DIFFTECH: Differencing Similar Technologies from Crowd-Scale Comparison Discussions

Michael Shell, *Member, IEEE,* John Doe, *Fellow, OSA,* and Jane Doe, *Life Fellow, IEEE*

Abstract—Developers can use different technologies for many software development tasks in their work. However, when faced with several technologies with comparable functionalities, it is not easy for developers to select the most appropriate one, as comparisons among technologies are time-consuming by trial and error. Instead, developers can resort to expert articles, read official documents or ask questions in Q&A sites for technology comparison, but it is opportunistic to get a comprehensive comparison as online information is often fragmented or contradictory. To overcome these limitations, we propose the *diffTech* system that exploits the crowdsourced discussions from Stack Overflow, and assists technology comparison with an informative summary of different comparison aspects. We first build a large database of comparable technologies in software engineering by mining tags in Stack Overflow, and then locate comparative sentences about comparable technologies with natural language processing methods. We further mine prominent comparison aspects by clustering similar comparative sentences and representing each cluster with its keywords. The evaluation demonstrates both the accuracy and usefulness of our model and we implement our approach into a practical website for public use.

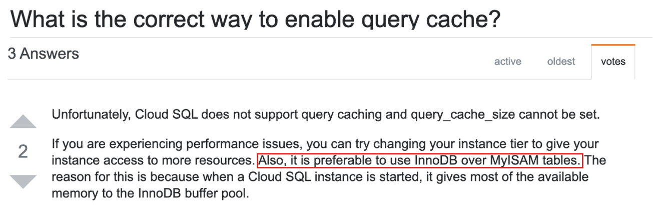
Index Terms—differencing similar technology, Stack Overflow, NLP

1 INTRODUCTION

A diverse set of technologies (e.g, algorithms, programming languages, platforms, libraries/frameworks, concepts for software engineering) [1], [2] is available for use by developers and that set continues growing. By adopting suitable technologies, it will significantly accelerate the software development process and also enhance the software quality. But when developers are looking for proper technologies for their tasks, they are likely to find several comparable 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 candidates, developers are expected to have a good understanding of different technologies in order to make a proper choice for their work. However, even for experienced developers, it can be difficult to keep pace with the rapid evolution of technologies. Developers can try each of the candidates in their work for the comparison. But such trial-and-error assessment is time-consuming and labor extensive. Instead, we find that the perceptions of developers about comparable technologies and the choices they make about 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 [3] to learn more about comparable technologies.

First, they read experts' articles about technology comparison like *"Intellij vs. Eclipse: Why IDEA is Better"*. Second, developers can seek answers on Q&A websites such as Stack Overflow or Quora (e.g., *"Apache OpenNLP vs NLTK"*). These expert articles and community answers are indexable by search engines, thus enabling developers to find answers to their technology comparison inquiries.


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.

Fig. 1. A comparative sentence in a post (#1008671) that is not explicitly for technology comparison.

However, there are two limitations with expert articles and community answers.

- *Fragmented view:* An expert article or community answer usually focuses on a specific aspect of some comparable technologies, and developers have to aggregate the fragmented information into a complete comparison in different aspects. For example, to compare *mysql* and *postgresql*, one article [4] contrasts their speed, while another [5] compares their reliability. Only after reading both articles, developers can have a relatively comprehensive overview of these two comparable technologies.
- *Diverse opinions:* One expert article or community answer is based on the author's knowledge and experience. However, the knowledge and experience of developers vary greatly. For example, one developer may prefer *Eclipse* over *Intellij* because *Eclipse* fits his project setting better. But that setting may not be extensible to other developers. At the same time, some developers may prefer *Intellij* over *Eclipse* for other reasons. Such contradictory preferences among different opinions may confuse developers.

The above two limitations create a high barrier for developers to effectively gather useful information about technology differences on the Web in order to tell apart comparable technologies. Although developers may manually

- *M. Shell was with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332. E-mail: see http://www.michaelshell.org/contact.html*
- *J. Doe and J. Doe are with Anonymous University.*

Manuscript received April 19, 2005; revised August 26, 2015.

aggregate relevant information by searching and reading many web pages, that would be very opportunistic and time consuming. To overcome the above limitations, we present the *diffTech* system that automatically distills and aggregates fragmented and trustworthy technology comparison information from the crowd-scale Q&A discussions in Stack Overflow, and assists technology comparison with an informative summary of different aspects of comparison information.

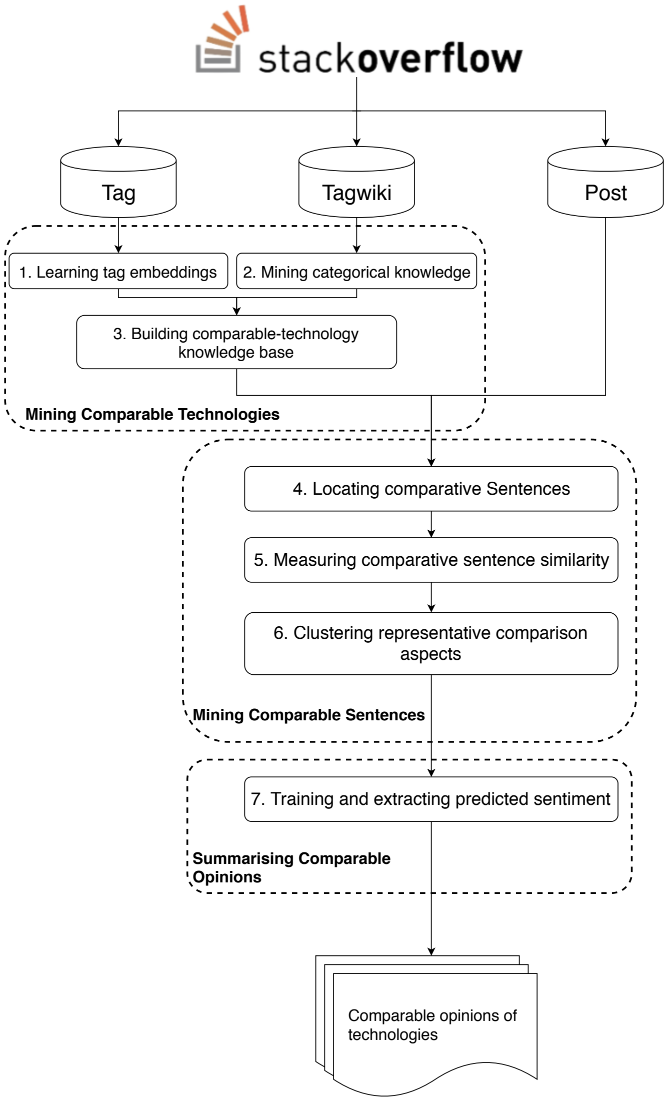
Our system is motivated by the fact that a wide range of technologies have been discussed by millions of users in Stack Overflow [6], and users often express their preferences toward a technology and compare one technology with the others in the discussions. Apart from posts explicitly about the comparison of some technologies, many comparative sentences **hide** in posts that are implicitly about technology comparison. Fig.1 shows such an example: the answer "accidentally" compares the security of *Innodb* and *Myisam*, while the question "What is the correct way to enable query cache?" does not explicit ask for this comparison. Inspired by such phenomenon, we then propose our system to mine and aggregate the comparative sentences in Stack Overflow discussions.

As shown in Fig. 2, we consider Stack Overflow tags as a collection of technology terms and first find comparable technologies by analyzing tag embeddings and categories. Our system then mines comparative opinions from Q&A discussions by analyzing sentence patterns, calculating word mover distance [7] and community detection [8] to cluster comparative sentences into different aspects by which users compare the two technologies for user inspection. Finally, we fine-tune the BERT [9] model to summarize overall sentiment from all the comparative sentences for each comparable technology pairs.

As there is no ground truth for technology comparison, we manually validate the performance of each step of our approach. The experiment results confirm the accuracy of comparable technology identification (90.7%), and distilling comparative sentences (83.7%) from Q&A discussions. By manually building the ground truth, we show that our clustering method (word mover distance and community detection) for comparative sentences significantly outperforms other baselines. Regarding the accuracy of overall sentiment summarization, our model also boost ??% improvement compared with the baselines. In addition, we also demonstrate the generality of our approach by successfully extracting comparative opinions of comparable technologies in other domain-specific datasets.

This paper is an extended version of our earlier study [10], the extension makes the following additional contributions:

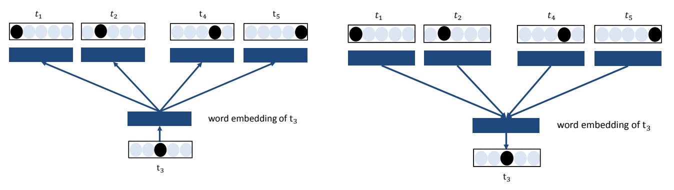
- Apart from the existing single-sentence patterns, we also take the context and coreference into the consideration for discovering more comparative sentences about similar technologies.
- To give developers an clear overview, we develop a classifier for identifying the sentiment towards each technology, and aggregate the crowd opinions in total.
- Based on the extracted comparative opinions, we implement a practical website[1] for developers looking for technology comparison knowledge. The analysis of site visiting logs demonstrate the usefulness of our tool.
- We not only update previous experiments by including new data and new baselines, but also add more detailed analysis of experiments results. We also demonstrate the generality of our method in other domain-specific datasets.


stackoverflow
Tag
Tagwiki
Post
1. Learning tag embeddings
2. Mining categorical knowledge
3. Building comparable-technology knowledge base
Mining Comparable Technologies
4. Locating comparative Sentences
5. Measuring comparative sentence similarity
6. Clustering representative comparison aspects
Mining Comparable Sentences
7. Training and extracting predicted sentiment
Summarising Comparable Opinions
Comparable opinions of technologies
Fig. 2. The overview of our approach

2 Mining Similar Technology

Studies [11], [12], [13] 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. We leverage word embedding techniques to infer semantically related tags, and develop natural language

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



(a) Continuous skip-gram model (b) Continuous bag-of-words model

Fig. 3. The architecture of the two word embeddings models. The continuous skip-gram model predicts surrounding words given the central word, and the CBOW model predicts the central word based on the context words. Note the differences in arrow direction between the two models.

methods to analyze each tag's TagWiki to determine the corresponding technology's category (e.g., algorithm, library, IDE). Finally, we build a knowledge base of comparable technologies by filtering the same-category, semantically-related tags.

2.1 Learning Tag Embeddings

Word embeddings are dense low-dimensional vector representations of words that are built on the assumption that words with similar meanings tend to be present in similar context. Studies [12], [14] show that word embeddings are able to capture rich semantic and syntactic properties of words for measuring word similarity. 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.

There are two kinds of widely-used word embedding methods [14], the continuous skip-gram model [15] and the continuous bag-of-words (CBOW) model. As illustrated in Fig. 3, 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 (Fig. 3(a)), while the CBOW is the opposite, that is, predicting the center word by the context words (Fig. 3(b)). Note that word order within the context window is not important for learning word embeddings.

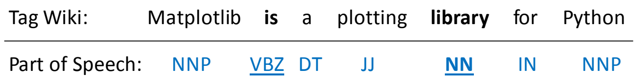
Specifically, given a sequence of training text stream $t_1, t_2, ..., t_k$, the objective of the continuous skip-gram model is to maximize the following average log probability:

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

while the objective of the CBOW model is:

$$L = \frac{1}{K}\sum_{k=1}^{K} \log p(t_k|(t_{k-N}, t_{k-N+1}, ..., t_{k+N})) \quad (2)$$

where t_k is the central word, t_{k+j} is its surrounding word with the distance j, and N indicates the window size. In our application of the word embedding, a tag sentence is a training text stream, and each tag is a word. As tag sentence is short (has at most 5 tags), we set N as 5 in our approach so that the context of one tag is all other tags in the current sentences. That is, the context window contains all other tags as the surrounding words for a given tag. Therefore,



Tag Wiki:	Matplotlib	**is**	a	plotting	**library**	for	Python
Part of Speech:	NNP	VBZ	DT	JJ	**NN**	IN	NNP

Fig. 4. POS tagging of the definition sentence of the tag *RSA*

tag order does not matter in this work for learning tag embeddings.

To determine which word-embedding model performs better in our comparable technology reasoning task , we carry out a comparison experiment, and the details are discussed in Section 6.1.3.

2.2 Mining Categorical Knowledge

In Stack Overflow, tags can be of different categories, such as programming language, library, framework, tool, API, algorithm, 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. For example, the tagWiki of the tag *Matplotlib* starts with the sentence "Matplotlib is a plotting library for Python". Typically, the first noun just after the *be* verb defines the category of the tag. For example, from the tag definition of *Matplotlib*, we can learn that the category of *Matplotlib* is *library*.

Based on the above observation of tag definitions, we use the NLP methods [12], [16] to extract such noun from the tag definition sentence as the category of a tag. Given the tagWiki of a tag in Stack Overflow, we extract the first sentence of the TagWiki description, and clean up the sentence by removing hyperlinks and brackets such as "{}", "()". Then, we apply Part of Speech (POS) tagging to the extracted sentence. POS tagging is the process of marking up a word in a text as corresponding to a particular part of speech, such as noun, verb, adjective. NLP tools usually agree on the POS tags of nouns, and we find that POS tagger in NLTK [17] is especially suitable for our task. In NLTK, the noun is annotated by different POS tags [18] including NN (Noun, singular or mass), NNS (Noun, plural), NNP (Proper noun, singular), NNPS (Proper noun, plural). Fig. 4 shows the results for the tag definition sentence of *Matplotlib*. Based on the POS tagging results, we extract the first noun (*library* in this example) after the be verb (*is* in this example) as the category of the tag. That is, the category of *Matplotlib* is *library*. Note that if the noun is some specific words such as *system*, *development*, we will further check its neighborhood words to see if it is *operating system* or *independent development environment*.

With this method, we obtain 318 categories for the 23,658 tags (about 67% of all the tags that have TagWiki). We manually normalize these 318 categories labels, such as merging *app* and *applications* as *application*, *libraries* and *lib* as *library*, and normalizing uppercase and lowercase (e.g., *API* and *api*). As a result, we obtain 167 categories. Furthermore, we manually categorize these 167 categories into five general categories: programming language, platform, library, API, and concept/standard [19]. This is because the meaning of the fine-grained categories is often overlapping, and there is no consistent rule for the usage of these terms in the TagWiki. This generalization step is necessary, especially for the library tags that broadly refer to the tags whose fine-grained categories can be library, framework, api, toolkit, wrapper, and so on. 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. All these tags are referred to as library tags in our approach.

Although the above method obtains the tag category for the majority of the tags, the first sentence of the TagWiki of some tags is not formatted in the standard "tag be noun phrase" form. For example, the first sentence of the TagWiki of the tag *itext* is "Library to create and manipulate PDF documents in Java", or for *markermanager*, the tag definition sentence is "A Google Maps tool", or for *ghc-pkg*, the tag definition sentence is "The command ghc-pkg can be used to handle GHC packages". As there is no *be* verb in this sentence, the above NLP method cannot return a noun phrase as the tag category. According to our observation, for most of such cases, the category of the tag is still present in the sentence, but often in many different ways. It is very likely that the category word appears as the first noun phrase that match the existing category words in the definition sentence. Therefore, we use a dictionary look-up method to determine the category of such tags. Specially, we use the 167 categories obtained using the above NLP method as a dictionary to recognize the category of the tags that have not been categorized using the NLP method. Given an uncategorized tag, we scan the first sentence of the tag's TagWiki from the beginning, and search for the first match of a category label in the sentence. If a match is found, the tag is categorized as the matched category. For example, the tag *itext* is categorized as *library* using this dictionary look-up method. Using the dictionary look-up method, we obtain the category for 9,648 more tags.

Note that we cannot categorize some (less than 15%) of the tags using the above NLP method and the dictionary look-up method. This is because these tags do not have a clear tag definition sentence, for example, the TagWiki of the tag *richtextbox* states that "The RichTextBox control enables you to display or edit RTF content". This sentence is not a clear definition of what *richtextbox* is. Or no category match can be found in the tag definition sentence of some tags. For example, the TagWiki of the tag *carousel* states that "A rotating display of content that can house a variety of content". Unfortunately, we do not have the category "display" in the 167 categories we collect using the NLP method. When building comparable-technologies knowledge base, we exclude these uncategorized tags as potential candidates.

2.3 Building Similar-technology Knowledge Base

Given a technology tag t_1 with its vector $vec(t_1)$, we first find most similar library t_2 whose vector $vec(t_2)$ is most closed to it, i.e.,

$$\underset{t_2 \in T}{\operatorname{argmax}} \cos(vec(t_1), vec(t_2)) \tag{3}$$

where T is the set of technology tags excluding t_1, and $cos(u, v)$ is the cosine similarity of the two vectors.

Note that tags whose tag embedding is similar to the vector $vec(t_1)$ may not always be in the same category. For example, tag embeddings of the tags *nlp*, *language-model* are similar to the vector $vec(nltk)$. These tags are relevant to the *nltk* library as they refer to some NLP concepts and tasks, but they are not comparable libraries to the *nltk*. In our approach, we rely on the category of tags (i.e., categorical knowledge) to return only tags within the same category as candidates. Some examples can be seen in Table 1.

TABLE 1
Examples of filtering results by categorical knowledge (in red)

Source	Top-5 recommendations from word embedding
nltk	~~nlp~~, opennlp, gate, ~~language-model~~, stanford-nlp
tcp	tcp-ip, ~~network-programming~~, udp, ~~packets~~, ~~tcpserver~~
vim	sublimetext, ~~vim-plugin~~, emacs, nano, gedit
swift	objective-c, ~~cocoa-touch~~, ~~storyboard~~, ~~launch-screen~~
bubble-sort	insertion-sort, selection-sort, mergesort, timsort, heapsort

In practice, there could be several comparable technologies t_2 to the technology t_1. Thus, we select tags t_2 with the cosine similarity in Eq. 3 above a threshold $Thresh$. Take the library *nltk* (a NLP library in python) as an example. We will preserve several candidates which are libraries such as *textblob*, *stanford-nlp*.

3 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. Note that the comparison of comparable technologies may lie in one sentence, but also consecutive sentences. So, we also take the context into consideration for comparison opinion extraction. Displaying all these sentences as a whole may make it difficult for developers to read and digest the comparison information. Therefore, we measure the similarity among the comparative sentences, and then cluster them into several groups, each of which may identify a prominent aspect of technology comparison that users are concerned with.

3.1 Extracting Comparative Sentences

There are three steps to extract comparative sentences of the two technologies. We first carry out some preprocessing of the Stack Overflow post content. Then we locate the sentences that contain the name of the two technologies, and further select the comparative sentences that satisfy a set of comparative sentence patterns.

3.1.1 Preprocessing

To extract trustworthy opinions about the comparison of technologies, we consider only answer posts with positive score points. Then we split the textual content of such answer posts into individual sentences by punctuations like ".", "!", "?". We remove all sentences ended with question mark, as we want to extract facts instead of doubts. We further exclude sentences from question post, as most of them are assumptions or questions to ask. We lowercase all sentences to make the sentence tokens consistent with the technology names because all tags are in lowercase.

Note that within the technical discussion in Stack Overflow, developers may adopt some pronoun like "it", "that", "which" to represent the technology. For example, given the

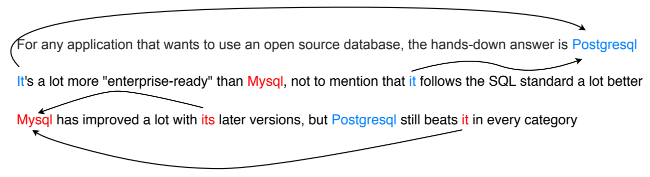

For any application that wants to use an open source database, the hands-down answer is Postgresql
It's a lot more "enterprise-ready" than Mysql, not to mention that it follows the SQL standard a lot better
Mysql has improved a lot with its later versions, but Postgresql still beats it in every category

Fig. 5. An example of coreference resolution in comparative sentence.

paragraph "*For any application that wants to use an open source database, the hands-down answer is postgresql. It's a lot more enterprise-ready than mysql, not ot mention that it follows the SQL standard a lot better. Mysql has improved a lot with its later versions, but Postgresql still beats it in every category.*" as seen in Figure 5, the two "it" in the second sentence refers to "postgresql" in the first sentence. And But with some cases, the reference may be too far from its source, leading to the negative influence to the location of comparative sentences. Therefore, we adopt the coreference resolution algorithm [20] to recover the pronoun before locating comparative sentences.

Conventionally, the coreference resolution is based on a set of manual-crafted rules by analyzing dependency tree of words in the sentence. However, those rules may not be scalable, therefore, we adopt the state-of-the-art neural network based coreference method, named NeuralCoref[2]. It feeds word embedding of potential words around each mention to two neural networks with one for giving score for finding a possible antecedent, and the other giving a score for a mention having no antecedent. Given two scores, we compare them and pick the highest score to determine whether the mention has an antecedent and which word it is.

3.1.2 Locating Candidate Sentences

The sentences mentioning a pair of comparable technologies may contain the comparison opinion of them. According to our observation, the comparison may occur in either one single sentence, or the contextual sentences. Therefore, the candidate comparison sentences may be single sentences mentioning two comparable technologies, or the consecutive sentences with each of it containing one comparable technology respectively.

Note that using only the tag names is not enough. As posts in Stack Overflow are informal discussions about programming-related issues, users often use alias to refer to the same technology. Aliases of technologies can be abbreviations, synonyms and some frequent misspellings. For example, there are several different alias of "visual studio" in many forms such as "visual-studio" (synonym), "vs" (abbreviation), and "visual studion" (misspelling) in the discussions. The presence of such aliases will lead to significant missing of comparative sentences, if we match technology mentions in a sentence with only tag names. Chen et al.'s work [21] builds a large thesaurus of morphological forms of software-specific terms, including abbreviations, synonyms and misspellings. Table 2 shows some examples of technologies aliases in this thesaurus. Based on this thesaurus, we find 7310 different alias for 3731 software technologies. These aliases help to locate more candidate comparative sentences that mention certain technologies.

TABLE 2
Examples of alias

Tech term	Synonyms	Abbreviation
photoshop	adobe_photoshop, photoshops	ps
dataframe	data-frame, pd.dataframe	df
libgd	gd_library	gd
microsoft sql server	ms-sql, msql, ms_sql	mssql
breadth-first search	breadth first search, breadth-first-search	bfs

3.1.3 Selecting Comparative Sentences

To identify comparative sentences from candidate sentences, we develop two sets of comparative sentence patterns: one for the single sentence and the other for contextual sentences. Single sentence pattern is a sequence of POS tags. For example, the sequence of POS tags "*RBR JJ IN*" is a pattern that consists of a comparative adverb (*RBR*), an adjective (*JJ*) and subsequently a preposition (*IN*), such as "more efficient than", "less friendly than", etc. 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).

Based on data observations of comparative sentences, we summarise four comparative patterns for single sentences as seen in Table 3. To make the patterns more flexible, we use a wildcard character to represent a list of arbitrary words to match the pattern. For each sentence mentioning the two comparable technologies, we obtain its POS tags and check if it matches any one of four patterns. If so, the sentence will be selected as a single comparative sentence.

For contextual candidate sentences, we develop three patterns for identifying comparative opinions in Table 4. The first two patterns are similar to the first two patterns in single sentence, as sometimes developers in Stack Overflow may tell the comparison in two consecutive sentences e.g., "**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." Therefore, for consecutive sentences containing comparable technologies separately, we will check if either sentence fit in these two patterns. Note that there are four single-sentence patterns, while we only take the first two of them into the contextual-sentence patterns. According to our observation, the last two single-sentence patterns are not very accurate (The details will be discussed in Section 6.2).

The other pattern is that one contextual sentences is the affirmation (AFF) while the other is negation (NEG) with at least one comparable technology as the subject of one sentence. For example, the prior sentence "Cassini does not support HTTPs" is the negation sentence, and the latter sentence "However, you can use IIS to do this" is the affirmation sentence, resulting in the comparison opinion that IIS can support the feature that is not owned by Cassini. To detect such pattern, we first create a list of negation words defined as "*NW*" in POS tag such as *??, ??, ??*[3]. For

2. https://github.com/huggingface/neuralcoref

3. The full list can be seen at ??

TABLE 3
Patterns of comparative single sentences 1. There should be two examples in the first pattern.

Single sentence			
No.	**Pattern**	**Sequence example**	**Original sentence**
1	*TECH * VBZ * (JJR ∨ RBR)*	innodb has 30 higher	InnoDB has 30% higher performance than MyISAM on average.
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.

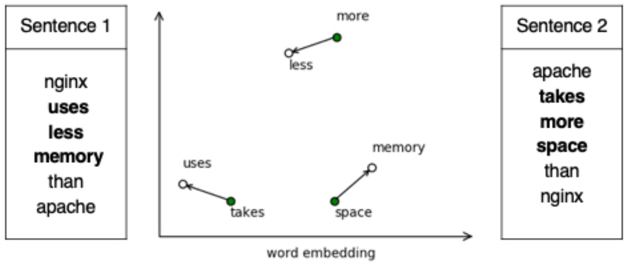


Fig. 6. An illustration of measuring similarity of two comparative sentences This figure is blurred

each pair of sentences we check if either of them match the pattern *"TECH NW"*, so that we locate the negation sentence. Then, for the other affirmation sentence, it must match any one of the following pattern:*"TECH VB NN"*, *"TECH VB JJ"*, *"TECH VB RB"*, and *"VB TECH TO VB"*.

3.2 Measure Sentence Similarity

To measure the similarity of two comparative sentences, we adopt the Word Mover's Distance [7] which is especially useful for short-text comparison. Given two sentences S_1 and S_2, we take one word i from S_1 and one word j from S_2. Let their word vectors be v_i and v_j. The distance between the word i and the word j is the Euclidean distance between their vectors, $c(i,j) = ||v_i - v_j||_2$. To avoid confusion between word and sentence distance, we will refer to $c(i,j)$ as the cost associated with "traveling" from one word to another. One word i in S_1 may move to several different words in the S_2, but its total weight is 1. So we use $T_{ij} \geq 0$ to denote how much of word i in S_1 travels to word j in S_2. It costs $\sum_j T_{ij}c(i,j)$ to move one word i entirely into S_2. We define the distance between the two sentences as the minimum (weighted) cumulative cost required to move all words from S_1 to S_2, i.e., $D(S_1, S_2) = \sum_{i,j} T_{ij}c(i,j)$. This problem is very similar to transportation problem i.e., how to spend less to transform all goods from source cities $A_1, A_2, ...$ to target cities $B_1, B_2, ...$. Getting such minimum cost actually is a well-studied optimization problem of earth mover distance [22], [23].

To use word mover's distance in our approach, 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. Word embedding has been shown to be able to capture rich semantic and syntactic information of words. Our approach does not consider word mover's distance for all words in a sentence. Instead, for each comparative sentence, we extract only keywords with POS tags that are most relevant to the comparison, including adjectives (JJ), comparative adjectives (JJR) and nouns (NN, NNS, NNP and NNPS), not including the technologies under comparison. 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

$$similarity\ score(S_1, S_2) = \frac{1}{1 + D(S_1, S_2)}$$

The similarity score is in range $(0, 1)$, and the higher the score, the more similar the two sentences. If the similarity score between the two sentences is larger than the threshold, we regard them as similar. The threshold is 0.55 in this work, determined heuristically by a small-scale pilot study. We show some similar comparative sentences by word mover's distance in Table 5.

To help reader understand word movers' distance, we show an example in Figure 6 with two comparative sentences for comparing *apache* and *nginx*: *"nginx uses less memory than apache"* and "'*Apache takes more space than nginx'*. The keywords in the two sentences that are most relevant to the comparison are highlighted in bold. We see that the minimum distance between the two sentences is mainly the accumulation of word distance between pairs of similar words *(uses, takes)*, *(less, more)*, and *(memory, space)*. As the distance between the two sentences is small, the similarity score is high even though the two sentences use rather different words and express the comparison in reverse directions.

3.3 Clustering Representative Comparison Aspects

For each pair of comparable technologies, we collect a set of comparative sentences about their comparison in Section 3.1. Within these comparative sentences, we find pairs of similar sentences in Section 3.2. We take each comparative sentence as one node in the graph. If the two sentences are determined as similar, we add an edge between them in the graph. In this way, we obtain a graph of comparative sentences for a given pair of comparative technologies.

Although some comparative sentences are very different in words or comparison directions (examples shown in Fig. 6 and Table 5), they may still share the same comparison opinions. In graph theory, a set of highly correlated nodes is referred to as a community (cluster) in the network. Based on the sentence similarity, we cluster similar opinions by applying the community detection algorithm to the graph of comparative sentences. In this work, we use the Girvan-Newman algorithm [8] which is a hierarchical community detection method. It uses an iterative modularity maximization method to partition the network into a finite number of disjoint clusters that will be considered as communities.

TABLE 4
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.

TABLE 5
Examples of similar comparative sentences by Word Mover's Distance

Comparable technology pair	Comparative sentences
quicksort & mergesort	Quicksort is done in place and doesn't require allocating memory, unlike mergesort. Mergesort would use more space than quicksort.
swing & awt	Consider using swing which has much better performance over the old heavyweight awt. Yes swing has newer and better apis than awt.
google-chrome & safari	But safari takes more time than google-chrome browser. I get a lot of results about how safari is slower than google-chrome.
get & post	Post is also more secure than get because you aren t sticking". When you use post data is a a lot more safer than get and you can send large no. of request parameters.
tcp & udp	Tcp is a slower more reliable protocol than udp is. This is the reason why udp is much faster than tcp.

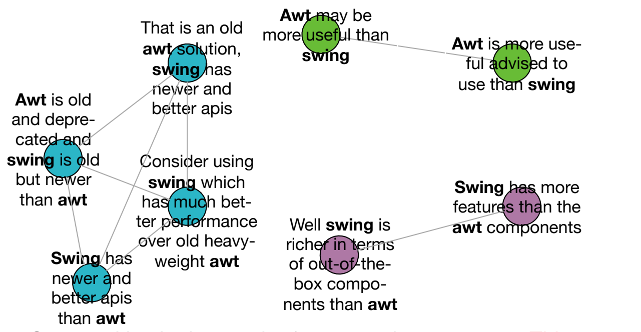

That is an old **awt** solution, **swing** has newer and better apis
Awt may be more useful than **swing**
Awt is more useful advised to use than **swing**
Awt is old and deprecated and **swing** is old but newer than **awt**
Consider using **swing** which has much better performance over old heavyweight **awt**
Swing has more features than the **awt** components
Well **swing** is richer in terms of out-of-the-box components than **awt**
Swing has newer and better apis than **awt**

Fig. 7. Communities in the graph of comparative sentences This example does not show the power of community detection, as each subgraph is regarded as a community.

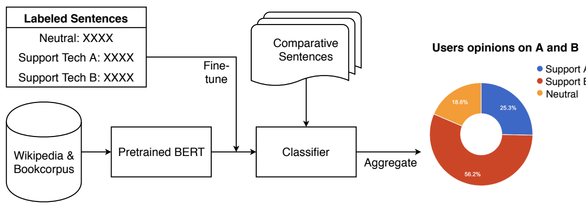


Fig. 8. Summerizing overall opinions towards each technology

Each node must be assigned to exactly one community. Fig. 7 shows the graph of comparative sentences for the comparison of *awt* and *swing* (two java toolkits), in which each node is a comparative sentence, and the detected communities are visualized in the same color.

As seen in Fig. 7, each community may represent a prominent comparison aspect of the two comparable technologies. But some communities may contain too many comparative sentences to understand easily. Therefore, we use TF-IDF (Term Frequency Inverse Document Frequency) to extract keywords from comparative sentence in one community to represent the comparison aspect of this community. TF-IDF is a statistical measure to evaluate the importance of a word to a document in a collection. It consists of two parts: term frequency (TF, the number occurrences of a term in a document) and inverse document frequency (IDF, the logarithm of the total number of documents in the collection divided by the number of documents in the collection that contain the specific term). For each community, we remove stop words in the sentences, and regard each community as a document. We take the top-3 words with largest TF-IDF scores as the representative aspect for the community. Table 6 shows the comparison aspects of four communities for comparing *UDP* with *TCP*. The representative keywords directly show that the comparison between *UDP* with *TCP* mainly focuses on four aspects: speed, header, usability, and fields that they are used for.

4 Summarizing Overall Opinion

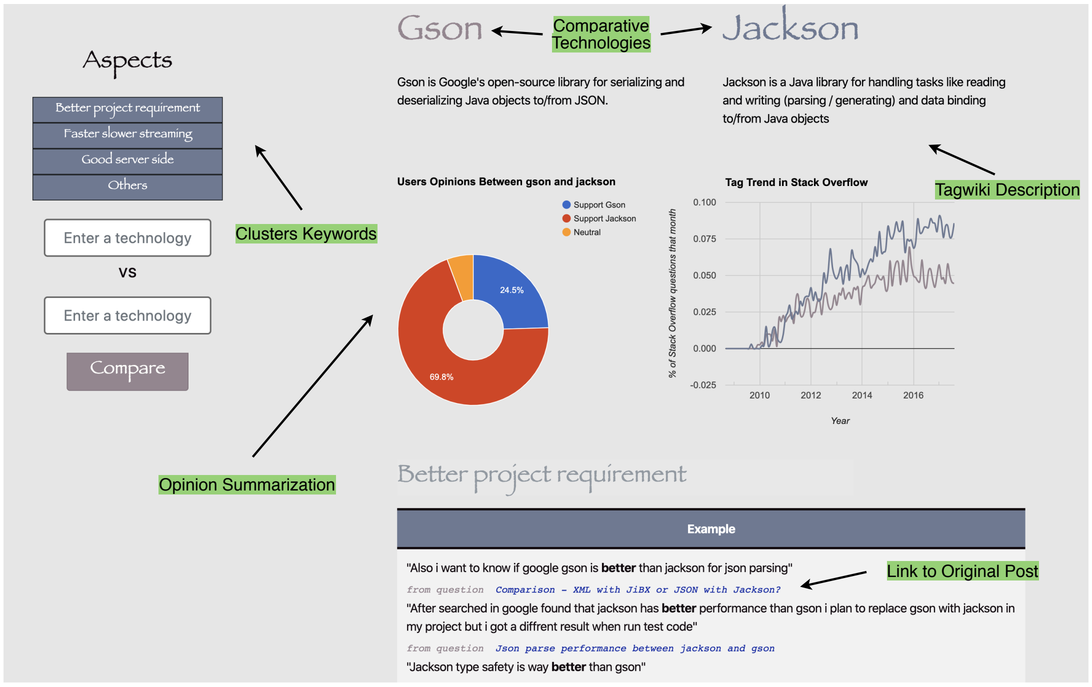
For some comparable technologies, there may be too many comparison opinions which are too time-consuming for developers to read all of them. Therefore, we further develop a sentiment classifier based on the BERT model [9] for automatically distilling the overall opinion towards the comparable technologies. That summarization can be an important criteria for developers to determine which technology for adopting. The general process of summarising overall opinion is shown in Fig. 8

For summarizing the overall opinions toward comparable technologies, we formulate it as a classification problem. Given a set of comparative sentences about them, we build a classifier for identifying which technology does each sentence support. By counting the total number of support sentences for each technology, we obtain the sentiment summarization of them. We replace the comparable technology pairs with two unique tokens i.e., TechA (first occurrence) and TechB (second occurrence) to generalize different technology pairs. We randomly select 801 comparative sentences and replace the comparable technology pairs with two unique tokens i.e., TechA (first occurrence) and TechB (second occurrence) to generalize different technology pairs.

However, the supervised learn requires a large-scale labeled dataset which labour-extensive and time-consuming.

TABLE 6
The representative keywords for clusters of *UDP* and *TCP*.

Representative keywords	Comparative sentences
faster slower reliable	One often finds the argument that udp is faster then tcp Udp is way lighter and faster but somewhat less reliable than tcp. Udp is generally faster than tcp as it does not have to do the overhead checking of consistency that tcp must deal with.
header connection size	You will notice that the tcp header has more fields than the udp header and many of those fields will be populated. Udp communication is connection less as compared to tcp which need a connection. The header size of udp is less than tcp.
harder, easier,travelsal	Doing p2p nat traversal over tcp is a bit harder than udp. Keep in mind that implementing udp traversal is easier than tcp. It was introduced since the nat traversal for tcp is much more complicated than udp.



Gson is Google's open-source library for serializing and deserializing Java objects to/from JSON.

Jackson is a Java library for handling tasks like reading and writing (parsing / generating) and data binding to/from Java objects

"Also i want to know if google gson is **better** than jackson for json parsing"

from question *Comparison - XML with JiBX or JSON with Jackson?*

"After searched in google found that jackson has **better** performance than gson i plan to replace gson with jackson in my project but i got a diffrent result when run test code"

from question *Json parse performance between jackson and gson*

"Jackson type safety is way **better** than gson"

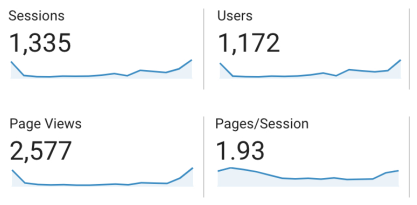
Fig. 9. The screenshot of our website *DiffTech* and some important elements in the website 1) The first sentence is a question actually that should be removed; 2) Can we find change another example that there is no explicit comparison posts? 3) For the webpage, can we add a line after the tagwiki, and a line after the trend? 4) For the trend graph, how to know which line is for which technology? 5) The clustering of this example is not good, so change another example.

To overcome this issue, we adopt the state-of-the-art model, BERT for this work. BERT (Bidirectional Encoder Representations) [9] is the first deeply bidirectional unsupervised language representation, pre-trained using only a plain text corpus [24]. For each word, BERT considers its different meanings in different contextual environments. For example, the word 'chair' would have same representation in 'seat a the chair' and 'the chair of the committee' for a context-free models. A contextual model like BERT would have one representation of each word that is based on the other words in its surroundings. BERT also introduce the "masked language modeling", which is to make sure BERT is a deep bidirectional representation. While doing the training, the model will randomly mask some percentage of the input tokens, and predict them. For example, the **Input** is "A Lady went to the [MASK1]. She bought a [MASK2] of pork belly." and the **Labels** are "MASK1=butcher,MASK2=tray". Those features make BERT different but powerful than other pre-trained representations. In addition, Google release a pretrained BERT model [25] based on the large-scale corpus including Wikipedia and BookCorpus [26] which contains 2,500 million words from English wikipedia sentences and 800 million words from 11,038 unpublished books. As the pretrained model is well trained based on such big data and long time, it already encodes a lot of information about our general language. To leverage that learned knowledge, we only fine tune the pretrained BERT model by adding a new layer on the top of it. We freeze the parameters of existing pretrained model but only train the final layer based on a small manually-labeled dataset for adapting it with domain-specific information.

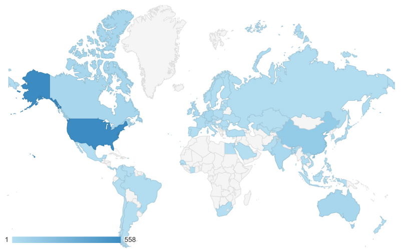
5 IMPLEMENTATION

5.1 Dataset

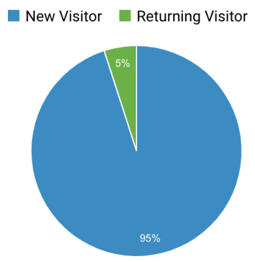
We take the latest Stack Overflow data dump (released on 4 September 2019) as the data source. It contains 18,154,493 questions with 55,665 unique tags, and 27,765,324 answers. With the approach in Section 2, we collect in total 14,876 pairs of comparable technologies. Among these technologies, we extract 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 technology comparison.



(a) Visiting Statistics



(b) Geo Map



(c) Returning Users

Fig. 10. The traffic of our website from Google AnalyticsThis is too large, make it to a single-column figure with (a) and (c) in the first row while the map in the second row.

5.2 Tool Support

Based on the our proposed approach, we implement a practical website[4] 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, the tool provides the link of each comparative sentence to its corresponding Stack Overflow post so that users can easily find more detailed content. For example, as shown in Fig 9, given a pair of comparable technologies *Gson* and *Jackson*, the definition of them from TagWiki is below each technology name. About 69.8% of the users support using *Jackson* instead of *Gson*. The post trend shows that although both technologies are more and more popular, there are more discussion about xxx... We could also see that the method clustered all the comparative sentences into four clusters and has been clarified at 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. User can click the link for retrieving more content.

5.3 Visitor Analysis

We initially released our website on September 2018, and has updated data and layout on September 2019. We posted the news on some websites such as StackApps [?] and Reddit [?] as well for propaganda purpose. You may update the following part if you have more data. We embedded Google Analytics into our website to monitor the site traffic. Fig. 10 shows that about 1,200 users from 75 different countries visited our website. These users viewed 2,577 different pages and in average 1.93 pages for each session. 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. By tracking their IP, we find that most users come from the U.S. (47.6%), and other countries such as China (10.8%), Australia (4%), Canada (4%), etc. Nearly 5% of the users come back to visit our website again, indicating the usefulness and attraction of our site.

Among all visiting technology pairs, users compared *Mysql* with *Postgresql* most frequently (197 times) and other frequent technology pairs like *tco vs udp*, *lisp vs scheme*. Apart

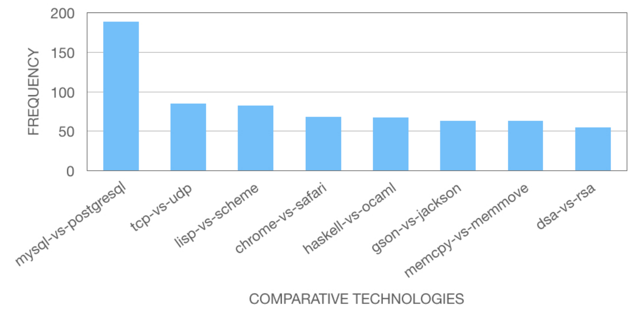


Fig. 11. Top 8 most visited pages 1)Make the labels larger 2)make it more flat for taking less vertical space.

from the visiting, some users also post comments under our advertisements in Reddit 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 suggestion for improving our tool like "??", and we will further add more features in our site to make it more powerful.

6 Experiment

In this section, we evaluate each step of our approach. As there is no ground truth for technology comparison, we have to manually check the results of each step or build the ground truth. And as it is clear to judge whether a tag is of a certain category from its tag description, whether two technologies are comparable, and whether a sentence is a comparative sentence, we recruit two Master students to manually check the results of these three steps. Only results that they both agree will be regarded as ground truth for computing relevant accuracy metrics, and those results without consensus will be given to the third judge who is a PhD student with more experience. All three students are majoring in computer science and computer engineering in our school, and they have diverse research and engineering background with different software tools and programming languages in their work. In addition, we release all experiment data and results in our website[5].

6.1 Accuracy of Extracting Comparable Technologies

This section reports our evaluation of the accuracy of tag category identification, the important of tag category for

4. https://difftech.herokuapp.com/

5. https://sites.google.com/view/difftech/

filtering out irrelevant technologies, and the impact of word embedding models and hyperparameters.

6.1.1 The Accuracy of Tag Category

From 33,306 tags with tag category extracted by our method, we randomly sample 1000 tags whose categories are determined using the NLP method, and the other 1000 tags whose categories are determined by the dictionary look-up method (see Section 2.2). Among the 1000 sampled tag categories by the NLP method, categories of 838 (83.8%) tags are correctly extracted by the proposed method. For the 1000 sampled tags by the dictionary look-up method, categories of 788 (78.8%) tags are correct.

According to our observation, two reasons lead to the erroneous tag categories. First, some tag definition sentences are complex which can lead to erroneous POS tagging results. For example, the tagWiki of the tag *rpy2* states that "RPy is a very simple, yet robust, Python interface to the R Programming Language". The default POS tagging recognizes *simple* as the noun which is then regarded as the category by our method. Second, the dictionary look-up method sometimes makes mistakes, as the matched category may not be the real category. For example, the TagWiki of the tag *honeypot* states "A trap set to detect or deflect attempts to hack a site or system". Our approach matches the *system* as the category of the *honeypot*.

6.1.2 The Importance of Tag Category

To check the importance of tag category for the accurate comparable technology extraction, we set up two methods, i.e., one is word embedding and tag category filtering, and the other is only with word embedding. The word embedding model in two methods are both skip-gram model with the word embedding dimension as 800. We randomly sample 150 technologies pairs extracted from each method, and manually check the if the extracted technology pair is comparable or not. It shows that the performance of model with tag category (90.7%) is much better than that without the tag category filtering (29.3%).

6.1.3 The impact of parameters of word embedding

There are two important parameters for the word embedding model, and we test its impact on the the performance of our method. First, we compare the performance of CBOW and Skip-gram mentioned in Section 2.1 by randomly sampling 150 technology pairs extracted by each method under the same parameter setting (the word embedding dimension is 400). The results show that Skip-gram model (90.7%) outperforms the CBOW model (88.7%), but the difference is marginal.

Second, we randomly sample 150 technologies pairs by the skip-gram model with different word embedding dimensions, and manually check the accuracy. From the dimension 200 to 1000 with the step as 200, the accuracy is 70.7%, 72.7%, 81.3%, 90.7%, 87.3%. We can see that the model with the word embedding dimension as 800 achieves the best performance. Finally, we take the Skip-gram model with 800 word-embedding dimension as the word embedding model to obtain the comparable technologies in this work.

TABLE 7
The accuracy of comparative sentences extraction

Single sentence				
No.	Pattern	**#right**	**#wrong**	**Accuracy**
1	*TECH * VBZ * JJR/RBR*	47	3	94%
2	*(RBR JJ) /JJR * CIN * TECH*	46	4	92%
3	*CV * CIN TECH*	42	8	82%
4	*CV VBG TECH*	38	12	76%
Contextual sentences				
No.	Pattern	**#right**	**#wrong**	**Accuracy**
1	*TECH * VBZ * JJR/RBR*	47	3	94%
2	*(RBR JJ) /JJR * CIN * TECH*	47	3	94%
3	*AFF-NEG*	45	5	90%
Total		312	38	89.1%

6.2 Accuracy and coverage of comparative sentences

We evaluate the accuracy and coverage of our approach in finding comparative sentences from the corpus. We first randomly sample 350 extracted comparative sentences (50 sentences for each comparative sentence pattern in Table 3 and Table 4). We manually check the accuracy of the sampled sentences and Table 7 shows the results. The overall accuracy of comparative sentence extraction is 89.1%, and our approach is especially accurate for the first 2 patterns for single sentence and all three patterns for contextual sentences. The last two patterns for single sentence do not get good performance due to the relatively loose conditions. That is also why we do not take those two patterns into extracting contextual comparative sentences.

We further check the wrong extraction of comparative sentences and find that most errors from single sentence patterns are caused when the two comparative technologies are listed together for a similar feature, i.e. "*Java framework awt or swing makes more sense for something this simple*" or when the two comparative technologies are used to compare with a third technology like "*I mean it came as a surprise to me that drupal is so much faster than wordpress and joomla*". In addition, although some sentences do not contain the question mark, they are actually interrogative sentence such as "*I also wonder if postgresql will be a win over mysql*".

6.3 Accuracy of clustering comparative sentences

We evaluate the performance of our opinion clustering method by comparing it with the baseline methods.

6.3.1 Baseline

We set up three baselines to compare with our comparative sentence clustering method. The first baseline is the traditional TF-IDF [27] with K-means [28], the second baseline is based on the document-to-vector deep learning model (i.e., Doc2vec [29]) with K-means. The last baseline is BERT [9] model with K-means, and the pretrained BERT model is directly downloaded from the Google Official Site[6]. All of methods first convert the comparative sentences into a list of vectors for each pair of comparable technologies . Then, we carry out K-means algorithms to cluster the sentence vectors into N clusters. To make the baseline as competitive as possible, we set N at the cluster number of the ground truth. In contrast, our method specifies its cluster number by community detection which may differ from the cluster number of the ground truth.

6. ??

TABLE 8
Ground truth for evaluating clustering results

No.	Technology pair	#comparative sentence	#cluster
1	*compiled & interpreted language*	34	4
2	*sortedlist & sorteddictionary*	18	4
3	*quicksort & heapsort*	51	5
4	*ant & maven*	83	9
5	*lxml & beautifulsoup*	52	6
6	*awt & swing*	53	7
7	*jackson & gson*	39	3
8	*jruby & mri*	25	4
9	*pypy & cpython*	72	7
10	*memmove & memcpy*	33	3

6.3.2 Ground Truth

As there is no ground truth for clustering comparative sentences, we ask two Master students mentioned before to manually build a small-scale ground truth. We randomly sample 15 pairs of comparable technologies with different number of comparative sentences. For each technology pair, the two students read each comparative sentence and each of them will individually create several clusters for these comparative sentences. Note some comparative sentences are unique without any similar comparative sentence, and we put all those sentences into one cluster. Then they will discuss with the Ph.D student about the clustering results, and change the clusters accordingly. Finally, they reach an agreement for 10 pairs of comparable technologies. We take these 10 pairs as the ground truth whose details can be seen in Table 8.

6.3.3 Evaluation Metrics

Given the ground truth clusters, many metrics have been proposed to evaluate the clustering performance in the literature. In this work, we take the Adjusted Rand Index (ARI) [30], Normalized Mutual Information(NMI) [31], homogeneity, completeness, V-measure [32], and Fowlkes-Mallows Index (FMI) [33]. For all six metrics, higher value represents better clustering performance. For each pair of comparable technologies, we take all comparative sentences as a fixed list, and G as a ground truth cluster assignment and C as the algorithm clustering assignment.

Adjusted Rand Index (ARI) measures the similarity between two partitions in a statistical way. It first calculates the raw Rand Index (RI) by $RI = \frac{a+b}{C_2^N}$ where a is the number of pairs of elements that are in the same cluster in G and also in the same cluster in C, and b is the number of pairs of elements that are in different clusters in G and also in different clusters in C. C_2^N is the total number of possible pairs in the dataset (without ordering) where N is the number of comparative sentences. To guarantee that random label assignments will get a value close to zero, ARI is defined as

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]}$$

where $E[RI]$ is the expected value of RI.

Normalized Mutual Information (NMI) measures the mutual information between the ground truth labels G and the algorithm clustering labels C, followed by a normalization operation:

$$NMI(G, C) = \frac{MI(G, C)}{\sqrt{H(G)H(C)}}$$

where $H(G)$ is the entropy of set G i.e., $H(G) = -\sum_{i=1}^{|G|} P(i)\log(P(i))$ and $P(i) = \frac{G_i}{N}$ is the probability than an objet picked at random falls into class G_i. The $MI(G, C)$ is the mutual information between G and C where $MI(G, C) = \sum_{i=1}^{|G|}\sum_{j=1}^{|C|} P(i, j)\log(\frac{P(i,j)}{P(i)P(j)})$

Homogeneity (HOM) is the proportion of clusters containing only members of a single class by

$$h = 1 - \frac{H(G|C)}{H(G)}$$

Completeness (COM) is the proportion of all members of a given class are assigned to the same cluster by

$$c = 1 - \frac{H(C|G)}{H(C)}$$

where $H(G|C)$ is the conditional entropy of the ground-truth classes given the algorithm clustering assignments.

V-measure (V-M) is the harmonic mean of homogeneity and completeness

$$v = 2 \times \frac{h \times c}{h + c}$$

Fowlkes-Mallows Index (FMI) is defined as the geometric mean of the pairwise precision and recall:

$$FMI = \frac{TP}{\sqrt{(TP + FP)(TP + FN)}}$$

where TP is the number of True Positive (i.e. the number of pairs of comparative sentences that belong to the same clusters in both the ground truth and the algorithm prediction), FP is the number of False Positive (i.e. the number of pairs of comparative sentences that belong to the same clusters in the ground-truth labels but not in the algorithm prediction) and FN is the number of False Negative (i.e the number of pairs of comparative sentences that belongs in the same clusters in the algorithm prediction but not in the ground truth labels).

6.3.4 Overall Performance

Table 9 shows the evaluation results. Three baseline methods has similar results whereas tf-idf and Sentence-Bert are slitly better, but our model significantly outperforms all models in all six metrics.

According to our inspection of detailed results, we find two reasons why our model outperforms the baselines. First, our model can capture the semantic meaning of comparative sentences. TF-IDF can only find similar sentences using the same words but count similar words like "secure" and "safe" as unrelated. While the sentence vector from Doc2vec is easily influenced by the noise as it takes all words in the sentence into consideration. The BERT model is trained by general corpus (like wikipeidia), which is different from our technology-specific dataset, resulting in the low-quality representation of domain-specific words. Second, constructing the similar sentences as a graph in our model explicitly encodes the sentence relationships. The community detection

TABLE 9
Clustering performance

Method	ARI	NMI	HOM	COM	V-M	FMI
TF-IDF+Kmeans	0.09	0.24	0.35	0.28	0.26	0.25
Doc2vec+Kmeans	0.01	0.17	0.29	0.20	0.18	0.18
Bert+Kmeans	0.10	0.24	0.35	0.28	0.26	0.26
DIFFTECH	**0.65**	**0.67**	**0.76**	**0.77**	**0.72**	**0.71**

TABLE 10
Examples of labeled sentences All three examples are not good.

Label	Sentence
Neutral	I am not sure if TechA server will be much better than TechB
Support Tech A	I do know TechA better than TechB
Support Tech B	Also have a look at TechB which is safer version of TechA

based on the graph can then easily put similar sentences into clusters. In contrast, for the four baselines, the error brought from them is accumulated and amplified to K-means in the clustering phase.

6.4 Accuracy of Opinion Summarization

In this section, we evaluate the accuracy of our overall opinion summarization. We compare our methods with five different baselines that are commonly used in sentence categorization. We implement those baselines and use four metrics to compare them with our methods, and our methods performs better in opinion summarization than the rest of them.

6.4.1 Baselines

We set up five baselines to compare with our method. The first baseline is TF-IDF [27] with SVM [34], whereas the second one is N-gram vectorizer with SVM. The two models firstly covert the train sentences into vectors. Then, we use the Support Vector Machines(SVM) to make predictions for the given sentences. Another two baselines are more advanced neural network based approaches including Convolutional Neural Network(CNN) and Long Short Term Memory(LSTM). The last baseline is FastText [35], which is a pre-trained model released by Facebook for text classification and representation learning. To keep the comparison fair, the dataset used for training and testing our model and baselines is the same.

6.4.2 Dataset

As we are adopting the supervised model for opinion summarization, we manually create a set of labels annotating which technology does the sentence support. We randomly select 801 and 150 comparative sentences as the training and testing datasets respectively. We replace the comparable technology pairs with two unique tokens i.e., TechA (first occurrence) and TechB (second occurrence) to generalize different technology pairs. For each sentence, two authors in this paper label it as one of three categories including supporting TechA, supporting TechB, or neutral, shown in Table 10 Note that the two annotators work individually and the sentence can be taken in to the consideration only when they reach the agreement. There are more sentences about supporting TechA or TechB, but fewer neutral sentences. To balance the data from three categories, we remove some comparative sentences about supporting certain technology, with 267 sentences for supporting TechA, 267 for supporting TechB and 267 neutral ones.

TABLE 11
Opinion summarization performance

Method	accuracy	precision	recall	f1-score
TF-IDF+SVM	0.533	0.506	0.513	0.507
n-gram+SVM	0.753	0.741	0.749	0.741
CNN	0.62	0.573	0.585	0.57
LSTM	0.633	0.613	0.622	0.614
FastText	0.68	0.658	0.663	0.659
DIFFTECH	**0.847**	**0.843**	**0.854**	**0.841**

6.4.3 Metrics

As this is a typical multi-class classification task, we adopt four metrics for measuring the performance of our model including accuracy, precision, recall, and F1-score. All of these metrics are based on the four statistics: TP (true positive) represents the number of sentences that are correctly classified as one label; TN (true negative) represents the number of sentences that are corrected classified as not that label; FP (false positive) represents the number of sentences that are predicted as the label, but the it's not; FN (false negative) represents the number of sentences that are predicted as not the label, but actually it is of the label;

Accuracy is the proportion of correct result among the whole test case: $Accuracy = \frac{TP+TN}{(TP+FP+TN+FN)}$

Precision is the ratio of the correctly predicted positive records to the whole positive records: $Precision = \frac{TP}{(TP+FP)}$

Recall is calculating the correctly predicted positive records among all predicted positive records: $Recall = \frac{TP}{(TP+FN)}$

F1-score is the harmonic mean of precision and recall, which can combine both of the two metrics above: $F1-score = 2 * \frac{Precision*Recall}{(Precision+Recall)}$

Note that these metrics are default for binary classification. As our task is a multi-class classification, we adopt the macro average [36] of these metrics for each class as the overall performance for this model. A higher value represents better performance for all the metrics.

6.4.4 Overall Performance

Table 11 shows the experiment results. We can see that TF-IDF with SVM has the worst performance. CNN, LSTM and FastText are both better than TF-IDF, while n-gram model with SVM has the highest among other baselines. But compared with all baselines, our model has the best performance with ??%, ??%, ??%, ??% increase than the best baseline in term of accuracy, precision, recall and f1-score. Note that TF-IDF has the worst performance as it can't capture the semantic information of complex sentences. For deep-learning based methods, CNN, LSTM, and FastText, their performance is not good as expected due to the small size of our training corpus. In contrast, our model is based on the pre-trained BERT model which both distill the knowledge from large-scale general corpus and also provide sentence-level semantic encoding.

Despite good performance of our model, there are still some wrong cases within our model. We manually check those wrongly predicted sentences, and find that most wrong predictions are caused by the neutral ones. For example, our model make mistake in the sentence "I am not sure if VMware Server will be much better than VirtualBox."

TABLE 12
The accuracy of comparative sentences extraction

Source	#right	#wrong	Accuracy
Unix and Linux	44	6	88%
Super User	42	8	82%

into supporting VMware. The sentence structure may be too complicated for the BERT model, but more labeled dataset in the future may help mitigate that effect. And some of the wrong predicted cases are the sentence is comparing two technologies, but they are not expressing their feelings on which they prefer, but focus on how different they are. For example in the sentence "char is guaranteed to be smaller than int" is simply indicate that char type takes less storage space than int type, it's a neutral express, but our model would take it into support int.

6.5 Generality of DiffTech

In order to show the generality of our method, we select another two Q&A sites from the Stack Exchange Networks. The two sites are Super User[7], which is the Q&A site for computer enthusiasts and power users, and Unix & Linux [8], which is for users of Linux, FreeBSD and other Unix-like operating systems. We adopt our approach to both sites, and collect 858 comparative sentences from Super User site as well as 611 comparative sentences from Unix & Linux. Note that the overall questions of Super User and Unix & Linux is less than 3% of that in Stack Overflow, hence only a small number of comparative sentences. We also cluster the comparative sentences based on technology pairs, and summarize the overall opinions.

As the extraction and clustering of the comparative sentences are the core parts of our DIFFTECH, we only evaluate the performance of these two steps. For the accuracy of extracting comparative sentences, we randomly select 50 sentences from each site and manually label them. For Super User site, the acucracy is 88%, and the accuracy for Unix & Linux is 84%. In regards of the accuracy of clustering, our model also outperforms the other baselines. Finally, similar to what we did in Section 6.4, we decrease the test samples from 150 to 50 for each site. The accuracy for Super User is 0.78. For Unix & Linux, the accuracy is 0.76. ???What do you mean by this? The detailed results of our experiments can be seen in our website[9].

Note that the performance of our model in this two datasets are not as good as that of Stack Overflow due to several reasons. First, the mistake could come from the very first step while we find similar technology pairs. For example, the pair *Linux* and *Ubuntu* are not used for comparing, *Ubuntu* is a subset of *Linux*, but they are often mentioned in the same sentence, which happens a lot in Unix & Linux. Second, it seems that users in the two sites are more likely to use the technology pairs side by side to compare with the third technology like "*Make sure with shortcuts etc that **opera firefox** is easier to start find than internet explorer*". These also explain the decrease of accuracy in summarize the overall opinion, for there are more neutral sentences which are harder to predict. Despite of all the decrease in accuracy, our model still has a relatively high accuracy and performs the best out of all the baselines. These show that given a corpus of domain-specific data, our approach can still be used to distill similar technologies. Actually these two reasons also occur much in the Stack Overflow, why will they degrade the performance?

7. ??
8. ??
9. ??

7 Related Works

7.1 Mining similar software artefacts

Finding similar software artefacts can help developers migrate from one tool to the other which is more suitable to their requirement. But it is a challenging task to identify similar software artefacts from the existing large pool of candidates. Therefore, much research effort has been put into this domain. Different methods has been adopted to mine similar artefacts ranging from high-level software [37], [38], mobile applications [39], [40], github projects [41] to low-level third-party libraries [12], [42], [43], APIs [44], [45], code snippets [46], or Q&A questions [47]. Compared with these research studies, the mined software technologies in this work has much broader scope including not only software-specific artefacts, but also general software concepts (e.g., algorithm, protocol), tools (e.g., IDE).

7.2 Extracting opinions about software technologies

Extraction opinions about technology preference is also important to developers, as the opinions from other developers may provide a general vision especially for novice developers on the technology selection for their own tasks. Uddin and Khomh [48], [49] extract API opinion sentences in different aspects to show developers' sentiment to that API. Ahasanuzzaman et.al [50] classify StackOverflow posts concerning API issues-only with Conditional Random Field (CRF) [51]. Lin et.al [52] produced a pattern-based approach that can identify overall quality, pros, and cons of APIs. Different from the works mentioned above for only extracting opinion about one certain API or library, our method tries to extract the comparison opinions of two similar technologies.

7.3 Comparison in Software Engineering

Given a list of similar technologies, developers may further compare and contrast them for the final selection. Some researcher investigate such comparison, the comparison is highly domain-specific such as software for traffic simulation [53], regression models [54], x86 virtualization [55], etc. Michail and Notkin [56] assess different third-party libraries by matching similar components (such as classes and functions) across similar libraries. But it can only work for library comparison without the possibility to be extended to other higher/lower-level technologies in Software Engineering. Instead, we find developers's preference of certain software technologies highly depends on other developers' usage experience and report of similar technology comparisons. Li et al. [57] adopt NLP methods to distill comparative user review about similar mobile Apps. De et al [58], [59] compare the software libraries by a set of metrics. Different from their works, we adopt a light-weight approach by extracting a large pool of comparable technologies and corresponding explicit comparison in natural-language sentences, rather than heavy-weight program analysis. In addition, apart from extracting comparative sentences, we further organize them into different clusters and represent each cluster with some keywords to help developers understand technology comparison more easily.

Finally, it is worth mentioning some related practical projects. SimilarWeb [60] is a website that provides both users engagement statistics and similar competitors for websites and mobile applications. AlternativeTo [61] is a social software recommendation website in which users can find alternatives to a given software based on user recommendations. SimilarTech [62] is a site to recommend analogical third-party libraries across different programming languages. These websites can help users find similar or alternative websites or software applications without detailed comparison.

8 Conclusion and Future Work

In this paper, we present an automatic approach to distill and aggregate comparative opinions of comparable technologies from Q&A websites. We first obtain a large pool of comparable technologies by incorporating categorical knowledge into word embedding of tags in Stack Overflow, and then locate comparative sentences about these technologies by Coreference and POS-tag based pattern matching, organize comparative sentences into clusters for easier understanding, and finally summarize comparative sentences to obtain a general opinion for each comparable technologies. The evaluation shows that our system covers a large set of comparable technologies and their corresponding comparative sentences with high accuracy. We implement our method not only on Stack Overflow, but also on Super User and Unix and Linux shows that our method has a general usage on most of the Q&A sites. We also demonstrate the potential of our system to answer questions about comparing comparable technologies, because the technology comparison knowledge mined using our system largely overlap with the original answers in Stack Overflow.

Maybe mention on using Bert in clustering or compare clusters of comparative techs instead of two techs only. Find relations between questions and link similar answers to certain questions Apart from comparative sentences explicitly mentioning both comparable technologies, some comparative opinions may hide 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. In addition, we also plan to summarize higher-level opinions or preferences from separated individual comparative sentences for easier understanding.

References

[1] C. Chen and Z. Xing, "Mining technology landscape from stack overflow," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, 2016, p. 14.

[2] C. Chen, Z. Xing, and L. Han, "Techland: Assisting technology landscape inquiries with insights from stack overflow," in *Software Maintenance and Evolution (ICSME), 2016 IEEE International Conference on*. IEEE, 2016, pp. 356–366.

[3] L. Bao, J. Li, Z. Xing, X. Wang, X. Xia, and B. Zhou, "Extracting and analyzing time-series hci data from screen-captured task videos," *Empirical Software Engineering*, vol. 22, no. 1, pp. 134–174, 2017.

[4] "Millions of queries per second: Postgresql and mysql's peaceful battle at today's demanding workloads," https://goo.gl/RXVjkB/, 2017, accessed: 2018-04-05.

[5] "Mysql vs postgres," https://www.upguard.com/articles/postgres-vs-mysql/, 2017, accessed: 2018-04-05.

[6] C. Chen and Z. Xing, "Towards correlating search on google and asking on stack overflow," in *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, vol. 1. IEEE, 2016, pp. 83–92.

[7] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *International Conference on Machine Learning*, 2015, pp. 957–966.

[8] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[10] Y. Huang, C. Chen, Z. Xing, T. Lin, and Y. Liu, "Tell them apart: distilling technology differences from crowd-scale comparison discussions." in *ASE*, 2018, pp. 214–224.

[11] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? an analysis of topics and trends in stack overflow," *Empirical Software Engineering*, vol. 19, no. 3, pp. 619–654, 2014.

[12] C. Chen, S. Gao, and Z. Xing, "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, 2016, pp. 338–348.

[13] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web?: Nier track," in *Software Engineering (ICSE), 2011 33rd International Conference on*. IEEE, 2011, pp. 804–807.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[16] J. Kazama and K. Torisawa, "Exploiting wikipedia as external knowledge for named entity recognition," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007, pp. 698–707.

[17] S. Bird and E. Loper, "Nltk: the natural language toolkit," in *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*. Association for Computational Linguistics, 2004, p. 31.

[18] "Alphabetical list of part-of-speech tags used in the penn treebank project," https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html, 2003, accessed: 2018-02-02.

[19] D. Ye, Z. Xing, C. Y. Foo, Z. Q. Ang, J. Li, and N. Kapre, "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, 2016, pp. 90–101.

[20] K. Clark and C. D. Manning, "Deep reinforcement learning for mention-ranking coreference models," *arXiv preprint arXiv:1609.08667*, 2016.

[21] X. Chen, C. Chen, D. Zhang, and Z. Xing, "Sethesaurus: Wordnet in software engineering," *IEEE Transactions on Software Engineering*, 2019.

[22] H. Ling and K. Okada, "An efficient earth mover's distance algorithm for robust histogram comparison," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 5, pp. 840–853, 2007.

[23] O. Pele and M. Werman, "Fast and robust earth mover's distances," in *Computer vision, 2009 IEEE 12th international conference on*. IEEE, 2009, pp. 460–467.

[24] "Open sourcing bert: State-of-the-art pre-training for natural language processing," https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html, 2018, accessed: 2019-09-12.

[25] "Bidirectional encoder representations from transformers (bert)," https://tfhub.dev/tensorflow/bert_en_wwm_uncased_L-24_H-1024_A-16/1, 2019, accessed: 2019-11-15.

[26] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *arXiv preprint arXiv:1506.06724*, 2015.

[27] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[28] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[29] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, 2014, pp. 1188–1196.

[30] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[31] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *Journal of Machine Learning Research*, vol. 11, no. Oct, pp. 2837–2854, 2010.

[32] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure," in *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007.

[33] E. B. Fowlkes and C. L. Mallows, "A method for comparing two hierarchical clusterings," *Journal of the American statistical association*, vol. 78, no. 383, pp. 553–569, 1983.

[34] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep 1995. [Online]. Available: https://doi.org/10.1023/A:1022627411411

[35] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "Fasttext.zip: Compressing text classification models," *arXiv preprint arXiv:1612.03651*, 2016.

[36] "sklearn.metrics.precision_recall_fscore_support," https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html, 2019, accessed: 2019-11-02.

[37] C. McMillan, M. Grechanik, and D. Poshyvanyk, "Detecting similar software applications," pp. 364–374, 2012.

[38] F. Thung, D. Lo, and L. Jiang, "Detecting similar applications with collaborative tagging," in *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*. IEEE, 2012, pp. 600–603.

[39] N. Chen, S. C. Hoi, S. Li, and X. Xiao, "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, 2015, pp. 305–314.

[40] M. Linares-Vásquez, A. Holtzhauer, and D. Poshyvanyk, "On automatically detecting similar android apps," in *Program Comprehension (ICPC), 2016 IEEE 24th International Conference on*. IEEE, 2016, pp. 1–10.

[41] Y. Zhang, D. Lo, P. S. Kochhar, X. Xia, Q. Li, and J. Sun, "Detecting similar repositories on github," in *Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference on*. IEEE, 2017, pp. 13–23.

[42] C. Teyton, J.-R. Falleri, and X. Blanc, "Automatic discovery of function mappings between similar libraries," in *Reverse Engineering (WCRE), 2013 20th Working Conference on*. IEEE, 2013, pp. 192–201.

[43] C. Chen and Z. Xing, "Similartech: automatically recommend analogical libraries across different programming languages," in *Automated Software Engineering (ASE), 2016 31st IEEE/ACM International Conference on*. IEEE, 2016, pp. 834–839.

[44] X. Gu, H. Zhang, D. Zhang, and S. Kim, "Deepam: Migrate apis with multi-modal sequence to sequence learning," *arXiv preprint arXiv:1704.07734*, 2017.

[45] T. D. Nguyen, A. T. Nguyen, H. D. Phan, and T. N. Nguyen, "Exploring api embedding for api usages and applications," in *Software Engineering (ICSE), 2017 IEEE/ACM 39th International Conference on*. IEEE, 2017, pp. 438–449.

[46] F.-H. Su, J. Bell, G. Kaiser, and S. Sethumadhavan, "Identifying functionally similar code in complex codebases," in *Program Comprehension (ICPC), 2016 IEEE 24th International Conference on*. IEEE, 2016, pp. 1–10.

[47] G. Chen, C. Chen, Z. Xing, and B. Xu, "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, 2016, pp. 744–755.

[48] G. Uddin and F. Khomh, "Opiner: an opinion search and summarization engine for apis," in *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press, 2017, pp. 978–983.

[49] ——, "Automatic summarization of api reviews," in *Automated Software Engineering (ASE), 2017 32nd IEEE/ACM International Conference on*. IEEE, 2017, pp. 159–170.

[50] M. Ahasanuzzaman, M. Asaduzzaman, C. K. Roy, and K. A. Schneider, "Caps: a supervised technique for classifying stack overflow posts concerning api issues," *Empirical Software Engineering*, pp. 1–40, 2019.

[51] C. Sutton, A. McCallum *et al.*, "An introduction to conditional random fields," *Foundations and Trends® in Machine Learning*, vol. 4, no. 4, pp. 267–373, 2012.

[52] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, and M. Lanza, "Pattern-based mining of opinions in q&a websites," in *Proceedings of the 41st International Conference on Software Engineering*. IEEE Press, 2019, pp. 548–559.

[53] S. L. Jones, A. J. Sullivan, N. Cheekoti, M. D. Anderson, and D. Malave, "Traffic simulation software comparison study," *UTCA report*, vol. 2217, 2004.

[54] N. J. Horton and S. R. Lipsitz, "Multiple imputation in practice: comparison of software packages for regression models with missing variables," *The American Statistician*, vol. 55, no. 3, pp. 244–254, 2001.

[55] K. Adams and O. Agesen, "A comparison of software and hardware techniques for x86 virtualization," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 5, pp. 2–13, 2006.

[56] A. Michail and D. Notkin, "Assessing software libraries by browsing similar classes, functions and relationships," in *Proceedings of the 21st international conference on Software engineering*. ACM, 1999, pp. 463–472.

[57] Y. Li, B. Jia, Y. Guo, and X. Chen, "Mining user reviews for mobile app comparisons," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 3, p. 75, 2017.

[58] F. L. de la Mora and S. Nadi, "Which library should i use?: a metric-based comparison of software libraries," in *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*. ACM, 2018, pp. 37–40.

[59] ——, "An empirical study of metric-based comparisons of software libraries," in *Proceedings of the 14th International Conference on Predictive Models and Data Analytics in Software Engineering*. ACM, 2018, pp. 22–31.

[60] "Similarweb," https://www.similarweb.com/, 2018, accessed: 2018-04-05.

[61] "Alternativeto - crowdsourced software recommendations," https://alternativeto.net/, 2018, accessed: 2018-04-05.

[62] "Similartech: Find alternative libraries across languages," https://graphofknowledge.appspot.com/similartech/, 2018, accessed: 2018-04-05.