World Wide Web (2019) 22:1699–1725
https://doi.org/10.1007/s11280-018-0621-y

# LinkLive: discovering Web learning resources for developers from Q&A discussions

**Jing Li[1] · Zhenchang Xing[2] · Aixin Sun[1]**

Received: 13 February 2017 / Revised: 27 June 2018 / Accepted: 28 June 2018 /
Published online: 23 July 2018

**Abstract** Software developers need access to correlated information (e.g., API documentation, Wikipedia pages, Stack Overflow questions and answers) which are often dispersed among different Web resources. This paper is concerned with the situation where a developer is visiting a Web page, but at the same time is willing to explore correlated Web resources to extend his/her knowledge or to satisfy his/her curiosity. Specifically, we present an item-based collaborative filtering technique, named LinkLive, for automatically recommending a list of correlated Web resources for a particular Web page. The recommendation is done by exploiting hyperlink associations from the crowdsourced knowledge on Stack Overflow. We motivate our research using an exploratory study of hyperlink dissemination patterns on Stack Overflow. We then present our LinkLive technique that uses multiple features, including hyperlink co-occurrences in Q&A discussions, locations (e.g., question, answer, or comment) in which hyperlinks are referenced, and votes for posts/comments in which hyperlinks are referenced. Experiments using 7 years of Stack Overflow data show that, our technique recommends correlated Web resources with promising accuracy in an open setting. A user study of 6 participants suggests that practitioners find the recommended Web resources useful for Web discovery.

**Keywords** API documentation · Recommendation systems · Web discovery · Social media

✉ Jing Li
jli030@e.ntu.edu.sg

Zhenchang Xing
zhenchang.xing@anu.edu.au

Aixin Sun
axsun@ntu.edu.sg

[1] School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore

[2] College of Engineering and Computer Science, Australian National University, Canberra, Australia

## 1 Introduction

Software developers perceive online programming resources as the "key information resource" in their learning and work [13]. The ability to search, understand, and use online programming resources is one of the key abilities affecting software developers' efficiency and success [75]. In general, Web users' information needs can be categorized at a high level as informational, navigational, and transactional [16]. In informational search, there are many situations in which users know the topics they are looking for, and are willing to explore correlated Web resources to extend their knowledge or to satisfy their curiosity [36, 50, 84].

For example, when visiting a Web page about Singleton design pattern, a developer may be willing to explore correlated Web resources, such as other design patterns (e.g., Abstract Factory pattern), concepts related to Singleton pattern (e.g., double-checked locking and enum-based singleton), or Singleton pattern implementations. As another example, consider a developer who visits the JUnit website. The developer may appreciate recommendations of correlated Web resources. Examples include alternatives to JUnit like TestNG, mocking framework for unit testing in Java like EasyMock, testing frameworks for Web development like Selenium, or code analysis tools like PMD.

Search-based methods often cannot help developers discover *correlated and new* Web resources [19], because search engines generally employ keyword matching or rely on certain content similarity of Web resources. Correlated Web resources, however, may not have similar content. For example, commercial search engines cannot return the four commonly-used visualization tools at once: D3,[1] Gephi,[2] Raphael[3] and Highchart.[4] The four webpages share only very few general concepts, i.e., "*library*" and "*visualization*". Another example is about software design patterns: Singleton pattern,[5] Abstract Factory pattern,[6] and Double-checked locking.[7] Furthermore, when developers have no or little knowledge about the correlated information that they may be interested in, it is difficult for them to formulate an effective search query. Particularly, for a developer who just starts learning design patterns, it is unlikely that he/she knows Double-checked locking is a concept related to Singleton pattern. Thus, an automatic technique that recommends correlated Web resources when developers visit a particular Web page can greatly assist them in discovering correlated Web resources. Our study addresses this particular need by exploiting the crowdsourced knowledge on Stack Overflow.

The idea of recommending correlated Web resources for a particular Web page is related to recommendation systems for e-commerce websites. In particular, item-based collaborative filtering approaches attempt to summarize product browsing or transaction history to recommend more products that are related to a particular product [34, 43, 65, 82]. Applied to Stack Overflow, the idea is to exploit the crowdsourced knowledge on the correlation of Web resources that are highly recognized by the community.

[1] https://d3js.org/

[2] https://gephi.org/

[3] http://dmitrybaranovskiy.github.io/raphael/

[4] https://www.highcharts.com/

[5] https://en.wikipedia.org/wiki/Singleton_pattern

[6] https://en.wikipedia.org/wiki/Abstract_factory_pattern

[7] https://en.wikipedia.org/wiki/Double-checked_locking

| | |
|---|---|
| Question Title | Singleton Pattern Interview |
| Question Body | I am recently asked about java related question in an interview with following code, since I am very new to java and barely code in Java so I really have no idea what the following code does. |
| Answer 1 | This is a Singleton Pattern.<br>Here's an example of Lazy Initialization, thread-safe singleton pattern from Wikipedia:<br>...<br>Setting the instance variable to volatile tells Java to read it from memory and to not set it in cache.<br>Synchronized statements or methods help with concurrency. Read more about double checked locking which is what happens for a "lazy initialization" singleton. |
| Comment 1 | double-checked locking |
| Answer 2 | Interviewer basically wants to check your knoweldge of Singleton pattern . Can the pattern be broken?. Ans is Yes. Check this or google - when singleton is not a singleton.<br>Best course is to use Enum based Singleton as suggested by Joshua Bloch. |
| Answer 3 | For singleton there are two standards that are being used: Double Checked locking, Enum based singleton pattern.<br>UPDATE: Here is another great article which discusses the double checked locking. |

**Figure 1** An example of Web linked resources on Stack Overflow. Hyperlinks are underlined.

Figure 1 shows an example Q&A on Singleton pattern[8]. In the discussion, users reference 11 Web resources related to Singleton pattern. Nine out of these 11 resources are referenced more than 5 times on Stack Overflow. Among frequently referenced Web resources on Stack Overflow, some are referenced hundreds or thousands of times. For example, the first Web resource for "Singleton Pattern" is referenced 1204 times and the Web resource for "Synchronized statements or methods" is referenced 313 times. More importantly, these frequently referenced Web resources are often referenced together in the same discussion threads. We hypothesize that taken in aggregate, Stack Overflow discussions can be mined to recommend community-recognized, correlated Web resources.

To validate our hypothesis, we conduct a large-scale exploratory study of 5.5 millions hyperlinks referenced on Stack Overflow, to investigate its hyperlink dissemination patterns. We observe that 1) Stack Overflow discussions contain a large number of hyperlinks, that cover a variety of online programming resources (e.g., official APIs, tutorials, code examples, forum discussions); 2) These hyperlinks are widely referenced in questions, answers and comments; 3) Millions of discussion threads contain two or more hyperlinks; 4) The presence of hyperlinks in posts correlates with the community votes on the posts.

Our exploratory study suggests the potential of Stack Overflow data for recommending correlated Web resources. Based on our observations of hyperlink dissemination patterns, we design an item-based collaborative filtering approach for recommending correlated Web resources. The recommendation takes advantage of multiple features, such as the location in which hyperlinks are referenced (i.e., in question, answer, or comment), the co-occurrence

[8] http://stackoverflow.com/questions/15425282/singleton-pattern-interview

frequency of hyperlinks, and the votes on the posts (or comments) in which hyperlinks are referenced. Given a set of Stack Overflow discussion threads, our approach produces a Hyperlink Associative Network (HAN) to model the community-recognized, correlated Web resources. Based on this HAN, our approach recommends the top-*k* most correlated Web resources.

To evaluate our approach, we mine a HAN using 6 years of Stack Overflow data (July 2008 - September 2014) and evaluate the accuracy of the recommendation using 9 months of Stack Overflow data (October 2014 - June 2015) as testing dataset. Our evaluation shows that our technique recommends correlated Web resources with satisfactory precision and recall in an open setting. As part of this work, we implement a proof-of-concept tool LinkLive[9]. A user study with 6 participants suggests that our LinkLive tool can recommend helpful and diverse Web resources.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 reports our exploratory study of hyperlink dissemination patterns on Stack Overflow. Section 4 discusses our item-based collaborative filtering approach. Section 5 introduces our LinkLive tool. Section 6 reports the empirical evaluation of our approach. Section 7 reports a user study to evaluate the helpfulness and diversity of LinkLive recommendation. Section 8 discusses our findings. Section 9 concludes the work and outlines our future plan.

## 2 Related work

We first review recommendation systems in general, and then review related work from the aspect of recommendation systems for software engineering.

### 2.1 Recommendation systems

**Content-based recommendation systems** Content-based recommendation systems make recommendations by analyzing the content of textual information and finding regularities in the content [72], but do not exploit users' rating history. Content-based systems are designed mostly to recommend text-based items [1], which can be divided into two categories: *keyword-based* systems and *ontology-based* systems.

In most content-based systems, item descriptions are textual features extracted from Web pages, emails, news articles, or product descriptions. The content in these systems is usually described with *keywords*. For example, Fab system [7] represents Web page content with 100 most important words and then recommends Web pages to users based on these keywords. Similarly, the Syskill & Webert system [59] represents documents with the 128 most informative words. Besides Web systems, keyword-based recommender systems are also widely used in other applications, such as news filtering [2, 69], book recommendation [51] and music recommendation [17]. Some simple retrieval models such as keyword matching [81] and the vector space model with TF-IDF weighting [71] are widely used in these keyword-based recommendation systems.

To better understand text-based items, some content-based systems learn more accurate item profiles that contain references to concepts defined in external knowledge bases.

[9]The LinkLive tool is implemented as a Web browser add-on, based on GreaseMonkey/TamperMonkey technique. It can be downloaded at http://128.199.241.136:9000/download/.

For examples, SiteIF [46] involves MultiWordNet, a multilingual lexical database, in representing documents; ITR system [21] integrates WordNet lexical ontology in the process of learning item profiles. Recently, Freebase, DBpedia and Google knowledge graph are incorporated in content-based recommendations [32, 52, 89]. In summary, these studies incorporate either linguistic or external knowledge to provide better and more accurate results compared to keyword-based methods.

**Collaborative filtering** Collaborative filtering (CF) uses the known preferences of a group of users/items to make recommendations or predictions of the unknown preferences for other users/items [72]. The main difference between CF and content-based recommendation systems is that CF uses the user-item interaction data to make predictions, while content-based systems rely on extracted features of users or items for predictions. According to [14], CF approaches can be categorized as: *memory-based* CF and *model-based* CF.

Memory-based CF methods usually use similarity metrics to obtain the distance between two users, or two items, based on each of their ratios [12]. Memory-based CF methods can be further divided into *item-based* CF and *user-based* CF. Item-based methods [8, 33] identify other items that are similar to the items that a user has liked or rated. User-based methods [48, 86] first find similar users based on users' rating history, then recommend items by comparing with the preferences of similar users. Similarity computation between items or users is a critical step in memory-based CF algorithms. Some most popular approaches to similarity computation are correlation-based [48], cosine-based [18] and conditional probability-based [30] similarity.

Model-based CF methods use the collection of ratings to learn a model, which is then used to make intelligent recommendations for test data or real-world data. Many algorithms, such as Bayesian model [56], matrix factorization [45], fuzzy system [85], genetic algorithm [4], clustering model [70], and dependency network [76], have been investigated to solve the shortcomings [12] of memory-based CF algorithms. In any recommender system, the number of ratings already obtained is usually very small compared to the number of ratings that need to be predicted. Some studies [54, 55] have developed dimensionality reduction techniques to address the problem of sparsity.

Studies show that item-based CF is more robust in the face of user interest changes [20, 43], and is less demanding on the quality of search queries [57, 65]. Therefore, we adopt item-based CF in this work. Unlike traditional recommendation systems that make recommendations by exploiting explicit ratings of items, our approach assumes that the frequency of hyperlink mentions and the votes on the mentioning posts implicitly reflect users' preferences over Web resources.

**Hybrid recommendation systems** Some recommendation systems use a *hybrid* approach by combining collaborative and content-based methods to make recommendations. Hybrid recommender system can be grouped into 4 categories: 1). implementing collaborative and content-based methods separately and combining their predictions [10]; 2). incorporating some content-based characteristics into a collaborative approach [7, 49]; 3). incorporating some collaborative characteristics into a content-based approach [53]; 4). constructing a general unifying model that incorporates both content-based and collaborative characteristics [66]. In conclusion, hybrid systems are designed to avoid certain limitations of content-based and collaborative systems.

**Deep learning based recommendation systems** In recent years, deep learning has achieved tremendous successes on speech recognition, computer vision, and natural language processing. Deep learning is able to effectively capture the non-linear and latent user-item relationships in recommendation systems [92]. For example, neural network-based collaborative filtering [27] is a general framework that learns the user-item interaction function using multilayer perceptron. Deep factorization machine [26] is able to model the high-order feature interactions via deep neural network and low-order interactions via factorization machine. Some other systems, such as those based on autoencoder [67], convolutional neural network [24], recurrent neural network [28], restricted boltzmann machine [63], and generative adversarial network [83], are developed to further enhance recommendation quality.

Although existing studies have investigated the effectiveness of deep learning in recommendation systems, they heavily rely on large-scale training data and do not utilize various side information (e.g., votes and location in our study) in a comprehensive manner. On the contrary, our approach is applicable to data on different scales and incorporates more side information in correlation-based item recommendation.

### 2.2 Recommendation systems for software engineering

Recommendation systems specific to software engineering are emerging to assist developers in a wide range of activities. Example activities include guiding software changes [94], recommending code examples [29], assisting code navigation [31], and augmenting API documentation [38–40, 78].

Social content in software engineering domain, such as Q&A discussions on Stack Overflow, has recently gained much research interest [3, 37, 79]. Some work focuses on content recommendation in Q&A sites. For example, Pedro et al. [64] propose RankSLDA, recommending question for collaborative Q&A systems. Wang et al. [80] present a tag recommendation system to improve the quality of tags in software information sites. Stack Overflow can also recommend related posts based on topic similarity [3]. Some work attempts to integrate social content into software development environments. For example, Ponzanelli et al. [61] present Prompter, a self-confident recommender system, that automatically searches and identifies relevant Stack Overflow discussions, given the code context in IDE. Zagalsky et al. [91] present a code search and recommendation tool which brings together social media and code recommendation systems. Others attempt to link information across different information sources. For example, Subramanian et al. [73] present a live API documentation tool, to link the official API documentation with user-generated content on Stack Overflow. Bagheri et al. [6] propose a semantic linking technique to recommend content from Reddit for Stack Overflow questions.

Existing studies mainly focus on recovering traceability between source code, Q&A discussions, and other forms of social content, to facilitate relevant information access. In contrast, our focus in this study is on mining hyperlink correlations in Q&A discussions to recommend correlated Web resources.

## 3 An exploratory study of hyperlinks on stack overflow

Our technique relies on the presence of hyperlinks in Stack Overflow discussions, and the presence of potential correlation patterns among these hyperlinks. Thus, an investigation is warranted if hyperlinks on Stack Overflow are a potential source for discovering correlated Web resources. We investigate three research questions:

**Table 1** Top-20 most referenced domains

| Rank | #Citations | Domain | Rank | #Citations | Domain |
|---|---|---|---|---|---|
| 1 | 1,916,405 | stackoverflow.com | 11 | 140,586 | apple.com |
| 2 | 772,453 | microsoft.com | 12 | 137,706 | python.org |
| 3 | 667,334 | jsfiddle.net | 13 | 124,171 | apache.org |
| 4 | 567,959 | github.com | 14 | 112,318 | mozilla.org |
| 5 | 370,405 | wikipedia.org | 15 | 101,733 | sourceforge.net |
| 6 | 341,174 | google.com | 16 | 83,880 | stackexchange.com |
| 7 | 264,459 | php.net | 17 | 79,181 | w3.org |
| 8 | 240,949 | oracle.com | 18 | 75,616 | mysql.com |
| 9 | 177,417 | android.com | 19 | 65,701 | msdn.com |
| 10 | 172,606 | jquery.com | 20 | 63,170 | facebook.com |

- **RQ1**: What hyperlinks are referenced? Where are they referenced?
- **RQ2**: How frequent a hyperlink is referenced? How frequent two or more hyperlinks are referenced together in the same discussion thread?
- **RQ3**: Does the presence of hyperlinks in posts correlate with number of votes received?

## 3.1 Dataset

We use Stack Overflow data dump (July 2008 to September 2014) in this exploratory study. The same dataset is also used as the training data for the evaluation of our technique. The dataset contains 7,990,787 questions, 13,684,117 answers, and 32,506,636 comments. We consider a question and all its answers and comments as a discussion thread. Thus, we have 7,990,787 discussion threads in the dataset, i.e., the same as the number of questions. We extract hyperlinks in questions, answers and comments from the *href* HTML tag. As many hyperlinks in comments are referenced as plain text, we also use regular expressions to parse plain text and to extract hyperlinks.

## 3.2 RQ1: what and where

We extract 5,522,886 distinct hyperlinks from the dataset. These hyperlinks are from 234,815 distinct domains. Table 1 lists the top-20 most referenced domains in Stack Overflow discussions. Observe that the domains cover a variety of online programming resources, including official API documents, tutorial websites, code example sites, code repositories, Wikipedia, and forum discussions, etc. Our results are consistent with earlier studies on hyperlink sharing practices on Stack Overflow [23].

Tables 2 and 3 summarize the statistics of hyperlinks that are referenced in questions, answers, and comments, respectively. We can observe that hyperlinks widely present in questions, answers, and comments. Stack Overflow encourages users to include hyperlinks to relevant Web resources in their discussions[10]. This explains the wide present of hyperlinks on Stack Overflow. Therefore, to analyze hyperlinks on Stack Overflow, we must consider all components that may contain hyperlinks in the discussion, e.g., questions, answers, and comments.

[10]http://stackoverflow.com/help/how-to-answer

**Table 2** Statistics of questions/answers/comments having hyperlinks

| | #Questions | | #Answers | | #Comments | |
|---|---|---|---|---|---|---|
| Total number | 7,990,787 | | 13,684,117 | | 32,506,636 | |
| Contain hyperlinks | 1,315,053 | (16.46%) | 4,457,576 | (32.57%) | 5,788,648 | (17.81%) |

## 3.3 RQ2: reference and co-occurrence frequency

Figure 2 plots hyperlink citation distribution and domain citation distribution. The chart shows the percentage of hyperlinks (or domains) that have been referenced for a certain number of times. About 21.83% hyperlinks and about 50.14% domains are referenced at least twice in discussions. The power-law distribution of hyperlink citations indicates that the hyperlinks shared on Stack Overflow are largely stable, although the number of distinct hyperlinks keeps growing. That is, a small portion of frequently referenced hyperlinks attracts a large portion of developers' attention. Therefore, the frequency of a hyperlink is an important indicator of the community's preference.

Figure 3 shows the distribution of hyperlinks per discussion thread. Observe that 28.65% of discussion threads (i.e., 2,289,360) contain two or more hyperlinks. This shows that millions of discussion threads can be a potential source for mining correlated Web resources. Sharing in the same discussion thread indicates the relatedness among hyperlinks.

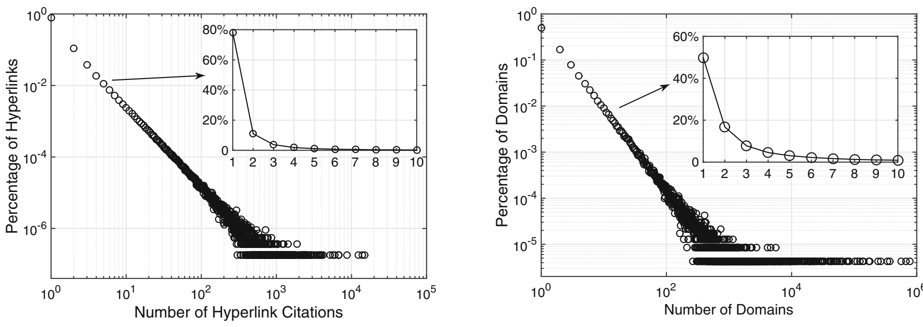
## 3.4 RQ3: hyperlink-vote correlation

Previous studies show that, the presence of hyperlinks in a post is a strong indicator of the post being more informative [22]. We want to further investigate the correlation between the presence of hyperlinks in a post and the number of votes it receives. To this end, we collect 4,596,855 accepted answers in our dataset. We then split these accepted answers into two groups: group1 (1,640,651 answers) where the accepted answers contain hyperlinks, and group2 (2,956,204 answers) where the accepted answers do not contain hyperlinks. We have the null hypothesis *H0: There is no statistically significant difference in the number of votes on the accepted answers from the two groups.*

Examination of distribution of number of votes on answers shows that the distribution does not obey normal distribution. Therefore, we use non-parametric statistical tests to study the significance of the vote differences in the two groups. In particular, we use the KolmogorovSmirnov test (KS-test) [41]. The KS-test has the advantage of making no assumption about the distribution of data. *p-value* of the KS-test result is below 0.001. Therefore, we reject the null hypothesis. In other words, there is a statistically significant difference in the number of votes on the accepted answers with and without hyperlinks. This analysis suggests that number of votes is another important indicator of community's preference of hyperlink.

**Table 3** Statistics of the sources of distinct hyperlinks

| Location of hyperlink | Questions | Answers | Comments | All |
|---|---|---|---|---|
| #Distinct hyperlinks | 1,379,007 | 3,120,816 | 1,913,474 | 5,522,886 |



(a) Hyperlink distribution (b) Domain distribution

**Figure 2** The hyperlink and domain citation distribution. The absolute numbers are plotted in log scale, and the percentages are plotted in bar chart

## 4 The linklive approach

Our exploratory study shows that hyperlinks on Stack Overflow are a good source for mining correlated Web resources. The analysis also leads to the design of our item-based collaborative filtering approach for mining correlated Web resources. In this work, we consider a hyperlink as a Resource-of-Interest (ROI). Given a hyperlink (referred to as a *seed hyperlink* or *seed ROI*), our goal is to recommend top-*k* correlated Web resources (referred to as *recommended hyperlinks* or *recommended ROIs*) that are highly recognized by the Stack Overflow community.

### 4.1 Hyperlink associative network

We construct a Hyperlink Associative Network (HAN) to model correlated hyperlinks on Stack Overflow as follows. The notations used in this paper are listed in Table 4.

**Definition 1** (**Discussion Thread**) A discussion thread consists of a question and all its answers in chronological order. Both questions and answers are also known as posts. A post may have zero or more comments.

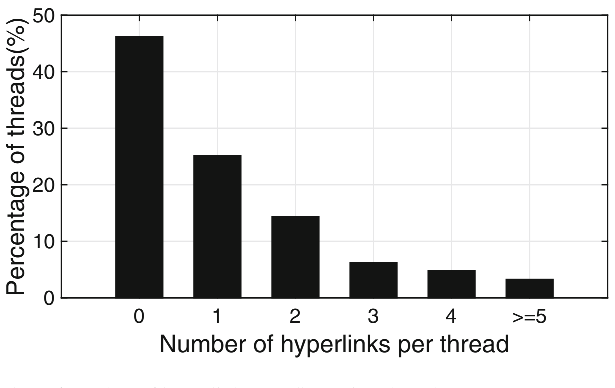


**Figure 3** Distribution of number of hyperlinks per discussion thread

**Table 4** The notations of this paper

| Symbol | Description |
|---|---|
| $T$ | A discussion thread |
| $h_s$ | A seed ROI |
| $h_t$ | A candidate ROI |
| $H$ | The set of distinct hyperlinks $H = h_s \bigcup h_t$ |
| $\mathcal{G}_{h_s} = (V, E)$ | Hyperlink Co-occurrence Graph for $h_s$, Vertex $V$, Edge $E$ |
| $e_{s,t}$ | The edge from node $h_s$ to node $h_t$ |
| $HAN = (H, X)$ | Hyperlink Associative Network. $HAN = \bigcup \mathcal{G}_{h_s}$ |
| $\omega_{s,t}$ | The weight of edge $e_{s,t}$ in $HAN$ |
| $S_{e_{s,t}}$ | The score of edge $e_{s,t}$ in $E$ |
| $S'_{e_{s,t}}$ | The normalized score of edge $e_{s,t}$ in $E$ |
| $\boldsymbol{S}_p$ | The score vector of posts (question and answers) in the discussion thread $T$ |
| $\boldsymbol{S}_c$ | The score vector of comments of a post in the discussion thread $T$ |

**Definition 2** (**ROI Location and Location Type**) The question, answer, or comment in which an ROI is referenced is the location of the ROI. Each of which is also known as the location type.

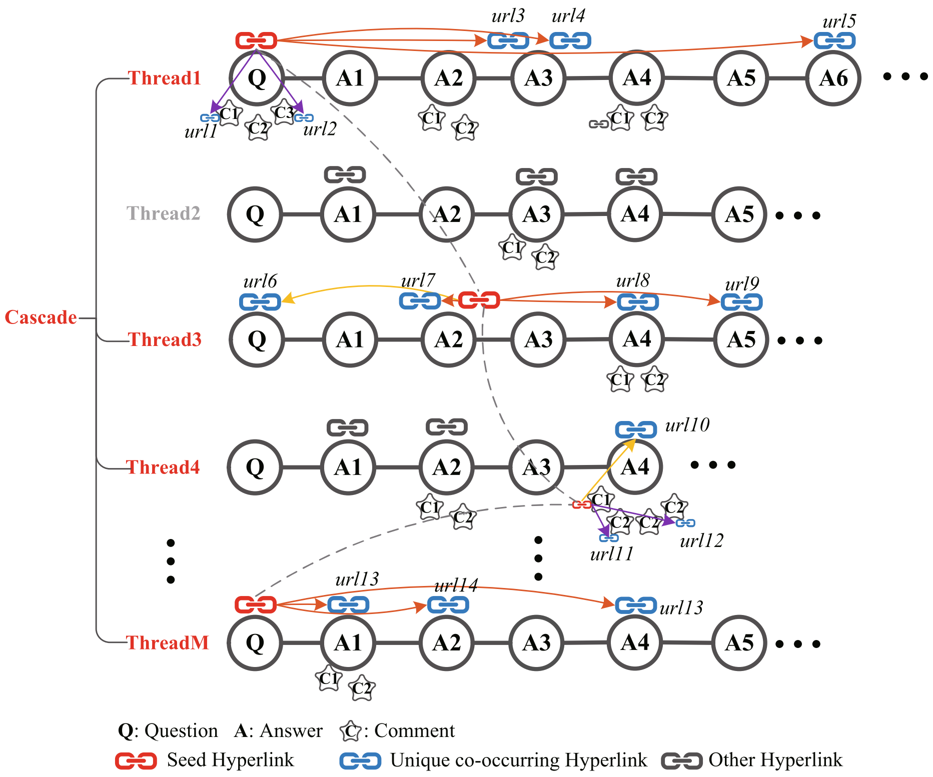


**Figure 4** Hyperlink co-occurrence graph. Thread2 does not contain the seed hyperlink, and thus is excluded in this cascade. The different arrows indicate different associative-edge weights

**Definition 3** (**ROI Cascade**) An ROI cascade consists of all the discussion threads that reference a seed ROI in chronological order.

In our model, a discussion thread is the basic information unit. Figure 4 illustrates multiple discussion threads, from Thread1 to ThreadM, in chronological order. Let $H$ be the set of distinct hyperlinks in all discussion threads. For seed ROI $h_s \in H$, we construct an ROI cascade, as shown in Figure 4, where the seed ROI is highlighted in red. The seed ROI is referenced in multiple locations including questions $q$ of Thread1 and ThreadM, answer $A2$ in Thread3, and a comment to answer A4 in Thread4. Note that, Thread2 does not contain the seed ROI, and thus is excluded in the ROI cascade.

**Definition 4** (**Hyperlink Co-occurrence Graph** $\mathcal{G}_{h_s}$) A $\mathcal{G}_{h_s} = (V, E)$ is a directed graph where vertex set $V$ is the set of all distinct hyperlinks referenced in ROI cascade of seed ROI $h_s$, and a co-occurrence edge $e_{s,t} \in E$ represents the co-occurrence of the seed hyperlinks $h_s$ and the other hyperlink $h_t$ in the same discussion thread. $h_t$ is referred to as *candidate hyperlink* or *candidate ROI*. The edge $e_{s,t} \in E$ is indexed by the locations of $h_s$ and $h_t$. The edge has a $score$ indicating the number of votes on the post (i.e., question or answer) or comment in which $h_t$ is referenced.

Given the ROI cascade of seed ROI $h_s$, the edge set $E$ of graph $\mathcal{G}_{h_s}$ is constructed as follows. For each discussion thread $T$ in the ROI cascade,

- If the location of $h_s$ is a post (i.e., question or answer), then for each distinct hyperlink $h_t$ in the posts of the discussion thread $T$, a co-occurrence edge $e_{s,t}$ (node $h_s$ to node $h_t$) is added to $E$. The red edges in Thread1, Thread3, and ThreadM and the orange edge between the seed ROI and $url6$ in question Q of Thread3 illustrate this scenario. The seed ROI $h_s$ and a candidate ROI $h_t$ may appear in the same post, e.g., $url7$ in answer A2 of Thread3. If a candidate ROI $h_t$ appears in $n$ ($n \geq 2$) different posts, e.g., $url13$ in answers A1 and A4 of ThreadM, then $n$ different edges will be added.
- If the location of $h_s$ is a post, then for each distinct hyperlink $h_t$ in the comments of the post, a co-occurrence edge $e_{s,t}$ (node $h_s$ to node $h_t$) is added to $E$. The purple edges between the seed ROI and $url1$ and $url2$ in comments of question $q$ of Thread1 illustrate this scenario. Note that we do not consider hyperlinks in comments of other posts (such as the one in a comment of A4 of Thread1) as candidate ROIs. The rationale is that the comments are usually related to only the post being commented.
- If the location of $h_s$ is a comment, then for each distinct hyperlink $h_t$ in the comments of the same post and in the post being commented, a co-occurrence edge $e_{s,t}$ (node $h_s$ to node $h_t$) is added to $E$. The orange and purple edges between the seed ROI and $url10$ in answer A4 of Thread4 and $url11$ and $url12$ in comments of answer A4 illustrate this scenario. Again, we do not consider hyperlinks in other posts (such those in answers A1 and A2 of Thread4) as candidate ROIs of the seed ROI in a comment of a post.

**Definition 5** (**Hyperlink Associative Network HAN**) A $HAN = (H, X)$ is a directed graph where vertex set $H$ is the set of hyperlinks in all discussion threads, and an associative edge $e_{s,t} \in X$ if there exist one or more co-occurrence edges $e_{s,t} \in E$ ($E$ is the edge set of $\mathcal{G}_{h_s}$). That is, $HAN = \bigcup \mathcal{G}_{h_s}$ for all $h_s$ and $H = h_t \bigcup h_s$.

Each associative edge $e_{s,t} \in X$ has a $weight$, to be described next. This weight measures the correlation similarity between seed ROI $h_s$ and a candidate ROI $h_t$.

### 4.2 Associative-edge weight computation

The *weight* of associative edge $e_{s,t} \in X$ (denoted by $\omega_{s,t}$) is computed based on the *score* of the corresponding co-occurrence edges $e_{s,t} \in E$ (denoted by $S_{e_{s,t}}$) in $\mathcal{G}_{h_s}$. For a candidate ROI $h_t$ of seed ROI $h_s$ in a discussion thread, the score of posts (or comments) in which the $h_t$ is referenced reflects the competition among the candidate ROIs in the discussion thread. The intuition is that ROIs in high-score posts (or comments) would be more valuable to users than those in low-score posts (or comments). Thus, a naive method to compute $\omega_{s,t}$ is to sum up $S_{e_{s,t}}$ of all co-occurrence edges $e_{s,t} \in E$, i.e., $\omega_{s,t} = \sum_{e_{s,t} \in E} S_{e_{s,t}}$. We refer to this straightforward method as the *baseline method* in our evaluation.

We now design an enhanced weight computation method (referred to as the *enhanced method* in our evaluation) based on the following two intuitions. First, the correlation between a seed ROI $h_s$ and a candidate ROI $h_t$ is different when $h_s$ is referenced in different types of locations, i.e., question, answer, and comment. This intuition is based on the consideration that the asker is very likely to follow up all answers, but an answerer or a commenter may not pay much attention to answers or comments from others. Second, scores of posts or comments vary greatly from one discussion thread to another. Thus, it would be necessary to normalize the scores within each discussion thread so that the scores are comparable across different discussion threads.

Based on above two intuitions, we normalize the *score* of co-occurrence edge $e_{s,t} \in E$ for the three different types of locations of $h_s$ as follows. We denote the normalized score as $S'_{e_{s,t}}$.

***$h_s$* is referenced in question** The seed ROI in question Q of Thread1 and ThreadM in Figure 4 illustrate this scenario. We only consider the competition among the candidate ROIs $h_t$ in comparable type of location, i.e., $h_t$ in posts or $h_t$ in comments, because the score of posts and comments can vary greatly in scale and most of comments have no score.

Let $\boldsymbol{S}_p$ be the score vector of all posts (question and answers) of the discussion thread $T$. If $h_t$ is referenced in a post in $T$, then the normalized score of the edge $e_{s,t} \in E$ is:

$$S'_{e_{s,t}} = \frac{S_{e_{s,t}} - \min(\boldsymbol{S}_p)}{\max(\boldsymbol{S}_p) - \min(\boldsymbol{S}_p)} \tag{1}$$

That is, $S'_{e_{s,t}}$ is a value in [0, 1] that reflects the relative score of $h_t$ compared with that of other candidate ROIs of the $h_s$ in the posts of the discussion thread $T$. If $\max(\boldsymbol{S}_p) = \min(\boldsymbol{S}_p) = 0$, we set $S'_{e_{s,t}} = 0.1$. The *score* of the red edges in Thread1 and ThreadM in Figure 4 is normalized using (1).

Let $\boldsymbol{S}_c$ be the score vector of all comments of question $q$ in which $h_s$ is referenced in discussion thread $T$. If $h_t$ is referenced in a comment of question $q$, then the normalized score of edge $S'_{e_{s,t}}$ is:

$$S'_{e_{s,t}} = \frac{S_{e_{s,t}} - \min(\boldsymbol{S}_c)}{\max(\boldsymbol{S}_c) - \min(\boldsymbol{S}_c)} \tag{2}$$

If $\max(\boldsymbol{S}_c) = \min(\boldsymbol{S}_c) = 0$, we set $S'_{e_{s,t}} = 0.1$. The *score* of the purple edges in Thread1 in Figure 4 is normalized using (2).

***$h_s$* is referenced in answer** The seed ROI in answer A2 of Thread3 in Figure 4 illustrates this scenario. Let the answer be in discussion thread $T$. If $h_t$ is referenced in question $q$ of $T$ (e.g., the orange edge between the seed ROI and *url*6 in question Q of Thread3), we set $S'_{e_{s,t}} = 1$. This is based on the consideration that the information in an answer should be directly related to the information in the question. If $h_t$ is referenced in an answer of discussion thread $T$ (e.g., the red edges between seed ROI and $url7$, $url8$ and $url9$ in Thread3), we compute $S'_{e_{s,t}}$ using the (1). If $h_t$ is referenced in a comment of an answer in which $h_s$ is referenced, we compute $S'_{e_{s,t}}$ using the (2).

**$h_s$ is referenced in comment** The seed ROI in a comment of answer A4 of Thread4 in Figure 4 illustrates this scenario. If $h_t$ is referenced in a post (A4 of Thread4) being commented (e.g., the orange between seed ROI and $url19$ in answer A4 in Thread4), we set $S'_{e_{s,t}} = 1$. This is based on the consideration that the information in a comment should be directly related to the information in the post being commented. If $h_t$ is referenced in the comments of the same post (e.g., the purple edges between seed ROI and $url11$ and $url12$ in the comments of answer A4 of Thread4), we compute $S'_{e_{s,t}}$ using (2).

Once we normalize the $score$ of co-occurrence edge $e_{s,t} \in E$ for all discussion threads, we compute the normalized weight of associative edge $e_{s,t} \in X$ as $\omega_{s,t} = \sum_{e_{s,t} \in X} S'_{e_{s,t}}$.

### 4.3 ROI recommendation

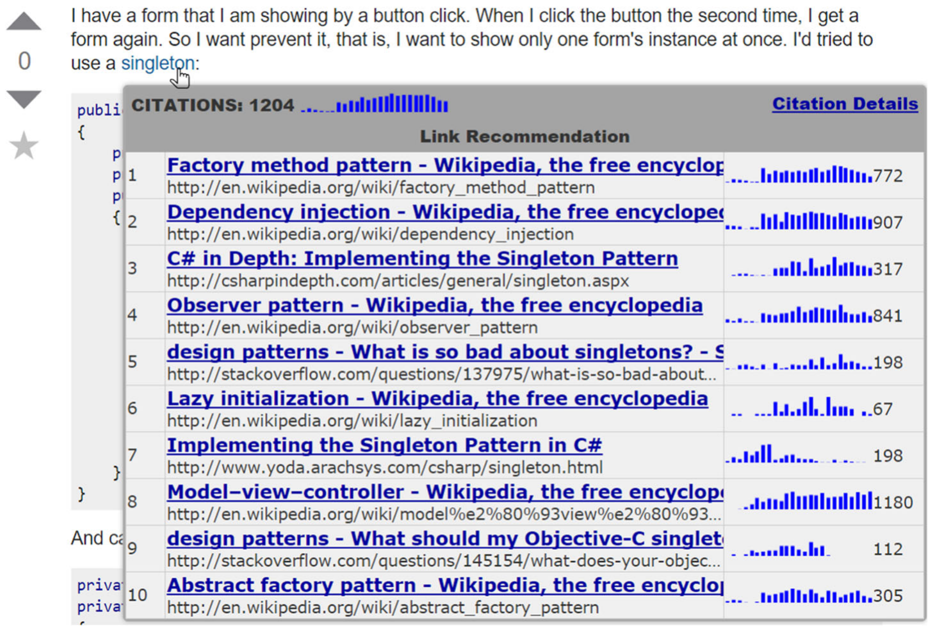
The HAN mined from a set of discussion threads serves as the underlying model for recommendation of correlated Web resources. Given an ROI, if it appears in the HAN, we recommend the top-$k$ candidate ROIs in the HAN that have the highest associative edge weight (e.g., correlation similarity) with the given ROI as the recommended ROIs. Formally, given an ROI $h_q$, we retrieve top-$k$ candidate ROIs from $HAN = (H, X)$ by

$$\begin{aligned} h_t &= \underset{h_t \in H}{\operatorname{argmax}} \sum_{e_{q,t} \in X} S'_{e_{q,t}} \\ s.t. &\quad e_{q,t} \in X \end{aligned} \tag{3}$$

where $e_{q,t}$ is a co-occurrence edge representing the co-occurrence of the seed hyperlink $h_q$ and the other hyperlink $h_t$ in the same discussion thread.

**Algorithm 1** ROI Recommendation

```
   Query: An ROI $h_q$
   Output: Top-$k$ ROIs
   // Offline Phase: Construct HAN
1  $HAN = \emptyset$;
2  foreach $h_s$ in ROI corpus do                         // iterate ROI corpus
3      Construct co-occurrence graph $\mathcal{G}_{h_s} = (V, E)$ ;
4      foreach $e_{s,t}$ in $E$ do                       // iterate edges in $\mathcal{G}_{h_s}$
5          Compute normalized weights $S'_{e_{s,t}}$;
6          $HAN \leftarrow HAN \bigcup \mathcal{G}_{h_s}$;
7  return $HAN$
   // Online Phase: Real time recommendations for $h_q$
8  begin Querying $HAN$ with $h_q$
9      Extract subgraph $\mathcal{G}_{h_q} = (V, E)$ from $HAN$ for $h_q$;
10     Retrieve top-$k$ ROIs based on (3);
11     return top-$k$ ROIs for $h_q$
```


0

I have a form that I am showing by a button click. When I click the button the second time, I get a form again. So I want prevent it, that is, I want to show only one form's instance at once. I'd tried to use a singleton:

CITATIONS: 1204 Citation Details

Link Recommendation

| | | | |
|---|---|---|---|
| 1 | Factory method pattern - Wikipedia, the free encyclop | http://en.wikipedia.org/wiki/factory_method_pattern | 772 |
| 2 | Dependency injection - Wikipedia, the free encyclope | http://en.wikipedia.org/wiki/dependency_injection | 907 |
| 3 | C# in Depth: Implementing the Singleton Pattern | http://csharpindepth.com/articles/general/singleton.aspx | 317 |
| 4 | Observer pattern - Wikipedia, the free encyclopedia | http://en.wikipedia.org/wiki/observer_pattern | 841 |
| 5 | design patterns - What is so bad about singletons? - S | http://stackoverflow.com/questions/137975/what-is-so-bad-about... | 198 |
| 6 | Lazy initialization - Wikipedia, the free encyclopedia | http://en.wikipedia.org/wiki/lazy_initialization | 67 |
| 7 | Implementing the Singleton Pattern in C# | http://www.yoda.arachsys.com/csharp/singleton.html | 198 |
| 8 | Model–view–controller - Wikipedia, the free encyclop | http://en.wikipedia.org/wiki/model%e2%80%93view%e2%80%93... | 1180 |
| 9 | design patterns - What should my Objective-C singlet | http://stackoverflow.com/questions/145154/what-does-your-objec... | 112 |
| 10 | Abstract factory pattern - Wikipedia, the free encyclo | http://en.wikipedia.org/wiki/abstract_factory_pattern | 305 |

**Figure 5** The LinkLive Tool. The recommendation is triggered when mouse hovers over hyperlink ‘singleton’

Algorithm 1 summarizes the procedure of ROI recommendation, which consists of an offline phase and an online phase. The offline phase first constructs co-occurrence graph for each seed ROI, then computes the associative-edge weights. The output of the offline phase is an HAN. Given an ROI, the online phase retrieves top-$k$ ROIs from the $HAN$. We assume that there are $n$ seed ROIs in our corpus and each ROI has $m$ associated ROIs in its ROI cascade. The time complexity of the offline phase is $O(nm)$. For the online phase, the time complexity of extracting subgraph is $O(n)$. The time complexity of ranking edges is $O(m \log m)$. Thus, the total time complexity of online phase is $O(n + m \log m)$.

## 5 The linklive tool

We implement a proof-of-concept tool of our LinkLive approach[11]. The backend hyperlink associative network (i.e., hyperlink correlation model) is mined from training data, which is the Stack Overflow data dump (July 2008 to September 2014) (see Section 3). When a user visits a Web page or mouse hovers over a hyperlink in the Web page, the LinkLive tool searches for hyperlinks from the backend model. If the hyperlink is found in the backend, and the hyperlink has been referenced 5 times or more in the training dataset, the tool recommends top-10 correlated Web resources for the given hyperlink. We set a minimal

[11]Video demonstration at https://youtu.be/PvgzJ-fslGs
The tool is available for downloading at http://128.199.241.136:9000/download/

reference frequency of the seed hyperlink for triggering the recommendation, because we observe that for seed hyperlinks that are referenced fewer than 5 times, the co-occurring hyperlinks are ad-hoc.

Figure 5 shows the LinkLive recommendation when mouse hovers over hyperlink (https://en.wikipedia.org/wiki/Singleton_pattern) in a Stack Overflow question[12]. In addition to the recommendation of correlated Web resources, the LinkLive tool also shows a bar chart of citation history for the seed hyperlink and each recommended hyperlink.

## 6 Evaluation of the linklive recommendation

We evaluate LinkLive from two perspectives: *accuracy* and *reliability*.

- **RQ4**: How accurate is the LinkLive recommendation of correlated Web resources compared with user-explicitly-referenced Web resources in Stack Overflow discussions?
- **RQ5**: How does the time elapsed after modeling training affect the accuracy of the LinkLive recommendation?

### 6.1 Experiment setup

#### 6.1.1 Testing dataset

To answer the above research questions, we use Stack Overflow data dump (October 2014 to June 2015) as the testing dataset. The testing dataset contains 1,835,754 discussion threads and 1,384,063 distinct hyperlinks. Similar to the training dataset, 28.36% of discussion threads (or 520,619) in the test dataset have two or more hyperlinks.

We build the "ground truth" to answer research questions RQ4 and RQ5 from these 28.36% discussion threads as follows. We collect the discussion threads in the testing dataset that reference at least two hyperlinks and at least one of them can trigger LinkLive recommendation. 260,141 discussion threads in the testing dataset satisfy this criteria, and 92,800 hyperlinks can trigger LinkLive recommendation. These 92,800 hyperlinks are referred to as seed hyperlinks in this evaluation. Given one of the 260,141 discussion threads and a seed hyperlink in the discussion thread, we collect all the co-occurring hyperlinks for the seed hyperlink in the discussion thread in the same way as we build Hyperlink Co-occurrence Graph (see Section 4.1). We consider this set of co-occurring hyperlinks as the ground truth of correlated Web resources for the given seed hyperlink in the test data. Hereafter, we refer to this ground truth as user-explicitly-referenced correlated Web resources.

#### 6.1.2 Metrics

We use two metrics to evaluate the accuracy of LinkLive recommendation: $Precision@k$ and $Recall@k$ ($Pr@k$ and $Re@k$ for short). $k = \{1, 5, 10, 20, 30\}$ is the number ofrecommended Web resources. Let $R^k_{h_s}$ be the set of top-$k$ recommended Web resources for a

[12]http://stackoverflow.com/questions/23360052

seed hyperlink $h_s$ using LinkLive. Let $GT_{h_s}$ be the set of user-explicitly-referenced correlated Web resources for $h_s$ in a discussion thread. $Pr@k$ for $h_s$ is $\frac{R^k_{h_s} \cap GT_{h_s}}{k}$. $Re@k$ for $h_s$ is $\frac{R^k_{h_s} \cap GT_{h_s}}{GT_{h_s}}$. We average the precision and recall values over all the 260,141 discussion threads.

## 6.2 The accuracy of linklive recommendation

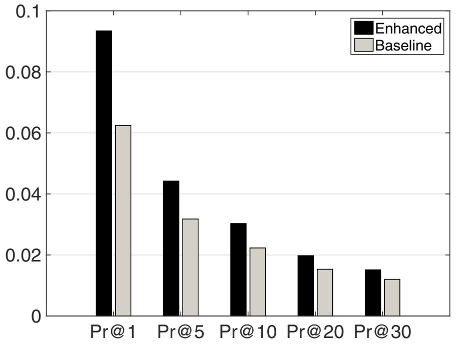
Recall that we have a baseline method and an enhanced method to compute associative edge weight (the correlation similarity between hyperlinks) for the recommendation of correlated Web resources (see Section 4.3). In this section, we compare the accuracy of the recommendation of the two methods using the $Precision@K$ and $Recall@K$ metrics.

### *6.2.1 Overall performance*

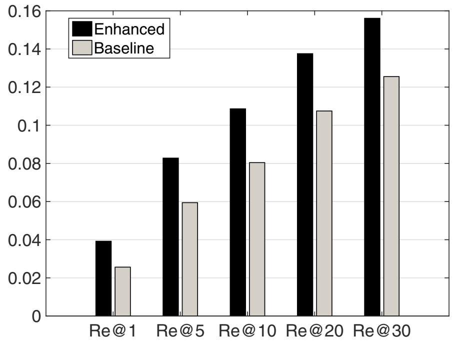
Figure 6 reports the $Pr@k$ and $Re@k$ of both baseline and enhanced methods. We observe that:

- The enhanced method outperforms the baseline method at all different $k$ values for both precision and recall. This shows that taking into account the location where hyperlinks are referenced and normalizing scores in discussion threads improves the accuracy of LinkLive recommendation.
- As expected, precision drops as the value of $k$ increases, while the recall increases along $k$ increases. The best precision is 9.34% at $k = 1$ and the best recall is 15.61% at $k = 30$. At $k = 10$, the precision is 3.03% and the recall is 10.86%.

Note that $Pr@k$ is calculated by $\frac{R^k_{h_s} \cap GT_{h_s}}{k}$. Most discussion threads only contain two hyperlinks. That is, $R^k_{h_s} \cap GT_{h_s} = 1$ in most cases. Thus, the $Pr@k$ is expected to be very small. On the other hand, in the testing dataset, 79% of hyperlinks have not been referenced in the training dataset.



(a) Precision@k



(b) Recall@k

**Figure 6** Overall performance of the LinkLive recommendation

As discussed in Section 3.3, these hyperlinks do not represent Web resources that the Stack Overflow community care the most about. However, the presence of such hyperlinks brings down the precision of the recommendation. Furthermore, there are only a small percentage (about 3.81%) of discussion threads referencing 5 or more hyperlinks. That is, for over 96% of the 260,141 discussion threads from which we collect the ground truth, the ground truth contains fewer than 5 user-explicitly-referenced Web resources. Therefore, even the LinkLive top-$k$ recommendations contain all the user-explicitly-referenced Web resources, the precision is low when $k$ is 5 or larger.

As the main goal of LinkLive recommendation is to help developers discover correlated Web resources that they may be interested in, we deem recall to be more important than precision. The reference of hyperlinks in the test discussion threads can be affected by many factors, such as variation of question topics, the expertise of askers and answerers. In such an open-ended setting, our enhanced recommendation method achieves satisfactory and acceptable recall (10.86% at top 10), on a par with the recall of the state-of-the-art recommendation systems for e-commerce or location-based services reported in the literature [25, 90].

### 6.2.2 *Impact of number of citations*

As shown in Figure 2, the distribution of hyperlinks obeys a power-law distribution. It reveals that most hyperlinks are referenced by a small number of times and a small fraction of hyperlinks are referenced frequently. In this evaluation, we split the hyperlinks into four citation levels according to the number of their citations in Q&A discussions: "$5 - 50$", "$51 - 100$", "$101 - 500$" and "$> 500$" (Table 5). In the testing dataset, the number of seed hyperlinks at these 4 citation levels are '82,190', '5,979', '4,171' and '460', respectively. These seed hyperlinks are referenced '233,707', '48,130', '79,396' and '59,125' times in the testing dataset, respectively.

Tables 6 and 7 show the precision and recall of LinkLive recommendation for seed hyperlinks at the 4 citation levels. From these two tables, we observe that:

- At all citation levels, the enhanced method outperforms the baseline method.
- Both precision and recall increase as the citation frequency of seed hyperlinks increases. For seed hyperlinks that are referenced "$5-50$" times, the precision and recall are worse than the overall performance. For seed hyperlinks that are referenced "$> 500$" times, the enhanced method achieves precision 4.81%@10 and recall 17.61%@10, which is better than the overall performance.
- Comparing the lowest and highest citation levels "$5 - 50$" versus "$> 500$", the performance at citation level "$> 500$" is significantly better than that of level "$5 - 50$". In most cases, the precision and recall values are doubled between the two levels.

**Table 5** Number of Seed ROIs and citations at different citation levels

| Citation Levels | 5-50 | 51-100 | 101-500 | >500 |
|---|---|---|---|---|
| #Seed ROIs | 82,190 | 5,979 | 4,171 | 460 |
| #Citations | 233,707 | 48,130 | 79,396 | 59,125 |

**Table 6** Precision at different citation levels

| Citation Levels | | 5-50 | 51-100 | 101-500 | >500 |
|---|---|---|---|---|---|
| Pr@1 | Enhanced | 7.02% | 9.72% | 11.46% | 15.31% |
| | Baseline | 4.42% | 5.24% | 7.24% | 12.86% |
| Pr@5 | Enhanced | 3.31% | 4.51% | 5.49% | 7.05% |
| | Baseline | 2.48% | 2.83% | 3.49% | 5.64% |
| Pr@10 | Enhanced | 2.25% | 2.79% | 3.69% | 4.81% |
| | Baseline | 1.86% | 1.93% | 2.37% | 3.47% |
| Pr@20 | Enhanced | 1.56% | 1.68% | 2.23% | 2.27% |
| | Baseline | 1.37% | 1.40% | 1.53% | 2.06% |
| Pr@30 | Enhanced | 1.23% | 1.24% | 1.63% | 1.98% |
| | Baseline | 1.12% | 1.13% | 1.19% | 1.49% |

This result suggests that for a seed hyperlink that is more frequently referenced in Q&A discussions, hyperlinks that previously co-occur with seed hyperlink are very likely to be referenced again, when the seed hyperlink is referenced again. As such, our approach makes more accurate recommendation for the seed hyperlinks that are more frequently referenced. This phenomena can be explained by using preferential attachment theory [15], which is the key intuition underlying the design of our LinkLive approach.

***RQ4**: The LinkLive recommendation is reasonably accurate in an open-ended setting, compared with user-explicitly-referenced correlated Web resources.*

## 6.3 The reliability of linklive recommendation

Stack Overflow receives thousands of questions and answers every day. As the data grows over time, the technology landscape also changes rapidly. LinkLive recommendation relies on hyperlink correlation patterns in Q&A discussions to make effective recommendation. The time elapsed after model training may invalidate the patterns learned from the past Q&A discussions. To study the impact of time elapsed on LinkLive recommendation, we split the

**Table 7** Recall at different citation levels

| Citation Levels | | 5-50 | 51-100 | 101-500 | >500 |
|---|---|---|---|---|---|
| Re@1 | Enhanced | 2.81% | 3.94% | 4.68% | 7.25% |
| | Baseline | 1.75% | 2.09% | 2.80% | 5.81% |
| Re@5 | Enhanced | 6.09% | 8.58% | 10.12% | 13.69% |
| | Baseline | 4.64% | 5.31% | 6.27% | 10.86% |
| Re@10 | Enhanced | 7.93% | 10.46% | 13.02% | 17.61% |
| | Baseline | 6.62% | 7.17% | 8.42% | 12.75% |
| Re@20 | Enhanced | 1.39% | 12.33% | 15.49% | 19.87% |
| | Baseline | 9.19% | 9.70% | 10.81% | 14.90% |
| Re@30 | Enhanced | 11.77% | 13.65% | 16.83% | 21.22% |
| | Baseline | 10.76% | 11.47% | 12.50% | 16.14% |

testing dataset into three subsets, each of which contains 3-month data (October 2014 to December 2014, January 2015 to March 2015, and April 2015 to June 2015). We use the first 3-month data as the baseline to compare the accuracy of LinkLive recommendation with the second and the third 3-month data. For the three subsets, we collect '54,654', '51,076' and '47,722' seed hyperlinks, respectively, and LinkLive tool recommends '181,385', '169,511' and '158,381' correlated Web resources, respectively. For each subset, we collect the ground truth and compute $Precision@k$ and $Recall@k$ in the same way as we process the full testing dataset.

Figure 7 shows $Precision@k$ and $Recall@k$ values ($k = 1, 5, 10, 20, 30$) for the three testing subsets. We observe that:

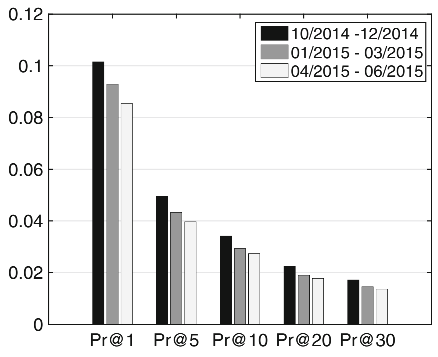
– Both precision and recall deteriorate as the time gap between the training dataset and the testing dataset increases. However, neither precision nor recall degrade significantly even after six months of HAN training.
– The time elapsed after model training has bigger impact on precision than on recall. Precision drops up to 15.4% after three months of model training, up to 20.9% after six months of model training. Recall drops 9.65% after three months of model training, 15.2% after six months of model training.

***RQ5**: The LinkLive tool makes reliable recommendation for new data after three months of model training. Recall is more reliable than precision which is a desirable goal of the tool. Users can update the backend model every three months to avoid significant drop in recommendation accuracy.*

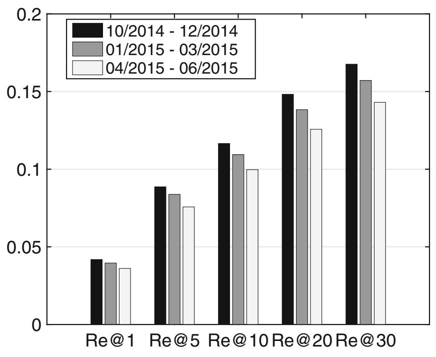
## 7 User study

We perform a user study to evaluate the helpfulness and diversity of LinkLive recommendation:

– **RQ6**: Can the LinkLive discover helpful and diverse Web resources for developers in practice?



(a) Precision@k



(b) Recall@k

**Figure 7** Performance on the three testing subsets

## 7.1 Study design

### 7.1.1 Participants

We recruit 6 graduate students. All participants have either computer science or computer engineering background. Participants have 1-3 years of programming experience in popular programming languages and tools, such as Java, Python, C++, and MySQL.

### 7.1.2 Data sampling

From the testing dataset, we randomly sample 45 seed hyperlinks that are referenced in 45 best answers (one seed hyperlink per answer) for each hyperlink citation level (“5 − 50”, “51−100”, “101−500” and “> 500”). We collect in total 180 (45x4) seed hyperlinks for this study. As the evaluation of the recommended Web resources requires certain background, we sample hyperlinks that are referenced in the discussion threads that are tagged with programming techniques (i.e., Java, Python, C++, MySQL) that participants are familiar with. Each participant is randomly assigned 30 seed hyperlinks to rate the helpfulness and diversity of recommended Web resources by the LinkLive tool.

### 7.1.3 Evaluation metrics

For each sampled seed hyperlink, we use the LinkLive tool to recommend the top-10 correlated Web resources. We implement a Web application for the participants to evaluate the recommended Web resources. For each seed hyperlink, the Web application presents the seed hyperlink, the answer in which the seed hyperlink is referenced, and the top-10 recommended Web resources for the seed hyperlink. Participants are asked to rate each recommended Web resources in terms of helpfulness and category.

Helpfulness is a 7-point likert scale (1 being least helpful to 7 being most helpful). Participants are asked to read carefully the information in the seed hyperlink, the answer in which the seed hyperlink is referenced, and the recommended Web resources to determine the level of helpfulness of the recommended Web resources for understanding the answer and/or the content of the seed hyperlink.

We predefine 5 categories for the Web resources, including official documentation (e.g., Java API documentation, jQuery library API), unofficial documentation (e.g., technical blogs, jsfiddle code examples), Q&A site (e.g., Stack Overflow, Quora), code repository (e.g., Github, Sourceforage), and encyclopedia (e.g., Wikipedia, Javapedia). Participants are asked to select one category for each recommended Web resource. They can enter “others” if they believe none of the predefined categories fit the recommended Web resources, for example dead links.

## 7.2 Perceived helpfulness

Among the 1,800 recommended Web resources (10 for each 180 seed hyperlinks), 71.8% are rated helpful (5=28.7%, 6=24.7%, or 7=18.3%), 12.05% are rated neutral (4), and 13.6% are rated as unhelpful (1=3.1%, 2=3.0% or 3=7.5%). Among the 13.6% unhelpful resources, 2.4% are dead links (e.g., due to changing URL), and most others are download links or home pages of API documentation or library, which offer little information.

For each seed hyperlink, we average the helpfulness score of the 10 recommended Web resources. Figure 8 shows the box plot of the average recommendation helpfulness score for

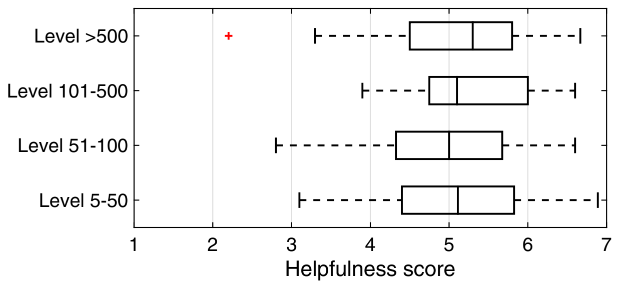


**Figure 8** Average recommendation helpfulness score at different citation levels

the 45 sampled seed hyperlinks at different levels of hyperlink citation. Observe that there is no significant difference in the mean perceived helpfulness score at different levels of hyperlink citation. The mean perceived helpfulness score is slightly above 5 (i.e., moderate helpful). There is one outlier seed hyperlink at level "> 500", i.e., http://stackoverflow.com/help/on-topic which describes questions that can be asked on Stack Overflow. For this seed hyperlink, 7 of the 10 recommended resources are about the norms or good practices to ask or answer questions on Stack Overflow. Although the recommend Web resources are very relevant to the seed hyperlink, the participant deems the recommendation as unhelpful, as they are not relevant to any specific programming issues.

Figure 9 presents the box plot of the average recommendation helpfulness score for the 30 sampled seed hyperlinks for each participant. We can see that Participant1 has least variation in his/her ratings, while Participant6 has most variation in his/her ratings. The other four participants have similar variations in their ratings. Five participants have similar mean perceived helpfulness score in their ratings (around 5, moderate helpful), while Participant1 have slight higher mean perceived helpfulness score (around 6). For Participant4, there is one outlier seed hyperlink, i.e., https://docs.python.org/2/library/struct.html. The hyperlink links to the Python module for conversions between Python values and C structs. Participant4 gives the score of 2 to 4 recommended Web resources (about audio, abstract syntax notation, and sqlite3 in Python, and the Wikipedia page for "Don't Repeat Yourself"), which makes the average recommendation helpfulness score for this seed hyperlink an outlier.

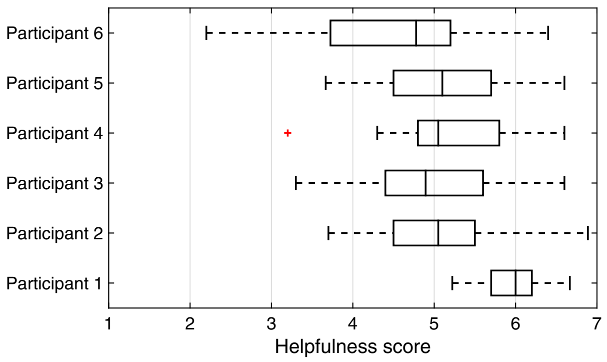


**Figure 9** Average recommendation helpfulness score for different participants

## 7.3 Perceived diversity

Figure 10 presents the percentage of different categories of the 1,800 recommended Web resources for the 180 seed hyperlinks. Among the 1,800 recommended Web resources, official documentation accounts are about 50.9%, and other types of documentation (i.e., encyclopedia (13.6%), unofficial documentation (13.2%), and Q&A site (10.4%)) account for about 37.2%. Code repository accounts for a small portion (4.5%). Others, including 44 dead links and some downloading links, account for 7.39%. Participants all rate Web resources in others category as unhelpful.

Among the 180 sampled seed hyperlinks, LinkLive recommends 1 category of Web resources for only 12.7% (23/180) seed hyperlinks, and recommends 2 or more categories of Web resources for 87.3% (157/180) seed hyperlinks. For 7 seed hyperlinks, the recommend Web resources cover all the five categories. For example, one of these 7 seed hyperlinks http://scikit-learn.org/stable/index.html links to Python *scikit-learn* machine learning library. LinkLive recommends 5 categories of Web resources, including the Wikepedia page for regression analysis, the *countmotifs.py* project on Github, Stack Overflow posts about machine learning, *scikit-learn* official documentation about feature extraction, and a professor home page for a list of handwritten, face, text and speech datasets.

***RQ6***: *The LinkLive tool helps developers discover helpful and diverse Web resources that may assist developers in understanding Stack Overflow discussions and/or content of seed hyperlinks.*

## 7.4 Threats to validity

Participants indicate that it is straightforward to select category for a recommended Web resource. However, participants (e.g., Participant1 versus Participant6) exhibit different rating behaviors for helpfulness of recommended Web resources, because the evaluation of the helpfulness is based on subjective assessment, and is affected by priori knowledge (or absence of knowledge) of the participants. A general feedback from the participants is that it is sometimes difficult to determine the helpfulness of the recommended Web resources because it depends on the information needs. For a developer looking for some basic knowledge, official documentation and encyclopedia-like information would be very helpful. But for a developer looking for a bug fix, a code example, a Stack Overflow post would be more

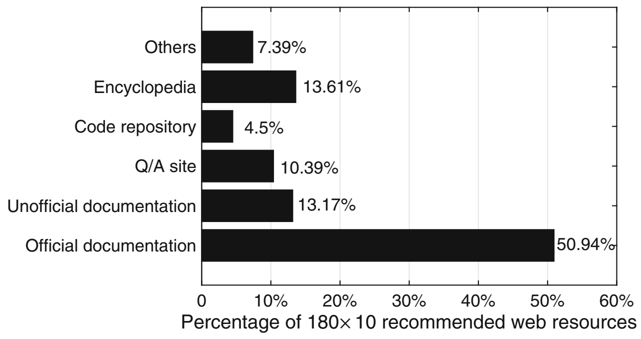


**Figure 10** Distribution of categories of recommended Web resources

helpful. That is, helpfulness is often context-sensitive. Due to these limitations, this study provides only initial evidence of the helpfulness of the LinkLive recommendation.

## 8 Discussion

In addition to hyperlinks, Stack Overflow discussions contain many other useful information, such as software-specific concepts like *machine learning*, *Observer pattern*, software tools and libraries like *Eclipse, Django*, and APIs like class names, method names. Several studies [5, 44, 73], including recent work from our team [87, 88], propose software-specific named entity recognition techniques to recognize software-specific entities in Q&A discussions and other informal documentations. Once a rich set of software-specific entities has been recognized, our item-based collaborative filtering approach can be extended to make recommendation for a variety of entities that developer may be interested in.

An innovation of our approach is that we exploit crowdsourced knowledge in Stack Overflow (i.e., hyperlink correlation patterns in this work) to support recommendation tasks beyond Q&A. This is different from existing recommendation systems [3, 35, 58, 60, 77] which mainly focus on facilitating access to online programming resources or social content in software development. The crowdsourced knowledge underlying our LinkLive recommendation could be enhanced to deliver entity-centric search services for software developers, similar to SimilarTech[13], AlternativeTo[14], or SimilarWeb[15]. Entity-centric search systems, such as serendipitous search system [47, 62], direct answers [9, 68], entity-centric recommendation [11, 42], have been actively researched in information retrieval community. Our work is an attempt along this line of research for software engineering data.

The LinkLive recommendation relies on frequent hyperlink co-occurrences in Q&A discussions. We are now investigating neural-network-based deep learning techniques (such as [74, 93]) to mine semantically related Web resources from the discussion context that they are referenced. Neural-network-based techniques have been successfully applied in many natural language processing applications to learn semantic representation of words based on the assumption that word with similar meaning tend to present in similar contexts. We hypothesize that semantically related Web resources could be referenced in similar discussion contexts, even though they may not be frequently referenced together in the same discussion threads. Co-occurrence based recommendation and neural-network based recommendation could be complementary. Furthermore, embedding hyperlinks with the discussion context could enable context-sensitive recommendation of relevant Web resources. For example, consider this sentence on Stack Overflow: *Jave use java.util.Collections.sort() implementation for ascending order*. This sentence creates a context for the link `java.util.Collections.sort()` through the surrounding words. We then can build two vocabularies: one for English words, and the other for links to API documentation. Then we use neural language models to generate low-dimensional, distributed embeddings of words [74]. Neural language models take the advantage of word order in text documents and capture both syntactic and semantic relationships between words. In the embedding space, the vectors of terms and APIs with

[13]https://www.similartech.com/

[14]http://alternativeto.net/

[15]http://www.similarweb.com/

the same intent have the shortest distance. For example, term "*ascending*" is close to API `java.util.Collections.sort()` in the embedding space.

## 9 Conclusion and future work

In this paper, we present an item-based collaborative filtering approach for recommending correlated Web resources like the ones that developers are interested in. Our approach exploits the fact that correlated Web resources have been frequently referenced in discussions on Stack Overflow. Taken in aggregation, hyperlink correlation patterns can be discovered from Stack Overflow discussions for recommendation of community-recognized, correlated Web resources. We implement a proof-of-concept tool, named LinkLive, and evaluate its recommendation quality in two studies. Our evaluation shows that LinkLive is able to recommend helpful and diverse correlated Web resources with satisfactory accuracy.

In the future, we will investigate deep learning techniques to discover semantically correlated Web resources, and extend our approach to recommendation of a variety of software-specific entities. Especially, API documentation is an important resource for programmers to learn unfamiliar APIs. Such official API documentation provides information about functionality, structure, and parameters, but not on specific issues or specific usage scenarios. On the other hand, programmers often face very specific issues which are not explicitly stated in software documentation. This mismatch leads to the overwhelming discussions online. However, waiting for answers from other programmers may take much time. As a part of the future work from this study, we are investigating this question: *can we answer a programmer's question by providing a link to the most relevant software documentation?* To answer this question, we need to bridge the semantic gap between programming questions and software documentation by taking information from multiple sources including the online disucssion.

## References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734–749 (2005)
2. Ahn, J.-W., Brusilovsky, P., Grady, J., He, D., Syn, S.Y.: Open user profiles for adaptive news systems: help or harm? In: Proceedings of WWW, pp. 11–20. ACM (2007)
3. Anderson, A., Huttenlocher, D., Kleinberg, J., Leskovec, J.: Discovering value from community activity on focused question answering sites: a case study of stack overflow. In: Proceedings of KDD, pp. 850–858 (2012)
4. Ar, Y., Bostanci, E.: A genetic algorithm solution to the collaborative filtering problem. Expert Syst. Appl. **61**, 122–128 (2016)
5. Baeza-Yates, R., Boldi, P., Chierichetti, F.: Essential Web pages are easy to find. In: Proceedings of WWW, pp. 97–107 (2015)
6. Bagheri, E., Ensan, F.: Semantic tagging and linking of software engineering social content. Autom. Softw. Eng. **23**(2), 147–190 (2016)
7. Balabanović, M., Shoham, Y.: Fab: content-based, collaborative recommendation. Commun. ACM **40**(3), 66–72 (1997)
8. Barragáns-Martínez, A.B., Costa-Montenegro, E., Burguillo, J.C., Rey-López, M., Mikic-Fonte, F.A., Peleteiro, A.: A hybrid content-based and item-based collaborative filtering approach to recommend tv programs enhanced with singular value decomposition. Inform. Sci. **180**(22), 4290–4311 (2010)

9. Bernstein, M.S., Teevan, J., Dumais, S., Liebling, D., Horvitz, E.: Direct answers for search queries in the long tail. In: Proceedings of SIGCHI, pp. 237–246 (2012)
10. Billsus, D., Pazzani, M.J.: User modeling for adaptive news access. User Model. User-Adap. Inter. **10**(2-3), 147–180 (2000)
11. Blanco, R., Cambazoglu, B.B., Mika, P., Torzec, N.: Entity recommendations in Web search. In: Proceedings of ISWC, pp. 33–48 (2013)
12. Bobadilla, J., Ortega, F., Hernando, A., Gutiérrez, A.: Recommender systems survey. Knowl.-Based Syst. **46**, 109–132 (2013)
13. Brandt, J., Guo, P.J., Lewenstein, J., Dontcheva, M., Klemmer, S.R.: Two studies of opportunistic programming: interleaving Web foraging, learning, and writing code. In: Proceedings of CHI, pp. 1589–1598 (2009)
14. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of UAI, pp. 43–52 (1998)
15. Bretherton, I.: Attachment theory: retrospect and prospect. Monographs Soc. Res. Child Develop. **50**(1/2), 3–35 (1985)
16. Broder, A.: A taxonomy of Web search. In: Proceedings of SIGIR, vol. 36, pp. 3–10 (2002)
17. Celma, Ò., Serra, X.: Foafing the music: bridging the semantic gap in music recommendation. Web Semant. Sci. Serv. Agents World Wide Web **6**(4), 250–256 (2008)
18. Chowdhury, G.G.: Introduction to Modern Information Retrieval. Facet Publishing, London (2010)
19. Cooley, R., Mobasher, B., Srivastava, J.: Web mining: information and pattern discovery on the World Wide Web. In: Proceedings of ICTAI, pp. 558–567 (1997)
20. Das, A.S., Datar, M., Garg, A., Rajaram, S.: Google news personalization: scalable online collaborative filtering. In: Proceedings of WWW, pp. 271–280 (2007)
21. Degemmis, M., Lops, P., Semeraro, G.: A content-collaborative recommender that exploits wordnet-based user profiles for neighborhood formation. Proceedings of UMAP **17**(3), 217–255 (2007)
22. Dou, Z., Song, R., Nie, J.-Y., Wen, J.-R.: Using anchor texts with their hyperlink structure for Web search. In: Proceedings of SIGIR, pp. 227–234 (2009)
23. Gomez, C., Cleary, B., Singer, L.: A study of innovation diffusion through link sharing on stack overflow. In: Proceedings of MSR, pp. 81–84 (2013)
24. Gong, Y., Zhang, Q.: Hashtag recommendation using attention-based convolutional neural network. In: Proceedings of IJCAI, pp. 2782–2788. AAAI Press (2016)
25. Grbovic, M., Radosavljevic, V., Djuric, N., Bhamidipati, N., Savla, J., Bhagwan, V., Sharp, D.: E-commerce in your inbox: product recommendations at scale. In: Proceedings of KDD, pp. 1809–1818 (2015)
26. Guo, H., Tang, R., Ye, Y., Li, Z., Deepfm, X.H.E.: A factorization-machine based neural network for ctr prediction. In: Proceedings of IJCAI (2017)
27. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.-S.: Neural collaborative filtering. In: Proceedings of WWW, pp. 173–182. International World Wide Web Conferences Steering Committee (2017)
28. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. In: Proceedings of ICLR (2015)
29. Holmes, R., Walker, R.J., Murphy, G.C.: Strathcona example recommendation tool. In: Proceedings of Software Engineering Notes, vol. 30, pp. 237–240 (2005)
30. Karypis, G.: Evaluation of item-based top-N recommendation algorithms. In: Proceedings of CIKM, pp. 247–254. ACM, New York (2001)
31. Klemmer, S.R., Sinha, A.K., Chen, J., Landay, J.A., Aboobaker, N., Wang, A.: Suede: a wizard of oz prototyping tool for speech user interfaces. In: Proceedings of UIST, pp. 1–10. ACM (2000)
32. Krompaß, D., Baier, S., Tresp, V.: Type-constrained representation learning in knowledge graphs. In: Proceedings of ISWC, pp. 640–655. Springer (2015)
33. Levandoski, J.J., Sarwat, M., Eldawy, A., Mokbel, M.F.: Lars: a location-aware recommender system. In: Proceedings of ICDE, pp. 450–461 (2012)
34. Li, Y., Lu, L., Xuefeng, L.: A hybrid collaborative filtering method for multiple-interests and multiple-content recommendation in e-commerce. Expert Syst. Appl. **28**(1), 67–77 (2005)
35. Li, H., Zhao, X., Xing, Z., Bao, L., Peng, X., Gao, D., Zhao, W.: amassist: in-ide ambient search of online programming resources. In: Proceedings of SANER, pp. 390–398 (2015)
36. Li, J., Bao, L., Xing, Z., Wang, X., Zhou, B.: Bpminer: mining developers' behavior patterns from screen-captured task videos. In: Proceedings of SAC, pp. 1371–1377. ACM (2016)
37. Li, J., Xing, Z., Ye, D., Zhao, X.: From discussion to wisdom: Web resource recommendation for hyperlinks in stack overflow. In: Proceedings of SAC, pp. 1127–1133. ACM (2016)
38. Li, J., Sun, A., Xing, Z.: Learning to answer programming questions with software documentation through social context embedding. Inform. Sci. **448-449**, 36–52 (2018)

39. Li, J., Sun, A., Xing, Z., Han, L.: Api caveat explorer: surfacing negative usages from practice. In: Proceedings of SIGIR, pp. 1293–1296 (2018). https://doi.org/10.1145/3209978.3210170
40. Li, J., Xing, Z., Kabir, A.: Leveraging official content and social context to recommend software documentation. IEEE Transactions on Services Computing. IEEE Early Access (2018), https://doi.org/10.1109/TSC.2018.2812729
41. Lilliefors, H.W.: On the kolmogorov-smirnov test for normality with mean and variance unknown. J. Am. Stat. Assoc. **62**(318), 399–402 (1967)
42. Lin, T., Pantel, P., Gamon, M., Kannan, A., Fuxman, A.: Active objects: actions for entity-centric search. In: Proceedings of WWW, pp. 589–598 (2012)
43. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. **7**(1), 76–80 (2003)
44. Liu, T.-Y.: Learning to rank for information retrieval. Found. Trend Inform. Retrieval **3**(3), 225–331 (2009)
45. Luo, X., Xia, Y., Zhu, Q.: Incremental collaborative filtering recommender based on regularized matrix factorization. Know.-Based Syst. **27**, 271–280 (2012)
46. Magnini, B., Strapparava, C.: Improving user modelling with content-based techniques. In: Proceedings of the International Conference on User Modeling, pp. 74–83. Springer (2001)
47. Marchionini, G.: Exploratory search: from finding to understanding. Commun. ACM **49**(4), 41–46 (2006)
48. McLaughlin, M.R., Herlocker, J.L.: A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In: Proceedings of SIGIR, pp. 329–336. ACM (2004)
49. Melville, P., Mooney, R.J., Nagarajan, R.: Content-boosted collaborative filtering for improved recommendations. In: Eighteenth National Conference on Artificial Intelligence, pp. 187–192. American Association for Artificial Intelligence, Menlo Park (2002)
50. Miliaraki, I., Blanco, R., Lalmas, M.: From Selena Gomez to Marlon Brando: understanding explorative entity search. In: Proceedings of WWW, pp. 765–775 (2015)
51. Mooney, R.J., Roy, L.: Content-based book recommending using learning for text categorization. In: Proceedings of the Fifth ACM Conference on Digital Libraries, pp. 195–204. ACM (2000)
52. Nguyen, P.T., Tomeo, P., Di Noia, T., Di Sciascio, E.: Content-based recommendations via dbpedia and freebase: a case study in the music domain. In: Proceedings of ISWC, pp. 605–621. Springer (2015)
53. Nicholas, I.S.C., Nicholas, C.K.: Combining content and collaboration in text filtering. In: Proceedings of IJCAI, pp. 86–91 (1999)
54. Nilashi, M., Ibrahim, O.B., Ithnin, N., Zakaria, R.: A multi-criteria recommendation system using dimensionality reduction and neuro-fuzzy techniques. Soft. Comput. **19**(11), 3173–3207 (2015)
55. Nilashi, M., Ibrahim, O., Bagherifard, K.: A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques. Expert Syst. Appl. **92**, 507–520 (2018)
56. Park, M.-H., Hong, J.-H., Cho, S.-B.: Location-based recommendation system using Bayesian user's preference model in mobile devices. In: Ubiquitous Intelligence and Computing, pp. 1130–1139. Springer (2007)
57. Park, S.-T., Pennock, D.M.: Applying collaborative filtering techniques to movie search for better ranking and browsing. In: Proceedings of KDD, pp. 550–559 (2007)
58. Parnin, C., Treude, C., Grammel, L., Storey, M.-A.: Crowd documentation: exploring the Coverage and the Dynamics of Api Discussions on Stack Overflow. Georgia Institute of Technology, Tech. Rep (2012)
59. Pazzani, M., Billsus, D.: Learning and revising user profiles: the identification of interesting Web sites. Mach. Learn. **27**(3), 313–331 (1997)
60. Ponzanelli, L., Bacchelli, A., Lanza. M.: Seahawk: stack overflow in the ide. In: Proceedings of ICSE, pp. 1295–1298 (2013)
61. Ponzanelli, L., Bavota, G., Di Penta, M., Oliveto, R., Lanza, M.: Prompter: a self-confident recommender system. In: Proceedings of ICSME, pp. 577–580 (2014)
62. Sakai, T., Nogami, K.: Serendipitous search via wikipedia: a query log analysis. In: Proceedings of SIGIR, pp. 780–781 (2009)
63. Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted Boltzmann machines for collaborative filtering. In: Proceedings of ICML, pp. 791–798. ACM, New York (2007)
64. San Pedro, J., Karatzoglou, A.: Question recommendation for collaborative question answering systems with rankslda. In: Proceedings of RecSys, pp. 193–200 (2014)
65. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of WWW, pp. 285–295 (2001)
66. Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M.: Methods and metrics for cold-start recommendations. In: Proceedings of SIGIR, pp. 253–260. ACM, New York (2002)

67. Sedhain, S., Menon, A.K., Sanner, S., Xie, L.: Autorec: autoencoders meet collaborative filtering. In: Proceedings of WWW, pp. 111–112. ACM (2015)
68. Seebach, C.: Searching for answers–knowledge exchange through social media in organizations. In: Proceedings of HICSS, pp. 3908–3917 (2012)
69. Sheth, B., Maes, P.: Evolving agents for personalized information filtering. In: Proceedings of the Ninth Conference on Artificial Intelligence for Applications, pp. 345–352. IEEE (1993)
70. Shinde, S.K., Kulkarni, U.: Hybrid personalized recommender system using centering-bunching based clustering algorithm. Expert Syst. Appl. **39**(1), 1381–1387 (2012)
71. Singhal, A. et al.: Modern information retrieval: a brief overview. IEEE Data Eng. Bull. **24**(4), 35–43 (2001)
72. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Advan. Artif. Intell. **2009**, 4 (2009)
73. Subramanian, S., Inozemtseva, L., Holmes, R.: Live api documentation. In: Proceedings of ICSE, pp. 643–652 (2014)
74. Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., Qin, B.: Learning sentiment-specific word embedding for twitter sentiment classification. In: Proceedings of ACL, vol. 1, pp. 1555–1565 (2014)
75. Tenopir, C., King, D.W.: Communication Patterns of Engineers. Wiley, New York (2004)
76. Tran, T., Phung, D., Venkatesh, S.: Collaborative filtering via sparse markov random fields. Inform. Sci. **369**, 221–237 (2016)
77. Treude, C., Barzilay, O., Storey, M.-A.: How do programmers ask and answer questions on the Web?: Nier track. In: Proceedings of ICSE, pp. 804–807 (2011)
78. Treude, C., Robillard, M.: Augmenting api documentation with insights from stack overflow. In: Proceedings of ICSE
79. Wang, S., Lo, D., Jiang, L.: An empirical study on developer interactions in stackoverflow. In: Proceedings of SAC, pp. 1019–1024 (2013)
80. Wang, S., Lo, D., Vasilescu, B., Serebrenik, A.: Entagrec: an enhanced tag recommendation system for software information sites. In: Proceedings of ICSME, pp. 291–300 (2014)
81. Wang, F.-H., Jian, S.-Y.: An effective content-based recommendation method for Web browsing based on keyword context matching. J. Inform. Electron. **1**(2), 49–59 (2006)
82. Wang, J., De Vries, A.P., Reinders, M.J.: Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In: Proceedings of SIGIR, pp. 501–508 (2006)
83. Wang, J., Yu, L., Zhang, W., Gong, Y., Xu, Y., Wang, B., Zhang, P., Zhang, D.: Irgan: a minimax game for unifying generative and discriminative information retrieval models. In: Proceedings of SIGIR, pp. 515–524. ACM (2017)
84. White, R.W., Roth, R.A.: Exploratory search: beyond the query-response paradigm. Synth. Lect. Inform. Concepts Retr. Serv. **1**(1), 1–98 (2009)
85. Yager, R.R.: Fuzzy logic methods in recommender systems. Fuzzy Sets Syst. **136**(2), 133–149 (2003)
86. Ye, M., Yin, P., Lee, W.-C., Lee, D.-L.: Exploiting geographical influence for collaborative point-of-interest recommendation. In: Proceedings of SIGIR, pp. 325–334. ACM, New York (2011)
87. Ye, D., Xing, Z., Foo, C.Y., Ang, Z.Q., Li, J., Kapre, N.: Software-specific named entity recognition in software engineering social content. In: Proceedings of SANER, vol. 1, pp. 90–101. IEEE (2016)
88. Ye, D., Xing, Z., Li, J., Kapre, N.: Software-specific part-of-speech tagging: an experimental study on stack overflow. In: Proceedings of SAC, pp. 1378–1385. ACM (2016)
89. Yu, X., Ma, H., Hsu, B.-J.P., Han, J.: On building entity recommender systems using user click log and freebase knowledge. In: Proceedings of WSDM, pp. 263–272. ACM (2014)
90. Yuan, Q., Cong, G., Sun, A.: Graph-based point-of-interest recommendation with geographical and temporal influences. In: Proceedings of CIKM, pp. 659–668 (2014)
91. Zagalsky, A., Barzilay, O., Yehudai, A.: Example overflow: using social media for code recommendation. In: Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering, pp. 38–42 (2012)
92. Zhang, S., Yao, L., Sun, A.: Deep learning based recommender system: a survey and new perspectives. arXiv:1707.07435 (2017)
93. Zhou, G., He, T., Zhao, J., Hu, P.: Learning continuous word embedding with metadata for question retrieval in community question answering. In: Proceedings of ACL, pp. 250–259 (2015)
94. Zimmermann, T., Dallmeier, V., Halachev, K., Zeller, A.: erose: guiding programmers in eclipse. In: Proceedings of SPLASH, pp. 186–187 (2005)