al. Resource description framework (rdf) model and syntax specification. *W3C recommendation*, 1999.

[19] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[20] B. Markines, C. Cattuto, F. Menczer, D. Benz, A. Hotho, and G. Stumme. Evaluating similarity measures for emergent semantics of social tagging. In *Proceedings of the 18th international conference on World wide web*, pages 641–650. ACM, 2009.

[21] A. Pal, S. Chang, and J. A. Konstan. Evolution of experts in question answering communities. In *ICWSM*, 2012.

[22] M. Papoutsoglou, N. Mittas, and L. Angelis. Mining people analytics from stackoverflow job advertisements. In *Software Engineering and Advanced Applications (SEAA), 2017 43rd Euromicro Conference on*, pages 108–115. IEEE, 2017.

[23] L. Ponzanelli, A. Mocci, A. Bacchelli, M. Lanza, and D. Fullerton. Improving low quality stack overflow post detection. In *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*, pages 541–544. IEEE, 2014.

[24] V. S. Rekha, N. Divya, and P. S. Bagavathi. A hybrid auto-tagging system for stackoverflow forum questions. In *Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing*, page 56. ACM, 2014.

[25] P. C. Rigby and M. P. Robillard. Discovering essential code elements in informal documentation. In *Software Engineering (ICSE), 2013 35th International Conference on*, pages 832–841. IEEE, 2013.

[26] G. Smith. *Tagging: people-powered metadata for the social web*. New Riders Publishing, 2007.

[27] C. Stanley and M. D. Byrne. Predicting tags for stackoverflow posts. In *Proceedings of ICCM*, volume 2013, 2013.

[28] M. Vojnović, J. Cruise, D. Gunawardena, and P. Marbach. Ranking and suggesting popular items. *IEEE Transactions on Knowledge and Data Engineering*, 21(8):1133–1146, 2009.