The Journal of Systems and Software 156 (2019) 283–299

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Modeling stack overflow tags and topics as a hierarchy of concepts

Hui Chen[a,*], John Coogle[b], Kostadin Damevski[b]

[a] Department of Computer and Information Science, Brooklyn College of the City University of New York, Brooklyn, NY 11210 United States
[b] Department of Computer Science, Virginia Commonwealth University, Richmond, Virginia 23284 United States

ARTICLE INFO

Article history:
Received 22 August 2018
Revised 13 July 2019
Accepted 15 July 2019
Available online 15 July 2019

Keywords:
Concept hierarchy
Hierarchical topic model
Stack overflow
Tag synonym identification
Tag prediction
Entropy-based search evaluation

ABSTRACT

Developers rely on online Q&A forums to look up technical solutions, to pose questions on implementation problems, and to enhance their community profile by contributing answers. Many popular developer communication platforms, such as the Stack Overflow Q&A forum, require threads of discussion to be tagged by their contributors for easier lookup in both asking and answering questions. In this paper, we propose to leverage Stack Overflow's tags to create a hierarchical organization of concepts discussed on this platform. The resulting concept hierarchy couples tags with a model of their relevancy to prospective questions and answers. For this purpose, we configure and apply a supervised multi-label hierarchical topic model to Stack Overflow questions and demonstrate the quality of the model in several ways: by identifying tag synonyms, by tagging previously unseen Stack Overflow posts, and by exploring how the hierarchy could aid exploratory searches of the corpus. The results suggest that when traversing the inferred hierarchical concept model of Stack Overflow the questions become more specific as one explores down the hierarchy and more diverse as one jumps to different branches. The results also indicate that the model is an improvement over the baseline for the detection of tag synonyms and that the model could enhance existing ensemble methods for suggesting tags for new questions. The paper indicates that the concept hierarchy as a modeling imperative can create a useful representation of the Stack Overflow corpus. This hierarchy can be in turn integrated into development tools which rely on information retrieval and natural language processing, and thereby help developers more efficiently navigate crowdsourced online documentation.

1. Introduction

A modern trend in software development is that developers increasingly rely on Web resources to acquire knowledge, seek snippets of code for reuse, disseminate ideas, and establish professional standing and reputation (Storey et al., 2017). For instance, a successful resource that is integrated in many developers' daily work is the Stack Overflow Q&A forum, where technically sound answers contribute to a developer's professional standing, while helping to solve many other developers' software development problems. This community-driven process has led to the accumulation of significant amount of knowledge about software development on Stack Overflow. However, the growing number of posts on this platform make it challenging for developers to quickly find relevant posts to inform their work or to contribute answers to, which some consider as worrisome to sustaining the platform (Srba and Bielikova, 2016). To address this problem, we aim at proposing automatic means to effectively group and relate the vast content on Stack Overflow in order to improve the rapid retrieval of relevant content on this platform. This can in turn become a foundation to build effective developer-relevant search tools. For instance, a number of recommendation tools targeting Stack Overflow have already been proposed with the aim of improving developer productivity by integrating relevant information from Stack Overflow into the IDE (Ponzanelli et al., 2014; Nguyen et al., 2016; Campbell and Treude, 2017; Greco et al., 2018).

To achieve this objective, we leverage Stack Overflow's use of tags. Tags aid exploratory information retrieval, which occurs in scenarios when a developer is unaware of the specific item she is searching (Baeza-Yates and Ribeiro-Neto, 1999; Shneiderman, 2003). This is contrast to known-item (or navigational) search, which occurs when developers are searching for specific posts (e.g. using an error message as a query). Each Stack Overflow question is tagged by the author with a small set of tags (minimum of one and maximum of five[1]), usually corresponding to the technology domain that the question belongs to (e.g. ANDROID, IOS, JAVA-THREADS). Organization via tags, labels, categories etc. is commonly

* Corresponding author.
E-mail addresses: huichen@ieee.org, hui.chen@brooklyn.cuny.edu (H. Chen).

[1] https://stackoverflow.com/help/tagging.

https://doi.org/10.1016/j.jss.2019.07.033
0164-1212/

used in our day-to-day lives to make finding items easier. For instance, when we walk into a library, we look for a relevant section; and when we visit a grocery store, we go down a specific isle. Different from these analogies, in Stack Overflow, a single question may belong to several tags, which is intuitive as questions often do not fit only a single category (e.g. a question on *What's the best way to share data between activities?* can belong to both ANDROID and INTENT). However, this does not suggest that tags themselves should not form a relationship between each other. At present, the Stack Overflow tags assigned to a question are independent of each other.

One particular view of the relationship between tags is hierarchical, i.e., given a more generic concept, such as, ANDROID, we may be interested in examining a more specific concept, such as, INTENT. A tag hierarchy can help a question author decide which set of tags are more appropriate, as often authors that want their question to be answered rapidly select both more generic and more specific tags (Treude et al., 2011). Conversely, when a developer uses tags to search for information on Stack Overflow, she or he can expand the list of returned posts by exploring either more specific or generic tags. Stack Overflow's answer contributors commonly search for new questions using a set of tags corresponding to, often related, technologies where they possess expertise. Empirical evidence from other fields also suggests that hierarchical tags help both contributors and readers as they perform exploratory searches (Teevan et al., 2004; Kairam et al., 2015; Athukorala et al., 2014). In addition, a hierarchical organization of tags helps exploratory information seekers to branch to different content from the initial set of explored results, which may be based on an imperfect query. By branching away from the current path of tags, the diversity of examined posts increases, which can be beneficial in viewing a broader set of results (Angel and Koudas, 2011).

In this paper, we propose to automatically construct a hierarchical tagged concept model from Stack Overflow posts. Exemplifying the ideas discussed above, we identify a modeling technique called the Label-to-Hierarchy model (L2H) (Nguyen et al., 2014). This model allows establishing a mapping between tags and associated topics extracted from the Stack Overflow post content. Such Stack Overflow tag-topics (or concepts) exist in a hierarchy and express a discrete probability distribution of the constituent terms occurring in the Stack Overflow corpus. We examine how this model can aid both question writers, by predicting tags for posts, and information seekers, by organizing tags and determining tags synonyms, in Stack Overflow. Specifically, the contributions of this paper are:

- approach for applying the L2H model to organize Stack Overflow posts and tags into a hierarchical concept model;
- evaluation of the concept models for tag synonym prediction;
- examination of the concept model for predicting the tags of an unseen post;
- novel technique for evaluating the concept hierarchy based on entropy for its potential to focus or broaden retrieved document sets; and
- evaluation of the L2H model based on the novel entropy-based metrics.

The rest of the paper is organized as follows. We begin with background on hierarchical models for textual data, including the model used in this paper, in Section 2, followed by the characteristics of the Stack Overflow dataset and the challenges to apply the model to the dataset in Section 3. We describe our extensive experimental plan along several dimensions (tag synonyms, tag prediction, and navigational search) in Section 4, while the results from this evaluation are presented in Section 5. Finally, in Section 6 we present a concise survey of the related work, and in Section 7, we conclude the paper and list the future work.

2. Background

Relative to traditional technical documentation, such as, official API documentation and reference books, crowd-sourced online documentation is typically uncurated and therefore less organized. In this paper, we propose a concept[2] hierarchy for tagged external documentation sources, such as Stack Overflow. With this purpose in mind, in this section, we review and compare relevant hierarchical clustering and hierarchical topic modeling methods. Following this, we provide background on hierarchical concept modeling, the primary modeling technique we employ for the task of building concept hierarchies for Stack Overflow data in this paper.

2.1. Hierarchical clustering and topic modeling

Hierarchical clustering algorithms are to extract a hierarchy of common data points, and have been frequently applied to clustering documents (Steinbach et al., 2000; Xu and Wunsch II, 2005). These algorithms divide documents into clusters in either an agglomerative fashion, i.e., merging smaller clusters to a larger one bottom-up, or a divisive fashion, i.e., dividing a larger cluster to smaller ones top-down. At completion, both types of algorithms yield a hierarchy of clusters.

Most hierarchical clustering algorithms have a characteristic that is undesirable for building a concept hierarchy. That is, these algorithms assign each document to one cluster path, from the root of the hierarchy to the leaf of the hierarchy, i.e., if a cluster corresponds to a concept, the document can only be associated with a single path of concepts, from more generic concepts to more specific concepts. However, a document cannot be associated to two disjoint concepts.

Considering that Stack Overflow documents, as well as other software documentation types, are tagged, we would also like to leverage the tags as human-recognizable expression of concepts. Hierarchical clustering algorithms typically do not take tags into account. To address this concern, one may also consider hierarchical classification algorithms where tags are the labels into which documents are classified (Dumais and Chen, 2000; Silla and Freitas, 2011).

Recently, a number of hierarchical probabilistic topic models have appeared in the literature. Although they are very popular for processing natural language texts (Blei et al., 2003), probabilistic topic models are applicable to any discrete data, including genome data and IDE interaction traces (Pritchard et al., 2000; Damevski et al., 2018). These probabilistic topic models are commonly referred to as mixed membership generative models (Erosheva et al., 2004), because a document is assigned to a set of groups probabilistically, i.e., a *mixture* of the set of group memberships for the document. The groups themselves are referred to as *topics*. By modeling group membership probabilistically, these models avoid the issue of a document only associating to a single path of concepts, present in hierarchical clustering.

Similar to the clustering algorithms whose learning of clusters from data is unsupervised, many topic models are unsupervised. Among these unsupervised topic models is Latent Dirichlet Allocation (LDA) (Blei et al., 2003). LDA is a flat topic model, in which the topics are independent and there is no structural relationship among the discovered topics. Hierarchical topic models are developed to overcome two challenging issues in these flat topic models. First, when using flat topic models, it is difficult or at least

[2] We use the term concept in this paper to refer to high-level development-related abstractions (e.g. techniques, libraries, languages, operating systems).

computationally expensive to discover the number of topics that should be modeled in a document collection. Second, since there is only a rudimentary relationship among topics, the meaning of the topics is difficult to interpret, in particular, when multiple topics look alike based on their probability distributions. Representative hierarchical topic models include Hierarchical Latent Dirichlet Allocation (HLDA) and Nested Hierarchical Dirichlet Process (Blei et al., 2010; Paisley et al., 2015). Hierarchical topic models also overcome the issue that a document is assigned only a path of concepts in the hierarchy; however, the topics still need to be labeled with human interpretable tags.

A set of tagged (or labeled) and partially tagged (or labeled) topic models have been developed to address the challenge (Ramage et al., 2011; Nguyen et al., 2014). Ramage et al. associate a document with multiple tags and each tag can be considered as an interpretable concept for the topic associated with the tag (Ramage et al., 2011), while Nguyen et al. not only associate a document with multiple interpretable tags, but the tags also form a tree-like hierarchy (Nguyen et al., 2014). Considering the above, in this paper we select Nguyen et al.'s L2H topic model Nguyen et al. (2014) to extract a concept hierarchy from a tagged corpus like Stack Overflow.

The Stack Overflow community, on its Meta Stack Exchange site, has discussed the idea of organizing tags in a hierarchy as a search aid (Stack Exchange, Inc., 2018b). While the conclusion of the discussion is unclear, with arguments both for and against such an organization, it is clear that a structure where tags belong to multiple groupings is preferable.

2.2. Hierarchical concept model

In this section, we formally describe hierarchical concept (tag-topic) models, a class of probabilistic modeling techniques suited for tagged textual corpora. We start with a set of definitions; the most basic unit of discrete data are *words*, and we denote a word as w. A sequence of words forms a *document*, i.e., $\vec{d} = (w_{d,1}, w_{d,2}, \ldots, w_{d,N_d})$ where N_d is the number of words in the document and $w_{d,i}$, $1 \le i \le N_d$ is a word in the document. A collection of documents form a corpus, i.e., $\mathbb{D} = \{\vec{d_1}, \vec{d_2}, \ldots, \vec{d_D}\}$ where $D = |\mathbb{D}|$ is the number of documents in the corpus, and $\vec{d_i}$, $1 \le i \le D$ is a document in the corpus. The set of all the unique words in the corpus is referred to as the vocabulary of the corpus. Here we refer to it as *word vocabulary* and denote it as $\mathbb{V} = \{w_1, w_2, \ldots, w_V\}$ where $V = |\mathbb{V}|$ is the number of unique words in the vocabulary. Each document is associated with one or more tags (or labels). The set of all the unique tags is the vocabulary of the tags of the corpus. Here we refer to it as *tag vocabulary* and denote it as $\mathbb{L} = \{l_1, l_2, \ldots, l_L\}$ where $L = |\mathbb{L}|$ is the number of unique tags. Specifically, we leverage the *Label-to-Hierarchy* model (L2H) (Nguyen et al., 2014), whose purpose is to build a concept hierarchy from a set of documents, where each document contains multiple tags.

Formally, a concept c is a pair $c = (l, \phi)$, where $l \in \mathbb{L}$ is a tag and $\phi \in \lneqq$ is a topic. A topic is defined as a discrete probability distribution over words in the fixed vocabulary $\mathbb{V}$, and can be defined by its probability mass function, i.e., for topic ϕ, $P_\phi(w = w_i) = P_{\phi,i}$, $1 \le i \le |\mathbb{V}|$ and $\sum_{i=1}^{|\mathbb{V}|} P_{\phi,i} = 1$. The model assumes that the tags and the topics have a one-to-one mapping, i.e., the number of topics $\Phi = |\lneqq|$ is identical to the number of unique tags L. As input, L2H requires a set of documents, each of which is assigned a list of tags, and outputs a tree-like hierarchy of concepts, representing relationships among the concepts. In L2H's tree-like structure, a concept on the parent node is more generic than its children while a concept in a child node more specific.

Initially, L2H constructs an weighted directed graph $\mathcal{G} = (\mathbb{V}, \mathbb{E})$, where vertices $\mathbb{V}$ are the set of concepts $\mathbb{C}$, i.e., $\mathbb{V} = \mathbb{C}$ while edges $\mathbb{E}$ are association relationships between concepts. Two concepts $c_i = (l_i, \phi_i)$ and $c_j = (l_j, \phi_j)$ are connected when there is an edge between them, and a concept c may have many directly connected neighbors. The edge weights of the neighbors to a concept c represent the strength of their relationship to the concept. In order to build a concept hierarchy from this graph, we consider the weights between two concepts c_i and c_j as latent variables, and in particular, the weight of edge e_{ij}, the edge pointing from concept c_i to c_j, as representative of the proportion of concept c_j in concept c_i. For instance, from the group of concepts ANDROID, APK (Android Package Kit), and ANDROID-STUDIO (IDE for building Android apps), we consider APK and ANDROID-STUDIO to be more specific concepts than ANDROID, while APK and ANDROID-STUDIO are two parallel, but unrelated, concepts. When we consider all the input documents that are associated with the concept ANDROID, the portion of the documents that are associated with APK would indicate the weight of edge $e_{android,apk}$, while the portion with ANDROID-STUDIO the weight $e_{android,android-studio}$. Since the edge weights are latent variables, we can only estimate them by observing a set of tagged documents where tags correspond to a set of concepts. Specifically, the association exists if and only if at least one document is tagged by the two tags in their respective concepts (Nguyen et al., 2014); furthermore, the *initial* weight of an edge, e.g., e_{ij} is estimated as the proportion of the documents tagged by both of the tags l_i and l_j among those documents tagged by tag l_i. Before inference of the L2H model, we build an initial graph $\mathcal{G}$ using the set of tagged documents.

A salient feature of L2H is that it does not assume that a document is tagged exhaustively with the concepts it is *innately* associated with, instead it learns the relationship from the entire collection of the documents. Obviously, this depends on the input dataset — a gap exists when one forgets to tag a document with a relevant concept using the concept's respective tag; however, when the dataset is sufficiently large, it is likely in the dataset there exist documents that others have tagged with the missing concept, and the gap can be filled.

The learning algorithm in effect determines the probabilistic distribution over the latent variables that include the topics, the probabilistic distributions over the words of each topic, and the tree structure, i.e., the topology of the tree and the weight of the edges. In this paper, we use the Markov chain Monte Carlo (MCMC) inference algorithm (Andrieu et al., 2003). Note that in the inference algorithm, a background concept is introduced, as the root of the concept tree with artificial edges to all of the vertices. To understand the latent and observable variables, how they are related, and what constraints they are subject to, we can examine the graphical structure of the model, and how a document may be generated from the model, since the L2H model is in effect a probabilistic graphical model for documents and commonly understood by describing its generative process. We present the graphical representation of the L2H model in Fig. 1.

Shaded nodes in Fig. 1 represent the observed data, the D documents and the tag set of each document (e.g., $\mathbb{L}_d$ for document d). From these two, using frequencies of tag occurrence we form a concept graph $\mathcal{G}$ as described above, which we can then also considered observed. The hyperparameters α, $\gamma = (\gamma_1, \gamma_2)$, and β represent prior belief or knowledge on how the generative process should be constrained. The rest are latent variables, which are learned. The training algorithm determines the joint probability distribution of the parameters and the variables given the observed data.

The generative process of a topic model typically consists of two major steps. The first step generates a set of K topics, while the second generates a set of D documents using the K topics. A document exhibits the K topics with different proportions. For each word in the document, the algorithm selects a topic out of the K

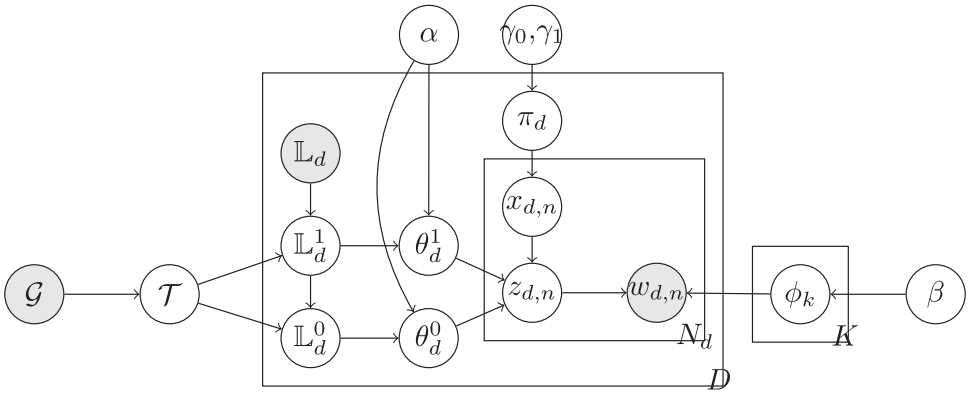


Fig. 1. Graphical representation of the L2H model. The model can be viewed as an extension to the Latent Dirichlet Allocation (LDA). The right hand side (starting at $w_{d,n}$) of the graph illustrates the process to generate K topics, identical to the graphical representation of LDA. The difference L2H from LDA is the selection of a topic out of the K topics for a word in a document, as illustrated in the left hand side of the graph (starting at $z_{d,n}$). In L2H, a topic is selected based on a tree-like hierarchical prior and tags of the document and the document collection, while in LDA it is selected from a distribution drawn from a Dirichlet prior.

topics, and generates the word by sampling the selected topic, a discrete probability distribution. The difference between L2H and flat topic models like LDA is at the second major step, when a topic is being selected for each word in a document, as shown in Fig. 1.

The process to select the topic indicator $z_{d,n}$ starts at sampling a spanning tree $\mathcal{T}$ from $\mathcal{G}$. Note that by introducing a background node, we ensure that a spanning tree exists in graph $\mathcal{G}$. How we sample a specific spanning tree from graph $\mathcal{G}$ affects computational efficiency since the graph is more likely to have many spanning trees. In the inference algorithm described in Nguyen et al. (2014), the tree is initialized as the *maximum spanning tree*. The structure of tree is updated during the training process. This choice is made to increase computational efficiency. The K tags are divided into two subsets for each document. Given document d's tags $\mathbb{L}_d$ and the concept tree $\mathcal{T}$, we divide the K tags into two subsets $\mathbb{L}_d^1$ and $\mathbb{L}_d^0$. The former are tags in $\mathbb{L}_d$ and the tags' ancestors in tree $\mathbb{T}$, and the later is the complement of $\mathbb{L}_d^1$. We form $\mathbb{L}_d^1$ in such a way because: (1) one may not exhaustively tag a document with all concepts it is associated with; and (2) some systems have constraints on the number of tags a document can be assigned to, e.g., in Stack Overflow, one can only tag at most 5 tags. This also implies that we divide the K topics into two subsets. In order to determine which set to choose from to select a topic, we introduce a hyperparameter $\gamma = (\gamma_1, \gamma_2)$, which parameterizes a *Beta* distribution, a draw from which is a probability called the switching probability π_d. The parameters should be chosen in a way that θ_d^1 would be selected with higher probability than θ_d^0, topic proportions of the two subset of topics. Drawing from the Bernoulli distribution parameterized by π_d, we get $x_{d,n}$ and use it to select one of the two subsets of topic proportions, from which we know which of the K topic a word is sampled from.

3. Building a concept hierarchy in stack overflow

Documents in development related communication channels (e.g. Q&A, tutorials, blogs posts) have a few common characteristics, differentiating them from typical natural language texts. First, the documents commonly contain a mix of source code and natural language text, which makes them a unique source for mining and analysis (Chatterjee et al., 2017; Binkley et al., 2017; Rigby and Robillard, 2013). Second, documents on many of these platforms contain noise, which arises from the informal nature of the communication. It is common that uncorrected misspelling are present, or for documents to be mislabeled or poorly organized. Specific to Q&A style documents, question writers may not clearly express their information request, while the provided answers may also be of ranging quality, with respect to their level of detail, ease of understanding, and even correctness. For analysis or mining of these online software development texts, numerous choices on how to pre-process the data are typically necessary, while particular analysis and modeling approaches can be very sensitive to these choices. Probabilistic topic models, such as the one we have used in this paper, are particularly capable of dealing with noisy discrete data, relying on probability to model uncertainty.

In the following, we describe different choices for data preprocessing and model construction. An overview of the specific selections we used is provided in Table 1.

3.1. Data preprocessing

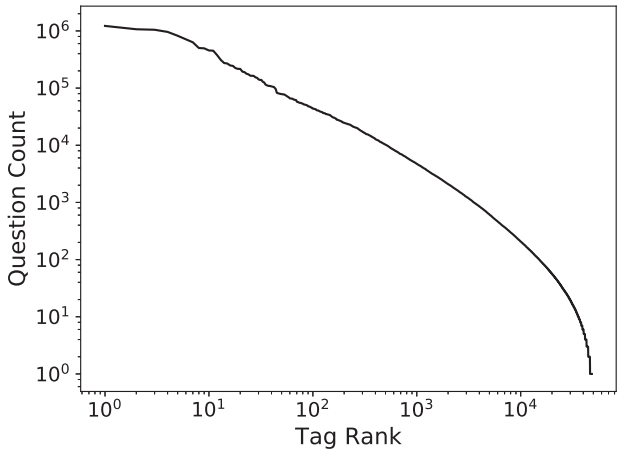
Data preprocessing is extremely important in applying complex models as the computational time to learn a model from a large dataset like Stack Overflow is nontrivial, even on modern hardware. More importantly, preprocessing should reflect the intended purpose of the model. For instance, if the objective is to help a user to tag questions or to help a user to browse and search questions, it may be more appropriate to consider only the questions when constructing a model. Here, we describe some of the important data preprocessing choices in using a topic model on Stack Overflow, while the next section describes the concrete pipeline we utilized in building our model.

Document Granularity. First, we need to indicate what exactly constitutes a document in Stack Overflow. There are a few choices, such as, (1) treating each question or answer as a document, (2) treating a thread including both the question and corresponding answers as a document, and (3) treating the question, answers, and surrounding comments on both as a document. Because on Stack Overflow tags are primarily associated with a question, we only select the questions and ignore all answers and comments.

Selecting Documents. It is computational costly and not always necessary to include all the posts present in Stack Overflow in constructing the model. In addition, the dataset has some ancillary attributes that we can consider in selecting a subset of documents from the dataset. Posts can be selected based on time stamps, on number of views, on votes of posts and questions, on post histories, and on the combinations of these attributes. In building a robust concept hierarchy the distribution of tags is the most relevant quantity. Documents containing extremely rare tags are unlikely to meaningfully contribute to our model.

Table 1
Parameters for data preprocessing and model building.

Data Source	Data Preprocessing *(by examining selected documents in input dataset)*	Data Filtering *(by examining tag and word frequency distributions in Stack Overflow)*	Model Hyperparameters *(by optimization using a separate held out sample of 3mos. of Stack Overflow data)*
Stack Overflow posts between January 1, 2016 and March 13, 2017	• questions only; • select tags. Use frequent tags as selection criteria, i.e., we filter out tags with which less than 250 questions are associated; • select questions. Maximally 5000 random questions are selected for each tag;	• remove words that appear in less than 300 questions, and those that appear in 40% or more questions	$\alpha = 10$, $\beta = 1000$, $\gamma_1 = 0.9$, $\gamma_2 = 0.1$



statistic	value
number of questions	13,472,796
number of answers	21,299,522
number of comments	55,852,373
questions with <code>	10,034,060
answers with <code>	15,188,226
comments with <code>	6,814
number of tags	48,373
tags used ≤ 10 times	12,955
tags used > 1000 times	3,537

Fig. 2. Statistics (right) and a plot of tags vs the number questions they appear in (left) for the Stack Overflow Data Dump from 2008-07-31 21:42:52.667 to 2017-03-13 21:56:17.233. Tag rank is the rank of a tag when the tags are ordered according to the number of Stack Overflow questions in which they are associated with, i.e., question count.

Probabilistic topic models are statistical, and to make estimated statistics meaningful, we should ignore infrequent tags or terms that do not have enough support in the dataset. This is a common approach when applying probabilistic topic models. For instance, when introducing the Labeled LDA model, Ramage et al. selected tags of medium to high frequency (Ramage et al., 2009a). Wang et al. ignored tags assigned to fewer than 50 documents when building a Stack Overflow tag prediction model (Wang et al., 2014). Also, since computational cost of topic models grows superlinearly with the number of tags (or topics) (Chen et al., 2018), by examining the tag frequency distribution in Fig. 2, we filter out tags assigned to fewer than 250 Stack Overflow questions.

Forming Words. A Stack Overflow post typically contains both natural language text and code. Although Stack Overflow uses the <code> tag to annotate code, since the mark's main purpose is proper visual formatting of the source code, it can sometimes be used for formatting non-source code text by the contributing developers, while novice contributors often forget to use it. To overcome this, researchers have recently developed more sophisticated mechanisms and similarity metrics to accurately identify code snippets (Baltes et al., 2018). Once the entire vocabulary is known, we vectorize each document into a term-frequency vector where term is a word and the frequency is the number of occurrences of the word in the document. We form words in a two-step approach. First, we divide each document into natural text and code using the <code> tag. Second, we divide the natural text of a document into word using word boundaries, such as, punctuation and spaces. A similar procedure is also applied to the code, while taking account for subtle differences in extracting words from the text and the code. For instance, in natural language text, the underscore character is an acceptable choice for word boundaries, while in code, it is not.

Selecting Tags. Tag popularity generally follows a power law, i.e., a number of tags are widely used, while many are used sparingly, as shown in Fig. 2 that graphs the number of questions with a specific tag versus the ordered rank of tags on a log-log scale. In particular, few of the tags are widely used and documented with extensive wiki pages. Examples of these are JAVASCRIPT, JAVA, C#, PHP, and ANDROID, which are assigned in 9.94%, 9.08%, 7.93%, 7.78%, and 7.13% of Stack Overflow questions, respectively. In addition, significant portions of tags are rarely used. As shown in the right side of Fig. 2, 12,955 tags or about 26.78% tags are used no more than 10 times. The value of these tags to the overall corpus organization is arguable. In addition, the rare occurrence results in insufficient support to model these tags effectively without incurring noise. To create a dataset where tags are consistently represented, we randomly sample questions for a tag if the tag is overly represented, such as, JAVASCRIPT that is associated with 9.94% of questions in Stack Overflow. For instance, Ramage et al. randomly sampled 4000 documents that contain at least one of the selected tags. In our work, we randomly select 5000 documents for a tag if there are more documents for the tag on Stack Overflow during the period we are examining.

3.2. Building the model

The processing pipeline we used to obtain a concept hierarchy from the Stack Overflow dataset consists of the following set of 4 steps.

1. In Step (1) of the pipeline, we select only the questions from the Stack Overflow dataset, a granularity with sufficient support for extracting commonly occurring concepts. Next, as mentioned above, we select a corpus of Stack Overflow questions based on their tags, ensuring that we use tags that occur frequently enough (i.e. with enough support) in the dataset, omitting tags that are too rare or too frequent. We also limit the number of questions per tag to a maximum of 5000 as discussed in Section 3.1.
2. During Step (2) we extract a corpus consisting of a set of questions, extracting constituent word lists as described above. It is common practice to filter out stop words, which are very common, or rare words that have insufficient statistical support, from a corpus (Schofield et al., 2017). Following best practices in topic modeling (Rehurek and Sojka, 2010), we choose to filter out rare words that appear in less than 300 questions and overly frequent words (i.e. stop words) that appear more than 40% questions. Frequent adjacent words are combined to form bigrams, and these bigrams are treated as additional words to the model, which are filtered using the same thresholds.
3. Step (3) specifies the hyperparameters of the model's priors, including α, β, γ_1, and γ_2, as shown in Fig. 1. We select the hyperparameters' values using both metric-based validation and by qualitatively examining the model to determine it expresses logical distributions of terms. As metrics, we use perplexity (or, alternatively, predictive likelihood) to examine the ability of the model to recognize unseen data. The choice of the hyperparameters can have significant impacts on the topics extracted (Agrawal et al., 2018). To choose the model's hyperparameters, we start with values based on previous work in hierarchical topic modeling (Paisley et al., 2015; Blei et al., 2010; Nguyen et al., 2014) and perform a limited parameter search based on a smaller dataset consisting of 3 months of Stack Overflow data. Full scale grid-search optimization was prohibitive due to the high runtime overhead of constructing the model on a substantial dataset. Prior research also indicates that the sensitivity of the modeled topics is reduced as the training dataset size increases resulting from improved statistical support from the data (Gelman et al., 2014). Another way to reduce the sensitivity to the hyperparameters is to ensure each word or tag has sufficient statistical support, which we have done by aggressively filtering our corpus.
4. Assuming a good model fit, we can use the learned concept hierarchy for predictive purposes, such as, e.g. to determine tag synonyms, or to predict tags for an unseen question. In addition, since probabilistic graphical models of this kind are also interpretable they can have empirical value to researchers that want to examine frequent terms occurring in a particular development community, which is represented by a set of tags.

The overall parameters used in constructing our model and their origin in listed in Table 1.

3.3. Example concept hierarchy

Fig. 5 visualizes a subset of the Stack Overflow concept hierarchy, focusing on concepts in the ANDROID sub-tree. This hierarchy expresses the structure of the concepts in the Stack Overflow document collection, and represents the basis of our learned model. Given this model and any (seen or unseen) document, we can infer a hierarchy that the specific document exhibits. For instance, Fig. 4 shows the concept tree for a single Stack Overflow question entitled *"How to immediately get id token from google play services Unity Plugin"* – listed in Fig. 3 – which was inferred from the ANDROID centered model. Unbeknownst to our model, this question was tagged with the tags C#, ANDROID, UNITY3D, and GOOGLE-PLAY-SERVICES by developers on the platform. We observe that the concept hierarchy has considerable overlap with the human selected tags, while providing additional context through the organization of the tags and additional relevant tags (e.g. ADAPTER, .NET).

4. Experimental design

We aim to answer the following set of research questions, which focus on the quality and the predictive power of the concept hierarchy (*RQ 1* and *RQ 2*) and on the potential usefulness of the concept hierarchy for search and navigation (*RQ* 3)[3]

RQ 1 Does the concept hierarchy identify tag synonyms matching human labeled tag synonyms on Stack Overflow?
RQ 2 Does the concept hierarchy predict tags for an unseen Stack Overflow document matching human assigned tags?
RQ 3 Does traversing the concept hierarchy towards the bottom focus, while traversing towards the top broaden the set of Stack Overflow posts?

For the evaluation of all three RQs, we use the dataset, preprocessing, filtering and hyperparameter choices described in Table 1. The resulting dataset has 369 tags and 196 pairs of synonyms as reported by the Stack Overflow community. We use 3 sets of evaluation metrics, described below, in Sections 4.1–4.3, in order to answer *RQ 1, RQ 2*, and *RQ 3*, respectively.

4.1. Tag synonym identification (RQ1)

The Stack Overflow dataset has tens of millions of unique posts and tens of thousands of tags. Many tags have community-contributed wiki pages containing descriptive information. While tagging is effective in helping search and navigation, allowing user to create tags results in the number of user-contributed tags growing over time. In part, this is because of the existence of tag synonyms, where a user has contributed a tag that already exists (e.g. AMAZON-ATHENA and AWS-ATHENA). Stack Overflow indicates "[w]henever you see questions being repeatedly tagged with the wrong or incorrect tag – or multiple tags that mean the same thing – it's a good idea to propose a tag synonym" (Stack Exchange, Inc., 2018a). Users can suggest and vote on tag synonyms (Stack Exchange, Inc., 2018a). A tag synonym pair is organized as a "master" and "synonym". Stack Overflow will map any questions tagged with a "synonym" tag to its corresponding "master" tag. The set of the Stack Overflow community mapped tag synonyms has gradually grown over time, and the number of pairs of tag synonyms stands at 3,650, retrieved on June 13, 2017 from the Stack Overflow Data Explorer (Stack Exchange, Inc., 2018c), compared to 2,765 pairs in September, 2014 from the same source as reported in Beyer and Pinzger (2015, 2016). By identifying tag synonyms, we can effectively prune the tag space from unnecessary tags, thereby improving search efficiency via tags. In this section, we aim to evaluate the concept hierarchy, which contains a hierarchy of tags and their associated topics (probability distributions of terms), for the purpose of tag synonym identification.

The intuition is that if two topics (i.e. their term distributions) are close to each other and their difference is indistinguishable, then their associated tags should be essentially indistinguishable. There are a large number of choices of metrics to

[3] Our code and scripts as well as instructions to obtain experiment data are publicly available at: https://doi.org/10.5281/zenodo.3234916.

I'm using https://github.com/playgameservices/play-games-plugin-for-unity plugin to sign in user with his google account , and then i want to get Id Token and send it to my server and register an account for this user in my own database.this is my code to get Id token :

```
PlayGamesPlatform.Instance.Authenticate(success =>
{
    if (success)
    {
        Debug.Log("Id Token :");
        Debug.LogFormat("{0}", PlayGamesPlatform.Instance.GetIdToken());
        Debug.Log("End Of Id Token");
    }
});
```

The problem is the first time it prints just empty string , when i call this for the second time (or a moment later) it prints the token. I want to get token **immediately** or with a **callback** to make sure that token is recieved .

How to make sure that token is recieved? is there any callback for this?

Thanks

Fig. 3. Stack Overflow question entitled *"How to immediately get id token from google play services Unity Plugin"* corresponding to the inferred concept tree in Fig. 4.

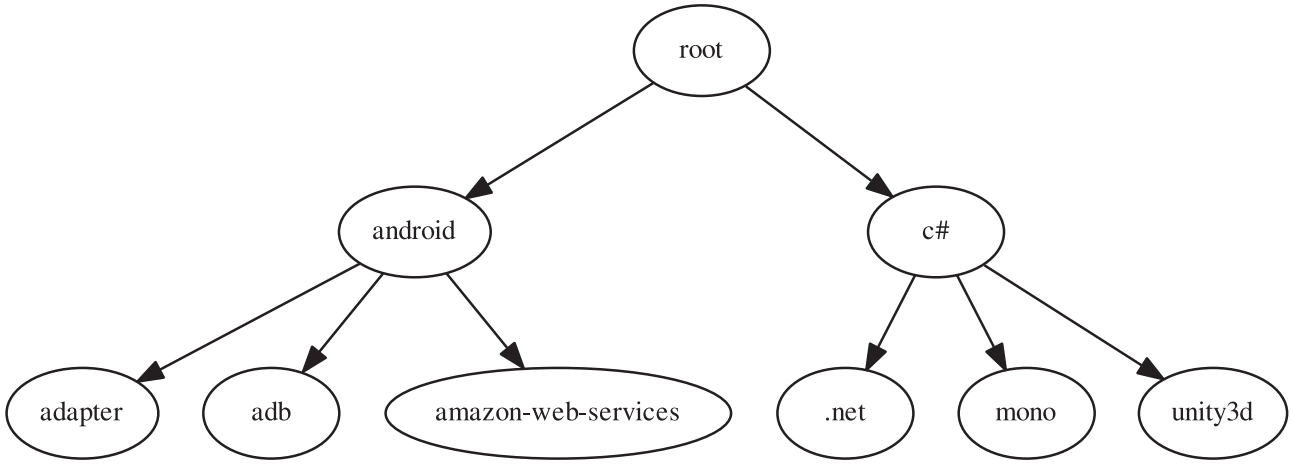


Fig. 4. Inferred concept tree of a Stack Overflow post entitled *"How to immediately get id token from google play services Unity Plugin"*. The corresponding question is in Fig. 3. The graph displays only the tags with significant weight > 0.008. The normalized weights (or probability) of tags ANDROID, C#, ADAPTER, ADB, AMAZON-WEB-SERVICES, .NET, .MONO, and .UNITY3D are, 0.5935, 0.1552, 0.0200, 0.0477, 0.0773, 0.0115, 0.0085, and 0.0573, respectively.

choose in comparing two topics (i.e. two discrete probability distributions) (Cha, 2007). In topic modeling, the Kullback-Leibler divergence (KL divergence) is widely used due to its probabilistic and information theoretic interpretation (Stevens et al., 2012; Mei et al., 2007; Lau et al., 2011). However, in this paper, we choose the Jensen-Shannon divergence, an improvement over the KL divergence (Lin, 1991; Endres and Schindelin, 2003) in that it is symmetric (i.e., the Jensen-Shannon divergence from X to Y is always equal to Y to X) and always a finite value, while KL divergence is not.

The topological closeness of two concepts in a tree may also be considered in comparing two concepts. It may also be computed in several ways. In this paper, we adopt the shortest path between two concepts p and q for $g(p, q)$. Since the graph is a tree, the distance of two nodes can be easily computed since we only need to traverse from p and from q toward the root until we reach a common ancestor. To determine the effectiveness of automatic synonym identification, we exhaustively examine all the pairs of tags in the evaluation dataset, i.e., $\binom{369}{2} = 67,896$ pairs, computing the Jenson-Shannon distance between any two tags. If the distance of the pair is less than a threshold, we consider the pair to be synonymous.

Based on our hierarchical concept hierarchy, we can consider two distinct tag synonym identification schemes:

Synonym Identification Scheme 1 [Topic-based tag synonym identification]: A topic is a discrete probability distribution over words. Given that there are V unique words in our data, we represent a topic indexed by z conveniently as a vector $\vec{\phi}_z = (\phi_{z,1}, \phi_{z,2}, \ldots, \phi_{z,i}, \ldots \phi_{z,N})$ where $\phi_i = P(w = w_i|z)$ and $\sum_{i=1}^{N} P(w_i|z) = 1$. In this approach, we consider two tags t_i and t_j are synonyms when $d(\vec{\phi}_{t_i}, \vec{\phi}_{t_j}) < \delta_\phi$ where $d(p, q)$ is a similarity metric measuring the closeness of two probability vectors p and q, and δ_ϕ is a threshold.

Synonym Identification Scheme 2 [Concept hierarchy-based tag synonym identification]: The above scheme does not consider the

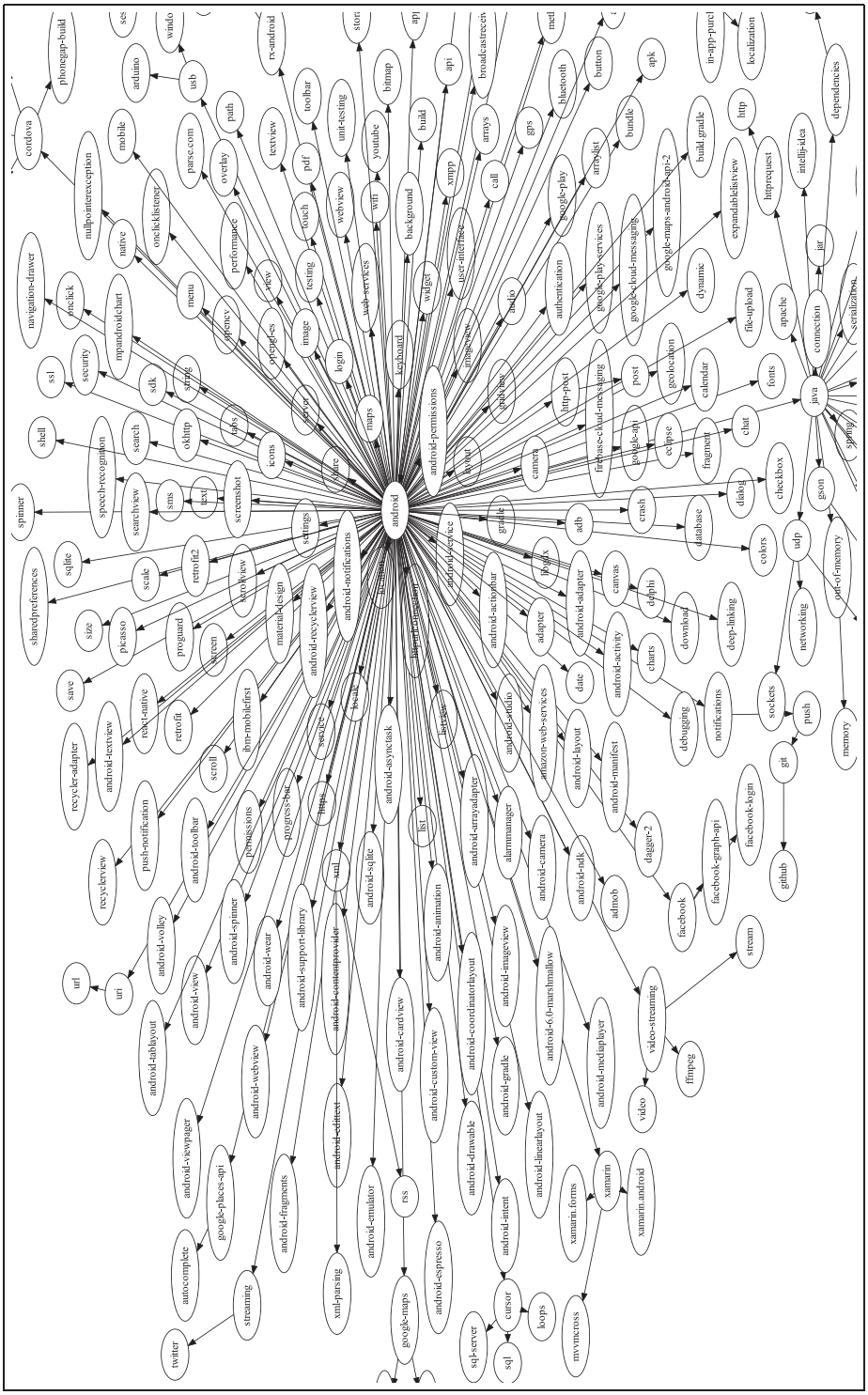


Fig. 5. A global concept hierarchy, centered on the ANDROID subset of Stack Overflow tags and questions.

hierarchical structure of the tag-topic model. The concepts, each of which is a pair of tag and its corresponding topic, form a tree structure. We denote the shortest graph distance of two topics in the tree as $d_t(\vec{\phi}_{t_i}, \vec{\phi}_{t_j})$. Utilizing the tree structure of the concepts, we consider two tags t_i and t_j are synonyms when $d(\vec{\phi}_{t_i}, \vec{\phi}_{t_j}) < \delta_\phi \wedge g(\vec{\phi}_{t_i}, \vec{\phi}_{t_j}) < \delta_g$ where $d(p, q)$ is a metric measuring the closeness of two probability vectors p and q, $g(p, q)$ is a similarity metric on the closeness of two concepts that correspond to p and q in the concept tree, δ_ϕ and δ_g are two thresholds.

4.2. Tag prediction (RQ2)

For the second research question, we examine how well we can predict the tags of an unseen (or held out) Stack Overflow question using our concept hierarchy, where an unseen question is one that is not in the training dataset. The target application that this evaluation mimics is recommending tags to a developer that has just finished composing a new question in Stack Overflow. The evaluation process is as follows. We start by randomly

selecting a set of unseen questions retrieved from Stack Overflow, and use the concept hierarchy to predict the probabilities of tags whose concepts the documents exhibit. We rank the list of tags ordered by their probabilities, where the greater the probability, the higher the rank. For simplicity, the tree structure is ignored. Next, we select a list of top ranked tags, given N length of list elements – the top-N tags. Since the unseen documents are tagged by the Stack Overflow contributors, we can compare the tags of a document with its top ranked documents, and count the number of matches. The ratio of total number of matches to the maximally possible matches (i.e. the number of total tags in the held out question) is the *matching tags accuracy*, which we compute as our primary metric. We compare our accuracy to the recently proposed tag prediction system EnTagRec. We searched but could not find any description by Stack Exchange of the existing algorithm to predict tags. Therefore, we have not yet devised a good method to compare directly to the Stack Overflow tag suggestion algorithm.

4.3. Exploratory search effectiveness (RQ3)

We evaluate whether it is possible for our concept hierarchy to serve as a search aid for Stack Overflow. Our aim to aid *exploratory* searches, when the developer seeks a document previously unknown to him or her. While a silent majority often seeks answers on Stack Overflow via Web searches, exploratory search via tags appear to be a common strategy by many question answers on Stack Overflow[4]. According to the berry picking model of information retrieval, a user satisfies their information need via a process of dynamically adapting and refining search queries (Baeza-Yates and Ribeiro-Neto, 1999). In this scenario, the user typically begins with a generic query, for which the search tool retrieves a large set of ranked results that are impossible to examine in their entirety. The concept hierarchy can then be used to further filter and explore the initial set of retrieved results. Alternatively, absent an initial query, the developer can begin navigating at the root of the concept tree. To clarify how the concept tree can be used for this purpose, we present Algorithm 1. The essential part of the algorithm is a function that computes a candidate document set, as determined by the learned global concept tree.

There are a two distinct objectives in navigating the concept tree: specificity and diversity (Martínez and Reyes-Valdés, 2008). Specificity is relevant as developers refine their searches by descending down the concept hierarchy. Ideally, we want to ensure that the documents presented to the developers are specific to the particular sub-tree they are currently exploring. However, when in doubt and unsure of the most profitable path down the hierarchy, a developer may need to take a different tack. Therefore, a second dimension is the diversity of a different branch in the tree that enables a departure from the current selections. More specifically, in the context of Algorithm 1, we expect that the returned candidate document set becomes more specific when a user selects a tag deeper along a path in the concept tree; however, we expect that the document set becomes more diverse when a user select a tag on a different branch of the tree. These two objectives appear in the information retrieval literature to express the quality of retrieved search results or a corpus (Shi et al., 2012; Angel and Koudas, 2011). In the following, we define specific metrics to evaluate these objectives.

Consider the bipartite graph in Fig. 6, where D documents collectively exhibit K concepts.The m-th document exhibits K concepts with proportions $\theta_m = (\theta_{m,1}, \ldots, \theta_{m,n}, \ldots_{m,K})$ where $\theta_{m,n}$ is proportion, or weight, of the n-th concept in the document. Following Martínez and Reyes-Valdés (2008), we begin to quantify the

[4] see "*How to find the right questions that I can answer?*" at https://meta.stackexchange.com/questions/44739/.

Algorithm 1: Compute a candidate document set from a navigational search.

Input: Inferred Global Model from the training dataset: $\mathcal{M}$;
Document set to search from: $\mathbb{D}$

```
while true do
    DisplayConceptTree (M)
    l_k ← SelectKeyTagFromTree (M)
    C ←ComputeCandidateDocumentSet (M, D, l_k)
    DisplayCandidateDocumentSet (C)  // the application
    orders the documents in some fashion
end
```

Input: Inferred Global Model from the training dataset: $\mathcal{M}$;
Document set to search from: $\mathbb{D}$;
Key tag: l_k
Output: Subset of documents in $\mathbb{D}$ based on l_k

```
Function ComputeCandidateDocumentSet (M, D, l_k)
    C ← ∅
    ε_w ←GetTagWeightThreshold (M, l_k)
    ε_p ←GetTagGraphPathThreshold (M, l_k)
    foreach d ∈ D \ d_k do
        T_d ← GetDocumentTree (M, d)
        w_{d,l} ← GetTagWeight (T_d, l_k)        // construct a
        selection criterion based on tag weights
        p_{d,l} ←GetTagGraphPathValue (M, T_d, l_k)        // a
        criterion on concept tree graph structure
        if w_{d,l} > ε_w ∧ p_{d,l} > ε_p then C ← C ∪ {d}
    end
    return C
end
```

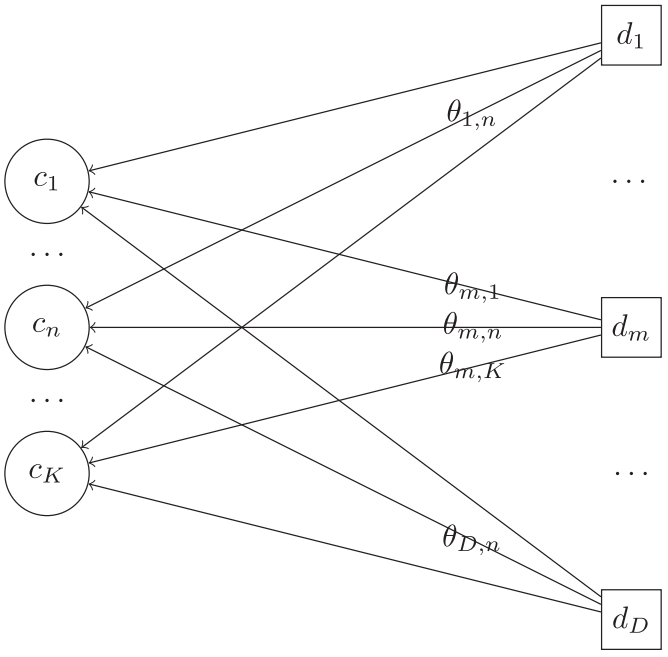


Fig. 6. Concepts, pairs of tags and topics and documents form a bipartite graph. The edge of the graph has a weight that is how strongly a document exhibits a concept. The graph shows K concepts and D documents where the m-th document exhibits the K concepts with proportions $\theta_m = (\theta_{m,1}, \ldots, \theta_{m,n}, \ldots_{m,K})$ where $\theta_{m,n}$ is proportion or weight of the n-th concept in the document.

diversity of a document exhibiting a concept by estimating Shannon's entropy.

$$H_m = -\sum_{n=1}^{K} \theta_{m,n} \log_2 \theta_{m,n}, 1 \le m \le D \tag{1}$$

Here, $1 \le n \le K$, n is an index of a concept (or a tag), and $1 \le m \le D$, m is an index of a document labeled by the tag. If the m-th document were to exhibit the K concepts equally, we would arrive at $H_m = -\sum_{n=1}^{K} 1/K \log_2 1/K = \log_2 K$, which would indicate that the document is the most diverse with regard to the K concepts. Conversely, if the m-th document were to exhibit only a single concept, e.g., the 1st concept, we would arrive at $H_m = -\lim_{\theta_{m,1} \to 1} \theta_{m,1} \log_2 \theta_{m,1} - \sum_{i=2}^{K} \lim_{\theta_{m,i} \to 0} \theta_{m,i} \log_2 \theta_{m,i} = 0$, which shows that the document is the least diverse with regard to the K concepts. We define another quantity,

$$V_m = \left(-\sum_{n=1}^{K} \theta_{m,n} \log_2 \overline{\theta}_n \right) - \left(-\sum_{n=1}^{K} \theta_{m,n} \log_2 \theta_{m,n} \right) \tag{2}$$

$$= \overline{H}_m - H_m \tag{3}$$

where $1 \le m \le D$ and

$$\overline{H}_m = -\sum_{n=1}^{K} \theta_{m,n} \log_2 \overline{\theta}_n \tag{4}$$

representing the weighted average of the K topics exhibited in the m-th document. This quantity measures how much a given document with regard to the exhibition of the K topics departs from the overall (or average) distribution of the K topics in the D documents (i.e., the entire document collection). We call this quantity the *document divergence*. The average divergence of the D document becomes,

$$\overline{V} = \frac{1}{D} \sum_{m=1}^{D} V_m \tag{5}$$

We now return turn our focus to specificity. Since a concept may be present in multiple documents with different proportion, the average proportion of the n-th concept the m-th document of a set of D documents is:

$$\overline{\theta}_n = \frac{1}{D} \sum_{j=1}^{D} \theta_{j,n} \tag{6}$$

Applying the Shannon entropy to the proportions of the n-th concept in the D documents, we define *concept specificity* as:

$$S_n = \frac{1}{D} \sum_{m=1}^{D} \frac{\theta_{m,n}}{\overline{\theta}_n} \log_2 \frac{\theta_{m,n}}{\overline{\theta}_n}, 1 \le n \le K \tag{7}$$

Consider the case that the n-th concept is present equally in the D documents, i.e., $\theta_{m,n} = \overline{\theta}_n$ for $1 \le m \le D$. It follows $S_n = \frac{1}{D} \sum_{m=1}^{D} \frac{\theta_{m,n}}{\overline{\theta}_n} \log_2 \frac{\theta_{m,n}}{\overline{\theta}_n} = \frac{1}{D} \sum_{m=1}^{D} \frac{\overline{\theta}_n}{\overline{\theta}_n} \log_2 \frac{\overline{\theta}_n}{\overline{\theta}_n} = \frac{1}{D} \sum_{m=1}^{D} 0 = 0$. Consider another case thet the n-th concept is only present in a single document in the D document, e.g., the 1st document. It follows that $\overline{\theta}_1 = 1/D$, $\theta_{1,n} = 1$, $\theta_{m,n} = 0$ for $2 \le m \le D$, and $S_n = \frac{1}{D} (\lim_{\theta_{1,n} \to 1} \frac{\theta_{1,n}}{\overline{\theta}_n} \log_2 \frac{\theta_{1,n}}{\overline{\theta}_n} + \sum_{m=2}^{D} \lim_{\theta_{m,n} \to 0} \frac{\theta_{m,n}}{\overline{\theta}_n} \log_2 \frac{\theta_{m,n}}{\overline{\theta}_n} = \frac{1}{D} (D \log_2 D + \sum_{m=2}^{D} 0) = \log_2 D$. The former represents the case that the concept is least specific with regarding to the D documents, and the later the concept is most specific with regarding to the D documents. We call the quantity in Eq. 7 as the *concept specificity* with regard to the documents.

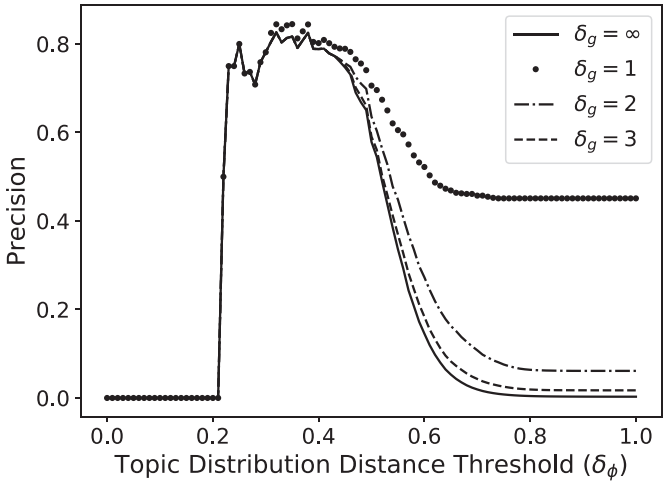


Fig. 7. Precision of tag synonym identification versus topic distribution distance where the precision is defined as $\frac{TP}{TP+FP}$.

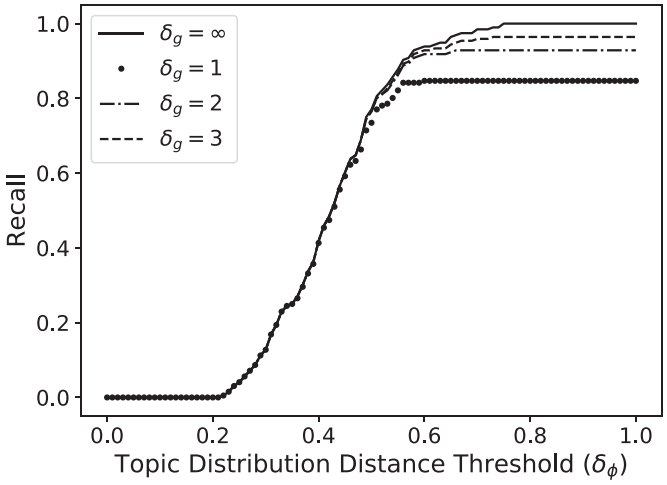


Fig. 8. Recall of tag synonym identification versus topic distribution distance where the recall is defined as $\frac{TP}{TP+FN}$.

5. Evaluation results

We now present the results of the evaluation we conducted to answer *RQ1, RQ2*, and *RQ3*, in turn, using the metrics detailed in the previous section.

5.1. Tag synonym identification (RQ1)

To answer the first research question, Figs. 7, 8, and 9 show the precision, recall, and F1 score of the synonym identification problem. The influence of the tree structure is encoded as δ_g, where a value of ∞ denotes no influence from the structure, only topics (i.e. Synonym Identification Scheme 1). The result shows that the identification algorithm that uses both topic distributions and the tree hierarchy (i.e. Synonym Identification Scheme 2) produces better results in precision and F1 score. However, this identification algorithm can produce more false negatives, i.e., there will be more cases the algorithm considers two tags are not synonyms while humans consider they are.

We draw further insight on whether we should consider the tree structure in identifying synonyms from observing Fig. 10. Generally, the topic distribution distance increases with the tree distance, which implies that these two quantities are correlated. How-

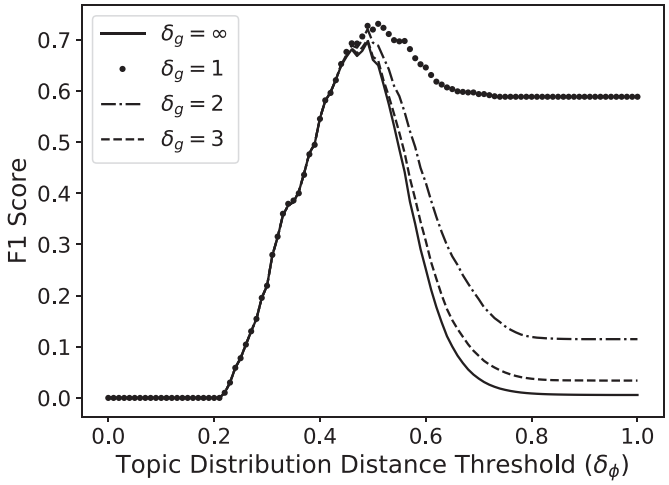


Fig. 9. F1 score of tag synonym identification versus topic distribution distance where the F1 score is defined as $\frac{2TP}{2TP+FN+FN}$.

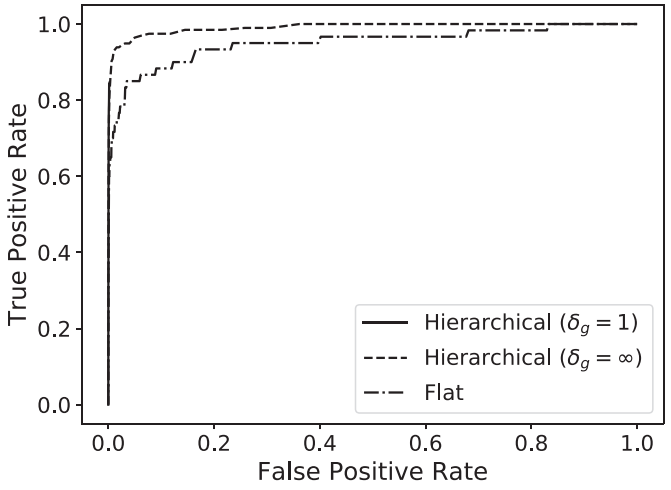


Fig. 11. The Receiver Operating Characteristics (ROC) for tag-synonym identification using hierarchical model (L2H) and flat model (LLDA), respectively.

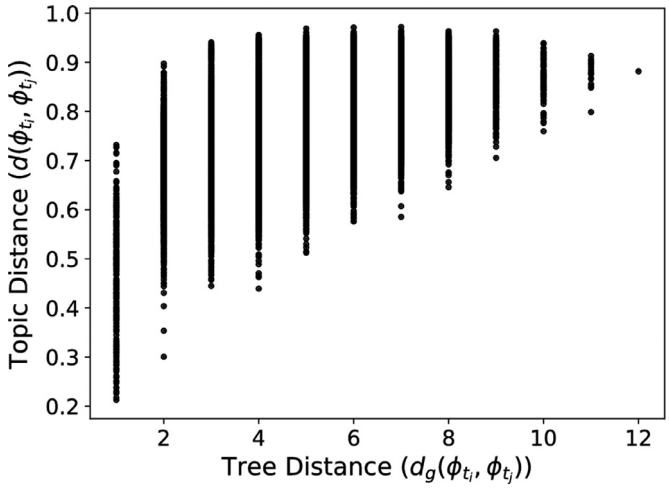


Fig. 10. Topic distribution distance versus tree distance.

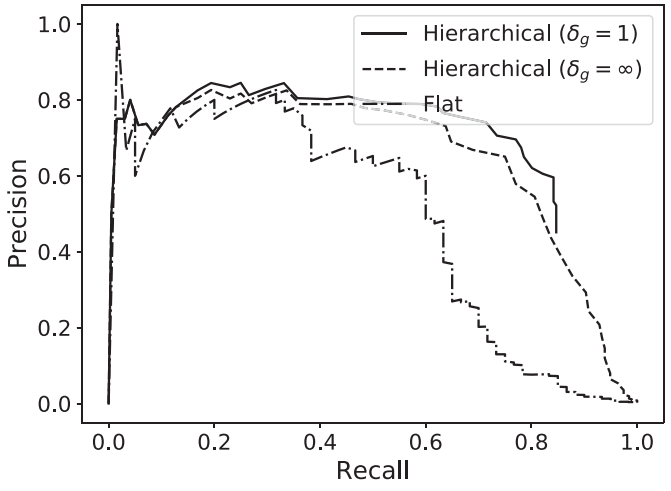


Fig. 12. The Precision-Recall Characteristics (PRC) for tag-synonym identification using hierarchical model (L2H) and flat model (LLDA), respectively.

ever, it can also be observed that for small distances in the tree, e.g. at 1, or 2, the topic distance varies greatly. We also observe that some tags that are far apart in the tree have similar topic distributions. Both of these indicate that when by using the tree structure and topic similarity in concert, we can avoid some likely misclassification that could occur if we considered each quantity in isolation. Figs. 7, 8, and 9 confirm this understanding.

Fig. 11 shows the true positive rate versus the false positive rate, commonly known as the Receiver Operating Characteristic for the binary classification problem. The curve illustrates that the true positive rate increases sharply as the false positive rate does, which indicates that we can select a threshold to obtain a high true positive rate at a low false negative rate. It is worth noting that this binary classification problem has very high class-imbalance, i.e., the synonym pairs is much fewer than non-synonym pairs. For the unbalanced classification problem, the Precision-Recall curve is often computed as a metric to measure the performance of the binary classification. Fig. 12 shows the precision-recall curve that shows that the hierarchy contributes to higher precision-recall curve.

An important observation from Fig. 12 that matches the observation in Fig. 10 is that two tags that are close in their topic distributions may not be close in the concept hierarchy topologically and vice versa. In Fig. 12, we use a fixed tree distance threshold of 1 ($\delta_g = 1$) to determine whether two tags are a pair of synonyms, which excludes many candidate pairs that are close in their topic distribution but further apart in their graph distance, which limits the recall to reach 1. This observation also suggests classification with more flexible class boundaries may help.

The above experiments indicate that effectively identifying tag synonyms from the Stack Overflow data depends heavily on choices of thresholds, which trade off between different types of identification errors. Table 2 shows a selection of precision, recall, accuracy, and F1 score values, when each of these metrics is optimal, respectively. The results indicate that we would arrive at 100% recall if we simply identify all pairs as tag synonyms. Without the remaining metrics, recall is clearly meaningless. We argue that accuracy is also not necessarily a good metrics for a novelty identification task like this, because the number of negatives ($67,896 - 196 = 67,700$ pairs) far out-weights that of positives, i.e., we can have high accuracy with very small true positives because the number of true negatives is large. For the tag synonym identification problem, precision is most appropriate, and the best precision is 84.48% as shown in Table 2 when we consider both topic distributions and concept hierarchy.

Beyer et al. initially suggested automated tag synonym identification for Stack Overflow based solely on the tags (Beyer and Pinzger, 2015), ignoring the content of questions to which tags are assigned. Their approach produces a ranked list of synonyms,

Table 2
Example results of tag synonym identification.

Distribution Threshold (δ_ϕ)	Graph Threshold (δ_g)	Optimal Metric	Performance Metrics
0.35	1	Precision=84.48%	Precision=84.48% Recall=25.00% Accuracy=62.49% F1 Score=38.58%
0.75	∞	Recall=100.00%	Precision=0.79% Recall=100.00% Accuracy=81.91% F1 Score=1.58%
0.60	∞	Accuracy=96.14%	Precision=14.55% Recall=84.69% Accuracy=96.14% F1 Score=64.59%
0.51	1	F1 Score=73.12%	Precision=69.58% Recall=77.04% Accuracy=88.47% F1 Score=73.12%

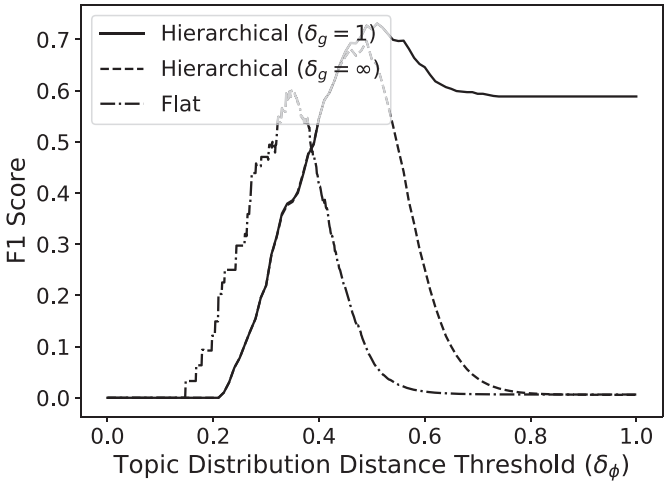


Fig. 13. The F1 Scores for tag-synonym identification using hierarchical model (L2H) and flat model (LLDA), respectively.

which, when compared to the user curated tag synonyms in Stack Overflow, has accuracy of 74.9% for suggesting a correct tag within the top 15 suggestions, and 45.9% for the top 1 suggestion (Beyer and Pinzger, 2015). However, since Beyer et al.'s rules for tag similarity are developed based on the user curated tags in Stack Overflow, the accuracy on a set of synonyms not part of the existing set drops to 20% for the top suggestion. The two challenges in directly comparing our results to Beyer et al. are that: 1) we use evaluation thresholds that are not based on a ranked list of tags, as shown in Table 2; 2) accuracy is not an appropriate metric for class imbalanced classification. However, overcoming these discrepancies, we observe that, if optimizing for accuracy, we can achieve a value for accuracy of 96.14%, which is significantly better than Beyer et al.; while the number of tag synonyms we select varies from instance to instance it is always less than 10 tags.

Beyer at al.'s approach for tag synonym identification does not consider the text of the Stack Overflow posts, so our improvements are to be expected. In order to compare to a model that relies on textual data, in Fig. 13 we also compare the F1 scores obtained using the hierarchical model and a popular flat labeled topic model (Labeled LDA (Ramage et al., 2009b)). The results clearly show that the hierarchical model is superior to the flat model for tag synonym identification.

Next, we examine the false positives produced by the model, i.e., the pairs of tags that the concept hierarchy-based tag synonym identification approach recognizes as synonyms, but are not (yet) manually tagged as synonyms on Stack Overflow. Using the two thresholds for F1-Score-optimal tag synonym identification, $\delta_\Phi = 0.51$ and $\delta_g = 1$ shown in Table 2, the set of suggested tag synonyms includes 66 false positive tag pairs (out of $\binom{369}{2} = 67,896$ potential pairs). Table 3 shows a randomly chosen half of those false positive tag pairs, a total of 33 pairs.

Two of the authors examined this subset of 33 false positives qualitatively, grouping them into three categories based on the type of relationship between the tags. The first category of false positive tag synonyms (pairs numbered 1–15) includes tag pairs that form a parent-child relationship, where one of the tags is a more general and the other more specific concept within the same technology space. These pairs have potential to be suggested as synonyms to the community as, according to Stack Overflow, "*[i]n some cases, tags that are subsets of other tags will also be considered synonyms, such as* JAVA-SE*for* JAVA" (Stack Exchange, Inc., 2019). Second, in rows 16 to 31, is a category of tag pairs that are representations of two closely related technologies. For instance, we will not be able to write a meaningful SQL JOIN statement without a WHERE clause and we often contrast OVERRIDING with OVERLOADING in an object-oriented programming language. While a few of the most closely related tags may be suggested as synonyms, the majority should remain separate as they represent related but distinct concepts. The last category is the smallest, consisting of pairs numbered 32 and 33, and includes tags that the two authors agree lack any significant relatedness between them.

5.2. Tag prediction (RQ2)

Figs. 14 and 15 show a set of results for predicting over 23,000 randomly chosen sets of unseen documents, contrasting the matching tags accuracy with a pair of relevant parameters, i.e. the number of tags in ranking (i.e. the top N) and the length of the question in number of words. As a baseline, we also show the accuracy obtained using Labeled LDA, a flat (i.e. not hierarchical) topic model.

Fig. 14 shows that the accuracy increases as the N in the top-N tags grows. This is expected, since we consider a match with a larger pool of tags when N grows. In Fig. 15 we see that the accuracy sharply increases when the questions (or documents) are longer. It is intuitive that when questions become longer, their meaning becomes more specific, and the algorithms successfully captures the improved statistical significance from the data.

Most importantly, the results in both figures show that the hierarchical model outperforms the flat topic model for tag prediction. This demonstrates the potential of the hierarchical model in building tag prediction systems for crowd-sourced information sites. For instance, a highly accurate tag predicting system called EnTagRec (Wang et al., 2014) outperforms other prior approaches (Xia et al., 2013). EnTagRec is based on an ensemble of a Bayesian (i.e. Labeled LDA) and a frequentist classifier. We anticipate that a hierarchical model would help improve the prediction accuracy of such a system by improving the Bayesian component (as indicated by the across the board improvement in accuracy of the hierarchical model vs. Labeled LDA in Fig. 14 and Fig. 15). To demonstrate this in part, we retrofitted EnTagRec to include the hierarchical topic model examined in this paper (calling the resulting ensemble hierarchical EnTagRec). Table 4 shows the Recall@5 for the dataset provided with EnTagRec using the two tag prediction methods. This result shows that the Hierarchical EnTagRec edges out the original EnTagRec when documents are long, but loses to the original EnTagRec when documents are short.

Table 3
Example of tag synonyms identified by the model that are not (yet) recognized by Stack Overflow (false positives).

Relationship Category	No.	Tag 1	Tag 2
Parent-Child	1	APPCELERATOR	APPCELERATOR-MOBILE
	2	ANGULARJS-DIRECTIVE	ANGULARJS-NG-REPEAT
	3	ANGULARJS-SERVICE	ANGULARJS-HTTP
	4	COMPLEXITY-THEORY	GRAPH-THEORY
	5	DOCKER	OPENSHIFT
	6	LARAVEL-5	ELOQUENT
	7	NLP	STANFORD-NLP
	8	PUBLISH	WEBDEPLOY
	9	PYTHON-IMAGING-LIBRARY	SKIMAGE
	10	SHAPE	POINTS
	11	TITANIUM	TITANIUM-MOBILE
	12	UISTORYBOARD	XCODE-STORYBOARD
	13	VIDEO-STREAMING	YOUTUBE
	14	WEB-SERVICES	APACHE-KARAF
	15	BUILD	PACKAGE
Peer-to-Peer	16	APACHE-CAMEL	APACHE-KARAF
	17	APACHE-SPARK-SQL	HIVE
	18	ALEXA-SKILLS-KIT	AWS-LAMBDA
	19	AMAZON-CLOUDFORMATION	AWS-LAMBDA
	20	AZURE-ACTIVE-DIRECTORY	OAUTH-2.0
	21	AZURE-ACTIVE-DIRECTORY	SQL-AZURE
	22	AZURE-MACHINE-LEARNING	AZURE-TABLE-STORAGE
	23	CELLS	FORMULA
	24	CHEF	DOCKER
	25	FOREIGN-KEYS	JOIN
	26	JOIN	WHERE
	27	NG-OPTIONS	NG-REPEAT
	28	OVERLOADING	OVERRIDING
	29	SEGUE	UIVIEWCONTROLLER
	30	SEGUE	VIEWCONTROLLER
	31	BIT	SIGN
Unrelated	32	BISON	SPACE
	33	MODULO	TIMESPAN

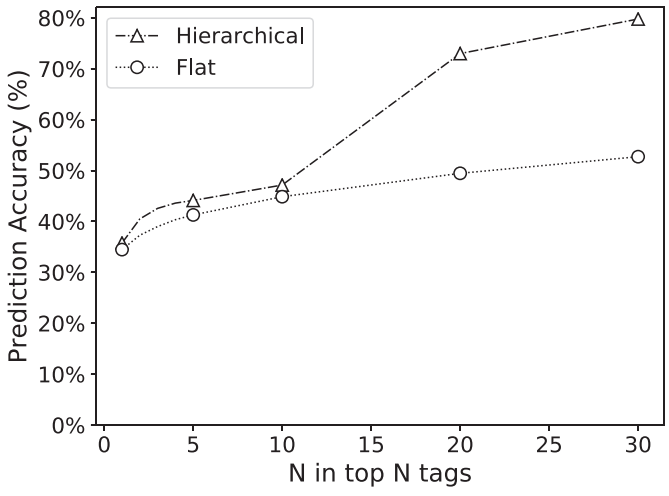


Fig. 14. Tag prediction accuracy using a flat model and that using a concept hierarchy. We consider a correct prediction when the tags of an unseen question appears in N most significant topics/concepts/tags (or top N tags).

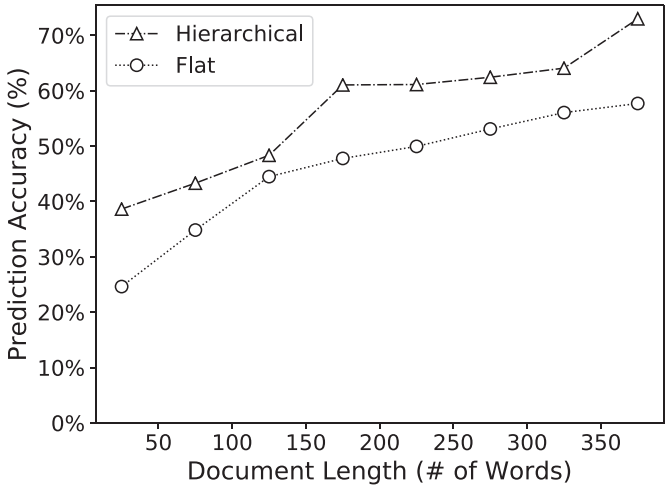


Fig. 15. Tag prediction accuracy using a flat model and that using a concept hierarchy. The accuracies shown are for top 5 tags (N=5). Each accuracy measure is estimated for unseen questions whose length has an upper bound of a specific length, e.g., (0 – 50], (50 – 100], etc.

Table 4
Recall@5 for EnTagRec and Hierarchical EnTagRec, which includes a hierarchical model in the tag prediction ensemble.

Tag Prediction Method	Document Length (# of Words)			
	[0, 50)	[50, 100)	[100, 150)	[150, ∞)
EnTagRec	0.7126	0.9011	0.9587	0.9841
Hierarchical EnTagRec	0.6619	0.9031	0.9762	0.9958

5.3. Exploratory search effectiveness (RQ3)

For this research question, Fig. 16 visualizes the specificity of the concept hierarchy extracted from our test dataset. Each line in the Figure corresponds to a branch in the tree from the root to the leaf. It clearly shows that specificity increases considerably from the root to level 2, and gradually levels off following that at a rate dependent on the specific tree branch. This is indicative of the hi-

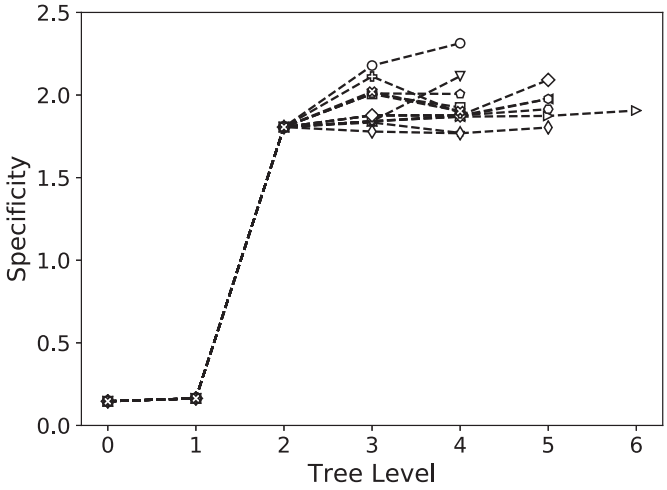


Fig. 16. Specificity of concepts in returned document set from using Algorithm 1, i.e., S_n where n indexes a concept. The figure only shows branches whose length is no less than 5 levels.

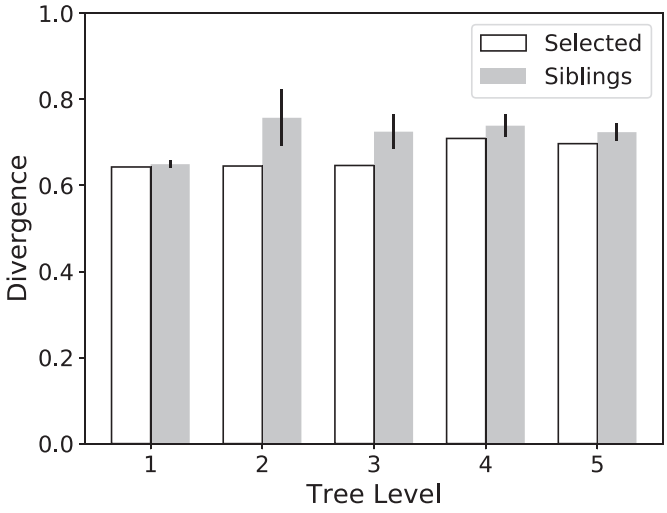


Fig. 17. Average document divergence within a subset of documents selected using a concept, and average divergences and standard errors to the subset of documents from using sibling concepts from using Algorithm 1.

erarchy aiding to navigate to a more specific set of documents, but that many deeper level of the hierarchy are roughly as specific as level 2.

We also compute average document divergence from the subset of documents from using any siblings of this concept. Fig. 17 is the comparison of the divergences. Since a concept has multiple siblings, we shows the range of the divergences corresponding to the sibling concepts. We observe that document divergence increases when we select a different sibling in the tree at almost all tree levels. However the difference in divergence between the selected subtree and the siblings is greatest at level 2 of the tree. Overall, we seen that the selection of Stack Overflow documents shown during navigational searches would be improved by our concept hierarchy and that this benefit is greatest at the higher levels of the tree, tapering off among the concepts in the deeper levels.

5.4. Threats to validity

Our approach for building a hierarchical concept model for Stack Overflow suffers from a few threats that impact the validity of our study and the ability to generalize its results.

A threat to the internal validity of the results are the specific parameter and configuration choices we used for our study. We mitigated this threat by systematically examining these parameters, by using a separate validation dataset, and by examining distributions of the input data to select reasonable thresholds. Another threat to internal validity is the sensitivity of topic models, and other similar probabilistic constructs, to a variety of random factors, including the ordering of the documents considered by the model (Agrawal et al., 2018). We mitigate this threat by examining tag synonym identification and tag prediction with a number of different randomly selected and ordered data sets.

A threat to construct validity is whether the measurements we used accurately reflect dimensions of interest in our study. To mitigate this, we used standard metrics and evaluation techniques that others have proposed and used in topic modeling and for the different purposes that we applied are model to.

The primary threat to external validity, which concerns the ability to generalize the results, is that we applied the modeling technique only on a snapshot of the Stack Overflow dataset. A mitigating factor to this threat is the considerable diversity and size of the datasets we used in our study.

6. Related work

Question and answer sites, like Stack Overflow, have grown to become an important enabler of collaborative knowledge sharing for software developers. Empirical evidence suggests that there has been a growing trend for developers to shift to Q&A sites for technical solutions to their development problems and for sharing knowledge with the community (Vasilescu et al., 2014). Popular Q&A sites have in effect become a crowd-sourced knowledge base for software developers and computer scientists (Parnin et al., 2012; Campos et al., 2016). For instance, Stack Overflow has significant coverage of the APIs of popular programming languages and platforms, and be effective for obtaining answers for code reviews and conceptual questions (Parnin et al., 2012).

Researches have proposed approaches to leverage Stack Overflow in various ways, including linking API documentation with posts and example code (Subramanian et al., 2014), building integrated recommendation tools within IDEs to Stack Overflow posts (Bacchelli et al., 2012), and integrating issue tracking systems with Stack Overflow (Correa and Sureka, 2013; Wang et al., 2017). These approaches demonstrate that the knowledge curated on Stack Overflow helps developers and software development both directly, when developers read and write questions and answers on the site, as well as indirectly, when the Stack Overflow data is processed by tools.

While applications of Stack Overflow are numerous in the literature, our focus is related work on approaches for tag recommendation and tag synonym identification, parts of the overall goal of this paper of organizing a large community-driven tagged corpus like Stack Overflow.

Several techniques have been proposed to recommend tags for a newly composed Stack Overflow question. Saha et al. proposed a support vector machine model that treats the tag recommendation problem as a binary classification problem Saha et al. (2013), by building a per-tag binary classifier that indicates whether a post should be classified as the tag, given a training dataset. For a new post the algorithm generates a list of classification decisions on all of the tags ranking each model based on the similarity to the class (e.g. the class center). The technique reports 70% prediction accuracy for the tags examined.

Fang et al. proposed nonlinear tensor factorization using Gaussian kernel for tag recommendation (Fang et al., 2015), a method similar to support vector machines. Feng and Wang utilized this tensor factorization method in conjunction with personal data to

recommend tags (Feng and Wang, 2012). Topic modeling, a probabilistic technique commonly applied to extracting concepts from a natural language corpus, has also been applied to the tag recommendation problem. For instance, variants of Labeled Latent Dirichlet Allocation (Labeled LDA) have been applied to tag prediction (Boudaer and Loeckx, 2016; Wu et al., 2016) with results showing better prediction for less popular tags and worse for more popular tags. The paper attributes this result to an observation that more popular tags are less meaningful to developers than less popular and more specific tags. Wang et al. proposed a tag recommendation system for software information sites and evaluated the system using the data from 4 software information sites including Stack Overflow (Wang et al., 2014). Their approach combines both Labeled LDA, a Bayesian inference technique and a frequentist modeling technique, and shows that both Bayesian and frequentist modeling techniques can be complementary and help achieve superior tag prediction performance.

Beyer and Pinzger devised multiple strategies to determine whether two tags should be considered as synonyms (Beyer and Pinzger, 2015; 2016). Synonyms proliferate on Stack Overflow because contributors are free to generate new tags, which can result in numerous duplicates. The site has a community driven mechanism to identify tag synonyms, but the process is manual and takes considerable time. In Beyer and Pinzger (2015), presented 9 strategies for tag synonym detection, and in Beyer and Pinzger (2016) 3 of the strategies were supplemented by 6 refined strategies. An important characteristic of these tag synonym detection strategies is that they are based on the tags alone and do not use the Stack Overflow post content. For instance, the *stemming* strategy compares stemmed tags, while strategy *metaphone* compares the pronunciation of two tags. Similarly, in Beyer and Pinzger (2016) Beyer et al. use graph community detection algorithms on a directed graph of Stack Overflow tag relationships. Since they are fairly straightforward, these *rule-based* strategies are computationally efficient to compute while producing reasonable accuracy and effectiveness. The combined strategies recognized 88.4% of tag synonym pairs, using the synonyms identified by Stack Overflow users as a gold set.

A recent article by Nassif et al. proposes an approach that groups Stack Overflow tags into a set of automatically extracted hypernyms Nassif et al. (2018). The approach is based on natural language processing and primarily leverages Wikipedia's corpus to create its groups of hypernyms (e.g. "programming languages", "integrated development environment") to which individual Stack Overflow tags belong to. While this approach is similar to ours in that it organizes Stack Overflow tags, it is different in many ways. Our tag categorization is multi-layered and only consists of Stack Overflow tags, not broader terms. More importantly, our approach also probabilistically models the word distribution of each tag, which can serve additional purposes, such as for tag prediction of an unseen document.

7. Conclusion and future work

Crowd-sourced online documentation has become popular among developers, as they increasingly rely on sources like the Stack Overflow Q&A Forum to solve problems during their daily tasks. In this paper, we describe a concept hierarchy as a convenient and effective means to help utilize the vast and growing resources on Stack Overflow. We posit that exploratory searching and developer-facing recommendation systems can benefit from this concept hierarchy.

The paper describes how to build a concept hierarchy using a hierarchical topic model, where a concept corresponds to a tag-topic pair. After applying a supervised hierarchical topic model to the Stack Overflow dataset, we examine two important problems. First, how do we determine the quality of model? Second, how do we apply the concept hierarchy to the design of search tools, and how do we quantify the potential for aiding search?

For the former, we apply the resulted hierarchical model to identify tag synonyms, and to predict tags for unseen documents. The results show that the model yields improvement in tag synonym identification relative to recently proposed rule based techniques, and an improvement in tag prediction performance to flat models commonly used in the literature. For the later, following recent research in information retrieval, we define both diversity and specificity for searches, terms inspired by work in genome diversity and specificity. We observe that searches generally become more specific following the concept hierarchy. We also observe that by visiting different branches, the searches would increase more diverse set of results. Both of these observations match the needs of information search.

Our work suggests a few future research directions. First, we intend to work towards parallelizing L2H in order to enable the construction of larger models, including more tags, which may reveal interesting associations among rare or underused tags. Second, the vast amount of external documentation resources (e.g. blogs, tutorials, books) are not tagged. We are interested in investigating how we may abstract a set of tags from the Stack Overflow content, which can be useful to build effective search tools for these other untagged sources. Third, a common phenomenon in human tagged documentation is tag explosion where tag spaces continues to grow, and the context of a tag may evolve over time. In this work, we apply a regimented synonym tag identification method (the class boundary is a rigid rectangular shape). A more sophisticated tag synonym identification system can not only improve identification performance but also detect changes of semantic meaning of tags over time, resulting from new trends in software development.

References

Agrawal, A., Fu, W., Menzies, T., 2018. What is wrong with topic modeling? and how to fix it using search-based software engineering. Inform. Softw. Technol. 98, 74–88. doi:10.1016/j.infsof.2018.02.005.

Andrieu, C., de Freitas, N., Doucet, A., Jordan, M.I., 2003. An introduction to MCMC for machine learning. Mach. Learn. 50 (1), 5–43. doi:10.1023/A:1020281327116.

Angel, A., Koudas, N., 2011. Efficient diversity-aware search. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data. ACM, New York, NY, USA, pp. 781–792. doi:10.1145/1989323.1989405.

Athukorala, K., Oulasvirta, A., Głowacka, D., Vreeken, J., Jacucci, G., 2014. Narrow or broad?: Estimating subjective specificity in exploratory search. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. ACM, New York, NY, USA, pp. 819–828. doi:10.1145/2661829.2661904.

Bacchelli, A., Ponzanelli, L., Lanza, M., 2012. Harnessing Stack Overflow for the IDE. In: Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering. IEEE Press, Piscataway, NJ, USA, pp. 26–30.

Baeza-Yates, R.A., Ribeiro-Neto, B., 1999. Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Baltes, S., Dumani, L., Treude, C., Diehl, S., 2018. SOTorrent: Reconstructing and analyzing the evolution of Stack Overflow posts. In: Proceedings of the 15th International Conference on Mining Software Repositories. ACM, New York, NY, USA, pp. 319–330. doi:10.1145/3196398.3196430.

Beyer, S., Pinzger, M., 2015. Synonym suggestion for tags on Stack Overflow. In: Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension. IEEE Press, Piscataway, NJ, USA, pp. 94–103.

Beyer, S., Pinzger, M., 2016. Grouping Android tag synonyms on Stack Overflow. In: 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR), pp. 430–440. doi:10.1109/MSR.2016.051.

Binkley, D., Lawrie, D., Morrell, C., 2017. The need for software specific natural language techniques. Empirical Software Engineering.

Blei, D.M., Griffiths, T.L., Jordan, M.I., 2010. The nested Chinese restaurant process and bayesian nonparametric inference of topic hierarchies. J. ACM 57 (2), 7:1–7:30. doi:10.1145/1667053.1667056.

Blei, D.M., Ng, A.Y., Jordan, M.I., 2003. Latent dirichlet allocation. J. Mach. Learn. Res. 3 (Jan), 993–1022.

Boudaer, G., Loeckx, J., 2016. Enriching topic modelling with users' histories for improving tag prediction in Q&A systems. In: Proceedings of the 25th International Conference Companion on World Wide Web. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, pp. 669–672. doi:10.1145/2872518.2890566.

Campbell, B.A., Treude, C., 2017. NLP2Code: Code snippet content assist via natural language tasks. In: Proceedings of the 2017 International Conference on Software Maintenance and Evolution.

Campos, E.C., de Souza, L.B.L., Maia, M.d.A., 2016. Searching crowd knowledge to recommend solutions for API usage tasks. J. Softw. Evol. Process 28 (10), 863–892 JSME-14-0119.R2 doi:10.1002/smr.1800.

Cha, S.-H., 2007. Comprehensive survey on distance/similarity measures between probability density functions. Int. J. Math. Model. Method. Appl. Sci. 1 (4), 300–307. Available: http://www.naun.org/main/NAUN/ijmmas/mmmas-49.pdf.

Chatterjee, P., Nishi, M.A., Damevski, K., Augustine, V., Pollock, L., Kraft, N.A., 2017. What information about code snippets is available in different software-related documents? An exploratory study. In: Proceedings of the 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 382–386. doi:10.1109/SANER.2017.7884638.

Chen, J., Zhu, J., Lu, J., Liu, S., 2018. Scalable training of hierarchical topic models. Proceed. VLDB Endow. 11 (7), 826–839.

Correa, D., Sureka, A., 2013. Integrating issue tracking systems with community-based question and answering websites. In: 2013 22nd Australian Software Engineering Conference, pp. 88–96. doi:10.1109/ASWEC.2013.20.

Damevski, K., Chen, H., Shepherd, D.C., Kraft, N.A., Pollock, L., 2018. Predicting future developer behavior in the IDE using topic models. IEEE Trans. Softw. Eng. 44 (11), 1100–1111. doi:10.1109/TSE.2017.2748134.

Dumais, S., Chen, H., 2000. Hierarchical classification of Web content. In: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, New York, NY, USA, pp. 256–263. doi:10.1145/345508.345593.

Endres, D.M., Schindelin, J.E., 2003. A new metric for probability distributions. IEEE Trans. Inform. Theory 49 (7), 1858–1860. doi:10.1109/TIT.2003.813506.

Erosheva, E., Fienberg, S., Lafferty, J., 2004. Mixed-membership models of scientific publications. Procee. Natl. Acad. Sci. 101 (suppl 1), 5220–5227. doi:10.1073/pnas.0307760101.

Fang, X., Pan, R., Cao, G., He, X., Dai, W., 2015. Personalized tag recommendation through nonlinear tensor factorization using gaussian kernel. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. AAAI Press, pp. 439–445.

Feng, W., Wang, J., 2012. Incorporating heterogeneous information for personalized tag recommendation in social tagging systems. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, New York, NY, USA, pp. 1276–1284. doi:10.1145/2339530.2339729.

Gelman, A., Carlin, J.B., Stern, H.S., Dunson, D.B., Vehtari, A., Rubin, D.B., 2014. Bayesian Data Analysis, 2. CRC press Boca Raton, FL.

Greco, C., Haden, T., Damevski, K., 2018. StackInTheFlow: behavior-driven recommendation system for stack overflow posts. In: Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings. ACM, New York, NY, USA, pp. 5–8. doi:10.1145/3183440.3183477.

Kairam, S., Riche, N.H., Drucker, S., Fernandez, R., Heer, J., 2015. Refinery: visual exploration of large, heterogeneous networks through associative browsing. Comput. Graph. Forum 34 (3), 301–310. doi:10.1111/cgf.12642.

Lau, J.H., Grieser, K., Newman, D., Baldwin, T., 2011. Automatic labelling of topic models. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1536–1545.

Lin, J., 1991. Divergence measures based on the Shannon entropy. IEEE Trans. Inform. Theory 37 (1), 145–151. doi:10.1109/18.61115.

Martínez, O., Reyes-Valdés, M.H., 2008. Defining diversity, specialization, and gene specificity in transcriptomes through information theory. Procee. Natl. Acad. Sci. 105 (28), 9709–9714.

Mei, Q., Shen, X., Zhai, C., 2007. Automatic labeling of multinomial topic models. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, New York, NY, USA, pp. 490–499. doi:10.1145/1281192.1281246.

Nassif, M., Treude, C., Robillard, M., 2018. Automatically categorizing software technologies. IEEE Trans. Softw. Eng. 1–1 doi:10.1109/TSE.2018.2836450.

Nguyen, T., Rigby, P.C., Nguyen, A.T., Karanfil, M., Nguyen, T.N., 2016. T2API: Synthesizing API code usage templates from english texts with statistical translation. In: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 1013–1017.

Nguyen, V.-A., Boyd-Graber, J., Resnik, P., Chang, J., 2014. Learning a concept hierarchy from multi-labeled documents. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2. MIT Press, Cambridge, MA, USA, pp. 3671–3679.

Paisley, J., Wang, C., Blei, D.M., Jordan, M.I., 2015. Nested hierarchical dirichlet processes. IEEE Trans. Pattern Anal. Mach. Intell. 37 (2), 256–270. doi:10.1109/TPAMI.2014.2318728.

Parnin, C., Treude, C., Grammel, L., Storey, M.-A., 2012. Crowd documentation: Exploring the coverage and the dynamics of API discussions on Stack Overflow. Technical Report. Georgia Institute of Technology. Available: http://chrisparnin.me/pdf/crowddoc.pdf.

Ponzanelli, L., Bavota, G., Di Penta, M., Oliveto, R., Lanza, M., 2014. Mining StackOverflow to turn the IDE into a self-confident programming prompter. In: Proceedings of the 11th Working Conference on Mining Software Repositories, pp. 102–111.

Pritchard, J.K., Stephens, M., Donnelly, P., 2000. Inference of population structure using multilocus genotype data. Genetics 155 (2), 945–959.

Ramage, D., Hall, D., Nallapati, R., Manning, C.D., 2009. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1. Association for Computational Linguistics, pp. 248–256.

Ramage, D., Hall, D., Nallapati, R., Manning, C.D., 2009. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 248–256.

Ramage, D., Manning, C.D., Dumais, S., 2011. Partially labeled topic models for interpretable text mining. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, New York, NY, USA, pp. 457–465. doi:10.1145/2020408.2020481.

Rehurek, R., Sojka, P., 2010. Software framework for topic modelling with large corpora. In: In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. Citeseer.

Rigby, P.C., Robillard, M.P., 2013. Discovering essential code elements in informal documentation. In: Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, Piscataway, NJ, USA, pp. 832–841.

Saha, A.K., Sahay, R.K., Schneider, K.A., 2013. A discriminative model approach for suggesting tags automatically for Stack Overflow questions. In: 2013 10th Working Conference on Mining Software Repositories (MSR), pp. 73–76. doi:10.1109/MSR.2013.6624009.

Schofield, A., Magnusson, M., Mimno, D., 2017. Pulling out the stops: Rethinking stopword removal for topic models. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pp. 432–436.

Shi, Y., Zhao, X., Wang, J., Larson, M., Hanjalic, A., 2012. Adaptive diversification of recommendation results via latent factor portfolio. In: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 175–184.

Shneiderman, B., 2003. The eyes have it: A task by data type taxonomy for information visualizations. In: The Craft of Information Visualization. Elsevier, pp. 364–371.

Silla Jr., C.N., Freitas, A.A., 2011. A survey of hierarchical classification across different application domains. Data Min. Knowl. Discov. 22 (1–2), 31–72. doi:10.1007/s10618-010-0175-9.

Srba, I., Bielikova, M., 2016. Why is Stack Overflow failing? Preserving sustainability in community question answering. IEEE Softw. 33 (4), 80–89. doi:10.1109/MS.2016.34.

Stack Exchange, Inc., 2018a. Create tag synonyms. Available: https://stackoverflow.com/help/privileges/suggest-tag-synonyms, retrieved on February 15, 2018.

Stack Exchange, Inc., 2018b. A proposal for tag hierarchy on Stack Exchange sites – Meta Stack Exchange. Available: https://meta.stackexchange.com/questions/45438/a-proposal-for-tag-hierarchy-on-stack-exchange-sites, retrieved on Feburary 15, 2018.

Stack Exchange, Inc., 2018c. Stack exchange data explorer. Available: https://data.stackexchange.com/, retrieved on February 15, 2018.

Stack Exchange, Inc., 2019. What are tag synonyms and merged tags? how do they work? Available: https://meta.stackexchange.com/questions/70710/what-are-tag-synonyms-and-merged-tags-how-do-they-work, retrieved on May, 2019.

Steinbach, M., Karypis, G., Kumar, V., et al., 2000. A comparison of document clustering techniques. In: KDD workshop on text mining, 400. Boston, pp. 525–526.

Stevens, K., Kegelmeyer, P., Andrzejewski, D., Buttler, D., 2012. Exploring topic coherence over many models and many topics. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 952–961.

Storey, M.-A., Zagalsky, A., Filho, F.F., Singer, L., German, D.M., 2017. How social and communication channels shape and challenge a participatory culture in software development. IEEE Trans. Softw. Eng. 43 (2), 185–204. doi:10.1109/TSE.2016.2584053.

Subramanian, S., Inozemtseva, L., Holmes, R., 2014. Live API documentation. In: Proceedings of the 36th International Conference on Software Engineering. ACM, New York, NY, USA, pp. 643–652. doi:10.1145/2568225.2568313.

Teevan, J., Alvarado, C., Ackerman, M.S., Karger, D.R., 2004. The perfect search engine is not enough: A study of orienteering behavior in directed search. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, New York, NY, USA, pp. 415–422. doi:10.1145/985692.985745.

Treude, C., Barzilay, O., Storey, M.-A., 2011. How do programmers ask and answer questions on the Web?: NIER track. In: 2011 33rd International Conference on Software Engineering (ICSE), pp. 804–807. doi:10.1145/1985793.1985907.

Vasilescu, B., Serebrenik, A., Devanbu, P., Filkov, V., 2014. How social q&a sites are changing knowledge sharing in open source software communities. In: Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing. ACM, New York, NY, USA, pp. 342–354. doi:10.1145/2531602.2531659.

Wang, H., Wang, T., Yin, G., Yang, C., 2017. Linking issue tracker with Q&A sites for knowledge sharing across communities. IEEE Trans. Serv. Comput. PP (99) 1–1 doi:10.1109/TSC.2015.2473847.

Wang, S., Lo, D., Vasilescu, B., Serebrenik, A., 2014. Entagrec: An enhanced tag recommendation system for software information sites. In: 2014 IEEE International Conference on Software Maintenance and Evolution, pp. 291–300. doi:10.1109/ICSME.2014.51.

Wu, Y., Yao, Y., Xu, F., Tong, H., Lu, J., 2016. Tag2Word: Using tags to generate words for content based tag recommendation. In: Proceedings of the 25th ACM Inter-

national on Conference on Information and Knowledge Management. ACM, New York, NY, USA, pp. 2287–2292. doi:10.1145/2983323.2983682.
Xia, X., Lo, D., Wang, X., Zhou, B., 2013. Tag recommendation in software information sites. In: 10th Working Conference on Mining Software Repositories (MSR), pp. 287–296. doi:10.1109/MSR.2013.6624040.
Xu, R., Wunsch II, D., 2005. Survey of clustering algorithms. IEEE Trans. Neural Netw. 16 (3), 645–678. doi:10.1109/TNN.2005.845141.

Hui Chen is an Assistant Professor at the Department of Computer and Information Science, Brooklyn College of the City University of New York and is on the faculty of the Computer Science Ph.D. program at the Graduate Center of the City University of New York. Before that, he was a computer science faculty member at Virginia State University. His research has been in a few research problems in wireless networks, system security, and software analytics. He served on various computer science and computer communications conference technical program committees and as reviewers for journals. He received a Ph.D. in computer science from the University of Memphis in Memphis, Tennessee.

John Coogle is a graduate student majoring in Computer Science at Virginia Commonwealth University. He received a B.S. in Computer Science from Virginia Commonwealth Unviersity and is currently pursuing a Master of Science degree under the supervision of Dr. Kostadin Damevski.

Kostadin Damevski is an Assistant Professor at the Department of Computer Science at Virginia Commonwealth University. Prior to that he was a faculty member at the Department of Computer Science at Virginia State University and a postdoctoral research assistant at the Scientific Computing and Imaging institute at the University of Utah. His research focuses on information retrieval techniques and recommendation systems for software maintenance. Dr. Damevski received a Ph.D. in Computer Science from the University of Utah in Salt Lake City.