SimilarTech: Automatically Recommend Analogical Libraries across Different Programming Languages

Chunyang Chen
chen0966@e.ntu.edu.sg

Zhenchang Xing
zcxing@ntu.edu.sg

School of Computer Science and Engineering
Nanyang Technological University, Singapore

ABSTRACT

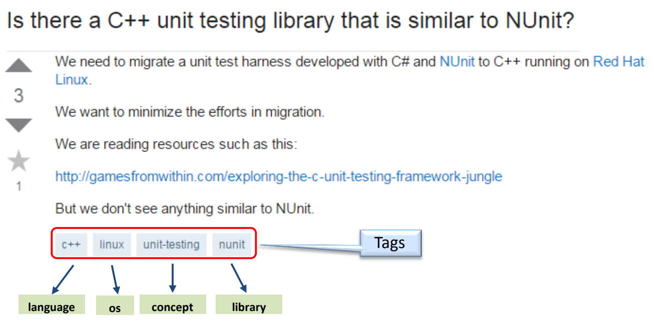
Third-party libraries are an integral part of many software projects. It often happens that developers need to find analogical libraries that can provide comparable features to the libraries they are already familiar with. Existing methods to find analogical libraries are limited by the community-curated list of libraries, blogs, or Q&A posts, which often contain overwhelming or out-of-date information. This paper presents our tool `SimilarTech`[1] that makes it possible to automatically recommend analogical libraries by incorporating tag embeddings and domain-specific relational and categorical knowledge mined from Stack Overflow. `SimilarTech` currently supports recommendation of 6,715 libraries across 6 different programming languages. We release our `SimilarTech` website for public use. The `SimilarTech` website attracts more than 2,400 users in the past 6 months. We observe two typical usage patterns of our website in the website visit logs which can satisfy different information needs of developers. The demo video can be seen at https://youtu.be/ubx8h4D4ieE.

1. INTRODUCTION

Third-party libraries are an integral part of many software systems. They allows developers to focus on the key business logic of their applications, and thus usually help developers work more efficiently. It often happens that a developer needs some analogical libraries that can provide features comparable to the libraries he is already familiar with. The reasons for such needs include that, for example, the current library lacks certain desired features or developers want to migrate to other languages because of the task requirements[2]. Figure 1 presents an example of such information need on Stack Overflow.

In addition to post such questions on Q&A websites, developers can search the Internet for information about analogical libraries by queries like "nltk similar libraries", "nltk java". The search results often include some community-curated list of libraries, many online articles, or Q&A discussions on Q&A site (e.g., Figure 1). However, many recommendations in the online documents may be out-of-date, or based on personal opinions and thus may be biased. Furthermore, developers often have to read many online documents, compare their information, and aggregate the answers to their analogical library queries.

To overcome such shortcomings, we develop a tool, `SimilarTech` that allows users to find analogical libraries across different languages easily and directly. Our `SimilarTech` tool extracts software libraries from Stack Overflow tags. `SimilarTech` determines semantic similarity of libraries based on the tag embeddings learned from tags of tens of millions of Stack Overflow questions. `SimilarTech` uses association rule mining and Natural Language Processing (NLP) methods to mine the relational and categorical information about tags. Finally, it reasons about analogical relationships between libraries based on the semantic similarity of libraries and the relational and categorical knowledge of libraries. Instead of listing dozens of crappy libraries or relying on blogs or Q&A posts, our approach directly recommends analogical libraries based on a knowledge base of analogical libraries mined from Stack Overflow. This knowledge base is like forever evolving blog posts about good analogical libraries to the libraries that one is familiar with.


Is there a C++ unit testing library that is similar to NUnit?

We need to migrate a unit test harness developed with C# and NUnit to C++ running on Red Hat Linux.

We want to minimize the efforts in migration.

We are reading resources such as this:

http://gamesfromwithin.com/exploring-the-c-unit-testing-framework-jungle

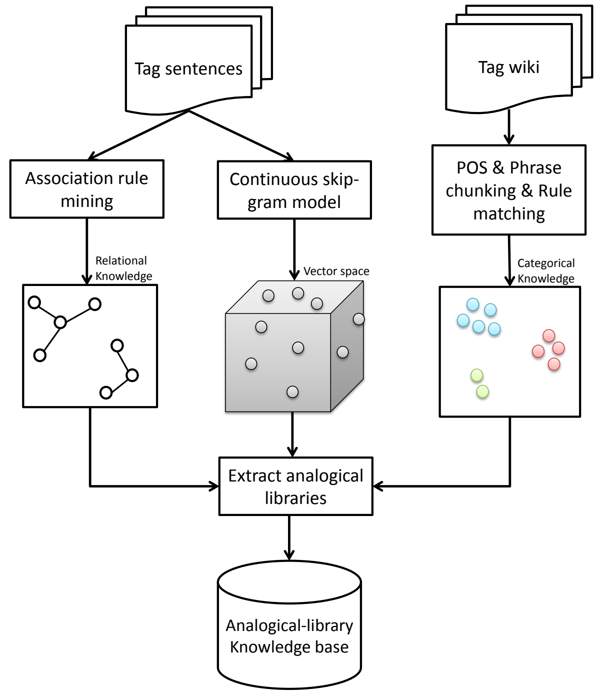
But we don't see anything similar to NUnit.

Figure 1: An exmaple of analogical-library question on Stack Overflow.

Currently, our `SimilarTech` tool supports recommendation of analogical libraries across six popular programming languages, and user can access the recommendations through a publicly available web application or through our browser plugin. In addition to analogical library recommendation, our website also provides the asking trend of each recommended library in Stack Overflow, i.e., the number of questions that are tagged with the library tag over time. This trend allows users to select popular libraries for their tasks.

By tracking and analyzing the website visit statistics via Google Analytics, it shows that more than 2,400 users have visited our site during the last 6 months. From the visiting logs, we also observe usage patterns of our website which can help address different information needs of developers in searching for software libraries. These quantitative and qualitative analysis demonstrate the usefulness of our tool.

[1]https://graphofknowledge.appspot.com/similartech
[2]http://stackoverflow.com/questions/212151/



Tag sentences
Tag wiki
Association rule mining
Continuous skip-gram model
POS & Phrase chunking & Rule matching
Relational Knowledge
Vector space
Categorical Knowledge
Extract analogical libraries
Analogical-library Knowledge base

Figure 2: The overview of our approach

2. THE APPROACH

Our approach takes as input the tags of each question in Stack Overflow and the TagWiki of each tag, and produces as output a knowledge base of analogical libraries (Figure 2). We considers the tags of a Stack Overflow question as a tag sentence, and each tag of the question as a word in the tag sentence. Therefore, we build a corpus of tag sentences, one tag sentence per question. We use association rule mining to ming the correlation among tag from tag sentences, and use POS tagging and phrase chunking to determine the tag category from its definition in TagWiki. Then we use continuous skip-gram model [9] to learn tag embeddings and incorporate relational and categorical knowledge of tags to build the knowledge base of analogical libraries.

2.1 Mining Relational Knowledge

In Stack Overflow, each question has up to 5 tags and co-occurring tags are always related, according to our observation. To reveal such implicit relationship, we use association rule mining [1] to discover important correlation between tags.

In our application of association rule mining, a Stack Overflow question is considered as a transaction and the question tags as items in the transaction. As we are interested in constructing a TAN, we need to find frequent pairs of technologies, i.e., frequent itemsets that consist of two tags. A pair of tags is frequent if the percentage of how many questions are tagged with this pair of tags compared with all the questions is above the minimal support threshold t_{sup}. Given a frequent pair of tags $\{t_1, t_2\}$, association rule mining generates an association rule $t_1 \Rightarrow t_2$ if the confidence of the rule is above the minimal confidence threshold t_{conf}. The confidence of the rule $t_1 \Rightarrow t_2$ is computed as the percentage of how many questions are tagged with the pair of tags compared with the questions that are tagged with the antecedent tag t_1. These rules indicate highly correlated tags

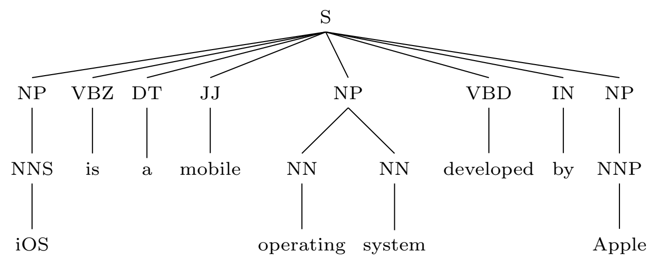


Figure 3: POS tagging and phrase chunking results of the definition sentence of the tag *iOS*

such as (*java, opennlp*) and (*python, nltk*). More details can be referred in our previous works [5, 6].

2.2 Mining Categorical Knowledge

In Fig. 1, we can see that the tags can be of different categories, such as programming language, library, concept, operating systems, etc. To determine the category of a tag, we resort to the tag definition in the TagWiki of the tag. The TagWiki of a tag is collaboratively edited by the Stack Overflow community. Although there are no strict formatting rules in Stack Overflow, the TagWiki description usually starts with a short sentence to define the tag (e.g., Fig. 3). Typically, the first noun phrase just after the *be* verb defines the category of the tag.

Based on this heuristic, we use the NLP methods (similar to the methods used in [8] for named entity recognition) to extract such noun phrase from the tag definition sentence as the category of a tag. POS tagging is the process of marking up a word in a text as corresponding to a particular part of speech, such as common noun, verb, adjective. Phrase chunking is the process of segmenting a sentence into its subconstituents, such as noun phrases, verb phrases. We first adopt the POS to annotate each word in the sentence and then figure out noun phrases by phrase chunking. Fig. 3 shows the results for the tag definition sentence of *iOS*. Based on the POS tagging and phrase chunking results, we extract the first noun phrase (NP) (*operating system* in this example) after the be verb (*is* in this example). We use this noun phrase as the category of the tag i.e., the category of *iOS* is *operating system*. But tag sentence may not always be complete which is not suitable for POS tagging. For these cases, we employ a heuristic label-matching algorithm to find their corresponding labels (more details can be found in [3])

2.3 Learning Tag Embeddings

Word embeddings are low-dimensional vector representations of words that are built on the assumption that words with similar meanings tend to present in similar contexts. Recently, Mikolov et al. [10, 9] demonstrate that the word embeddings encode similarities between pairs of words, for example, the $capital-of$ relation in "Paris : France", "Madrid : Spain". Remarkably, they show that such relations are reflected in vector offsets between word pairs i.e., using simple vector arithmetic could solve analogy questions of the form "a is to A as ? is to B" in which the nature of the relation is hidden.

In our approach, give a corpus of tag sentences, we use continuous skip-gram learning algorithm [9] to learn the word representation of each tag using the surrounding context of the tag in the corpus of tag sentences. In the resulting word embedding space, the vector offsets between analogical libraries and their corresponding programming languages would exhibit *relational similarity*, for example, $nltk - python \approx opennlp - java$. Thus, given a library *nltk* and its corresponding programming language *python* and another programming language *java*, we can transfer the problem of finding analogical libraries to a trivial K-nearest-neighbor search for the tags (e.g., *opennlp*) whose word representations is the most similar to the vector $nltk - python + java$ in the resulting word embedding space.

In this work, we adopt continuous skip-gram model [9] proposed by Mikolov. The objective of the continuous skip-gram model is to learn the word representation of each word that is good at predicting the co-occurring words in the same sentence. Specifically, given a sequence of training text stream $t_1, t_2, ..., t_k$, the objective of the continuous skip-gram model is to maximize the following average log probability:

$$L = \frac{1}{K}\sum_{k=1}^{K}\sum_{-N \preceq j \preceq N, j \neq 0} \log p(t_{k+j}|t_k) \qquad (1)$$

where t_k is the central word, t_{k+j} is its surrounding word with the distance j, and N indicates the context window size to be $2N + 1$.

The probability $p(t_{k+j}|t_k)$ in Eq. 1 can be formulated as a log-linear softmax function which can be efficiently solved by the negative sampling method [9]. After the iterative feed-forward and back propagation, the training process finally converges, and each tag obtains a low-dimension vector as its word representation (i.e., tag embedding) in the resulting vector space.

2.4 Building Analogical-Libraries Knowledge Base

Given the relational and categorical knowledge of the tags and the tag embeddings of the tags, we build a knowledge base of analogical libraries as follows.

In our approach, the library tags broadly refer to the tags whose categories are library, framework, api, toolkit and so on because the meaning of these categories is often overlapping, and there is no consistent rule for the usage of these terms in the TagWiki. For example, in Stack Overflow's TagWiki, *junit* is defined as a framework, *google-visualization* is defined as an API, and *wxpython* is defined as a wrapper. We regard all these tags as library tags in this work.

Given a library tag t_1, we first examine its correlated tags to determine the programming language tag PL_1. Let PL_2 be a programming language tag which can be the same as PL_1 or be different from PL_1. Let $vec(x)$ be the tag embedding of the tag x. To find the analogical library t_2 for the programming language PL_2 as the library t_1 for PL_1, we find the library tags t_2 whose tag embedding $vec(t_2)$ is most similar (by cosine similarity in this work) to the vector $vec(t_1) - vec(PL_1) + vec(PL_2)$, i.e.,

$$\operatorname*{argmax}_{t_2 \in T} \cos(vec(t_2), vec(t_1) - vec(PL_1) + vec(PL_2)) \qquad (2)$$

where T is the set of library tags excluding t_1, and $cos(u, v)$ is the cosine similarity of the two vectors. In practice, there could be several analogical libraries t_2 for the programming language PL_2 as the library t_1 for the programming language PL_1. Thus, we select library tags t_2 with the cosine similarity in Eq. 2 above a threshold t_{al}.

Take the library *nltk* (a NLP library in python) as an example. As shown in the Fig. 4, for *python*, our approach returns the analogical libraries such as *textblob* and *gensim*; for *java*, our approach returns the analogical libraries such as *stanford-nlp*, *opennlp*, and *gate*.

Note that tags whose tag embedding is similar to the vector $vec(t_1) - vec(PL_1) + vec(PL_2)$ may not always be library tags. In our approach, we rely on the category of tags (i.e., categorical knowledge) to return only library tags. Similarly, The returned library tags sometimes include libraries that are not for the given programming language PL_2. We rely on the correlation between a library and the programming language(s) (i.e., relational knowledge) to select the libraries for a given programming language.

3. QUALITY OF SIMILARTECH RECOMMENDATIONS

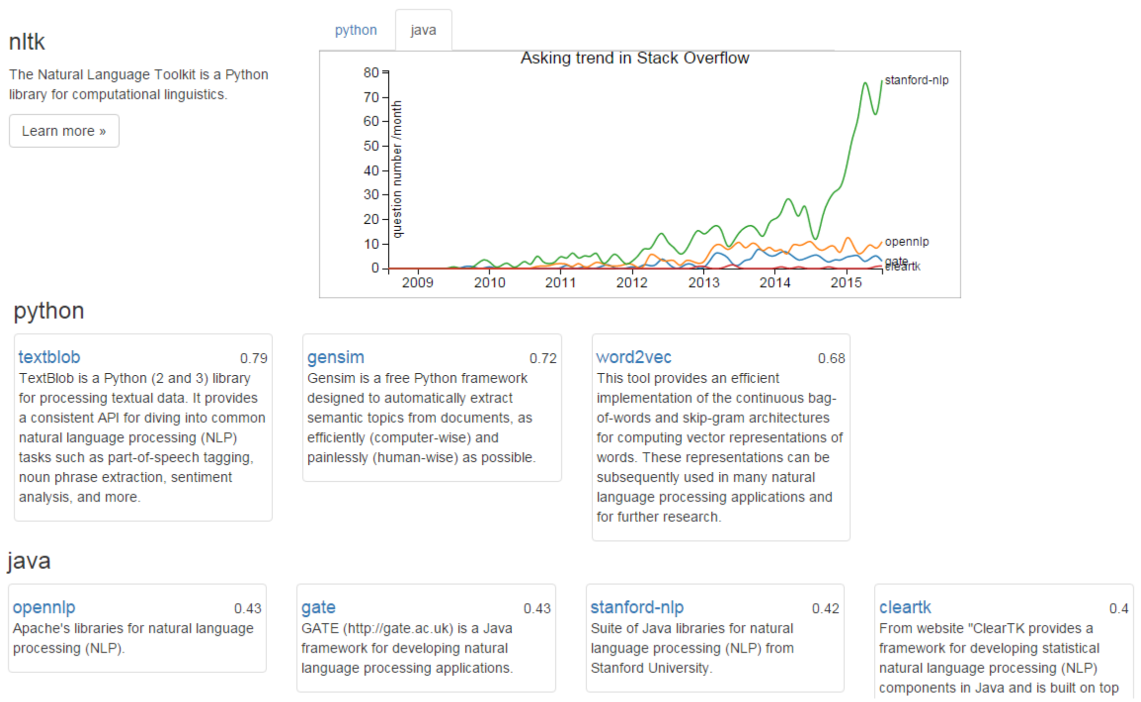
To evaluate the accuracy of our tool, we use Precision@k metric i.e., given a library, we manually check top K recommendation in different programming languages. For example, given *nltk*, our tool recommends analogical libraries from the two languages (*python* and *java*). We check whether each of the recommendations is correct according to our knowledge or by referring to the library information on the Web. We randomly sample 100 libraries and manually check their corresponding recommendations. The overall precision@1 is 0.81 and the precision@5 is 0.67. In addition, we also demonstrate that incorporating categorical and relational information can significantly improve the accuracy of the recommendations than relying solely on tag embeddings. More experiment details, including the evaluation of the quality of mined relational and categorical knowledge, can be found in our full research paper [3].

4. TOOL SUPPORT

We develop a web application called `SimilarTech` (https://graphofknowledge.appspot.com/similartech). Given a library name, our website recommends its analogical libraries in six different programming languages. The backend of `SimilarTech` is an analogical-libraries knowledge base built with the Stack Overflow data dump that contains Stack Overflow post data from July 31, 2008 to Aug 16, 2015. To avoid the out-of-date results, the knowledge base can be automatically updated periodically as the new data dump is released.

The data dump contains 9,970,064 questions and 41,856 different tags. Among 36,197 tags in our dataset with tag-Wiki, 7,783 tags are categorized as library tags. Our database only stores 6,715 library tags which are used frequently enough in Stack Overflow (more than 10 times in the current implementation) and their corresponding analogical libraries for the top-six most frequently-asked programming languages in Stack Overflow, i.e., *java*, *javascript*, *c#*, *php*, *python* and *c++*.

We built this site using the Django framework in Python and plot the asking trend of each library in D3.js. Figure 4 shows a screen shot of our web application. Given a library, our tool attempts to recommend analogical libraries (with the similarity to the given library above the minimal sim-


nltk

The Natural Language Toolkit is a Python library for computational linguistics.

Learn more »

python

textblob 0.79
TextBlob is a Python (2 and 3) library for processing textual data. It provides a consistent API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, and more.

gensim 0.72
Gensim is a free Python framework designed to automatically extract semantic topics from documents, as efficiently (computer-wise) and painlessly (human-wise) as possible.

word2vec 0.68
This tool provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. These representations can be subsequently used in many natural language processing applications and for further research.

java

opennlp 0.43
Apache's libraries for natural language processing (NLP).

gate 0.43
GATE (http://gate.ac.uk) is a Java framework for developing natural language processing applications.

stanford-nlp 0.42
Suite of Java libraries for natural language processing (NLP) from Stanford University.

cleartk 0.4
From website "ClearTK provides a framework for developing statistical natural language processing (NLP) components in Java and is built on top

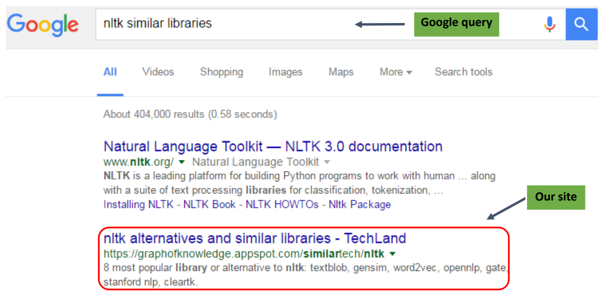
Figure 4: The scrrenshot of our website *SimilarTech*

ilarity threshold 0.4[3]) for the six programming languages. In the current implementation, `SimilarTech` presents up to four libraries with the highest similarity for each programming language. The rationale is that developers would be unlikely to look through a long list of recommendations and there are usually just a few most popular libraries for each programming language. Note that listing up to four libraries is only an implementation decision, not a limitation of our approach.

For each recommended analogical library, `SimilarTech` shows a brief definition extracted from the corresponding TagWiki. Clicking a library name navigates to the analogical-libraries page for the clicked library. This allows the user to interactively explore the underlying analogical-libraries knowledge base. *SimilarTech* also summarizes the number of questions tagged with a library per month, and plots the metrics over time in a so-called asking trend. The asking trends [4] of analogical libraries allow the user to easily compare the amount of the questions for each library on Stack Overflow.

We carry out search engine optimization to our `SimilarTech` site so that it can be indexed by search engines such as Google and Bing. We develop a program to automatically summarize our recommendation results for each library and put all these information into the metadata of each `SimilarTech` page. When search engines crawl our sites, they will not only index the content inside the page, but also the metadata of our site. As a result, search engines can rank our websites at the top of the search results for some queries. One example is shown in Figure 5 in which our site is ranked the second position for the query "nltk similar libraries" on May 5th, 2016. The metadata of this page is "8 most popular libraries or alternatives to *nltk*: *textblob* ...". The metadata

[3]We experimentally select this value as lower threshold leads to more errors while higher one results in fewer results.



nltk similar libraries

About 404,000 results (0.58 seconds)

Natural Language Toolkit — NLTK 3.0 documentation
www.nltk.org/ ▾ Natural Language Toolkit ▾
NLTK is a leading platform for building Python programs to work with human ... along with a suite of text processing libraries for classification, tokenization, ...
Installing NLTK - NLTK Book - NLTK HOWTOs - Nltk Package

nltk alternatives and similar libraries - TechLand
https://graphofknowledge.appspot.com/similartech/nltk ▾
8 most popular library or alternative to nltk: textblob, gensim, word2vec, opennlp, gate, stanford nlp, cleartk.

Figure 5: Our site ranks at top 2 in the Google search for the query "nltk similar libraries"

is intuitive and user-friendly to web users as they can understand what information our website provides even without clicking. Note that only the official site of *nltk* ranks higher than us. However, the *nltk* official site has nothing to do with the similar libraries.

5. TOOL USAGE

We report the visit statistics of our website by Google Analytics, and demonstrate the two typical usage scenarios of our `SimilarTech` website.

5.1 The Usage of the SimilarTech Website

We release our website to the public and post this news on several programming-related websites (e.g., http://stackapps.com/questions/6667/). According to the Google Analytics of the website traffic, more than 2,400 users from 99 countries visit our site, from Nov 11, 2015 to May 5, 2016. These users on average browse 3.7 pages in each session and they browse 10,377 pages in total[4]. It indicates that users are interested in our results so that they explore and interact with

[4]As most search engine robots do not activate Javascript,

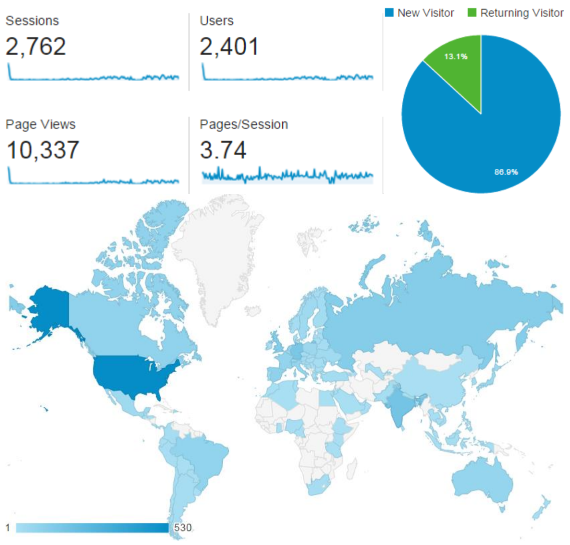


Figure 6: The visit statistics of our website from Google Analytics

our site by visiting several pages. 13% users come back after their first visit. Analysis of the site logs shows that users in total browse 1,802 libraries for analogical-libraries recommendation. The top 15 most-frequently visited libraries are listed in Figure 7. The usage data of our website, albeit very limited, demonstrates both the needs and interests in such analogical-library recommendation that our tool supports.

5.2 Usage scenarios

After quantitative analysis of the website visit data, we further analyze user navigation patterns in our site. According to our observation, there are two different navigation patterns of user visits. These two usage patterns help address two different information needs for software libraries.

5.2.1 Find libraries in the other languages

The primary goal of our tool is to help developers find analogical libraries to a library for different programming languages. The log of our website visits show that users indeed come to our site for such information needs, because their first search and the subsequent click are mostly for libraries in different languages. Such actions represent that given a library, users are interested in its analogical libraries in another language. As such, they click the recommended libraries for more details. For example, one user first types *pdfminer* which is one of the most popular pdf-parsing library in Python. Then, according to the visit log, he/she clicks *pdfbox* and *icepdf* which has the similar function but for a different language, i.e., java.

Apart from recommending such analogical libraries, our system sometimes may provide some serendipitous recommendations, i.e., information which is useful but may be out of the users' expectation. For example, one user first searches *opencv* (most popular computer-vision library in c++), and he/she then clicks one of the recommendation results, *marvin-framework* (a image-processing library in java). Then, he/she not only sees some similar libraries to *marvin-framework* such as *javacv*, but also a library *opencv4android* which is also a java library but specially adopted for the

robot traffic is not counted in Google Analytics [7].

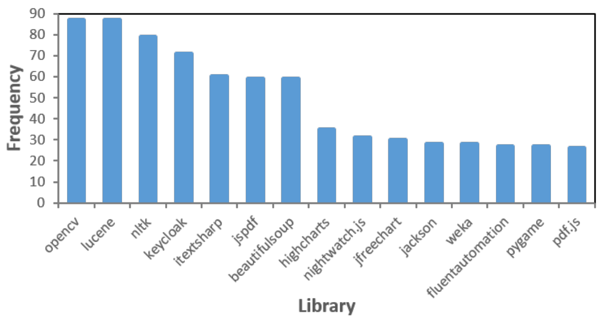


Figure 7: The top 15 most-frequently visited libraries on SimilarTech

Android mobile system. If the user is a mobile application developer, he/she could benefit from our recommendation. In fact, this user shows his/her interest in the recommended *opencv4android* by clicking it for more information.

5.2.2 Find libraries in one language

Besides the recommendation across different languages, many users also use our website to find similar libraries in the same language, which is out of our expectation. By observing such visit logs, we find two goals for their behaviors. First, some users are looking for libraries with better performance. For example, one user clicks *jspdf, pdf.js* and *pdfjs* consecutively. As these libraries have almost the same functionality, the user is likely comparing their performance according their description and trend data.

Second, user can refine their initial searches and find what they need by exploring the recommendation results. For example, one user first searches *pythonmagick* (an image-editing library in python) and then clicks *python-imaging-library* (a more popular library for image processing lib in python). Finally, he/she clicks a more specific lib called *pypng* (PNG image en/decoding) which is much smaller than the other two general libraries but may have the specific features just satisfying his/her needs. This shows that the process of navigating through our site can also be the process of search refinement to find better solutions to some specific tasks.

6. RELATED WORK

The aim of `SimilarTech` is to recommend analogical libraries across different programming languages. In contrast to this, the great majority of research work for recommendation in Software Engineering focuses on code level [17, 19] and API level [2, 16] recommendation.

Language migration is a common phenomenon for developers as they may have to switch from one programming language to another according to the task requirements. The biggest challenge is usually the code and library migration, rather than learning a new language itself[5]. Many researchers have proposed methods to overcome the code migration challenge, such as code mapping [12], function mapping [14], and API migration [18, 11]. Compared with to these code-level migration approaches, our approach supports library-level migration.

Thung et al. [15] analyze the library co-occurrence patterns in software projects to recommend relevant libraries for a software project. Teyton et al. [13] analyze the evolution

[5]http://stackoverflow.com/questions/212151/

of projects' dependencies on third-party libraries to recommend libraries that can replace an existing library in a software project. Different from these approaches, our approach does not rely on the information about the projects' dependencies on third-party libraries. Instead, we mine analogical libraries from the crowdsourced knowledge in domain-specific Q&A sites (such as Stack Overflow). Furthermore, existing approaches are limited to recommend libraries for the same programming language, while our system can recommend alternative, comparable libraries across different programming languages.

There are also some websites which support similar-tool recommendation. SimilarWeb[6] is a website that provides both users engagement statistics and similar competitors for websites and mobile applications. AlternativeTo[7] is a social software recommendation website in which users can find alternatives to a given software based on user recommendations. These websites can help regular web users to find similar or alternative websites or software applications. But their content is not useful for domain-specific information needs of software developers, for example, to find analogical libraries for different programming languages. In contrast, our web application is built on software-engineering data and is specifically designed for software developers.

7. CONCLUSION AND FUTURE WORK

In this paper, we describe a tool `SimilarTech`, which automatically recommends analogical libraries across different programming languages. The automatic recommendation is made possible by mining and incorporating semantic, relational and categorical knowledge from Stack Overflow. We adopt the cutting-edge deep learning method in NLP applications (also known as word embeddings) to learning tag embeddings, which encode the semantic similarity of tags. We further enhance the tag embeddings with software-specific relational and categorical knowledge to better answer analogy questions in software engineering context. Given a library, our tool can recommend several most salient analogical libraries for six different popular programming languages.

In the future, we will improve our web application and analyze the website traffic and user behaviors in our website to enhance the accuracy of analogical-libraries recommendation. Furthermore, we are very interested in extending our approach to fine-grained level of analogy relationships, for example, mining analogical APIs across different libraries or programming languages in Q&A discussions or other online resources (e.g., Github). Tens of thousands of API analogy questions can be found on Stack Overflow, which indicates the urgent needs for the automatic tool support at the API level. The ability to easily find analogical APIs and their usage examples can boost developers' productivity and efficiency when they migrate from one programming language to another unfamiliar language.

8. REFERENCES

[1] R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules. In *20th VLDB*, volume 1215, pages 487–499, 1994.

[2] W.-K. Chan, H. Cheng, and D. Lo. Searching connected api subgraph via text phrases. In *FSE*, page 10. ACM, 2012.

[3] C. Chen, S. Gao, and Z. Xing. Mining analogical libraries in q&a discussions -incorporating relational and categorical knowledge into word embedding. In *23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 338–348. IEEE, 2016.

[4] C. Chen and Z. Xin. Towards correlating search on google and asking on stack overflow. In *40th COMPSAC*, pages 83–92. IEEE, 2016.

[5] C. Chen and Z. Xing. Mining technology landscape from stack overflow. In *10th ESEM*. IEEE/ACM, 2016.

[6] C. Chen, Z. Xing, and L. Han. Techland: Assisting technology landscape inquiries with insights from stack overflow. In *32nd ICSME*. IEEE, 2016.

[7] Google analytics policy. https://support.google.com/analytics/answer/1315708?hl=en.

[8] J. Kazama and K. Torisawa. Exploiting wikipedia as external knowledge for named entity recognition. In *Proceedings of the 2007 EMNLP-CoNLL*, pages 698–707, 2007.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[10] T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013.

[11] A. T. Nguyen, H. A. Nguyen, T. T. Nguyen, and T. N. Nguyen. Statistical learning approach for mining api usage mappings for code migration. In *Proceedings of the 29th ASE*, pages 457–468. ACM, 2014.

[12] T. T. Nguyen, A. T. Nguyen, H. A. Nguyen, and T. N. Nguyen. A statistical semantic language model for source code. In *9th FSE*, pages 532–542. ACM, 2013.

[13] C. Teyton, J.-R. Falleri, and X. Blanc. Mining library migration graphs. In *19th WCRE*, pages 289–298. IEEE, 2012.

[14] C. Teyton, J.-R. Falleri, and X. Blanc. Automatic discovery of function mappings between similar libraries. In *20th WCRE*, pages 192–201. IEEE, 2013.

[15] F. Thung, D. Lo, and J. Lawall. Automated library recommendation. In *WCRE*, pages 182–191. IEEE, 2013.

[16] F. Thung, S. Wang, D. Lo, and J. Lawall. Automatic recommendation of api methods from feature requests. In *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*, pages 290–300. IEEE, 2013.

[17] D. M. L. Xu, R. Bodık, and D. Kimelman. Jungloid mining: Helping to navigate the api jungle. In *POPL*, 2005.

[18] H. Zhong, S. Thummalapenta, T. Xie, L. Zhang, and Q. Wang. Mining api mapping for language migration. In *32nd ICSE*, pages 195–204. ACM, 2010.

[19] H. Zhong, T. Xie, L. Zhang, J. Pei, and H. Mei. Mapo: Mining and recommending api usage patterns. In *ECOOP*, pages 318–343. Springer, 2009.

[6]www.similarweb.com/

[7]http://alternativeto.net/