Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Discovering semantically related technical terms and web resources in Q&A discussions[*]

Junfang JIA[1], Valeriia TUMANIAN[2], Guoqiang LI[†‡2]

[1]*School of Computer and Network Engineering, Shanxi Datong University, Datong 037009, China*
[2]*School of Software, Shanghai Jiao Tong University, Shanghai 200240, China*
[†]E-mail: li.g@sjtu.edu.cn

Received Apr. 21, 2020; Revision accepted Dec. 23, 2020; Crosschecked June 16, 2021

Abstract: A sheer number of techniques and web resources are available for software engineering practice and this number continues to grow. Discovering semantically similar or related technical terms and web resources offers the opportunity to design appealing services to facilitate information retrieval and information discovery. In this study, we extract technical terms and web resources from a community of question and answer (Q&A) discussions and propose an approach based on a neural language model to learn the semantic representations of technical terms and web resources in a joint low-dimensional vector space. Our approach maps technical terms and web resources to a semantic vector space based only on the surrounding technical terms and web resources of a technical term (or web resource) in a discussion thread, without the need for mining the text content of the discussion. We apply our approach to Stack Overflow data dump of March 2018. Through both quantitative and qualitative analyses in the clustering, search, and semantic reasoning tasks, we show that the learnt technical-term and web-resource vector representations can capture the semantic relatedness of technical terms and web resources, and they can be exploited to support various search and semantic reasoning tasks, by means of simple K-nearest neighbor search and simple algebraic operations on the learnt vector representations in the embedding space.

Key words: Technical terms; Web resources; Word embedding; Q&A web site; Clustering tasks; Recommendation tasks
https://doi.org/10.1631/FITEE.2000186 **CLC number:** TP311

1 Introduction

In software engineering, developers often use a set of related techniques and seek information from relevant web resources. Table 1 shows six clusters of semantically related technical terms and web resources. Discovering such semantically related technical terms and web resources offers the opportunity to design appealing services to facilitate information retrieval and discovery.

For example, we can cluster forum discussions of a set of related techniques and web resources to study the needs and thoughts of developers or to compile the frequently asked questions (FAQs) of common technical issues. Furthermore, we can develop recommendation systems for serendipitous discovery of information. For example, when users search for the natural language processing (NLP) techniques, we can recommend NLP libraries (e.g., NLTK) or related concepts (e.g., text mining) for query reformulation, or recommend web sites specializing in NLP (e.g., http://alias-i.com/lingpipe). As another example, when users visit the web site http://d3js.org, we can recommend similar web sites (e.g., http://highcharts.com) or suggest related

[‡] Corresponding author

[*] Project supported by the National Natural Science Foundation of China (No. 61872232)

ORCID: Junfang JIA, https://orcid.org/0000-0002-3451-8487; Valeriia TUMANIAN, https://orcid.org/0000-0001-6651-191X; Guoqiang LI, https://orcid.org/0000-0001-9005-7112

Table 1 Examples of three categories of clusters

Category	Cluster ID	Technical term	Web resource
Concept-centric	Cluster 98	Oop	http://en.wikipedia.org/wiki/singleton_pattern
		Class	http://en.wikipedia.org/wiki/single_responsibility_principle
		Design-patterns	http://en.wikipedia.org/wiki/dependency_injection
		Design	http://en.wikipedia.org/wiki/observer_pattern
		Inheritance	http://en.wikipedia.org/wiki/don%27t_repeat_yourself
	Cluster 56	Regex	http://www.regular-expressions.info/lookaround.html
		String	http://gskinner.com/regexr
		Replace	http://www.regular-expressions.info/charclass.html
		Split	http://docs.python.org/library/re.html
		Preg-match	http://www.regular-expressions.info/brackets.html
Technique-centric	Cluster 59	Django-models	https://docs.djangoproject.com/en/dev/howto/static-files
		Django-templates	https://docs.djangoproject.com/en/dev/howto/custom-template-tags
		Django-forms	https://docs.djangoproject.com/en/dev/topics/forms/modelforms
		Django-admin	https://docs.djangoproject.com/en/dev/topics/db/queries/#complex-lookups-with-q-objects
		Django-views	https://docs.djangoproject.com/en/dev/topics/signals
	Cluster 81	PHP	http://www.smarty.net
		Codeigniter	http://framework.zend.com
		Cakephp	http://www.doctrine-project.org
		Magento	http://codeigniter.com
		Zend-framework	http://cakephp.org
Task-centric	Cluster 130	Automation	http://phantomjs.org
		Selenium-webdriver	http://htmlunit.sourceforge.net
		Webdriver	http://msdn.microsoft.com/en-us/library/system.windows.forms.webbrowser.aspx
		Jmeter	http://docs.seleniumhq.org
		Phantomjs	https://github.com/jnicklas/capybara
	Cluster 146	Matlab	http://numpy.scipy.org
		Numpy	http://www.numpy.org
		Matrix	http://en.wikipedia.org/wiki/matlab
		Scipy	http://eigen.tuxfamily.org/index.php?title=main_page
		Fortran	http://www.gnu.org/software/gsl

libraries that the users may not be aware of (e.g., Dimplejs). Or when users search Java and D3.js, we can infer that users need some popular visualization library for Java equivalent to the D3.js library for JavaScript, and thus recommend libraries such as JFreeChart. Thus, our recommendations can become more satisfiable for users.

To support these information retrieval and discovery tasks, we need to represent technical terms and web resources in a semantic space in order to reason their semantic relatedness. In this study, we present a neural-language-model-based approach to learn the semantic representations of technical terms and web resources in a joint vector space. We choose programming-related question and answer (Q&A) web sites as our information source, from which we extract technical terms and web resources and learn their vector representations from millions of discussion threads.

Programming-related Q&A web sites, such as Stack Overflow, have become a tremendous knowledge repository for software engineering professionals. In Q&A discussions, web users tag their questions with main technical terms around which the questions revolve (Sillito et al., 2012), and share hyperlinks to useful online programming resources. LinkLive (Li et al., 2019) uses multiple features, including hyperlink co-occurrences in Q&A discussions. Several studies have used topic modeling methods such as latent Dirichlet allocation (LDA) (Blei et al., 2003) to discover the overarching topics in Q&A discussions. These studies analyzed the discussion content as a whole but not the specific technical terms or web resources in the discussions. Search-based methods help users find relevant information using similar attributes or content. However, they cannot discover semantically related but heterogeneous technical terms and web resources that are not similar in attributes or content. Correlation-similarity-based analyses, e.g., based on association rules (Agrawal et al., 1993), can discover correlated items that co-occur frequently, but many semantically related technical terms and web resources, such as graph-database and non-relational-database, may not be frequently mentioned together in the same discussion threads.

Different from keyword-based search methods and correlation-similarity-based methods, our approach is inspired by the recent success of neural language models in NLP applications (Mikolov et al., 2013a, 2013b; Chen et al., 2014; Xie et al., 2016, 2017; Gummidi et al., 2019). In these NLP applications, neural language models are able to learn word representations (or word embeddings) in a low-dimensional continuous vector space using the surrounding context of the word in a sentence, where semantically similar or related words are close to each other in the resulting embedding space.

Our objective is taking advantage of this property for the discovery of semantically related technical terms and web resources. Given a corpus of discussion threads, we extract question tags as technical terms and hyperlinks in Q&A discussions as web resources to construct a corpus of pseudo-documents. Then, we adopt the continuous skip-gram model (Mikolov et al., 2013a) to learn vector representations of technical terms and web resources in a low-dimensional space where semantically related technical terms and web resources are close in the resulting vector space.

Our approach reduces the tasks of clustering or recommending semantically related technical terms and web resources to a trivial K-nearest neighbor search in the embedding space. As a result and in contrast to existing search-based or correlation-similarity-based methods, related technical terms and web resources could have a high similarity score even if they do not have similar content or are not mentioned together. Furthermore, the learnt vector representations can be exploited to provide intuitive results for semantic reasoning tasks, for example, finding analogical libraries like JavaScript's D3.js and Java's JFreeChart.

We evaluate our approach using the Stack Overflow data dump of March 2018. To evaluate whether the vector representations learnt by our approach can capture the semantic relatedness of technical terms and web resources, we conduct three studies: clustering, search, and semantic reasoning tasks. Our studies show that the learnt vector representations of technical terms and web resources are in accordance with the semantic meanings of technical terms and web resources, and these vector representations have a wide potential for numerous recommendation and reasoning tasks. The recommendations based on the learnt technical-term and web-resource vectors can complement existing keyword-based search methods and correlation-similarity-based recommendations.

This paper makes the following contributions:

1. We formulate the discovery of semantically related technologies and web resources in Q&A discussions as an NLP word embedding task.

2. We adopt the cutting-edge word embedding technique to learn the semantic representations of technical terms and web resources in a joint vector space.

3. We evaluate quantitatively and qualitatively the vector representations of technical terms and web resources in clustering and recommendation tasks.

2 Related works

As a fast growing body of software engineering knowledge, Stack Overflow has attracted much research attention. Our previous work (Jia et al., 2020) did a quantitative study on multi-lingual sites of Stack Overflow, showing three major problems of establishing such multi-lingual versions. Another one (Jia and Li, 2021) focused on the natural ordering of tags in domain-specific Q&A sites.

Several studies (Wang et al., 2013; Barua et al., 2014; Rosen and Shihab, 2015) used topic modeling methods (such as LDA) to discover the overarching topics in Q&A discussions. For example, Barua et al. (2014) used LDA to discover the topics in developer discussions on Stack Overflow and studied the relations between topics and their trends. Wang et al. (2013) reported exploratory analyses of mobile development issues, which were discussed on Stack Overflow. They also used LDA to analyze Q&A

discussions. Topic models can extract the main concerns from the discussion content, but they are coarse-grained analyses and, unlike our work, do not deal with the semantics of specific technical terms or web resources. For example, to find specific techniques related to a topic, Barua et al. (2014) had to manually link the question tags to the topics. In contrast, our approach automatically maps technical terms and web resources to a semantic space based on a neural language model.

Traditional search-based methods model the descriptions of technical terms and content of web sites to help users retrieve relevant information using keywords. Xia et al. (2017) showed the difficulties that developers from all over the world face when searching for the needed information. Xu BW et al. (2017) also focused on the relevance of question retrieval, while Ren et al. (2019) posed an issue on filtering of the retrieved relevant information. Search-based methods cannot help users find new relevant and interesting items (Robillard et al., 2010), because recommendations are based on some similar attributes or contents. However, semantically related technical terms and web resources may not be described in similar content, and such heterogeneous information cannot be discovered using search-based methods. For example, Openxml is an extensible markup language (XML) based file format for representing Microsoft Office documents. Apache POI is a Java library for reading and writing Microsoft Office documents. Clearly, the two techniques are semantically related. However, Google search engine cannot find the other technique using one technique as a query. In contrast, our approach can help users find such semantically related technical terms and web resources as they would be close to each other in a learnt vector space.

Correlation similarity has often been used to discover correlated words that are frequently mentioned together (Tian et al., 2014b). For example, Tian et al. (2014a) proposed a similarity metric based on weighted co-occurrence to measure the similarity of words in the textual contents of posts on Stack Overflow. They developed a software-specific WordNet for finding similar words. Yang and Tan (2014) detected semantically related words from the source code and code comments in source code files using clustering and a co-occurrence-based method. Wang et al. (2012) inferred semantically related tags from FreeCode based on tag co-occurrence. Moreover, Huang et al. (2018) calculated the similarity score between two text descriptions on Stack Overflow using the word embedding technique. Our work is more general than that of Wang et al. (2012), as we consider both tags and uniform resource locators (URLs) in the Q&A discussions, and learn vector representations of tags and URLs in the same semantic space. Furthermore, existing methods infer related words from the direct co-occurrence of the words in the same documents. In contrast, our approach can discover semantically related technical terms and web resources that appear in the similar discussion context, but may not be mentioned together in the same discussion, for example, analogy libraries NLTK for Python versus OpenNLP for Java.

3 Word embedding techniques

3.1 General background

Word embeddings are low-dimensional vector representations of words which are built on the assumption that words with similar meanings tend to appear in similar contexts (Harris, 1954). Learnt vectors are able to encode rich semantic and syntactic regularities (Harris, 1954). Compared with one hot vector representation whose dimensionality is usually as large as the size of vocabulary (i.e., making no assumption about word similarity), word embedding techniques are capable of relieving the curse of dimensionality and improving the generalization of word representations. Word embedding has been widely used in several NLP tasks, such as part-of-speech tagging (Collobert et al., 2011), named entity recognition (Passos et al., 2014), and dependency parsing (Bansal et al., 2014).

Typically, word embedding techniques fall into two categories: count-based models and context-predicting models (Baroni et al., 2014). Count-based models, for example, as proposed in Bullinaria and Levy (2012), first calculate the word-document co-occurrence matrix and then use dimension reduction techniques such as singular value decomposition (SVD) to map the words to a lower-dimensional space. Context-predicting models, which are usually facilitated by neural networks, are trained by predicting the surrounding contexts of a given word.

Owing to their simplicity, context-predicting models are amazingly efficient in training with large corpora that contain billions of words. Our dataset consists of millions of discussion threads, which mention tens of thousands of technical terms and web resources millions of times. Therefore, in our work, we decide to use the continuous skip-gram model (Mikolov et al., 2013a), one of the most popular context-predicting models.

3.2 Continuous skip-gram model

The skip-gram model is a type of neural language model. Its basic idea is to learn a mapping from a central word to a continuous vector, which is then used to predict the surrounding words in the context window. The central word and context word are called the input and output layers of the neural network, respectively. Specifically, the objective function of the skip-gram model is defined to maximize the sum of log probabilities of surrounding context words conditioned on the central word:

$$\sum_{i=1}^{N} \sum_{-k\leq j\leq k, j\neq 0} \log p\left(w_{i+j}|w_i\right),$$

where w_i denotes the central word in a sliding window of k words, w_{i+j} denotes the context word surrounding w_i within the sliding window, and N denotes the length of the word sequence. Furthermore, $\log p\left(w_{i+j}|w_i\right)$ is the conditional probability defined using the softmax function:

$$p\left(w_{i+j}|w_i\right) = \frac{\exp((\boldsymbol{v}'_{w_{i+j}})^{\mathrm{T}} \cdot \boldsymbol{v}_{w_i})}{\sum_{w\in W} \exp((\boldsymbol{v}'_{w})^{\mathrm{T}} \cdot \boldsymbol{v}_{w_i})},$$

where $\boldsymbol{v}_{w_i}$ and $\boldsymbol{v}'_{w_{i+j}}$ are the vector representations, also known as the embeddings, of the central word w_i and the context word w_{i+j}, respectively, and W is the vocabulary of all words. Here, $\exp((\boldsymbol{v}'_{w_{i+j}})^{\mathrm{T}} \cdot \boldsymbol{v}_{w_i})$ is the exponent of the dot product of the context word vector and central word vector. Intuitively, $p\left(w_{i+j}|w_i\right)$ estimates the probability of a word appearing in a context of a central word by its normalized dot product with the central word. After the training process, we can obtain vector representations of central words and context words, respectively.

Note that, based on the definition of $p(w_{i+j}|w_i)$, calculating the derivatives of the objective function with regard to $\boldsymbol{v}'_{w_{i+j}}$ and $\boldsymbol{v}_{w_i}$ during the training process requires iterating over the whole vocabulary for each word w_i in the sequence. Such learning then becomes a very time-consuming process for a large vocabulary. Since people are usually interested only in learning meaningful word representations but not the accuracy of prediction, it is acceptable to speed up the training process with an approximation of the original model. With this aim, Mikolov et al. (2013a) used negative sampling, a special case of "contrastive estimation" (Gutmann and Hyvärinen, 2012), as an alternative objective function to the original softmax function:

$$\sum_{i=1}^{N} \sum_{\substack{-k\leq j\leq k\\ j\neq 0}} \Bigg(\log \sigma\Big((\boldsymbol{v}'_{w_{i+j}})^{\mathrm{T}} \cdot \boldsymbol{v}_{w_i} \Big) + \sum_{w'\in N(K)} \log \sigma\Big(-(\boldsymbol{v}'_{w'})^{\mathrm{T}} \cdot \boldsymbol{v}_{w_i} \Big) \Bigg), \tag{1}$$

where $N(k)$ is a sampling function that randomly samples K words from a unigram distribution as the negative training instances. $\sigma(x)$ is the sigmoid function:

$$\sigma(x) = \frac{1}{1+\exp(-x)}.$$

For each word-context pair (w_i, w_{i+j}), the negative sampling process produces N random negative samples (w_i, w'). Instead of iterating over the whole vocabulary, the learning process now needs only to go through the set of negative examples.

As suggested by Levy and Goldberg (2014c), a skip-gram model in fact implicitly factors the word-context co-occurrence matrix. This co-occurrence information is necessary in analyzing the relatedness between words, and it can be easily extended to incorporate heterogeneous input and output data. For example, Levy and Goldberg (2014a) used dependency links in place of context words in the output layer so that the resulting word embeddings encoded more syntactic information. We are therefore inspired to adopt the continuous skip-gram model to analyze the tags and URLs of discussion threads, which turns out to be an effective method for discovering software-specific semantically related technical terms and web resources.

4 Approach

The goal of this work is to propose a neural-language-model-based framework for discovering semantically similar technical terms and web resources from community Q&A discussions such as Stack Overflow discussion threads. The whole framework is composed of three modules: (1) pseudo-document generation from the community Q&A discussions; (2) customized skip-gram model for technical-term and web-resource embedding learning; (3) similarity computation.

4.1 Pseudo-document generation

In this study, we consider a question and all its answers as a discussion thread. Directly analyzing the text content of a discussion thread is a difficult and complicated task because it requires the understanding of not only the natural language but also programming language used to compose the code embedded in a thread. On the other hand, each discussion thread has a set of tags, edited by a question author, which captures the main technologies or constructs around which the question revolves (Sillito et al., 2012). Furthermore, salient online programming resources have been frequently referenced in many discussion threads. Therefore, analyzing the tags and URLs involved in the discussion threads could help discover semantically related technical terms and web resources.

Given a discussion thread, we generate a pseudo-document that consists of a set of technical terms and web resources. We extract the question tags as technical terms around which the discussion thread revolves. In this study, we use Stack Overflow as a test bed. As a Stack Overflow question can have 1–5 tags, a pseudo-document can have 1–5 technical terms. We extract the hyperlinks referenced in the discussion thread as web resources. A pseudo-document can have zero or more web resources. In this study, we would like to learn vector representations of technical terms and web resources in a joint vector space. Therefore, we consider only discussion threads that contain both tags and URLs.

4.2 Customized skip-gram model

Given a corpus of pseudo-documents of technical terms and web resources, we aim to learn low-dimensional vector representations of technical terms and web resources in a joint vector space using word embedding techniques.

The skip-gram model, as discussed in Section 3, uses the co-occurrence information about a central word (i.e., input layer) and the context words (i.e., output layer) in a sentence to train a neural network. To learn the technical-term and web-resource embeddings, we extend the skip-gram model to take as input the technical terms and web resources in a pseudo-document. Accordingly, the output layer of the skip-gram model is also set to predict the technical terms and web resources in the pseudo-document.

Unlike word sequence in a sentence or a document, technical terms and web resources in a pseudo-document are not constrained by a syntactic structure, and the orders of technical terms and web resources are random. Therefore, in our application of the skip-gram model, instead of using a sliding window to constrain the selection of context, we define the context of a given input (technical term or web resource) as all other technical terms and web resources in the pseudo-document. Fig. 1 shows the architecture of the customized skip-gram model for technical-term and web-resource embeddings. The key idea of our model is to use each technical term and web resource in the pseudo-document to predict all other technical terms and web resources in the same document.

A naive customization of the original skip-gram model is to treat technical terms and web resources equally as words in the document and set the size of the sliding window to the length of the document. However, since technical terms and web resources are heterogeneous, it makes no sense to assume that technical terms and web resources should share the same distribution. Therefore, we consider two types of contexts in the output layer: tag and URL context. Considering the two types of inputs, tag or URL, we have four types of input-output predictions in our customized skip-gram model: the input tag predicts tag and URL contexts (Fig. 1a), and the input URL predicts tag and URL context (Fig. 1b).

Therefore, the objective function of our skip-gram model is

$$\sum_{i=1}^{M}\left[\sum_{t\in T_{D_i}}\left(\sum_{\mathrm{ct}\in T_{D_i}\setminus\{t\}}\log p(\mathrm{ct}|t)+\sum_{\mathrm{cu}\in U_{D_i}}\log p(\mathrm{cu}|t)\right)\right.$$
$$\left.+\sum_{u\in U_{D_i}}\left(\sum_{\mathrm{ct}\in T_{D_i}}\log p(\mathrm{ct}|u)+\sum_{\mathrm{cu}\in U_{D_i}\setminus\{u\}}\log p(\mathrm{cu}|u)\right)\right],$$

where M is the number of pseudo-documents, D_i is the i^{th} pseudo-document, T_{D_i} is a set of technical

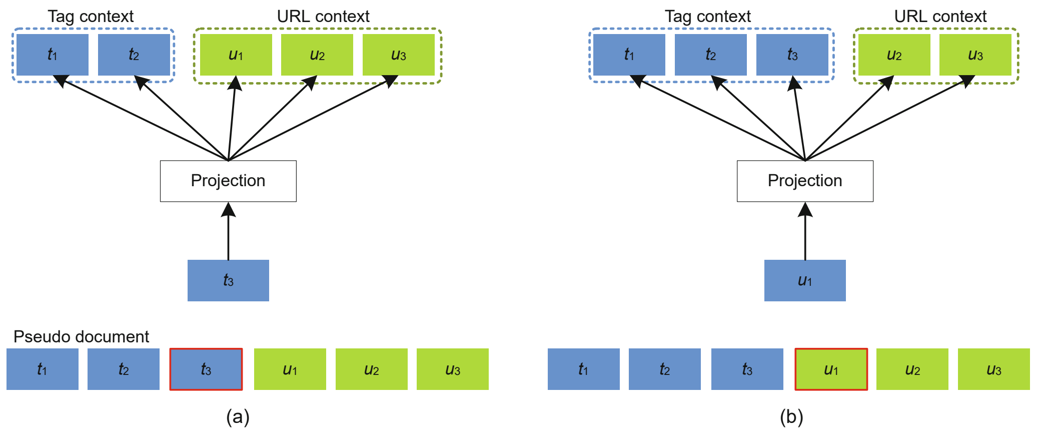

Fig. 1 Customized skip-gram model for technical-term and web-resource embedding: (a) input tag predicts tag and URL contexts; (b) input URL predicts tag and URL contexts

terms (i.e., tags) contained in D_i, and U_{D_i} is a set of web resources (i.e., URLs) contained in D_i. Here, ct and cu are the tag context and URL tag context of a given tag t or URL u.

We then define the conditional probability for the four types of input-output predictions independently. For example, we define $p(\text{cu}|u)$ (i.e., the conditional probability for URL-URL prediction) as

$$p\left(\text{cu}|u\right) = \frac{\exp((\boldsymbol{v}'_{\text{cu}})^{\text{T}} \cdot \boldsymbol{v}_u)}{\sum_{\text{cu}' \in W_u} \exp((\boldsymbol{v}'_{\text{cu}'})^{\text{T}} \cdot \boldsymbol{v}_u)},$$

where W_u is the URL vocabulary. Note that this definition is almost identical to the definition used by the original softmax function (see Section 3), except that this definition is for URLs. The conditional probability for the other three types of input-output prediction can be defined in the same way. For learning the parameters in the model, we apply the same negative sampling approach used in Mikolov et al. (2013a). As we have separated the URL and tag contexts, we generate negative instances by sampling only from the corresponding vocabulary during negative sampling.

4.3 Similarity computation

The resulting technical-term and web-resource embeddings learnt from the pseudo-documents map the technical terms and web resources to the same low-dimensional vector space, where their similarity can be simply measured as the cosine similarity between their vectors:

$$\text{Sim}(u,t) = \cos(\boldsymbol{v}_u, \boldsymbol{v}_t) = \frac{\boldsymbol{v}_u \cdot \boldsymbol{v}_t}{\|\boldsymbol{v}_u\|\|\boldsymbol{v}_t\|},$$

where t and u represent a tag (technical term) and a URL (web resource), respectively. The same measure can be applied to measure the similarity between tag-tag and URL-URL pairs. In the resulting embedding space, semantically similar or related technical terms and web resources would be close to each other (i.e., having high cosine similarity).

4.4 Web site embedding

We use simple weighted sum to generate web site embedding $\boldsymbol{v}_D$ based on the learnt web-resource vectors:

$$\boldsymbol{v}_D = \frac{\sum_{u \in D} f_u \boldsymbol{v}_u}{\sum_{u \in D} f_u},$$

where D is a set of web resources from the same web site, f_u is a reference frequency of a particular web resource $u \in D$, and $\boldsymbol{v}_u$ is a learnt vector representation of u. In our empirical study, we use web site embeddings to recommend web sites given a technical term or a web resource in search tasks.

5 Empirical study

To evaluate the vector representations of technical terms and web resources, we conduct three studies: clustering, search, and semantic reasoning tasks. In this section, we first give an overview of the three studies, describe the dataset that we use for the studies, and finally report the results and further analyses of the three tasks.

5.1 Study overview

5.1.1 Clustering task

Our skip-gram model learns vector representations of technical terms and web resources by predicting the surrounding tag and URL context of a technical term or web resource in a pseudo-document, without considering any content information of technical terms and web resources. Our first research question is "can the technical-term and web-resource embeddings learnt using this simple neural network capture semantic regularities of the related technical terms and web resources?" To answer this question, we cluster technical terms and web resources using K-means clustering (MacQueen, 1967) on the learnt vector representations. We manually inspect the semantic relatedness of technical terms and web resources in the resulting clusters, and identify three categories of clusters: concept-, technique-, and task-centric clusters. Using the description of technical terms from TagWiki and the titles of web pages, we quantitatively analyze the semantic relatedness of technical terms and web resources within and across the clusters using the intra- and inter-cluster content similarity. We also comparatively study the clusters obtained using K-means on the learnt technical-term and web-resource embeddings with the clusters obtained using the LDA method (Blei et al., 2003) on technical-term descriptions and web page titles.

5.1.2 Search task

Our second research question is "can the learnt technical-term and web-resource embeddings support the recommendation of semantically related technical terms and web resources?" To answer this question, we design four search tasks that could be useful for various online applications: (1) Given a technical term, search for similar or related technical terms (useful in query reformulation); (2) Given a technical term, search for relevant web resources (useful in web resource retrieval); (3) Given a web resource, search for relevant technical terms (useful in semantic tagging); (4) Given a web resource, search for similar or related web resources (useful in web resource recommendation).

Our approach reduces these search tasks to a simple K-nearest neighbor search in the joint vector space between technical terms and web resources. We report representative examples for each search task. To understand whether the recommended technical terms (or web resources) could also be retrieved using association rule mining (Agrawal et al., 1993), we quantitatively analyze the co-occurrence frequency of a given technical term (or web resource) and the recommended technical terms and/or web resources.

5.1.3 Semantic reasoning task

It has been shown that simple algebraic operations on the word vectors can produce intuitive results in semantic reasoning tasks (Mikolov et al., 2013a; Levy and Goldberg, 2014b; Mitra, 2015), such as vector(chicago) + vector(newspaper) $\approx$ vector(chicago sumtimes) and vector(king) $-$ vector(man) + vector(woman) $\approx$ vector(queen). Inspired by these studies, we conduct an exploratory study of two types of semantic reasoning tasks using the learnt vector representations of technical terms and web resources: (1) semantic addition—given two technical terms, term_1 and term_2, find technical terms (or web resources) whose vector representations are similar to the vector $\text{vector}(\text{term}_1) + \text{vector}(\text{term}_2)$; (2) analogical reasoning—given three technical terms, term_1, term_2, and term_3, find technical terms (or web resources) whose vector representations are similar to the vector $\text{vector}(\text{term}_1) - \text{vector}(\text{term}_2)+\text{vector}(\text{term}_3)$; (3) given a web resource URL and two technical terms, term_1 and term_2, find technical terms (or web resources) whose vector representations are similar to the vector $\text{vector}(\text{URL}) - \text{vector}(\text{term}_1) + \text{vector}(\text{term}_2)$.

5.2 Dataset and model training

We apply our approach to the Stack Overflow data dump which archives Q&A discussions from August 2009 to March 2018. We generate a corpus of pseudo-documents of technical terms and web resources as described in Section 4. Because the goal of this work is to learn technical-term and web-resource embeddings together, we remove the pseudo-documents that contain only tags or URLs. After this operation, we obtain a corpus of 3 724 259 pseudo-documents that contain both tags and URLs. In total, the corpus of pseudo-documents contains 40 993 unique tags (technical terms) and 3 883 171 unique URLs (web resources).

Similar to the rare-word treatment in many NLP applications (Levy et al., 2015), we remove rare technical terms and web resources (in this work, tags are used less than 20 times and URLs are referenced less than 20 times) from the pseudo-documents. After removing the rare technical terms and web resources, we obtain 812 035 non-empty pseudo-documents, which contain 16 598 unique technical terms and 35 124 unique web resources as our vocabulary.

We conduct experiments with the dimension of technical-term and web-resource embeddings and the count of random negative samples. We empirically set the embedding dimension to 400 and the count of random negative samples to 15 for the balance of efficiency and performance. After model training, we obtain the vector representations for each technical term and web resource in our vocabulary. All empirical studies below are based on these vectors.

5.3 Clustering task

We use the K-means clustering method (MacQueen, 1967) to cluster technical terms and web resources in our vocabulary based on the cosine similarity of the learnt vector representations. In this study, we set $K = 200$ and obtain 200 clusters. These 200 clusters have 258.61 ± 125.67 (mean $\pm$ standard deviation (SD)) technical terms and web resources. We evaluate the semantic relatedness of technical terms and web resources in the resulting clusters by qualitative manual inspection and quantitative analyses of intra- and inter-cluster content similarity.

5.3.1 Qualitative observation of clusters

We manually inspect the 200 clusters and a few observations are made. First, we observe three general categories of technical terms and web resources: concept-, technique-, and task-centric. Table 1 presents two examples for each category.

Second, by reading the TagWiki of technical terms, whereby each WikiWord is transformed automatically into a link, and visiting the web pages, we observe that a small number of clusters contain technical terms and web resources with homogeneous content (e.g., cluster 98 about object-oriented design and cluster 56 about regular expression), but most of the clusters contain technical terms and web resources with more heterogeneous content (e.g., cluster 59 about Django features, cluster 81 about hypertext preprocessor (PHP) web development frameworks, cluster 130 about web automation techniques, and cluster 146 about numerical and scientific computing).

Third, we observe different web-resource dissemination patterns. For concept- and technique-centric clusters, there are often some authoritative web resources that Stack Overflow users frequently reference, for example, Wikipedia pages for some software engineering concepts (cluster 98), web sites dedicated to some concepts (cluster 56), and official documentations on some techniques (cluster 59). For task- and technique-centric clusters (e.g., clusters 81, 130, and 146), the clusters often contain a diverse set of techniques and web resources.

Remark 1 Technical terms and web resources can be well clustered around concepts, techniques, and tasks. Some clusters contain homogeneous information, but most of the clusters contain heterogeneous information.

5.3.2 Intra- versus inter-cluster similarity

To confirm our qualitative observations, we quantitatively compare the similarity between intra- and inter-cluster contents, which is a commonly used metric to evaluate the clustering results (Rand, 1971). The well-clustered items should have low inter-cluster similarity but high intra-cluster similarity.

In this study, we crawl the web page title as the web-resource content, and the first sentence of the TagWiki of the corresponding tags as the technical-term content. The title of a web page and the first sentence of a tag's TagWiki usually provide a concise summary of the corresponding web resource and technical term. We manage to crawl the titles of 30 004 web resources and the TagWiki definitions of 14 554 technical terms in our vocabulary. We use the crawled content of these technical terms and web resources to compute the intra- and inter-cluster content similarity. We consider the crawled content of each technical term and web resource as a text document. After removing stop words (e.g., is and a), each document is represented as a term frequency-inverse document frequency (TF-IDF) vector using the standard text processing method. We compute the cosine similarity of the TF-IDF vectors of the two documents to measure the content similarity of the corresponding technical terms and web resources.

Considering the size of the clusters (258.61 ± 125.67 technical terms and web resources), computing the content similarity of all the pairs of technical terms and web resources within and across the clusters would be extremely time-consuming. Therefore, we adopt a random sampling approach. Given a cluster, we randomly select 40 items (technical terms and web resources) in the cluster and generate 780 pairs of different items for the cluster. We calculate the average cosine similarity of these 780 pairs of items as the intra-cluster similarity of the given cluster. For the two clusters that have fewer than 40 items, we generate pairs from all their items. To compute inter-cluster similarity, we randomly select 100 items from all other clusters and generate 4000 pairs of different items. We calculate the average cosine similarity of these 4000 pairs of item as the inter-cluster similarity of the given cluster.

Fig. 2 shows the comparison between the intra- and inter-cluster similarities of the 200 clusters. The x axis shows the rank of the clusters, sorted by descending order of their intra-cluster similarities. The y axis denotes the similarity metric. We can see that for all clusters, the inter-cluster similarity is very low (close to zero). For all clusters except a small number of the lowest-ranked clusters, the intra-cluster similarity is consistently and significantly (3–80 times) higher than that of the average inter-cluster similarity with other clusters. Only 40 clusters have intra-cluster content similarity larger than 0.1. These clusters usually contain homogeneous information from one or two authoritative web sites. For example, cluster 135 contains jquery-related technical terms. Almost all the web resources in this cluster are from http://api.jquery.com, and the titles of these web pages are all very similar. As such, the intra-cluster similarities of these clusters are high. In contrast, 152 clusters have intra-cluster content similarity between 0.015 and 0.1. These clusters contain more heterogeneous technical terms and web resources. This quantitative result is consistent with our qualitative observations.

We manually inspect the eight clusters whose intra-cluster similarity is smaller than 0.015, which is equivalent to the highest average inter-cluster similarity for all clusters. We observe two reasons for the low intra-cluster similarity. First, four clusters contain rare technical terms and web resources that have not been frequently used. For example, the average usage frequency of the tags and URLs in cluster 15 is only 28. The tags and URLs in cluster 15, for example, Syntax-Error, LiveCode, kdb, and http://php.net/heredoc, have very low content similarity. Earlier research (Qiu et al., 2014) showed that word embedding techniques perform poorly on rare words. Second, four clusters contain actually semantically related technical terms and web resources. For example, cluster 132 contains editor-related technical terms and web resources, such as xhtml, tinymce, and http://ckeditor.com. Due to the diversity of these technical terms and web resources, their content similarities are very low.

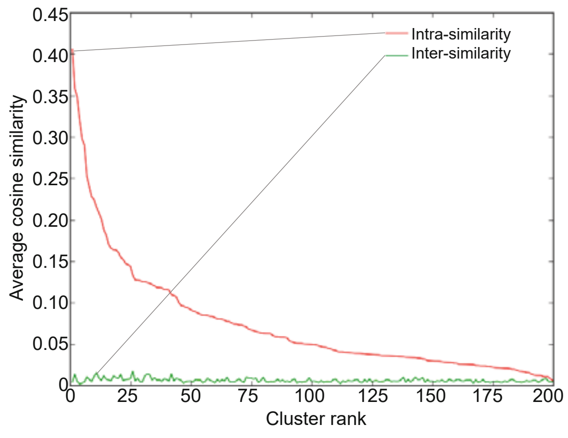

Fig. 2 Intra- and inter-cluster similarities: embedding-based *K*-means

Remark 2 The learnt vector representations can capture the semantic relatedness of the corresponding technical terms and web resources.

5.3.3 Comparison with LDA topics

We show that the learnt technical-term and web-resourced embeddings can produce high-quality clusters around software concepts, techniques, and tasks. A related question is "can content-based methods like topic models produce clusters comparable to clusters obtained using technical-term and web-resourced embeddings?" To study the differences between the content- and embedding-based clusters, we compare our embedding-based clusters with the topics discovered using the LDA method (Blei et al., 2003). In this experiment, we consider only technical terms, because technical-term definitions from TagWiki are usually complete natural language sentences with a certain level of details. In contrast, the web page titles are usually short and incomplete phrases. Furthermore, web page titles are usually inconsistent across different web sites, while TagWiki is clean and uniform, which makes technical-term definitions suitable for the comparative study with the LDA method.

Recall that we have 14 554 technical terms with the TagWiki definition in our vocabulary. We use K-means clustering to cluster these 14 554 technical terms into 100 clusters. We apply the LDA implemented in the Gensim tool (https://radimrehurek.com/gensim/) to the corpus of the 14 554 technical-term definitions, following the standard text mining process (stop word removal, tokenization, and stemming). We mine 100 topics and assign each technical-term to a topic in which the technical-term definition has a large probability, similar to Wang et al. (2013).

For the two sets of 100 clusters obtained using our embedding-based K-means clustering and the LDA method, respectively, we compute the intra- and inter-cluster similarity as in the previous subsection. Fig. 3 shows the comparison of the intra- and inter-cluster similarity of the technical-term clusters obtained using two different methods. The clusters are sorted by descending order of the intra-cluster similarity. We can see that the clusters obtained using our technical-term and web-resource embeddings have higher intra-cluster similarity than those obtained using the LDA method, except for the 10 lowest-ranked clusters.

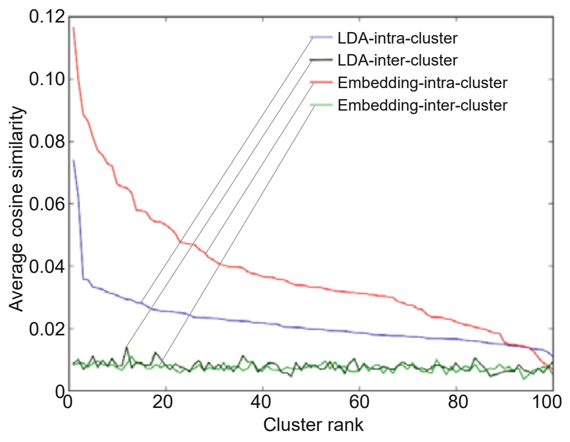

Fig. 3 Intra- and inter-cluster similarities: LDA versus embedding-based K-means

The results contradict the intuition in that the LDA method mines the topics from the document content, and intuitively it should produce document clusters with high intra-cluster content similarity. One plausible explanation is that the TagWiki definitions of the technical terms are still too short to train a high-quality LDA model. Earlier research in Hong and Davison (2010) found that topic models, such as LDA, perform poorly on short text documents. Certainly, it is possible to use the LDA method to obtain high-quality clusters by crawling more TagWiki content, but preparing a large corpus of high-quality text documents is a tedious and time-consuming task. We also manually examine the 10 lowest-ranked clusters obtained using our approach. We observe the same two reasons for the low intra-cluster similarity: clusters contain rare technical terms, and clusters contain semantically related but dissimilar-content technical terms.

Remark 3 Our technical-term and web-resource embeddings provide an alternative to the traditional topic models to obtain high-quality clusters of semantically related technical terms and web resources, without crawling and processing of a large volume of textual content, which is usually tedious and time-consuming.

5.4 Search task

We have shown that the learnt technical-term and web-resource embeddings capture the semantic relatedness of technical terms and web resources. In this subsection, we demonstrate that the learnt vector representations can support a wide range of search tasks by the simple K-nearest neighbor search in the learnt embedding space. We also perform quantitative co-occurrence analyses of the search results to learn the relationship between the technical-term and web-resource co-occurrence and semantic relatedness.

5.4.1 Four types of search tasks

As we have two types of information (technical-term and web-resource), we design four types of search tasks that could be useful for various online applications, denoted as term $->$? term, term $->$? URL, URL $->$? term, and URL $->$? URL. The left-hand item represents the query input, and the right-hand item represents the query result. Table 2 shows example queries of technical terms (or web resources) from the vocabulary, together with their top-5 nearest technical-term or web-resource neighbors in the embedding space. The technical terms and URLs in a gray background are the query inputs, while those in a white background are the query results.

1. Keyword suggestion

An important task in many online applications is finding similar or related words given an input word as a query. For example, when users browse questions of a specific tag on Stack Overflow, Stack Overflow suggests a list of related tags on the right side of the page. For the two examples of term − > ? term search, meaningful semantic relationships can be observed within the closest distance of the input technical-term. For the technical term NLP, the model finds NLP tool NLTK, related concepts such as text-mining and text-analysis, and concepts related to NLP in general, such as machine-learning and linguistics.

2. Semantic tagging

Automatic web-resource tagging can assign a list of relevant technical terms that explain the web-resource content. This is very useful in improving web-resource recommendation. Our model is suitable for such tasks because technical terms and web resources reside in the same vector space. Using the learnt embeddings, we can easily retrieve the nearest technical terms given a URL as an input. In Table 2, we can see that the retrieved technical terms often summarize and further explain the web resources. For example, for the URL http://d3js.org (the website of a popular JavaScript library for data visualization), the tags include the general concepts, graph- and data-visualization, force-layout algorithm, and two other visualization tools, gephi and flot.

3. Locating specialty web resources

Given a technical term, one may be interested in finding the most relevant web resources, which is a typical task of a search engine. Unlike the comprehensive list returned by a search engine, our model can find several most semantically related web resources in which

Table 2 Examples of search tasks

Term	− > ?	Term
loops		nlp
for-loop		nltk
while-loop		text-mining
nested-loops		linguistics
continue		text-analysis
break		machine-learning
URL	− > ?	Term
http://d3js.org		http://www.google.com/analytics
graph-visualization		web-analytics
data-visualization		analytics
flot		web-analytics-tools
gephi		ab-testing
force-layout		universal-analytics
Term	− > ?	URL
visualization		youtube
http://www.graphviz.org		https://developers.google.com/youtube/v3
http://gephi.org		https://developers.google.com/youtube/v3/docs/videos/list
http://www.gnuplot.info		https://developers.google.com/youtube/v3/docs/channels/list
http://thejit.org		https://developers.google.com/youtube/iframe_api_reference
http://highcharts.com		https://developers.google.com/youtube/player_parameters
URL	− > ?	URL
http://wordpress.org		http://en.wikipedia.org/wiki/utf-8
http://www.opensourcecms.com		http://en.wikipedia.org/wiki/ISO/IEC_8859-1
http://www.concrete5.org		http://en.wikipedia.org/wiki/utf-16
http://www.cushycms.com		http://en.wikipedia.org/wiki/unicode
http://www.cmsmatrix.org		http://en.wikipedia.org/wiki/numeric_character_reference
http://expressionengine.com		http://en.wikipedia.org/wiki/diacritic

developers are most interested. As shown in Table 2, for YouTube, our model does not return https://www.youtube.com/, but the YouTube data application programming interface (API) for Google developers. Furthermore, our model can find the web pages that do not necessarily contain the queried technical term. For example, our model returns http://www.highcharts.com/ for visualization, despite the fact that the web page does not mention word visualization.

4. Finding similar web resources

One may be interested in finding related web resources similar to the ones you are reading. In Table 2, we can see that given the Wikipedia 8-bit unicode transformation format (UTF-8) page, relevant and semantically related Wikipedia pages on character coding are located close in the vector space. Interestingly, in some cases the recommended web resources may not be similar to the queried URL, but they are still semantically related. For example, for the URL http://wordpress.org, the recommended web site http://www.cmsmatrix.org is not really a content management product, but it is a web site for comparing the features in more than 1200 content management system products.

5.4.2 Co-occurrence analysis of search results

To understand the relationship between co-occurrence and the semantic relatedness of technical terms and web resources, we randomly sample 1000 technical terms and web resources in our vocabulary. For each sampled technical term or web resource, we collect the top-10 technical terms or web resources in the relevant types of search tasks. For a pair of query item and search result, we examine the co-occurrence frequency of the pair in the Stack Overflow discussion threads. The statistics show that about 71% of the pairs of query items and search results never co-occur in the discussion threads, for example, graph-databases and non-relational-database, profile and profiles, and facebook-login and http://developers.facebook.com/docs/authentication.

About 24% of the pairs co-occur fewer than 10 times, for example, storage and data-storage, and python-requests and https://github.com/kennethreitz/requests. Fewer than 1% of the pairs co-occur more than 100 times in the discussion threads, for example, licensing and gpl, and facebook-graph-api and https://developers.facebook.com/apps. For those pairs that never co-occur or co-occur only a few times, the relationship between the two items may not be discovered using frequent itemset mining or association rule mining. However, our skip-gram model does not rely on the direct co-occurrence of the two items. Instead, it learns the embeddings from the co-occurrence of a technical term or web resource and the surrounding technical terms and web resources in the pseudo-document. Although a pair of items, such as fix-width and margin, never directly co-occur in the discussion threads, as long as they frequently appear in the similar context, the learnt embeddings can still be close in the vector space. As such, items that do not frequently co-occur with the query item can be recommended. Specifically, for the technical terms and web resources with low usage frequency, their semantic relatedness can be bridged by their context, while association rule mining methods cannot discover such relations.

For frequently co-occurring technical terms and web resources, their associations can be discovered. However, recommendations based on the learnt technical-term and web-resource embeddings can be very different from those based on associations. For example, Table 3 shows five examples of tags recommended using our learnt embeddings and related tags collected from the Stack Overflow web site. Stack Overflow recommends related tags based on the direct co-occurrence between frequent tags. Using our method, the tags are ordered by their cosine similarities with the query term. The tags by Stack Overflow are related to the tags that are ordered by their co-occurrence frequency with the query tag. We can see that Stack Overflow related tags are dominated by the most frequently used tags, such as programming languages and platforms. In contrast, our recommendations are more semantically related, comparable to the query term. Furthermore, the most related techniques are ranked higher using the range of the two techniques in the embedding space, rather than using the co-occurrence frequency, for example, gmail and imap, or hadoop and hdfs.

Remark 4 The learnt technical-term and web-resource embeddings can complement existing keyword-based and co-occurrence-based search systems in numerous online applications.

5.5 Semantic reasoning task

In this subsection, we report an exploratory study using simple algebraic operations on the learnt technical-term and web-resource embeddings in two types of semantic reasoning tasks.

5.5.1 Semantic addition

Developers are often interested in something related to two or more techniques. To answer such queries, we need to add the semantics of the two techniques together. The learnt technical-term embeddings can be exploited to solve the semantic addition questions by finding the technical terms and web resources whose embeddings are the most similar to the addition of the technical-term vectors. Table 4 shows four examples of semantic addition queries. Using simple vector addition, our model can recommend highly related technical terms and web resources for the two given technical terms.

5.5.2 Analogical reasoning

It often happens that developers need to find some analogical libraries that can provide features comparable to the libraries with which they are already familiar, for example, the NLP libraries for Java to the NLTK library for Python, and the Office software for Linux to the Microsoft Office for Windows. The traditional search systems do not support such analogical queries. Taking advantage of

Table 3 Embedding-based recommendation versus Stack Overflow related tags

Query tag	Cordova	Http	Algebra	Gmail	Hadoop
Tags using our method	Phonegap-plugins	Httprequest	Polynomial-math	Imap	MapReduce
	Appcelerator	Http-status-codes	Quadratic	Hotmail	Hdfs
	Phonegap-build	Http-headers	Symbolic-math	Gmail-imap	Hive
	Sencha-touch-2	Content-length	Calculus	Email-integration	Hbase
	Jquery-mobile	Http-request	Discrete-mathematics	Pop3	Cloudera
Stack Overflow related tags	Android	Java	Math	Email	MapReduce
	JavaScript	PHP	Algorithm	PHP	Java
	IoS	Android	Python	Smtp	Hive
	Jquery	JavaScript	Java	Android	Hdfs
	Jquery	Post	C++	Imap	Apache-pig

Table 4 Examples of semantic addition

Semantic algebraic operation	Technical term	Web resource
IoS + facebook	Facebook-ios-sdk	https://github.com/facebook/facebook-ios-sdk
	Sharekit	https://dev.twitter.com/docs/ios
	Fbconnect	http://getsharekit.com
	Mgtwitterengine	https://github.com/sharekit/sharekit
	Three20	http://www.getsharekit.com
Python + svm	Scikits	http://pybrain.org
	Scikit-learn	http://scikit-learn.org
	Training-data	http://orange.biolab.si
	Pybrain	http://www.heatonresearch.com
	Cross-validation	http://www.ailab.si
Symfony + orm	Doctrine	http://www.orm-designer.com
	Doctrine2	http://propel.phpdb.org
	Propel	http://www.phpandstuff.com
	Symfony-2.1	http://docs.doctrine-project.org
	Doctrine-1.2	http://www.propelorm.org
D3.js + scientific − computing	Networkx	http://mathgl.sf.net
	Graph-visualization	http://www.vtk.org
	Gephi	http://networkx.lanl.gov
	Ogr	http://www.rforge.net
	Curve-fitting	http://matplotlib.org

the learnt technical-term and web-resource embeddings, we can solve an analogy question "a is to A as ? is to B." The unknown word "?" can be inferred from the words whose embedding is most similar to the vector $a - A + B$.

Table 5 shows two examples of such analogy questions. Using simple algebraic operations URL − Python + Java, our model can recommend code convention related terms and web resources for Java, similar to the web page https://www.python.org/dev/peps/pep-0008/, which is a style guide for Python code. For the query "Microsoft Office is to Windows as ? is to Linux," our model recommends Office software for Linux (like openoffice and liboffice) and relevant web sites. Meanwhile, we observe some software and web resources related to Microsoft Office but not for Linux, such as Openxml, powerpoint, and http://poi.apache.org. We also observe some unrelated web resources in this example (http://www.artofsolving.com and dag.wieers.com, two personal blogs). Such unrelated web resources (or technical terms) could be filtered out by incorporating domain knowledge into a word embedding (Xu C et al., 2014) or by further considering the discussion context in which they appear and the content of the web resources (or technical terms) in the neural network as proposed in Grbovic et al. (2015).

6 Conclusions and future work

In this study, we present a neural-language-model-based approach to learn semantic representations of technical terms and web resources extracted from community Q&A discussions. Different from existing approaches that either mine textual content of discussions or rely on direct co-occurrence of technical terms and web resources, the underlying assumption of our approach is that semantically similar or related technical terms and web resources would be presented in similar technical-term and web-resource contexts.

Our evaluation of the large-scale Stack Overflow data dump shows that the learnt technical-term and web-resource vector representations work surprisingly well for clustering semantically related technical terms and web resources, even when the technical terms and web resources are not similar in content. Furthermore, the learnt vector representations have a good potential in reducing complex search and semantic reasoning tasks to simple K-nearest neighbor search and simple algebraic operations in the technical-term and web-resource embedding space.

In the future, we are interested in studying web-resource dissemination patterns for different categories of technical-term and web-resource clusters. We will incorporate context (e.g., anchor text of hyperlinks) and content (e.g., definition of technical terms) information into the technical-term and web-resource embeddings. We will also develop semantic search systems that exploit the learnt embeddings for recommending semantically related technical terms and web resources in tasks such as query reformulation and direct answers.

Table 5 Examples of analogical reasoning

Semantic algebraic operation	Technical term	Web resource
http://www.python.org/dev/peps/pep-0008 − python + java	Annotation-processing	http://www.oracle.com/technetwork/java/codeconventions-135099.html#367
	Checkstyle	http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html
	Apache-commons-lang	http://www.oracle.com/technetwork/java/javase/documentation/codeconventions-135099.html#367
	Anonymous-class	http://code.google.com/p/javadude/wiki/annotations
	Xtend	http://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html
Microsoft Office − windows + linux	Openxml	http://www.artofsolving.com
	Powerpoint	http://poi.apache.org
	Openxml-sdk	http://wiki.services.openoffice.org
	Openoffice.org	http://dag.wieers.com
	Libreoffice	http://api.openoffice.org

Contributors

Guoqiang LI designed the research. Junfang JIA and Valeriia TUMANIAN processed the data and programmed the system. Guoqiang LI drafted the manuscript. Junfang JIA and Valeriia TUMANIAN helped organize the manuscript. Guoqiang LI and Valeriia TUMANIAN revised and finalized the paper.

Compliance with ethics guidelines

Junfang JIA, Valeriia TUMANIAN, and Guoqiang LI declare that they have no conflict of interest.

References

Agrawal R, Imieliński T, Swami A, 1993. Mining association rules between sets of items in large databases. *ACM SIGMOD Rec*, 22(2):207-216. https://doi.org/10.1145/170036.170072

Bansal M, Gimpel K, Livescu K, 2014. Tailoring continuous word representations for dependency parsing. Proc 52nd Annual Meeting of the Association for Computational Linguistics, p.809-815. https://doi.org/10.3115/v1/P14-2131

Baroni M, Dinu G, Kruszewski G, 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. Proc 52nd Annual Meeting of the Association for Computational Linguistics, p.238-247. https://doi.org/10.3115/v1/P14-1023

Barua A, Thomas SW, Hassan AE, 2014. What are developers talking about? An analysis of topics and trends in Stack Overflow. *Empir Softw Eng*, 19(3):619-654. https://doi.org/10.1007/s10664-012-9231-y

Blei DM, Ng AY, Jordan MI, 2003. Latent Dirichlet allocation. *J Mach Learn Res*, 3(4-5):993-1022.

Bullinaria JA, Levy JP, 2012. Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and SVD. *Behav Res Methods*, 44(3):890-907. https://doi.org/10.3758/s13428-011-0183-8

Chen WL, Zhang Y, Zhang M, 2014. Feature embedding for dependency parsing. Proc 25th Int Conf on Computational Linguistics, p.816-826.

Collobert R, Weston J, Bottou L, et al., 2011. Natural language processing (almost) from scratch. *J Mach Learn Res*, 12:2493-2537.

Grbovic M, Djuric N, Radosavljevic V, et al., 2015. Context- and content-aware embeddings for query rewriting in sponsored search. Proc 38th Int ACM SIGIR Conf on Research and Development in Information Retrieval, p.383-392. https://doi.org/10.1145/2766462.2767709

Gummidi SRB, Xie XK, Pedersen TB, 2019. A survey of spatial crowdsourcing. *ACM Trans Database Syst*, 44(2):8. https://doi.org/10.1145/3291933

Gutmann MU, Hyvärinen A, 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J Mach Learn Res*, 13(1):307-361.

Harris ZS, 1954. Distributional structure. *Word*, 10:146-162.

Hong LJ, Davison BD, 2010. Empirical study of topic modeling in Twitter. Proc 1st Workshop on Social Media Analytics, p.80-88. https://doi.org/10.1145/1964858.1964870

Huang Q, Xia X, Xing ZC, et al., 2018. API method recommendation without worrying about the task-API knowledge gap. Proc 33rd ACM/IEEE Int Conf on Automated Software Engineering, p.293-304. https://doi.org/10.1145/3238147.3238191

Jia JF, Li GQ, 2021. Learning natural ordering of tags in domain-specific Q&A sites. *Front Inform Technol Electron Eng*, 22(2):170-184. https://doi.org/10.1631/FITEE.1900645

Jia JF, Tumanian V, Li GQ, 2020. In favour of or against multi-lingual Q&A sites? Exploring the evidence from user and knowledge perspectives. *Behav Inform Technol*, p.1-16. https://doi.org/10.1080/0144929X.2020.1752308

Levy O, Goldberg Y, 2014a. Dependency-based word embeddings. Proc 52nd Annual Meeting of the Association for Computational Linguistics, p.302-308. https://doi.org/10.3115/v1/P14-2050

Levy O, Goldberg Y, 2014b. Linguistic regularities in sparse and explicit word representations. Proc 18th Conf on Computational Natural Language Learning, p.171-180. https://doi.org/10.3115/v1/W14-1618

Levy O, Goldberg Y, 2014c. Neural word embedding as implicit matrix factorization. Proc 27th Int Conf on Neural Information Processing Systems, p.2177-2185.

Levy O, Goldberg Y, Dagan I, 2015. Improving distributional similarity with lessons learned from word embeddings. *Trans Assoc Comput Ling*, 3:211-225. https://doi.org/10.1162/tacl_a_00134

Li J, Xing ZC, Sun AX, 2019. LinkLive: discovering web learning resources for developers from Q&A discussions. *World Wide Web*, 22(4):1699-1725. https://doi.org/10.1007/s11280-018-0621-y

MacQueen J, 1967. Some methods for classification and analysis of multivariate observations. Proc 5th Berkeley Symp on Mathematical Statistics and Probability, p.281-297.

Mikolov T, Sutskever I, Chen K, et al., 2013a. Distributed representations of words and phrases and their compositionality. Proc 26th Int Conf on Neural Information Processing Systems, p.3111-3119.

Mikolov T, Chen K, Corrado G, et al., 2013b. Efficient estimation of word representations in vector space. https://arxiv.org/abs/1301.3781

Mitra B, 2015. Exploring session context using distributed representations of queries and reformulations. Proc 38th Int ACM SIGIR Conf on Research and Development in Information Retrieval, p.3-12. https://doi.org/10.1145/2766462.2767702

Passos A, Kumar V, McCallum A, 2014. Lexicon infused phrase embeddings for named entity resolution. https://arxiv.org/abs/1404.5367

Qiu SY, Cui Q, Bian J, et al., 2014. Co-learning of word representations and morpheme representations. Proc 25th Int Conf on Computational Linguistics, p.141-150.

Rand WM, 1971. Objective criteria for the evaluation of clustering methods. *J Am Stat Assoc*, 66(336):846-850.

Ren XX, Xing ZC, Xia X, et al., 2019. Discovering, explaining and summarizing controversial discussions in community Q&A sites. Proc 34th IEEE/ACM Int Conf on Automated Software Engineering, p.151-162. https://doi.org/10.1109/ASE.2019.00024

Robillard M, Walker R, Zimmermann T, 2010. Recommendation systems for software engineering. *IEEE Softw*, 27(4):80-86. https://doi.org/10.1109/MS.2009.161

Rosen C, Shihab E, 2015. What are mobile developers asking about? A large scale study using Stack OverFlow. *Empir Softw Eng*, 21(3):1192-1223. https://doi.org/10.1007/s10664-015-9379-3

Sillito J, Maurer F, Nasehi SM, et al., 2012. What makes a good code example?: a study of programming Q&A in StackOverflow. Proc IEEE Int Conf on Software Maintenance, p.25-34. https://doi.org/10.1109/ICSM.2012.6405249

Tian Y, Lo D, Lawall J, 2014a. Automated construction of a software-specific word similarity database. Proc Software Evolution Week-IEEE Conf on Software Maintenance, Reengineering, and Reverse Engineering, p.44-53. https://doi.org/10.1109/CSMR-WCRE.2014.6747213

Tian Y, Lo D, Lawall J, 2014b. SEWordSim: software-specific word similarity database. Companion Proc 36th Int Conf on Software Engineering, p.568-571. https://doi.org/10.1145/2591062.2591071

Wang SW, Lo D, Jiang LX, 2012. Inferring semantically related software terms and their taxonomy by leveraging collaborative tagging. Proc 28th IEEE Int Conf on Software Maintenance, p.604-607. https://doi.org/10.1109/ICSM.2012.6405332

Wang SW, Lo D, Jiang LX, 2013. An empirical study on developer interactions in Stack Overflow. Proc 28th Annual ACM Symp on Applied Computing, p.1019-1024. https://doi.org/10.1145/2480362.2480557

Xia X, Bao LF, Lo D, et al., 2017. What do developers search for on the web? *Empir Softw Eng*, 22(6):3149-3185. https://doi.org/10.1007/s10664-017-9514-4

Xie XK, Jin P, Yiu ML, et al., 2016. Enabling scalable geographic service sharing with weighted imprecise Voronoi cells. *IEEE Trans Knowl Data Eng*, 28(2):439-453. https://doi.org/10.1109/TKDE.2015.2464804

Xie XK, Lin X, Xu JL, et al., 2017. Reverse keyword-based location search. Proc IEEE 33rd Int Conf on Data Engineering, p.375-386. https://doi.org/10.1109/ICDE.2017.96

Xu BW, Xing ZC, Xia X, et al., 2017. AnswerBot: automated generation of answer summary to developers' technical questions. Proc 32nd IEEE/ACM Int Conf on Automated Software Engineering, p.706-716. https://doi.org/10.1109/ASE.2017.8115681

Xu C, Bai YL, Bian J, et al., 2014. RC-NET: a general framework for incorporating knowledge into word representations. Proc 23rd ACM Int Conf on Information and Knowledge Management, p.1219-1228. https://doi.org/10.1145/2661829.2662038

Yang JQ, Tan L, 2014. SWordNet: inferring semantically related words from software context. *Empir Softw Eng*, 19(6):1856-1886. https://doi.org/10.1007/s10664-013-9264-x