DiffTech: A Tool for Differencing Similar Technologies from Question-and-Answer Discussions

Han Wang
freddie.wanah@gmail.com
Faculty of Information Technology, Monash University
Australia

Chunyang Chen
chunyang.chen@monash.edu
Faculty of Information Technology, Monash University
Australia

Zhenchang Xing
zhenchang.xing@anu.edu.au
College of Engineering & Computer Science, Australian National University
Australia

John Grundy
john.grundy@monash.edu
Faculty of Information Technology, Monash University
Australia

ABSTRACT

Developers can use different technologies for different software development tasks in their work. However, when faced with several technologies with comparable functionalities, it can be challenging for developers to select the most appropriate one, as trial and error comparisons among such technologies are time-consuming. Instead, developers resort to expert articles, read official documents or ask questions in Q&A sites for technology comparison. However, it is still very opportunistic whether they will get a comprehensive comparison, as online information is often fragmented, contradictory and biased. To overcome these limitations, we propose the DiffTech system that exploits the crowd sourced discussions from Stack Overflow, and assists technology comparison with an informative summary of different comparison aspects. We found 19,118 comparative sentences from 2,410 pairs of comparable technologies. We released our DiffTech website for public use. Our website attracts over 1800 users and we also receive some positive comments on social media. A walkthrough video of the tool demo: *https://www.youtube.com/watch?v=ixX41DXRNsI*
Website link: *https://difftech.herokuapp.com/*

CCS CONCEPTS

• **Software and its engineering**;

KEYWORDS

differencing similar technology, Stack Overflow, NLP

ACM Reference Format:
Han Wang, Chunyang Chen, Zhenchang Xing, and John Grundy. 2020. DiffTech: A Tool for Differencing Similar Technologies from Question-and-Answer Discussions. In *Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '20), November 8–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3368089.3417931

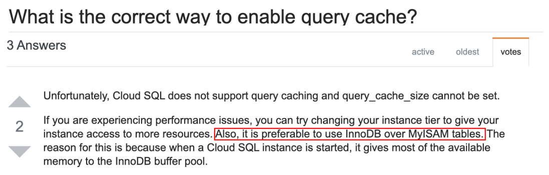
1 INTRODUCTION

A diverse set of technologies (e.g, algorithms, programming languages, platforms, libraries/frameworks, concepts for software engineering) [9, 12] is available for use by developers and that set continues growing. Adopting suitable technologies will significantly accelerate the software development process and also enhance the software quality. However, when developers are looking for the right technologies for their tasks, they are likely to find several candidates. For example, they will find *bubble sort* and *quick sort* algorithms for sorting, *nltk* and *opennlp* libraries for NLP, *Eclipse* and *Intellij* for developing Java applications.

Faced with so many compatible candidates, even for experienced developers, it can be difficult to keep pace with the rapid evolution of technologies. We find that the perceptions of developers about comparable technologies and the choices they make which technology to use are very likely to be influenced by how other developers *see* and *evaluate* the technologies. So developers often turn to the two information sources on the Web [4] to learn more about comparable technologies.

They may read experts' articles about technology comparison like *"Intellij vs. Eclipse: Why IDEA is Better"*. And search on Q&A websites such as Stack Overflow or Quora (e.g., *"Apache OpenNLP vs NLTK"*). However, there are two limitations with expert articles and community answers, which are *Fragment view* (each post can only focuses on one specific aspect, which leads developers to search everywhere) and *Diverse opinions* (developers have different opinions on the same technology). The above two limitations create a high barrier for developers to effectively gather useful information about technology differences on the Web.

Different methods have been adopted to mine similar artefacts ranging from high-level software [27], mobile applications [17, 25], github projects [20, 31] to low-level third-party libraries [8, 10], APIs [6, 14, 22, 28], or Q&A questions [7, 13, 16]. However, most of them only focus on specific domain, while we are developing a systematical way to not only extract different software-specific artefacts like general software concepts (e.g., algorithm, protocol), tools (e.g., IDE) but also the user opinions to them.


What is the correct way to enable query cache?

3 Answers

Unfortunately, Cloud SQL does not support query caching and query_cache_size cannot be set.

If you are experiencing performance issues, you can try changing your instance tier to give your instance access to more resources. Also, it is preferable to use InnoDB over MyISAM tables. The reason for this is because when a Cloud SQL instance is started, it gives most of the available memory to the InnoDB buffer pool.

Figure 1: A comparative sentence in a question (#30654296) that is not explicitly for technology comparison.

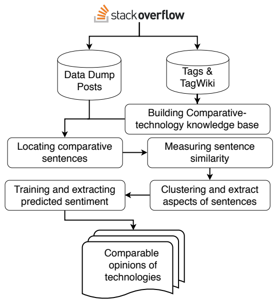

stackoverflow
Data Dump Posts
Tags & TagWiki
Building Comparative-technology knowledge base
Locating comparative sentences
Measuring sentence similarity
Training and extracting predicted sentiment
Clustering and extract aspects of sentences
Comparable opinions of technologies

Figure 2: The overview of our approach

Our work is motivated by the fact that a wide range of technologies have been discussed by millions of users in Stack Overflow, and users often express their preferences toward a technology and compare one technology with the others in the discussions. Apart from posts that are explicitly about the comparison of some technologies, many comparative sentences hide in posts that are implicitly about technology comparison. Figure.1 shows such an example: the answer "accidentally" compares *Innodb* and *Myisam*, while the question does not explicitly ask for this comparison. Inspired by such phenomenon, we then propose our system to mine and aggregate the comparative sentences in Stack Overflow discussions. We collect a large number of comparable technologies and corresponding comparative information from Stack Overflow. We then build a website called DiffTech[1] and make it public to real-world developers.

2 APPROACH

We consider Stack Overflow tags as a collection of technology terms and first build comparable technology knowledge base by analyzing tag embeddings and categories, as shown in Figure. 2. Our system then mines comparative opinions from Q&A discussions by analyzing sentence patterns, calculating word mover distance [24] and community detection [21] to cluster comparative sentences. We then fine-tune the BERT [19] model to summarize overall sentiment from all the comparative sentences for each comparable technology pairs. Finally, we built a developer-oriented information website using the data we collected and analysed.

2.1 Mining Similar Technology

Studies [5, 8, 29] show that Stack Overflow tags identify computer programming technologies that questions and answers revolve around. They cover a wide range of technologies, from algorithms (e.g., *dijkstra, rsa*), programming languages (e.g., *golang, javascript*), libraries and frameworks (e.g., *gson, flask*), and development tools (e.g., *sublime, vmware*). In this work, we regard Stack Overflow tags as a collection of technologies that developers would like to compare.

1) Learning Tag Embeddings: Word embeddings [26] are dense low-dimensional vector representations of words that are built on the assumption that words with similar meanings tend to be present in a similar context. In our approach, given a corpus of tag sentences, we use word embedding methods to learn the word representation of each tag using the surrounding context of the tag in the corpus of tag sentences. We use the continuous bag-of-words (CBOW) model as the word embedding methods and collect a corpus of tags and their corresponding vectors.

2) Mining Categorical Knowledge: To determine the category of a tag, we resort to the tag definition in the TagWiki of the tag. We find out that normally, the first noun just after the be verb defines the category of the tag. For example, *Matplotlib is a plotting library for Python.* Based on the pattern, we extract the first sentence of the TagWiki description, and then apply Part of Speech (POS) tagging to the extracted sentence. With this method, we obtain 318 categories for the 23,658 tags (about 67% of all the tags that have a TagWiki). We manually normalize and categorized these 318 categories into five general categories: programming language, platform, library, API, and concept/standard [30].

3) Building Similar-technology Knowledge Base: We've had the vectors of each tag and their categories after the two steps above. For a technology tag *t1*, and another technology tag *t2*, if their cosine similarity is less than a threshold, and they are identified as same the category, then we consider them as comparable technology pairs.

2.2 Mining Comparative Opinions

For each pair of comparable technologies in the knowledge base, we analyze the Q&A discussions in Stack Overflow to extract plausible comparative sentences by which Stack Overflow users express their opinions on the comparable technologies.

1) Locate Comparative Sentences: Similar to how we mine Categorical Knowledge, we also use the POS tagging when locating the comparative sentences. We divide the sentences into two categories: *single sentences* and *contextual sentences*. Then, we extend the list of common POS tags to enhance the identification of comparative sentences. More specifically, we create four comparative POS tags: *CV* (comparative verbs, e.g. prefer, compare, beat), *CIN* (comparative prepositions, e.g. than, over), *NW* (negation words, e.g. wouldn't, doesn't), and *TECH* (technology reference, including the name and aliases of a technology, e.g. python, eclipse). We use a large thesaurus of morphological forms of software-specific terms [15, 18] to replace the abbreviation, synonym and misspelling technologies. We summarize 4 patterns for single sentences and 3 patterns for contextual sentences. If the POS tags of a single sentence match the patterns, we identify it as a *single sentences* type comparative opinion. If the POS tags of two or three sequential sentences match the patterns, we identify them as a *contextual sentences* type comparative opinion. Table 1 and Table 2 show the patterns we use to

[1] https://difftech.herokuapp.com/

Table 1: Patterns of comparative single sentences

No.	Pattern	Sequence example	Original sentence
1	*TECH * VBZ * (JJR ∨ RBR)*	innodb has 30% higher ubuntu is better	InnoDB has 30% higher performance than MyISAM on average. i believe that ubuntu is better than centos.
2	*((RBR JJ) ∨ JJR) * CIN * TECH*	faster than coalesce more complex than mysql	Isnull is faster than coalesce. Triggers in postgresql have a syntax a bit more complex than mysql.
3	*CV * CIN TECH*	recommend scylla over cassandra	I would recommend scylla over cassandra.
4	*CV VBG TECH*	recommend using html5lib	I strongly recommend using html5lib instead of beautifulsoup.

Table 2: Patterns of comparative contextual sentences

No.	Pattern	Sequence example	Original sentence
1	*TECH * VBZ * (JJR ∨ RBR)*	triple des is generally better	Triple des is generally better but there are some known theoretical attacks. If you have a choice of cipher you might want to look at aes instead.
2	*((RBR JJ) ∨ JJR) * CIN * TECH*	faster than MySQL	Postgres has a richer set of abilities and a better optimizer. Its ability to do hash joins often makes it much faster than MySQL for joins.
3	*AFF | NEG*	cassini does not iis to do	Cassini does not support https. However you can use iis to do this.

locate the comparative sentences and some sample sentences. Note that the *AFF* and *NEG* refer to the affirmation and negation sentences. In this pattern the two sentences need to be one affirmation sentence and one negation sentence, and each of them need to talk about one technology that are comparable.

2) Measuring Sentence Similarity: To measure the similarity of two comparative sentences, we adopt the Word Mover's Distance [24] which is especially useful for short-text comparison. We first train a word embedding model based on the post content of Stack Overflow so that we get a dense vector representation for each word in Stack Overflow. Then, we compute the minimal word movers' distance between the keywords in one sentence and those in the other sentences. Base on the distance, we further compute the similarity score of the two sentences by $similarity\ score(S_1, S_2) = \frac{1}{1+D(S_1,S_2)}$.

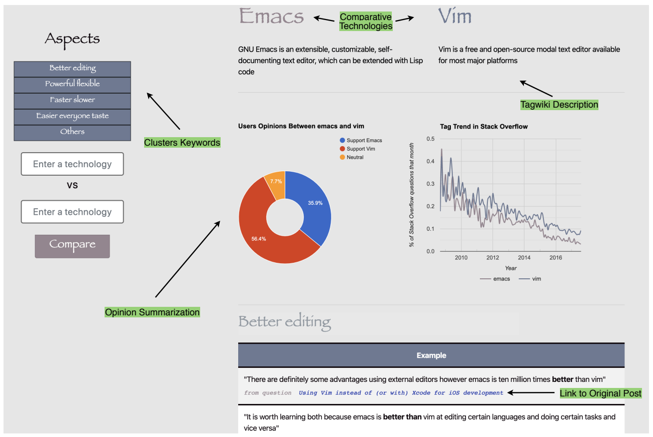
3) Clustering Representative Comparison Aspects: Based on comparative sentences and their similarities, we take each sentence as one node in the graph. If two sentences are determined as similar enough (similar score > 0.55), we add an edge between them in the graph. We cluster similar opinions by applying a community detection algorithm [21] to the graph of comparative sentences. Within each cluster, we take the top-3 words with largest TF-IDF (Term Frequency Inverse Document Frequency) scores as the representative aspect for the community.

2.3 Summarizing Overall Opinion

There may be too many comparison opinions that are too time-consuming for developers to read. So we further develop a sentiment classifier based on the BERT model [19] for automatically distilling the overall opinion towards the comparable technologies. That summarization can be an important criteria for developers to determine which technology to adopt.

2.4 Tool Implementation

We take the Stack Overflow data dump (released on 4 September 2019) as the data source. It contains 18,154,493 questions with 55,665 unique tags, and 27,765,324 answers. We collect in total 14,876 pairs of comparable technologies. Among these technologies, we extract



GNU Emacs is an extensible, customizable, self-documenting text editor, which can be extended with Lisp code

Vim is a free and open-source modal text editor available for most major platforms

"There are definitely some advantages using external editors however emacs is ten million times **better** than vim"

from question Using Vim instead of (or with) Xcode for iOS development

"It is worth learning both because emacs is **better than** vim at editing certain languages and doing certain tasks and vice versa"

Figure 3: The screenshot of our website DiffTech and some important elements in the website

19,118 comparative sentences for 2,410 pairs of comparable technologies. We use these technology pairs and comparative sentences to build a knowledge base for comparison.

Based on our proposed approach, we implemented a practical website (https://difftech.herokuapp.com/) for developers. With the knowledge base of comparable technologies and their comparative sentences mined from Stack Overflow, our site can return an informative and aggregated view of comparative sentences in different comparison aspects for comparable technology queries. In addition, it provides a link for each comparative sentence to its corresponding Stack Overflow post so that users can easily find detailed content. Figure 3 shows a screenshot from our website.

For each comparable technology pair, we mine the first sentence from the Tagwiki as the description. Below the descriptions, are two charts. The left pie chart is the overall opinion we summarized in Section 2.3. From the page, we can see about 56.4% of the users support using *Vim* instead of *Emacs*. The right side is the trend chart [11] shows that compared with *Emacs*, more people talks about *Vim*. We can also see that the method has grouped all of the comparative sentences into four clusters, shown in the top left corner. For each clustered aspect, we list the comparative sentences below and attach the direct link for each comparative sentence to its original post. Users can click the link to get more content.

The website was initially created in September 2018 for our previous study [23]. We updated it October 2019 with new contributed features. It is built with Django, and hosted on Heroku. We stored all the comparative information which we collect in the above steps in the Postgresql Database also on Heroku.

3 TOOL STUDY

The accuracy of our approach has been demonstrated in our previous work [23]. Here we demonstrate the usefulness of our tool by analyzing the usage scenarios and visitation log.

3.1 Usage Scenarios

According to our observation of users' visiting log, we categorize them into two groups: Developers and Programming Learners/Students. We discuss use cases for the two parties separately.

1) Developers: For industry developers, it can be challenging to choose the correct technology for their product. For example whether to use *ant* or *maven* for Java project management. They can visit our website and get to know which technology most users recommend from the opinion pie chart. They can find out which is more popular based on the trend chart. As they scroll down, they have views of detailed comparisons based on different aspects, learn under which scenarios the technology performs the best. If they find certain sentences they are interested in, they can easily access the detained Stack Overflow posts.

2) Programming Learners/Students: For this type of users, our website is like a learning material beyond lecture notes. DiffTech gives them an opportunity to compare similar technologies in different aspects, refer the seniors' experience, and learn the new technology quickly. For instance, if a student just learned *TCP* and *UDP* protocols, and wanted to know more about the differences in between. The comparative sentences from our site not just tell the fact that *UDP* is normally faster than *TCP*, but also give the reasons like *UDP* doesn't need handshakes and it doesn't require the arrival sequence. Moreover, the comparisons are not just conceptual, some of them also involved in practical part like under which circumstance suits which protocol better, for example "*Udp is more suitable for streaming media;but if you are sensitive with your music streaming tcp is more secure*". Another scenario would be a programming learner try to pick a machine learning tool. From DiffTech, he could know that "*I recommend you check this page also i think keras is better to begin with than tensorflow*" and "*Keras is a higher level library that is much easier to learn than tensorflow and you have more sample code online*".

3.2 Field Study

We embedded Google Analytics in our site to help with tracking usage data. In addition to the general site traffic statistics, we further investigate two questions regarding what users are interested in and what they think of our website:

1) Traffic Statistics: Figure 4 shows the website traffic from 3rd October 2019 to 13th Jun 2020. There are about 1811 users from 81 different countries visited our website. These users viewed 3,332

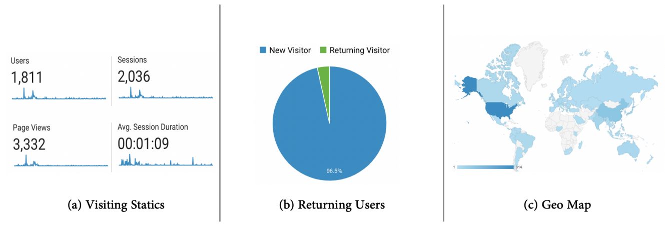


Figure 4: The traffic of our website from Google Analytics

pages and had an average session time over 1 minute. Note that most users come to our site with very specific target, i.e., comparing a pair of similar technologies, so, the average number of visited pages in each session is rather small. Most users came from the U.S. (43.58%), with other countries such as China (12.31%), Japan (4.87%), etc. Nearly 5% of the users come back to visit our website again, indicating the usefulness and attraction of our site.

2) Most viewed Comparisons: According to the analytics data, the comparison between *gson* and *jackson* are viewed the most by users, they have 547 views. *Mysql* and *Postgresql* are in the second place, whereas *lisp* and *scheme* are in the third place. All of which are viewed over 100 times in two months.

3) Users comments and suggestions: We posted links to the website on some websites such as StackApps [1] and Reddit [2] [3] for advertisement purpose and to collect user feedback. Apart from the visiting, some users also post comments under our posts such as "*Thank you ...will share the word*", "*It was actually better than I expected, at least for the suggested comparisons. Good job!*". They also provide some constructive suggestions for improving our tool like "*This sounds interesting. However I think it would be better if there were options to add to the technologies, pros, cons and usage.*". There are also some comments about the bug that the website has. We are happy to see the compliment from users and will improve based on the suggests.

4 CONCLUSION AND FUTURE WORK

In this demo we presented DiffTech, a system to distill and aggregate opinions of comparable technologies from Q&A websites. We first obtained a large pool of comparable technologies the word embedding of tags in Stack Overflow. We then located comparative sentences about these technologies by POS-tag based pattern matching, organized comparative sentences into clusters, and finally summarized comparative sentences to obtain an aggregated opinion for each comparable technologies. We used this to construct a proof-of-concept tool of it for developers.

In spite of comparative sentences explicitly mentioning both comparable technologies, some comparative opinions may be deeper. For example, one developer expresses his opinions about one technology in one paragraph while discussing the other technology in the next paragraph. Therefore, we will improve our system to distill technology comparison knowledge from the current sentence level to post level.

ACKNOWLEDGEMENTS

Prof. Grundy is supported by ARC Laureate Fellowship FL190100035.

REFERENCES

[1] 2019. DiffTech: Compare similar technologies based on StackOverflow data. https://stackapps.com/questions/8469/difftech-compare-similar-technologies-based-on-stackoverflow-data. Accessed: 2019-11-18.

[2] 2019. Website for developers compare similar technologies(i.e. postgresql vs mysql). https://www.reddit.com/r/programming/comments/drba2s/website_for_developers_compare_similar/. Accessed: 2019-11-18.

[3] 2019. Website the could be helpful for programming learners. https://www.reddit.com/r/learnprogramming/comments/dspnwc/website_the_could_be_helpful_for_programming/. Accessed: 2019-11-18.

[4] Lingfeng Bao, Jing Li, Zhenchang Xing, Xinyu Wang, Xin Xia, and Bo Zhou. 2017. Extracting and analyzing time-series HCI data from screen-captured task videos. *Empirical Software Engineering* 22, 1 (2017), 134–174.

[5] Anton Barua, Stephen W Thomas, and Ahmed E Hassan. 2014. What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Software Engineering* 19, 3 (2014), 619–654.

[6] Chunyang Chen. 2020. SimilarAPI: Mining Analogical APIs for Library Migration. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE.

[7] Chunyang Chen, Xi Chen, Jiamou Sun, Zhenchang Xing, and Guoqiang Li. 2018. Data-driven proactive policy assurance of post quality in community q&a sites. *Proceedings of the ACM on human-computer interaction* 2, CSCW (2018), 1–22.

[8] Chunyang Chen, Sa Gao, and Zhenchang Xing. 2016. Mining analogical libraries in q&a discussions–incorporating relational and categorical knowledge into word embedding. In *Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference on*, Vol. 1. IEEE, 338–348.

[9] Chunyang Chen and Zhenchang Xing. 2016. Mining technology landscape from stack overflow. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, 14.

[10] Chunyang Chen and Zhenchang Xing. 2016. Similartech: automatically recommend analogical libraries across different programming languages. In *Automated Software Engineering (ASE), 2016 31st IEEE/ACM International Conference on*. IEEE, 834–839.

[11] Chunyang Chen and Zhenchang Xing. 2016. Towards correlating search on google and asking on stack overflow. In *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, Vol. 1. IEEE, 83–92.

[12] Chunyang Chen, Zhenchang Xing, and Lei Han. 2016. Techland: Assisting technology landscape inquiries with insights from stack overflow. In *Software Maintenance and Evolution (ICSME), 2016 IEEE International Conference on*. IEEE, 356–366.

[13] Chunyang Chen, Zhenchang Xing, and Yang Liu. 2017. By the Community & For the Community: A Deep Learning Approach to Assist Collaborative Editing in Q&A Sites. *Proceedings of the ACM on Human-Computer Interaction* 1, 32 (2017), 1–32.

[14] Chunyang Chen, Zhenchang Xing, Yang Liu, and Kent Long Xiong Ong. 2019. Mining likely analogical apis across third-party libraries via large-scale unsupervised api semantics embedding. *IEEE Transactions on Software Engineering* (2019).

[15] Chunyang Chen, Zhenchang Xing, and Ximing Wang. 2017. Unsupervised software-specific morphological forms inference from informal discussions. In *Proceedings of the 39th International Conference on Software Engineering*. IEEE Press, 450–461.

[16] Guibin Chen, Chunyang Chen, Zhenchang Xing, and Bowen Xu. 2016. Learning a dual-language vector space for domain-specific cross-lingual question retrieval. In *Automated Software Engineering (ASE), 2016 31st IEEE/ACM International Conference on*. IEEE, 744–755.

[17] Ning Chen, Steven CH Hoi, Shaohua Li, and Xiaokui Xiao. 2015. SimApp: A framework for detecting similar mobile applications by online kernel learning. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. ACM, 305–314.

[18] Xiang Chen, Chunyang Chen, Dun Zhang, and Zhenchang Xing. 2019. SEthesaurus: WordNet in Software Engineering. *IEEE Transactions on Software Engineering* (2019).

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805* (2018).

[20] Sa Gao, Chunyang Chen, Zhenchang Xing, Yukun Ma, Wen Song, and Shang-Wei Lin. 2019. A neural model for method name generation from functional description. In *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 414–421.

[21] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *Proceedings of the national academy of sciences* 99, 12 (2002), 7821–7826.

[22] Xiaodong Gu, Hongyu Zhang, Dongmei Zhang, and Sunghun Kim. 2017. DeepAM: Migrate APIs with multi-modal sequence to sequence learning. *arXiv preprint arXiv:1704.07734* (2017).

[23] Yi Huang, Chunyang Chen, Zhenchang Xing, Tian Lin, and Yang Liu. 2018. Tell them apart: distilling technology differences from crowd-scale comparison discussions.. In *ASE*. 214–224.

[24] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*. 957–966.

[25] Mario Linares-Vásquez, Andrew Holtzhauer, and Denys Poshyvanyk. 2016. On automatically detecting similar android apps. In *Program Comprehension (ICPC), 2016 IEEE 24th International Conference on*. IEEE, 1–10.

[26] Suyu Ma, Zhenchang Xing, Chunyang Chen, Cheng Chen, Lizhen Qu, and Guoqiang Li. 2019. Easy-to-Deploy API Extraction by Multi-Level Feature Embedding and Transfer Learning. *IEEE Transactions on Software Engineering* (2019).

[27] Collin McMillan, Mark Grechanik, and Denys Poshyvanyk. 2012. Detecting similar software applications. In *Proceedings of the 34th International Conference on Software Engineering*. IEEE Press, 364–374.

[28] Trong Duc Nguyen, Anh Tuan Nguyen, Hung Dang Phan, and Tien N Nguyen. 2017. Exploring API embedding for API usages and applications. In *Software Engineering (ICSE), 2017 IEEE/ACM 39th International Conference on*. IEEE, 438–449.

[29] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. 2011. How do programmers ask and answer questions on the web?: Nier track. In *Software Engineering (ICSE), 2011 33rd International Conference on*. IEEE, 804–807.

[30] Deheng Ye, Zhenchang Xing, Chee Yong Foo, Zi Qun Ang, Jing Li, and Nachiket Kapre. 2016. Software-specific named entity recognition in software engineering social content. In *Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference on*, Vol. 1. IEEE, 90–101.

[31] Yun Zhang, David Lo, Pavneet Singh Kochhar, Xin Xia, Quanlai Li, and Jianling Sun. 2017. Detecting similar repositories on GitHub. In *Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference on*. IEEE, 13–23.