Empirical Software Engineering manuscript No.
(will be inserted by the editor)

What Are Developers Talking About? An Analysis of Topics and Trends in Stack Overflow

Anton Barua · Stephen W. Thomas · Ahmed E. Hassan

the date of receipt and acceptance should be inserted later

Abstract Programming question and answer (Q&A) websites, such as Stack Overflow, leverage the knowledge and expertise of users to provide answers to technical questions. Over time, these websites turn into repositories of software engineering knowledge. Such knowledge repositories can be invaluable for gaining insight into the use of specific technologies and the trends of developer discussions. Previous work has focused on analyzing the user activities or the social interactions in Q&A websites. However, analyzing the actual textual content of these websites can help the software engineering community to better understand the thoughts and needs of developers. In the article, we present a methodology to analyze the textual content of Stack Overflow discussions. We use latent Dirichlet allocation (LDA), a statistical topic modeling technique, to automatically discover the main topics present in developer discussions. We analyze these discovered topics, as well as their relationships and trends over time, to gain insights into the development community. Our analysis allows us to make a number of interesting observations, including: the topics of interest to developers range widely from jobs to version control systems to C# syntax; questions in some topics lead to discussions in other topics; and the topics gaining the most popularity over time are web development (especially jQuery), mobile applications (especially Android), Git, and MySQL.

Keywords Q&A Websites · Knowledge Repository · Topic Models · Trend Analysis

Anton Barua
School of Computing, Queen's University, Kingston, Canada K7L3N6
E-mail: barua@cs.queensu.ca

Stephen W. Thomas
School of Computing, Queen's University, Kingston, Canada K7L3N6
E-mail: sthomas@cs.queensu.ca

Ahmed E. Hassan
School of Computing, Queen's University, Kingston, Canada K7L3N6
E-mail: ahmed@cs.queensu.ca

1 Introduction

Today's software engineering field is composed of a diverse array of technologies, tools, languages, and platforms. Developers are expected to be proficient in a wide range of skill sets and paradigms. Even for experienced developers, it can be difficult to keep pace with the rapid speed in which these paradigms evolve.

For these reasons, developers often turn to programming question and answer (Q&A) websites, such as Stack Overflow (2012a), to seek help and advice from their peers about the technical challenges they face. Stack Overflow moderates hundreds of thousands of posts each month from developers with a variety of backgrounds, asking questions about a wide range of topics. Taken in aggregate, these posts act as a record of the history of the needs and thoughts of developers—a knowledge repository of programmers' needs. Analyzing and understanding this knowledge repository could provide key insights about the topics of interest to the developers, such as which development frameworks they prefer and why, what kind of working environments best suits their needs, and what are their favourite target platforms.

Researchers have started to analyze Q&A websites, such as Stack Overflow and Yahoo! Answers (2012), to discover patterns of user behavior (Gyöngyi et al 2008), determine the quality of answers (Shah and Pomerantz 2010), and analyze design features of successful Q&A websites (Mamykina et al 2011). In this work, we propose to analyze the actual text content of the posts—a rich, and so far untapped, source of information—to determine the overarching topics of developer discussion, the trends of the topics, and the interaction patterns between the topics. Understanding these topics could allow programming language and tool developers to understand usage trends, commercial vendors to assess the adoption rate of their products, and Q&A sites to perceive the usage patterns of their information content.

However, analyzing the textual content of such a large knowledge repository poses a number of challenges. The sheer volume of the data prohibits manual analysis, as Stack Overflow contains millions of posts created by tens of thousands of developers. In addition, the unstructured nature of the posts, which are written in natural language, prohibits most conventional data mining techniques from being effective (Hassan 2008).

In this article, we propose a semi-automatic methodology for analyzing such a knowledge repository, with the specific goal of uncovering the main discussion topics, their underlying dependencies, and trends over time. Our methodology is based on latent Dirichlet allocation (Blei et al 2003; Blei and Lafferty 2009), a *statistical topic model* used to automatically recover *topics* (i.e., groups of related words that approximate a real-world concept) from a corpus of text documents. Latent Dirichlet allocation (LDA) has been used successfully in many domains and is routinely applied to millions of documents (Hall et al 2008; Griffiths and Steyvers 2004). We define and apply metrics to the topics discovered by LDA, allowing us to perform quantitative and qualitative evaluations of the knowledge repository. For example, we find that: mobile and web platforms are becoming ever more popular; the .NET framework is dropping in popularity; and the Git version control system is (at the time of writing) just as popular as SVN. We also find that developers discuss a broad range of topics, from C# syntax to mobile development platforms to job-related experience. We provide our dataset and results online to facilitate the replication of our findings (Thomas 2012a).

Our research methodology can be useful to many parties, as we describe in more detail throughout the article. Tool developers can use our technique to fine-tune their tools and decide which technologies and languages to better support. Product managers can use our technique to perform basic and inexpensive market analyses. The Stack Overflow team itself can use our results to better moderate their website. Our approach can also be extended to help automatically generate contextual tags for posts, making searching and archiving of the text content more intuitive for users. Also, our technique can be used to locate posts of similar content, so that users can read all questions and answers that are relevant to their issue.

The rest of the article is organized as follows. We present our research questions, research data, and research methodology in Section 2. The results of our research questions are presented in Section 3. We discuss our findings in a broader context in Section 4. The potential threats to validity are discussed in Section 5. Section 6 outlines related work. Finally, Section 7 offers concluding remarks and avenues for future work.

2 Research Setting

In this section, we detail our research setting. First, we motivate and present four research questions. We then describe our research data, which we use to answer our research questions. Finally, we present our research methodology, including the metrics that we use to analyze our research data.

2.1 Research Questions

RQ1. What are the main discussion topics in Stack Overflow?

Modern developers work on multiple platforms, have a wide array of programming language choices, and use different tools and technologies to fulfill their needs. Programming Q&A websites are designed to cater to the needs of developers facing challenges. Identifying the major discussion topics in such knowledge sharing platforms can help us pinpoint the major areas of interest for developers. This information can be of help to companies so that they can fine tune various aspects of their products. Such improvements can be made in the design of APIs, product documentation, and supporting tools. This information also assists publishers of technical books, as it points out areas in which practitioners have the most interest and questions. Moreover, this information provides software engineering researchers with firsthand knowledge of some of the troublesome areas that are possible future research areas with a high chance of having an impact on practice. The major discussion topics can also highlight the most popular programming languages, tools and technologies.

RQ2. Does a question in one topic trigger answers in another?

We also investigate whether some topics are related to other topics in terms of questions and answers. This can help us identify *closely-coupled* topics, where questions in one topic tend to generate answers in seemingly unrelated topics. Moreover, this can help point out the cross-cutting areas of concerns for developers across different topics: problems so common that they span across multiple domains. For instance, if many questions regarding both mobile application development and web development generate answers related to user interfaces, it hints that user interface development is a cross-cutting concern faced by developers across two different platforms.

RQ3. How does developer interest change over time?

By analyzing the rise and fall of interests in different topics, product developers will be able to assess the relative popularity of their products. This will also help in identifying marketing and research opportunities and trends. For example, if interest in .NET Framework topic is rising while interest in Java topic is dropping, then companies, book publishers, and researchers might want to direct their attention to .NET problems and challenges. The trend analysis also helps in reasoning about the rise or fall of certain topics in developer discussions.

RQ4. How do the interests in specific technologies change over time?

While investigating topic impact over time provides insight into the broad trends of developers' thoughts, we are also interested in the trends of specific technologies, and how interests in related/competing technologies differ over time. For instance, we wish to directly compare the popularity of two scripting languages, Perl and Python. This would give us a more focused view about certain products, and also allow us to compare the impact of a particular technology across multiple topics.

2.2 Research Data

Stack Overflow (2012a) is an online platform where users can exchange knowledge related to programming and software engineering tasks. The website features the ability for users to ask new questions and answer existing questions, as well as to "vote" questions and answers up or down, based on the perceived value of the post. Users of Stack Overflow can earn reputation points and "badges" through various activities. For example, a person is awarded 10 reputation points for receiving an "up" vote on any of their answers, and receives a badge for getting voted 300 times.

Stack Overflow makes its data publicly available in XML format under the Creative Commons license (Stack Overflow 2012b). The dataset is divided into five XML documents: `badges.xml`, `comments.xml`, `posts.xml`, `users.xml` and `votes.xml`. For our purposes, we use `posts.xml`, which contains the actual text content of the posts, as well as the view count, favorite count, post type, creation date, and ID of the user who created each post.

Our Stack Overflow dataset spans 27 months, from July 2008 to September 2010. After pre-processing (described in Section 2.3.2), the lengths of the posts vary between 1 and 3,844 words. The majority of the posts (99.7%) are less than 500 words. In the month of July 2008, Stack Overflow had only 7 posts. This is due to the fact that there was only one day (July 31) of discussion included in the

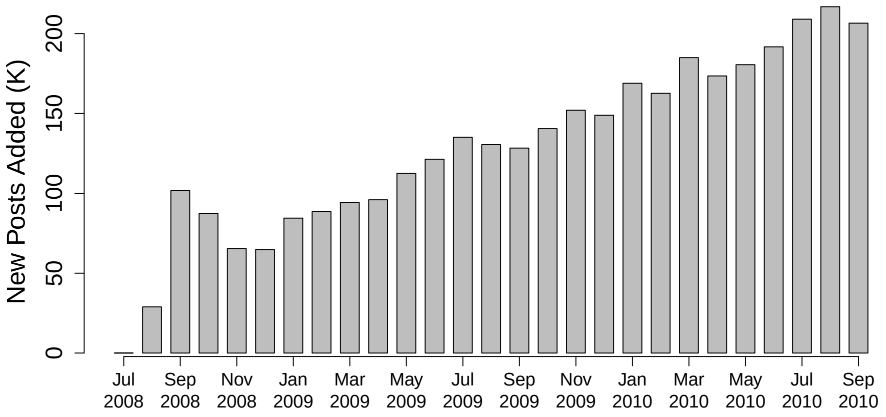


Fig. 1 The number of new posts per month in our Stack Overflow dataset.

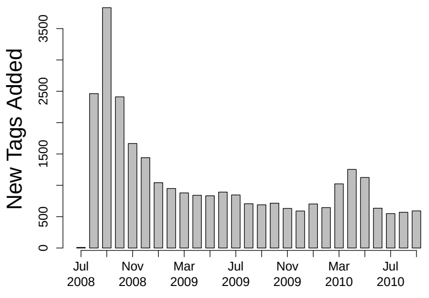
dataset. In contrast, there were 206,371 posts in September 2010. The number of posts per month is shown in Fig. 1. In total, our dataset contains 3,474,987 posts, of which 973,267 (28%) are questions and 2,501,720 (72%) are answers.

2.3 Research Methodology

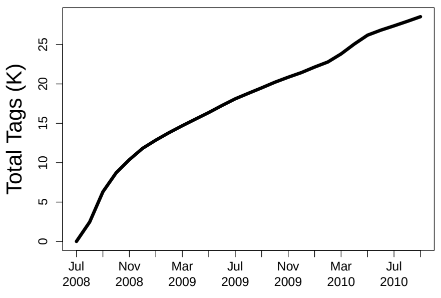
As outlined in our research questions, our research goals involve finding the main discussion topics in Stack Overflow. Stack Overflow currently categorizes each question with user-defined *tags*, which we considered using as a starting point to measure the main discussion topics. However, these tags have several drawbacks which prohibit them from serving our research goals.

First, tags are not predefined and are supplied by the user when posting a question, which can lead to erroneous and inconsistent tags. For example, one user can use the tag “iphone api” and another can use “iphone sdk”, and although the intention is the same (i.e., the posts are related), the posts have distinct tags and will not be grouped together. In fact, this problem has lead to a “tag explosion” (Fig. 2), as the number of tags grow rapidly every month (on average 1,097 tags added per month), but only a fraction (4%) are used to categorize most (90%) of the questions. In addition, since tags can only be applied to the question posts, and not the answer posts, much of the text content is left untagged. Moreover, tags can fail at providing anything but a general sense of the question post. For example, a question tagged “C#” can only tell us that the question is somehow related to the C# language; it cannot tell us whether it is a GUI development problem or a networking related issue. These drawbacks motivate us to evaluate a different approach to discover the discussion topics in Stack Overflow.

In this paper, we use *topic modeling* to discover the discussion topics from the Stack Overflow posts. Topic modeling is an advanced information retrieval technique that automatically finds the overarching *topics* from a given text corpus, without the need for tags, training data, or predefined taxonomies (Griffiths et al 2007; Blei and Lafferty 2009). Topic modeling only uses the word frequencies and



(a) New tags created each month.



(b) Cumulative growth of unique tags.

Fig. 2 Characteristics of the user-generated tags in Stack Overflow. (a) Many new tags are created each month. (b) The number of unique tags continues to grow over time.

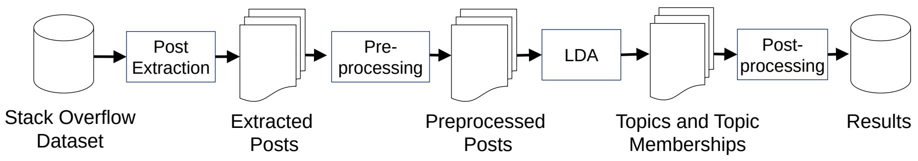


Fig. 3 An overview of our research methodology.

co-occurrence frequencies in the documents themselves to build a model of related words. Using this simple approach, topic modeling has been successfully used in other domains to automatically organize and analyze millions of unstructured documents, for example analyzing scientific literature trends (Griffiths and Steyvers 2004), studying the evolution of a software system (Linstead et al 2008; Thomas et al 2011), and even computer vision (Barnard et al 2003).

We base our research methodology on topic modeling, using the discovered topics as approximations of the discussion topics in which we are interested. Our methodology consists of the following steps (see Fig. 3). First, we extract and preprocess the posts from the Stack Overflow dataset. Second, we apply the topic modeling technique to the extracted and preprocessed posts. Finally, we analyze the discovered topics by defining metrics on their usage and inspecting the metrics over time. We now discuss each step in more detail.

2.3.1 Data Extraction

To extract the posts from the Stack Overflow dataset, we start with the `posts.xml` file from the Stack Overflow data dump, which contains all the user posts (i.e., questions and answers) on Stack Overflow. We separate each individual post into its own document, creating a total of 3,474,987 documents.

In addition to question posts, we consider answer posts in our analysis for a couple of reasons. First, answer posts comprise ~72% of the posts in the dataset, meaning that the majority of text content (which we wish to analyze) is located in the answer posts. Second, one of our research questions aims to discover the

<p>I've been having issues getting the C sockets API to work properly in C++. Specifically, although I am including sys/socket.h, I still get compile time errors telling me that AF_INET is not defined. Am I missing something obvious, or could this be related to the fact that I'm doing this coding on z/OS and my problems are much more complicated? </p>

(a) Before pre-processing.

issu c socket api work properli c++ specif includ sy socket.h compil time error af_inet defin miss obviou relat fact code z os problem complic

(b) After pre-processing.

Fig. 4 (a) An example post from the Stack Overflow dataset (`posts.xml`). (b) The preprocessing step removes HTML tags, removes stop words, and applies a stemming algorithm to each word.

relationship between question topics and answer topics; we must include answer posts to discover these relationships.

For each extracted post, we maintain a record of its metadata, which includes a timestamp, a post type (i.e., either question or answer), the user-specified tags, and, for answer posts, a pointer to the question it is answering and for question posts, a pointer to all its answers.

2.3.2 Data Preprocessing

We cleanse the textual content of the extracted posts in four steps. First, we discard any code snippets that are present in the posts (i.e., enclosed in <`code`> HTML tags), because source code syntax (e.g., `if` statements and `for` loops) introduces noise into the analysis phase. Since all code snippets contain similar programming language syntax and keywords, these do not help topic models to find useful topics (Thomas 2012b). Also, as most source code on Stack Overflow is only shown in small snippets, there is not enough context to allow the extraction of meaningful content from the snippets (Kuhn et al 2007). Next, we remove all HTML tags (e.g., `<b>` and `<a href="...">`), since these are not the focus of our analysis. Third, we remove common English-language stop words such as "a", "the" and "is", which do not help to create meaningful topics (Manning et al 2008). Finally, we apply the Porter stemming algorithm (Porter 1997), which maps words to their base form (e.g.,"programming", and "programmer" both get mapped to "program"). Fig. 4 shows an example post before and after preprocessing. We provide our extracted and preprocessed dataset online (Thomas 2012a).

2.3.3 Topic Modeling

Researchers have developed many topic modeling techniques to account for different data types, assumptions, and goals. In this paper, we use the popular topic modeling technique called *latent Dirichlet allocation* (LDA), as it is best suited for our research goal of finding discussion topics in natural language text documents (Blei et al 2003).

LDA is a *statistical* topic modeling technique, which means that LDA represents topics as probability distributions over the words in the corpus, and it represents documents as probability distributions over the discovered topics. LDA creates topics when it finds sets of words that tend to co-occur frequently in the documents of the corpus. Often, the words in a discovered topic are semantically related, which gives meaning to the topic as a whole. For example, the words with highest probability in a topic might be "planet", "space", "star", and "orbit" (because these words tend to occur together in documents), indicating that this topic is related to astronomy. Further, LDA might tell us that a particular document contains both this astronomy-related topic as well as a mathematics-related topic. Thus, it is now easy to collect all documents related to astronomy, or about mathematics, or about both, without using any training data or manually-created tags or labels.

LDA Implementation. LDA is probabilistic in nature. Given a set of documents, LDA uses machine learning algorithms to infer the topics and topic memberships for each document. In this paper, we use the implementation of the LDA model provided by MALLET version 2.0.6 (McCallum 2002), which is an implementation of the Gibbs sampling algorithm (Geman and Geman 1984). We ran MALLET for 500 Gibbs sampling iterations, after which the Gibbs sampling algorithm has stabilized (Griffiths and Steyvers 2004).

Number of Topics. The number of topics, denoted K, is a user-specified parameter that provides control over the granularity of the discovered topics. Larger values of K will produce finer-grained, more detailed topics while smaller values of K will produce coarser-grained, more general topics. There is no single value of K that is appropriate in all situations and all datasets (Wallach et al 2009; Grant and Cordy 2010). In this paper, we aim for topics of medium granularity, so that the topics capture the broad trends in our dataset while remaining distinct from each other. After experimentation with various values, we set K to 40, which provided the characterization we desired.

Bi-grams. LDA can operate on either the uni-grams (i.e., single words) or n-grams (i.e., sequences of n adjacent words) in the dataset. For example, given the text "compile time error", there are three uni-grams ({"compile", "time", "error"}) and two 2-grams ({"compile_time", "time_error"}). Since 2-grams (equivalently, *bi-grams*) have been shown to increase the quality of text analysis (Tan et al 2002), we use both uni-grams and bi-grams in the creation of the topics. (We accomplish this by passing the `--gram-sizes 1,2` option to MALLET.)

Output of LDA. The result of applying LDA to our preprocessed data is (a) a set of *topics*, defined as distributions over the unique words in the dataset and (b) a set of *topic membership vectors*, one for each post, indicating the percentage of words in the post that came from each of the K topics. As mentioned before, the highest-probable words in a topic are semantically related, which together reveal the nature, or concept, of the topic. For ease of readability, we also manually provide a short label for each topic, for example "SQL" for the topic that has top words "query", "table", "sql", and "row". We choose labels based on the top words in the topics, as well as examining a sample of posts that contain the topics. We also note that automated methods have recently been proposed to assign labels to topics (Mei et al 2007).

2.3.4 Metrics and Analysis

LDA discovers K topics, $z_1, \ldots, z_k$. We denote the membership of a particular topic z_k in document d_i as $\theta(d_i, z_k)$. We note that $\forall i, k : 0 \leq \theta(d_i, z_k) \leq 1$ and $\forall i : \sum_1^k \theta(d_i, z_k) = 1$. Using this notation, we compute the following metrics of interest to help us answer our research questions.

We first define a threshold, δ, to indicate whether a particular topics is "in" a document. Usually, a document will have between 1 and 5 dominant topics, each with memberships of 0.10 or higher (Blei et al 2003). These constitute the main topics in the document. However, due to the probabilistic nature of LDA, sometimes topics are assigned small but non-zero (e.g., 0.01) memberships to a document, and are not relevant to our analysis. Thus, by using the δ threshold as a membership cutoff, we keep only the main topics in each document and discard the probabilistic errors. In this paper, we set δ to 0.10, which we found to remove noisy topic memberships while still allowing only the dominant topics to be present in each document.

Topic share (RQ1). We define the overall ***share*** of a topic z_k across all posts as

$$share(z_k) = \frac{1}{|D|} \sum_{\substack{d_i \in D \\ \theta(d_i, z_k) \geq \delta}} \theta(d_i, z_k) \tag{1}$$

where D is the set of all posts in our dataset. The share metric measures the proportion of posts that contain the topic z_k. For example, if a topic has a share metric of 10%, then 10% of all posts contain this topic. The share metric allows us to measure the relative popularity of a topic across all the posts.

Topic relationships (RQ2). We wish to determine if there is a relationship between topics found in questions and topics found in the corresponding answers. We define a discussion as a single question post along with its answer posts. We define the relationship ***rel*** between two topics in one discussion as z_q and z_a as

$$rel(z_q, z_a) = \sum_{\substack{d_i \in Q, d_j \in A(d_i) \\ \theta(d_i, z_q) \geq \delta \\ \theta(d_j, z_a) \geq \delta}} \theta(d_i, z_q) \times \theta(d_j, z_a) \tag{2}$$

where Q is the set of all question posts and $A(d_i)$ is the set of all answers related to question d_i. We then aggregate across all discussions to generate aggregate ***rel*** values for each answer topic with respect to a certain question topic. We use a weighted metric to quantify the influence of *divergent* answer topics, i.e., those not present in the question. For example, if $\theta(d_q, z_k) = 0.8$ for a question d_q and $\theta(d_a, z_k) = 0.1$ for an answer d_a to the question, then the relatively small metric value of 0.08 would indicate that d_a is divergent from the dominant topic z_k of d_q.

Topic trends over time (RQ3). We also wish to analyze the temporal trends of topics. To do so, we define the ***impact*** of a topic z_k in month m as

$$impact(z_k, m) = \frac{1}{|D(m)|} \sum_{d_i \in D(m)} \theta(d_i, z_k) \tag{3}$$

where $D(m)$ is the set of all posts in month m. The ***impact*** metric measures the relative proportion of posts related to that topic compared to the other topics in that particular month.

Technology trends over time (RQ4). We wish to compare and contrast the trends of related technologies, such as Android vs. iPhone or C# vs. Java. Our topic modeling methodology created K=40 topics, which were appropriate for our previous research questions but are too broad to allow this detailed level of analysis. Further, the user-created tags are too detailed for this analysis (for example, the iPhone technology has many tags, such as "iphone-sdk-3.2" and "iphone-3gs"), in addition to having the limitations listed in Section 2.3. We propose an analysis technique that uses both topic modeling and the user-created tags, which overcomes the aforementioned limitations while enabling us to study the technology trends.

We define a *technology* as a cluster of tags related to a certain technical concept (e.g., iPhone) which are all related to a given topic (e.g., a mobile-app-related topic). To identify a technology, c, related to topic z_k, we manually select the group of most popular tags which are related to technology c and are related to topic z_k. For example, for the iPhone technology, we select tags such as "iphone-sdk-3.2", "iphone-3gs", "iphoneapp", "iphone-sdk-documentation", "iphone", "iphone-os-4.0", and "cocos2d-iphone" from the mobile-app-related topic. Our approach has two key advantages over simply counting the posts related to a tag. First, our approach helps remove noise caused by inappropriately tagged posts, since we only consider a post if it is tagged appropriately *and* actually contains the topic of interest. Second, it provides a more accurate account of trends, since we consider only the proportion of a post related to a topic, instead of the whole post (which can be skewed by posts that are only partially about a given tag).

Specifically, we first identify all tags related to a given topic. We say that a tag g is related to topic z_k if $tag\_score(g, zk) > 0$, where

$$tag\_score(g, z_k) = \sum_{d_i \in D(g)} \theta(d_i, z_k) \tag{4}$$

where $D(g)$ is the set of posts that have been tagged with g.

Next, we intuitively choose a selection of tags that are related to a certain technology. For example, for the iPhone technology, we perform a text search over all tags containing keywords such as: "iphone", "cocoa" (iPhone's UI framework), etc.

Finally, for each month m, with questions $Q(m)$, we define the ***tech_impact*** of a technology c as:

$$tech\_impact(c, z_k, m) = \sum_{d_i \in G(c, z_k, m)} \theta(d_i, z_k) \div \sum_{d_i \in Q(m)} \theta(d_i, z_k) \tag{5}$$

where $G(c, z_k, m)$ denotes all tags related to technology c under topic z_k in month m. This metric measures the amount of discussion about a technology during a month, relative to all discussions in that month. Note that this analysis is restricted to question posts, because Stack Overflow only allows question posts to be tagged. Our approach highlights the disadvantages of the user-created tags that are available with the Stack Overflow dataset. Here, we quantify the *contextual* *value* of a tag by using its membership value assigned by LDA, something that is unavailable with tags alone. In other words, if a user wrongly tags a "Java" question as "C#", then our topic modeling approach would automatically assign a lower value for that tag *in context* of that post. In this way, we minimize the noise created by wrong tags. Moreover, it allows us to see the impact of a technology across different topics.

3 Results

We now present the results of applying our research methodology on our research dataset. Our focus in this section is to succinctly answer each of our research questions. In Section 4, we summarize and discuss these results in a broader context.

3.1 RQ1. What are the main discussion topics in Stack Overflow?

We present the 40 topics discovered by our methodology from Stack Overflow in Tables 1 and 2 (in Appendix A). The topics are arranged in descending order according to their ***share*** metric (Equation 1). For each topic, we also show the proportion of posts that were questions and proportion of posts that were answers. To get a better feel for the topics, we present a subset of the topics along with representative example posts for each.

Web-related discussions. Out of the 40 topics, three topics are specifically about the web, namely: Website Design/CSS, Web Development, and Web Service/Application. Some example posts from these topics are:

> *I want to style the last td in a table without using a css class on the particular td. I want the td containing the text five to not have a border but again I do not want to use a class.*
>
> Topics: Website Design/CSS: 0.92

> *As we know we have JavaScript frameworks like Dojo and some more others. Which is best to use with ASP.NET?*
>
> Topics: Web Development: 1.0

> *Just use add service reference and point it to the wsdl. See how to consume web service.*
>
> Topics: Web Service/Application: 1.0

(In these examples, we show the raw post content, followed by the name and membership of the topics the post contains.)

Data management. We find multiple data management topics in Stack Overflow, including Database Platform, SQL, XML, and Object-Relational Mapping. Database Platform-specific posts contain more specific issues related to the underlying platforms such as MySQL (2012) and SQL Server (2012), than the SQL topic, which focuses on general concepts such as queries and stored procedures. The widespread usage of the XML data format can be seen by the existence of an entire XML topic. The following example posts are from the Database Platform, SQL, and XML topics, respectively:

I want to make a table in SQL Server that will add on insert a auto incremented primary key. This should be an autoincremented id similar to MySQL.

Topics: Database Platform: 0.91

Below is my stored procedure. I want to use stored procedure to select all row of date from tbl_member and insert table but it's not working. Someone can help me?

Topics: SQL: 1.0

Hi, I need to validate a xml file against DTD schema. I found out that I need to pass the source of schema file for validation. Is that possible to make libxml2 find the declaration of schema in XML file and do the validation on its own, or do I have to retrieve the declaration manually? Thanks in advance

Topics: XML: 0.98

Platform-specific discussions. We find discussions pertaining to both development and deployment platforms. Development platforms are the tools, IDEs and languages developers use to build their products, such as Visual Studio (2012) and Java (2012). To this end, we find topics such as .NET Framework, Java, and Windows/Visual Studio. Deployment platforms are the target environments or hardware where the developed software would be deployed (e.g., Windows, mobile devices etc.). To this end, we find the Mobile App Development topic and Web-related topics (Website Design/CSS, Web Development, and Web Service/Application). Note that the Windows/Visual Studio topic is also representative of the Windows deployment platform. Below we present some example posts from these topics.

I have visual studio professional installed. If I install visual studio express for phone will visual studio still work?

Topics: Windows/Visual Studio: 0.95

What would be some compelling reasons to upgrade a web app from .Net 1.1 to .Net 3.5?

Topics: .NET Framework: 0.87, Web Service/Application: 0.13

There's no difference between jdk and java sdk. Both of them mean the same thing. I think it was a PR decision at SUN to change over from jdk to java sdk.

Topics: Java: 0.84, Coding Style/Practice: 0.11

The android sdk contains an emulator. Does your app fail in that?

Topics: Mobile App Development: 0.89, Error/Exception: 0.11

Security. We find two distinct topics that are related to security features of software products: Authentication/Security and Cryptography. Two example posts from these topics are:

Is it possible to get login credentials such as name id if user does login by openid?

Topics: Authentication/Security: 1.0

Use a strong cryptographic hash function like md5 or sha1 but make sure you use a good salt. Otherwise you'll be susceptible to rainbow table attacks.

Topics: Cryptography: 0.94

Quality assurance and collaboration. During the development of a product, revision control (such as Apache SVN (2012) or Git (2012)) is often used to facilitate collaboration and to provide historical tracking of the source code. To this extent, we find a Version Control topic. After developing a software product, post-development tasks, such as testing, are very important for quality assurance. Here we find a Testing topic. Example posts are:

> *I have a Git repository with multiple branches. How can I know which branches are already merged into the master branch?*
>
> Topics: Version Control: 1.0

> *I would like to know how to do performance testing for the old asp pages. Any tools out there that you've used?*
>
> Topics: Testing: 0.83, .NET Framework: 0.17

Knowledge/Experience. We find two topics in Stack Overfow that represents the discussions of programmers about sharing of knowledge and experience. The first such topic is the Job/Experience topic, where programmers discuss the particular processes and practices followed in their workplace, and other experiences related to programming, such as the following question:

> *Do you still see yourself programming in the next years? Who is the oldest developer you know?*
>
> Topics: Job/Experience: 1.0

The second such topic is the Learning topic. This topic contains mostly questions regarding learning languages, technologies or algorithms, as illustrated by the following example:

> *I want to design a website but I don't know from where to start. Is there a beginners guide to start with?*
>
> Topics: Learning: 1.0

General discussions. The top three topics by share are the Coding Style/Practice, Problem/Solution, and QA and Links topics. All three topics are generic English-language topics and do not fit into any technical category. For example, consider the following posts:

> *Is your algorithm readable? Maybe dividing it into several functions would be beneficial for readability and hence maintainability even if it will not reduce duplication.*
>
> Topics: Coding Style/Practice: 0.81, Performance/Optimization: 0.19

> *While googling I found this link. It might be what you're looking for.*
>
> Topics: QA and Links: 1.0

> *I believe something along this line would solve your problem.*
>
> Topics: Problem/Solution: 1.0

The reason behind the higher share for these topics is the natural language conversation style of the Stack Overflow posts. It is expected that the LDA model would discover such topics, as many of the posts contain generic words used in everyday life. However, these topics are not the focus of our analysis, and thus we do not highlight examples from these topics in the rest of our analysis.

Answer Topic	Score
Web Development	12019.7
UI Development	3306.6
Website Design/CSS	3133.1
Function	2283.2
Code Snippets	1901.6

(a) Web Development

Answer Topic	Score
Website Design/CSS	19035.5
Web Development	3295.6
Image/Display	2408.6
UI Development	2157.6
Text-related	2038.3

(b) Website Design/CSS

Answer Topic	Score
Mobile App Development	10008.3
Job/Experience	1700.5
UI Development	1610.8
Image/Display	1541.4
File Operation	1384.0

(c) Mobile App Development

Answer Topic	Score
UI Development	16879.2
Web Development	3475.4
OO Programming	3051.4
Website Design/CSS	2572.8
Image/Display	2488.9

(d) UI Development

Answer Topic	Score
Testing	3888.3
OO Programming	919.6
Job/Experience	887.7
Project/Open source	543.4
Function	505.9

(e) Testing

Answer Topic	Score
Version Control	8786.9
File Operations	1465.4
Project/Open source	1033.3
Job/Experience	940.2
Networking	484.1

(f) Version Control

Answer Topic	Score
Authentication/Security	10811.4
Networking	2516.7
Job/Experience	1638.1
Web Development	1578.5
Database Platform	1446.5

(g) Authentication/security

Answer Topic	Score
Database Platform	15110.4
SQL	5037.6
Object-relational Mapping	2099.9
Performance/Optimization	2055.3
Job/Experience	1554.3

(h) Database Platform

Answer Topic	Score
Job/Experience	28081.7
Learning	6620.1
Project/Open Source	3247.4
Performance/Optimization	2411.8
.NET Framework	1955.5

(i) Job/Experience

Answer Topic	Score
Learning	19429.8
Job/Experience	7168.4
Compiling	2908.9
Function	2766.3
Performance/Optimization	2193.5

(j) Learning

Fig. 5 Selected question topics and the top five triggered answer topics, sorted by their ***rel*** metric score (Equation 2). For each answer topic, the aggregate ***rel*** score over all answer posts related to that question topic is reported.

3.2 RQ2. Does a question in one topic trigger answers in another?

Our first research question discovered the topics of discussion among the users of Stack Overflow. While the results give us a general overview of the discussion topics, we are also interested in discovering the relationships and inter-dependencies among various topics. This can help us group together closely-coupled topics: topics that are apparently distinct from each other, but a question in one topic has the potential to raise answers belonging to a separate topic. This analysis can help us to identify common issues faced by developers across multiple topic domains. It also points out the diversity of a single topic: a topic that groups together knowledge and resources from multiple topics and has the power to generate answers in other topics.

We use the ***rel*** metric (Equation 2) to determine the relationship between pairs of topics. Fig. 5 highlights some of the more interesting relationships. For each question topic, we list the five most related answer topics, sorted in descending order according to their ***rel*** metric score. For all question topics, the highest proportion of answers belong to the question topic itself. However, the more interesting findings are the remaining four answer topics.

For example, from Figs. 5(a), 5(b), and 5(c), it is clear that User Interface-related issues are more pressing than other issues in web and mobile application development, as the top five answer topics of these three question topics (Web Development, Website Design/CSS, Mobile App Development) contain the UI Development topic. Consider the following examples.

Question: *I have an activity in my app, when i call the line EditText username = (EditText)findViewById(R.id.usernameText); the app crashes, why is this?*

Topics: Mobile App Development: 0.89, OO Programming: 0.11

Answer: *The view has not been inflated, or you are missing setContentView(R.layout.layout_file)*

Topics: UI Development: 1.0

Question: *I have a submit button and back button in my asp.net webform. I need to use the submit button when pressing enter, but it's going to the back button instead. Please help...*

Topics: Web Development: 0.85, Problem/Solution: 0.15

Answer: *Wrap your section in a Panel - then you can use the DefaultButton property to set your submit button to be the default when enter is pressed.*

Topics: UI Development: 0.77, Web Development: 0.23

In the first question-answer pair, the asker wanted to determine why his application crashed. In response, one developer pointed out that the crash was indeed due to a missing call to a UI API. In the second question, the asker was confused about the behavior of his ASP.NET web application regarding an "Enter" key input. In response, the answerer shows a way in which a UI widget (i.e., "`Panel`") can be used to solve the problem. This shows that developers might be unaware of certain UI APIs while developing their software. Our findings in Figs. 5(a), 5(b), and 5(c) show that such cases are common among Stack Overflow users. Moreover, this problem is spread over two entirely separate developement platforms: mobile devices and the web.

The UI Development topic itself generates answers in the Object-Oriented Programming topic, indicating to the usage of object-oriented philosophy and design patterns utilized for building user interface widgets (Gamma et al 1995). This may be because many UI widgets, such as buttons, menus, and combo boxes, are implemented as class libraries. Usually, a programmer must use objects created from these class libraries to implement the desired widgets, as the following example illustrates.

Question: *hello all, i want to use a UIview and a UIImage ontop of it with a text box and button i want the user to drag it when the UIview or anything in the view is touched, what would be the best way to implement this method? the UIView is on a single View Controller and using Interface Builder to make my UIviews and UIimages.*

Topics: UI Development: 0.84, Image/Display: 0.13

Answer: *You need to subclass UIView and implement*
*- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event,*
*- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event,*
*- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event, and*
*- (void)touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event.*

Topics: OO Programming: 1.0

Another interesting finding is that the Mobile App Development topic generates answers in the Job/Experience topic (Fig. 5(c)), indicating the rise of mobile devices as a lucrative deployment platform for developers. Consider the following example.

Question: *How do I get paid for developing an app that is free to the public.*

Topics: Mobile App Development: 1.0

Answer: *You usually don't until your project is big enough to bring in hits. Then you make revenue off of advertising (on your site or in your product) or big companies buy the project (i.e. Qt / Novell). But typically free software is free. You may land a job if you're good enough and people recognize you but without a truly unique idea for your software, you're likely to have users find a non-advertisement enabled alternative if they can. It's not beyond the realm of possibility to profit off of free software, but it's no walk in the park. Additionally you can open up donations for your project, but typically those are given under the stipulation that it goes towards making the product better.*

Topics: Job/Experience: 0.74, Project/Open Source: 0.24

Here, a developer wants to know how to generate revenue from free mobile apps. In response, another developer outlines an approach based on his experience. Note that the answerer also hints at a job opportunity.

Two integral components in modern software development process are testing and version control (Pressman 2005; OSS Watch 2012). As we can see from Figs. 5(e) and 5(f), both of these question topics (Testing, Version Control) generate answers in the Project/Open Source topic, which is an indicator of the usage of open source tools in these phases. Also, open source projects themselves thrive on the version control systems. Answers often include the version control practices at the answerer's workplace, explaining the presence of Job/Experience topic in answers to Version Control questions. Most version control systems deal with files and are distributed over a network, explaining the presence of File Operation and Networking topics in the answers. The following two examples illustrate the usage of open source tools and sharing of experience regarding testing and version control.

Question: *Is it possible to get code coverage done by tests using google test framework?*

Topics: Testing: 0.85, QA and Links: 0.15

Answer: *Yes, I've successfully used both free (gcov) and commercial (CTC++) tools. No special steps are needed, just follow the documentation.*

Topics: Project/Open Source: 1.0

Question: *When using Subversion, should developers be working off the trunk or should the trunk be only used for merges from each individual developer's branch and watched by a continuous integration service?*

Topics: Version Control: 0.91

Answer: *I've almost always worked on teams that developed on the trunk–works fine. I'm not saying it's the best idea or anything, just not something that's worth opposing if it's going to get you fired. However, our system is always buildble and often uses CI as well. Each developer must be aware of the update/rebuild/retest/commit cycle (not that it's foolproof, but it works well enough). Hmph, it hurts when I think of how many of our software practices work "Well Enough".*

Topics: Job/Experience: 0.75, Problem/Solution: 0.17

Authentication/Security is a diverse topic, which finds applications to many fields of computer science. From Fig. 5(g) we can see the top answer topics related to Authentication/Security are Networking, Web Development, Jobs/Experience and Database Platform. This highlights the need for experienced security experts across multiple domains as the frequency and severity of attacks such as SQL Injection and Cross Site Scripting (XSS) are on the rise (Neuhaus and Zimmermann 2010). The following example illustrates the concern among Stack Overflow developers about security breaches and the need of qualified personnels.

Question: *I know several people who were in a situation like this. Let's say, you were trying out random sites for basic XSS/SQL Injection vulnerabilities, and you've found one that could be easily compromised. You email the admin/webmaster, but they don't reply. What would you do?*

Topics: Authentication/Security: 0.81, Coding Style/Practice: 0.15

Answer: *I usually look for a way to contact developers or QA personnel and tell them instead. This is easier done when you're active in the developer community. Sometimes admins/webmasters/whatever are actually helpdesk/marketing people who have little or no understanding of the vulnerability, or are otherwise loaded with so much work that they don't realize the gravity of such bugs.*

Topics: Job/Experience: 0.84

The Database Platform topic features answers from the SQL topic (Fig. 5(h)). More interestingly, we also see that the Object-Relational Mapping topic is related to the Database Platform topic, which is a sign of the rising popularity of object oriented databases. We can also see that Database Platform gives rise to answers in the Performance/Optimization topic. This outcome can be due to the large impact of query processing performance on the efficiency of modern data intensive applications. For example, consider the following question-asnwer pair regarding database performance:

Question: *MySQL has special table type MyISAM that does not support transactions. Does Oracle has something like this? I'd like to create write-only database(for logging) that needs to be very fast(will store a lot of data) and doesnt need transactions.*

Topics: Database Platform: 0.89, Performance/Optimization: 0.11

Answer: *Another option if you need extremely high performance is to consider Oracle's TimesTen In-Memory database: http://www.oracle.com/technology/products/timesten/index.html*

Topics: Performance/Optimization: 0.65, Database Platform: 0.35

Most of the aforementioned question topics (Mobile App Development, Testing, Version Control, Authentication/Security, Database Platform) feature answers in the Job/Experience topic (Figs. 5(c), 5(e), 5(f), 5(g), and 5(h)). In fact, Job/Experience related posts themselves are found in a topic of their own (see Fig. 5(i)). We see Performance/Optimization as another answer topic related to Job/Experience, which might actually indicate posts where developers talk about their performances in the workplace. Interestingly, we see both Project/Open Source and the .NET framework, indicating the varied mix of both open source and commercial tools used in the industry. We also see that Job/Experience questions generate answers in the Learning topic, hinting at the fact that the learning process for software engineers continue even after formal education, as the following example shows:

Question: *If you could go back and give yourself one piece of advice at the start of your programming life/career to help you on your way what would it be ?*

Topics: Job/Experience: 1.0

Answer: *Learn Smalltalk. The earlier you really appreciate and understanded object-oriented design the better!*

Topics: Learning: 1.0

We notice that the converse is also true: Learning questions generate answers in the Job/Experience topic (Fig. 5(j)). One reason for this relationship could be questions coming from novice developers being answered by relatively experienced developers, who share insight into their work experience and practices:

Question: *I see a lot of talk on here about functional languages and stuff. Why would you use one over a "traditional" language? What do they do better? What are they worse at? What's the ideal functional programming application?*

Topics: Learning: 0.80, Functions: 0.15

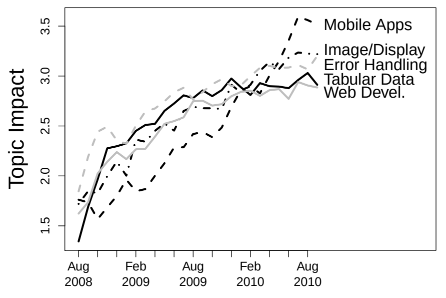
Answer: *My view is that it will catch on now that Microsoft have pushed it much further into the mainstream. For me it's attractive because of what it can do for us, because it's a new challenge and because of the job opportunities it resents for the future. Once mastered it will be another tool to further help make us more productive as programmers.*

Topics: Job/Experience: 0.95

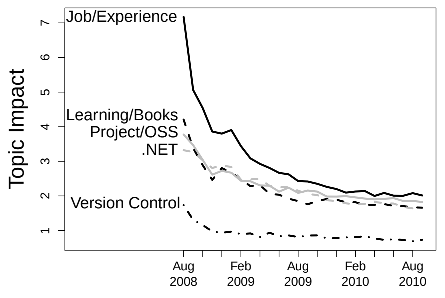
The example highlights the knowledge-sharing nature of Stack Overflow.

3.3 RQ3. How does developer interest change over time?

Computer science is a rapidly changing field, driven by both innovation and consumer demand. Software developers need to be aware of the recent advances in tools and technologies, or risk playing catch-up later. Consequently, we expect developer discussions trends to be symbiotic with the market trends. Table 2

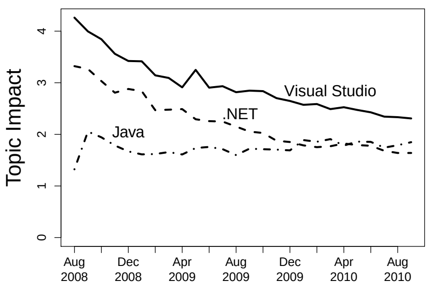


(a) Top 5 increasing trends.

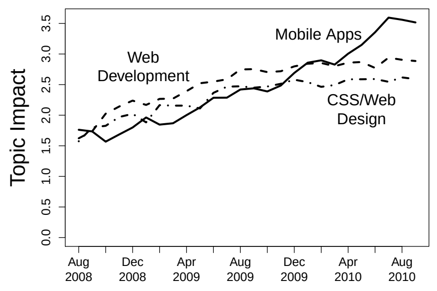


(b) Top 5 decreasing trends.

Fig. 6 The top five increasing and decreasing trends in Stack Overflow, as measured by the percentage change in impact between August 2008 and September 2010.



(a) Development platforms.



(b) Deployment platforms.

Fig. 7 Comparative trend analysis of topics. The impact of a topic for a particular month is calculated using the ***impact*** metric (Equation 3). (a) The .NET platform shows a negative trend compared to the steady trend of Java. (b) Interest in mobile application development is growing at a faster rate than web development.

(Appendix A) displays the trendlines of each topic, using the ***impact*** metric (Equation 3). These trendlines give us an indication of the rise or fall of interest in a particular topic. We now analyze the topics' trends in more detail.

First, we use the Cox Stuart trend test (Cox and Stuart 1955) to determine if each topic's impact metric is *increasing* or *decreasing* over time, to a statistically significant degree, using the standard 95% confidence level. Briefly, the Cox Stuart trend test compares the earlier data points against the later data points in a time series to determine whether its trend is increasing or decreasing, and uses the magnitudes of the differences to determine if the trend is significant. The results of the test are shown in Table 2; we find that 17 topics have an increasing trend, seven have a constant trend (i.e., neither increasing nor decreasing to a significant degree), and 16 have a decreasing trend.

Fig. 6 plots the top 5 increasing and decreasing trends, measured by the largest and smallest percentage differences in ***impact*** scores between the first time period and last time period. Fig. 6(a) shows that the topics with the largest increases over time are Tabular Data-related, Mobile App Development, Image/Display, Web Development, and Error Handling. Each of these topics increased by more than 73%

over the 27 month period, indicating a large increase in developer interest. On the other hand, Fig. 6(b) shows that the topics with the largest decreases over time are Job/Experience, Learning/Books, Version Control, Project/Open Source, and .NET Framework. While these five topics each had a relatively high impact in the beginning of the dataset, they were pushed out by other topics over time, each decreasing in impact by at least 50%. We further discuss these trends in Section 4.

Next, we use the trend data to make a detailed comparison between two major areas of development: development platforms and deployment platforms.

Development platform comparison. Two major development platforms used today are Microsoft .NET and Java (Evans Data Corporation 2011). We compare these two platforms in Fig. 7(a). In the beginning of Stack Overflow, the .NET Framework-related discussions (Windows/Visual Studio Tips and .NET Framework) outnumbered Java related discussions, but over time the trend of .NET related discussions is decreasing.

Target platform comparison. We compare two target platforms, web and mobile devices. As can be seen from Fig. 7(b), both platforms have upward trends. However, despite being a newer platform, the Mobile App Development topic surpasses web related topics in April 2010, signaling the rapid rise of developer interest in mobile application development.

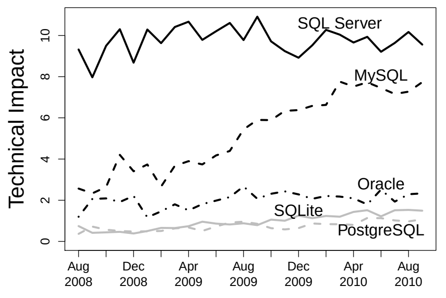
3.4 RQ4. How do the interests in specific technologies change over time?

Comparing the trends of topics against each other gives us valuable insights about the relative popularity of those topics. However, each topic can also be thought as a cluster of competing technologies. Each of these distinct technologies related to a topic has a trend of its own. Comparing trends of technologies related to a particular topic gives us insight of developer interests at a finer grain. In order to perform this fine-grained analysis, we employ the ***tech_impact*** metric (Equation 5) to compare the impact of technologies related to a certain topic.

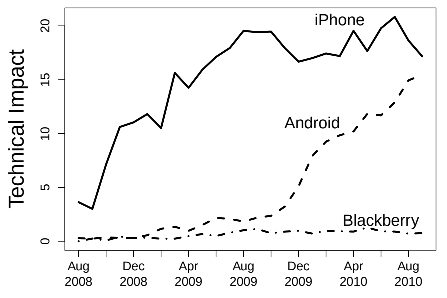
Fig. 8 presents a subset of topics to illustrate the results, namely Database Platform, Mobile App Development, Version Control, Learning, Scripting Language and Web Development. We make a few observations below.

First, we observe that SQL Server and MySQL are the dominant Database Platform technologies (Fig. 8(a)). Moreover, we see an upward trend in the case of MySQL, whereas the trend of SQL Server is relatively constant. Other platforms have relatively steady trends but have lower overall impact.

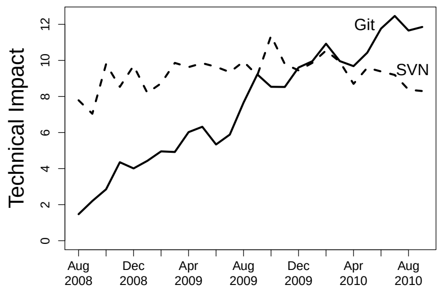
In the case of Mobile App Development, we can clearly see the rise of the Android platform since September 2009 (Fig. 8(b)). This coincides with the release date of the Android 1.5 platform. The iPhone is an older platform than Android, and has a higher impact before September 2009. However, we can see the impact of both platforms converges around September 2010. From this we can say that in about one year, the Android platform has achieved the same impact as iPhone, which may be attributed to the open source nature of the project. In contrast to these two platforms, the Blackberry platform has a steady but very low impact.



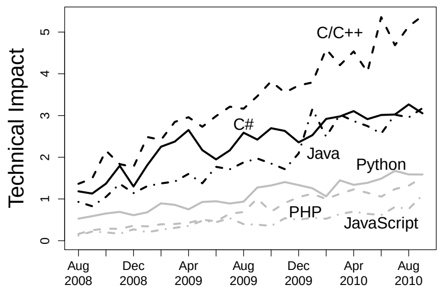
(a) Database platform



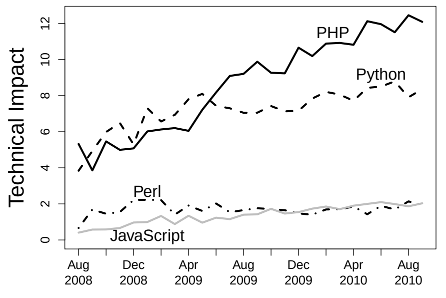
(b) Mobile application



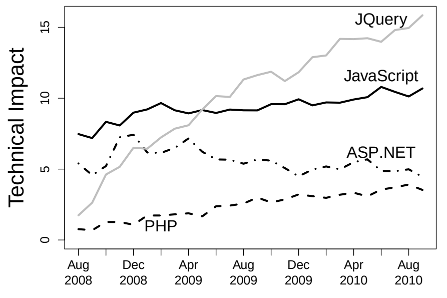
(c) Version control



(d) Learning



(e) Scripting language



(f) Web development

Fig. 8 Focused trend analysis of technologies within specific topics. (a) MySQL has a rising trend and is quickly catching up with SQL Server. (b) Android is rapidly rising. (c) Git has surpassed SVN. (d) Beginners ask more questions about C/C++, C#, and Java than Python, PHP and JavaScript. (e) Python and PHP are the more popular scripting languages. (f) JavaScript has a steady trend, whereas jQuery has a rapidly rising trend in Web Development.

In Fig. 8(c), we plot Git against SVN, two major version control systems. We observe that the impact of Git has steadily increased since 2008, whereas the impact of SVN has been relatively steady. Git surpasses SVN at around January 2010.

Users in Stack Overflow range from novice to professional programmers. Novice programmers often ask questions about learning programming languages. Therefore, we can see which languages novices ask more questions about by plotting the impact of various languages in the Learning topic. We see that C/C++, C#, and Java have higher impact than PHP, Python, and JavaScript (Fig. 8(d)).

We also analyze Scripting Languages (Fig. 8(e)). To be precise, we plot the trends of PHP: a server-side scripting language, Python and Perl: two general-purpose scripting languages and JavaScript: a scripting language for web browsers. Here we see that PHP and Python are the languages of choice. We also note the relative unpopularity of Perl compared to the other general purpose scripting language: Python. However, one surprising result is the relative unpopularity of JavaScript in the Scripting Language topic. This motivates us to further investigate the impact of JavaScript in another topic.

We analyze the different client and server side technologies under the Web Development topic (Fig. 8(f)). Here we can see that JavaScript has higher overall impact than ASP.NET and PHP. It is also interesting to see the rapid rise of jQuery, which is a popular JavaScript library. The added benefits of jQuery over normal JavaScript, such as CSS selectors and ease of adding visual effects to a web page might have contributed to its popularity (jQuery 2012).

4 Discussion

Our analysis of Stack Overflow allows us to gain some insight into the thought process of developers in current software industry. We find that web and mobile applications are emerging as the target platform for developers, yet tool-support in the mobile platform arena may still need to mature. Some common cross-cutting concerns across different platforms demand further research and analysis of those platforms. Also, the knowledge and experience sharing nature of Stack Overflow should motivate researchers and vendors alike to focus on innovative ways to generate documentation by utilizing the knowledge repository. Finally, Stack Overflow itself might benefit from contextual tagging of posts which would lead to a better categorization of the posts.

4.1 The Dominance of the Web

Of all the major themes apparent in technical topics, we find that web-related discussions are the most popular. This is bolstered by the fact that web-related discussions alone are distributed among three topics: Web Development, Website Design/CSS and Web Service/Application, whereas no other theme is spread across more than one topic. We also found web-related keywords in other topics (see Table 1 in Appendix A). The Authentication/Security topic has posts related to secure login, session management, and website cookies; the Scripting Language topic has PHP-related posts; the Java topic has JSP-related posts. It is also clear that JavaScript is currently the main building tool for modern Web 2.0 websites, as we can see several words related to JavaScript ("javascript", "jqueri" and "ajax" in the Web Development topic. The emergence of the MVC (Model-View-Controller) design pattern (Gamma et al 1995) in web applications is also apparent, as we find "mvc" as one of the top ten words in the Web Service/Application topic.

4.2 Towards Mobile Development

We notice that recent surges (Nielsen Company 2012) in the usage of mobile devices, such as iPhone and Android, have created a new deployment platform for the developers, as suggested by the Mobile Application topic. Moreover, we also see the sharp rise of the Mobile Application topic around March 2010, which coincides with the release of iOS 3.2 SDK for the iPad (see Fig. 7(b)). This signifies that developers are increasingly choosing mobile devices as their target platforms over traditional platforms. Together with the three web application topics, this suggests a trend towards deploying software for the "thin client" architecture.

The features of future platforms such as Google Play (2012) are also synonymous with our findings. Usually, these applications leverage the computing power of cloud servers and use the user's computer only as an interface. The "apps" in the mobile devices domain also follow a similar design principle, where small, lightweight software is preferred as the computational power of mobile devices are limited. Even though these results show a rapid growth of developer interest in mobile applications, we note the very limited academic research being done today in developer support for mobile applications.

4.3 Insight into the Usage of Development Tools, Platforms and Technologies

Software developers use a variety of tools, including IDEs, version control systems, and testing frameworks, to support their development process. The selection of these tools are guided by their choice of target platforms, which is often determined by the market trends. Below we discuss our viewpoints based on the analysis of Stack Overflow data.

Among the popular IDEs, we find an entire topic dedicated to Microsoft's flagship IDE: Visual Studio. Moreover, the .NET framework is also discussed in a topic of its own. However, we do not find any other IDEs or frameworks in their own topic. We notice Java-based IDEs, such as Eclipse and Netbeans, and Java-based frameworks, such as Spring and Hibernate, as the top keywords and tags in the Java topic. However, none of these IDEs and frameworks are discussed as much as Visual Studio or .NET and thus does not have a topic dedicated to themselves. The same is applicable for XCode, Apple's IDE for developing iOS apps, and Android, Google's SDK for developing mobile apps. Although we notice "xcode" and "android" in the top tags and keywords, they are yet to become topics themselves, despite the rise of mobile applications. We expect mobile application development tools would generate more discussions in the near future that would create distinct topics for the tools and frameworks.

Although Windows/Visual Studio and .NET Framework are distinct topics, we see a relatively downward trend for these two topics compared to Java when we investigate our third research question (see Fig. 7(a)). One reason for this might be proliferation of the MSDN network (2012), a hub for .NET related documentation, which provides ample resources for a .NET developer. It could be the case that .NET developers do not find satisfactory answers to their problems on Stack Overflow and thus gets redirected elsewhere. On the other hand, perhaps developers are migrating away from .NET because of the commercial, closed-source nature of the framework compared to the free and open source alternatives like Java.

In comparison, Java related discussions have maintained a relatively steady trend after their initial spike in September 2008.

4.4 Common Developer Concerns Across Multiple Domains

We find a number of developer concerns in Stack Overflow discussions that are common across multiple topics. For example, the user interface of a software is a concern that is spread across both web and mobile application development, as we discovered while investigating the second research question. This potentially indicates that user interface design is a general, cross-cutting concern for developers across various platforms. Tool developers and researchers might want to investigate this issue further, based on both user-centric analysis and developer discussions to come up with better tools for UI development. Another cross-cutting concern is authentication and security. We find that it is a concern across multiple application areas such as networks, web applications and database platforms. However, we see the absence of discussions related to security of mobile devices, although Mobile App Development is itself a topic in Stack Overflow. Researchers have already shown interests in the area of mobile device security (Becher et al 2011). While developing truly secure software is a research problem in itself (McGraw 2002), we believe that specifically tailored support built into IDEs, and augmenting the API documentations with additional measures for security would make the process easier for developers and establish some degree of confidence on the security and reliability of the products.

4.5 Embracing Open Source

We find references to open source software across many of the topics that we discovered. For instance, we find that discussions about that the open source Android platform has rapidly caught up with the commercial iPhone SDK after its release. This suggests that developers are quick to embrace an open source platform over a commercial platform. We find similar results in the case of Java and its related tools, which have a slow upward trend compared to the downward trend of the commercial .NET framework. In the database field, we see the rise of the open source MySQL DBMS. In the Version Control topic, we find keywords related to Git and SVN, two open source version control systems. In contrast, we do not find any keywords related to other commercial alternatives, such as Perforce (2012). Moreover, we only find one tag in the top ten tags (`tfs`, referring to Microsoft's Team Foundation Server), whereas there are five tags related to open source version control systems (`svn`, `git`, `mercurial`, `tortoisesvn`, `git-svn`) (see Table 3 in Appendix A). This is an indicator of the maturation of open source software in some of the related fields of software development, namely: Database, IDEs, Version Control and Mobile Applications.

4.6 Platform for Knowledge Sharing and Learning

Although by nature Stack Overflow is dominated by technical topics, we also discover non-technical topics. One of these topics is the Job/Experience topic, where developers discuss about their workplace environment, the current job market and about their career choices. Another topic is the Learning topic, where usually novice developers ask for "beginner's guide" to a particular problem or about certain languages or platform. We began our analysis of Stack Overflow expecting to find solely technical topics. However, these two topics show the diversity and breadth of topics developers talk about. Moreover, in our analysis for the second research question, we found Job/Experience as a secondary answer topic beside the primary answer topic in 6 out of the 10 question topic that we analyzed. This highlights that developers are likely to relate a particular problem to their own experience and try to answer based on that experience.

The topics related to the Learning questions are indicators of the limited availability of knowledge and training for such topics (see Fig. 5(j)). For example, surveys of undergraduate curriculums have shown that there is a dire need for Performance/Optimizations topics (Díaz-Herrera 2005; Dugan 2004). Through our analysis of Stack Overflow, we are able to see firsthand the impact of such a need on everyday practice. Moreover, we observe the limited availability of knowledge about build systems (i.e., the Compiling topic) with many questions leading to questions about them. This discovered relation shows the need for research in build systems as they are an important part of developers work practices, even though there is limited research being conducted on them (McIntosh et al 2011).

4.7 Directions for Generating Better Documentation

Developers come to Stack Overflow in order to find answers for particular problems they face. For example, a developer might post a question in Stack Overflow due to struggling with a problem with his Visual Studio installation. This might motivate tool developers to provide clear and comprehensive documentation for their products. From our analysis of the first research question, we get a bird's-eye view of the areas where developers face problems. For example, the .NET Framework and Windows/Visual Studio are two topics found by our analysis. This indicates that despite a near-comprehensive documentation like MSDN (2012), developers still rely on answers from their peers. This is a possible motivator for generating more user-friendly documentation for the .NET framework. We also get a glimpse of the cross-cutting concerns during our investigation of the second research question. These relations help uncover unexpected relations between topics and point out the need for additional research and documentation for aspects such as UI development and security. In the documentation of frameworks and tools, more information on user interface development might be helpful. We also find Job/Experience as one of the secondary topics in answers to Version Control questions. This is an indicator of the need to document best practices of the use of Version Control systems across software organizations so that developers can benefit from such varied experiences across the industry. Finally, from our broad and focused trend analysis of the third and the fourth research questions, we get a view of the relative popularity of platforms and technologies. This information can be further investigated. For example, by further analyzing the questions related to a particular technology (e.g., PHP) in a domain, researchers and tool developers can discover the common problems faced by developers while using the technology in that domain. This information can be used to generate more focused documentation to address those problems.

4.8 Contextual tagging

Stack overflow allows its users to tag a question with keywords in order to categorize the questions. However, user-created tags lead to a large number of tags that is difficult to manage. By using our topic modeling technique, we categorize posts into broad categories. Moreover, this gives us the basis to investigate the impact of certain technologies across multiple topics. The trend analysis of JavaScript in both Scripting Language and Web Development topic strengthens our assumption that tags alone might not be a viable way to identify topics and track changes in a technology as the tags related to a technology might span multiple topics. By using topic modeling, we can thus identify and compare the impact of a technology in different domains. Sites like Stack Overflow can benefit from this approach by using a hierarchical tagging scheme (Heymann and Garcia-Molina 2006), where the context derived from a question is used to put it in a broader category, and then user-specified tags are used to further categorize that question.

5 Threats to Validity

We now outline the potential internal and external threats to the validity of our results.

Internal Validity. We attempted to minimize the internal threats to validity by using mature tools for extracting and preprocessing the data (i.e., PHP's built-in XML parser, Perl's `Lingua::Stem` stemming library), executing LDA (i.e., MALLET (McCallum 2002)), and computing our metrics of interest (i.e., the R programming environment).

Choosing the optimal parameter settings for LDA is a difficult task with no known general solution (Grant and Cordy 2010; Wallach et al 2009). Although we experimented with different values for the number of topics (K), we cannot be sure that our chosen value was optimal for the analysis we performed. Additionally, our cut-off threshold for document membership, δ, affects the results of our study. However, experimentation found that our results are not particularly sensitive to the exact value of δ, and we note that all topics are subjected to the same threshold.

External Validity. One potential threat to the external validity of our results is that our analysis only includes the dataset of Stack Overflow. Although Stack Overflow is one of the top Q&A websites for developers, further investigation is required before we can determine whether our findings generalize well for other Q&A websites, programming forums, and different developer populations.

The results of our trend analysis are based only on the timeline of the dataset that we have studied. Our methodology does not incorporate a predictive model, and thus can not be used to make predictions about future data (e.g., changes brought about by radical new technologies or the demise of certain products).

The increase in a topic's activity over time could be caused by a number of different factors: (a) the topic is new/popular and developers enjoy discussing it; (b) the topic has poor documentation and is difficult to understand, so developers must seek help online. (Note that in the latter case, the topic must still be popular enough for developers to still consider it worth discussing.) Our methodology and results do not distinguish between these two related cases, although it may be useful to some parties to do so.

6 Related Work

We summarize related work in four categories: the general study of Q&A websites; the study of Stack Overflow specifically; the study of other social platforms for developers; and the use of LDA to study trends in software engineering data.

General Q&A Websites. Previous work has focused on analyzing general Q&A websites based on users' social interactions. Gyöngyi et al. (2008) analyze several aspects of user behavior in Yahoo! Answers, a Q&A website for the general public. The authors use the number of questions and answers in each predefined top-level category to determine the popularity of each category. Adamic et al. (2008) also analyze Yahoo! Answers to cluster the top-level categories into three broader categories using both content and user interactions. In contrast to these efforts, instead of using existing tags, we use a statistical topic model, LDA, to automatically discover topics from the textual content of the posts and employ temporal measures to identify a topic's popularity over time.

Stack Overflow. Researchers have started to analyze Stack Overflow to categorize its questions (Treude et al 2011) and identify its design features (Mamykina et al 2011). Treude et al. (2011) analyze Stack Overflow to find topics and to categorize the question into distinct types, such as "how-to", "discrepancy", "environment", etc. The authors apply their analysis to 15 days worth of posts, using user-created tags to identify the topics. They manually code the questions based on a random sample of the data. In contrast, we apply our automated method, based on LDA, to 27 month's worth of posts and use tags as a secondary basis to deduce trends of different technologies under broader topic categories. Mamykina et al. (2011) identify the core design features that led to the popularity of Stack Overflow: the reputation system based on points, the strong involvement of the design team with the community, and the single-domain focus. Further, the authors categorize users based on their frequency of activity on Stack Overflow (e.g., community activists, low profile users). Instead of user activity, we focus on the textual content generated by the users in order to extract the major topics of discussion.

Other Social Platforms. Researchers have analyzed other datasets and social platforms, such as code search engine usage logs and developer blogs, to find out the topics in which developers are interested. In particular, Bajracharya and Lopes (2012) analyze the log of a popular code search engine to discover major code search topics. Similar to our technique, they apply LDA on the usage log of the code search engine. Some of the topics found by their analysis are aligned with our findings (e.g., data structures, files, GUI, networking, parsing/compiling, security, and string). However, their analysis is based on a specific group of developers, namely Java programmers. In contrast, our analysis takes into account developers using a myriad of different programming languages and platforms. Further, we analyze both questions and answers, whereas the aforementioned work analyzes only the question content (i.e., search queries). Moreover, their study is focused on a specific need of developers: finding source code examples for a particular problem. In contrast, our analysis of a community based Q&A website addresses developer needs from a broader perspective.

In another study, Pagano and Maleej (2012) analyze the blogging activity of developers using topic models to find topics in those blogs. In particular, they analyze blogs written by developers who are active committers to a certain code base (e.g., Eclipse, PostgreSQL, GNOME, and Python). In contrast, our study takes into account developers with diverse backgrounds and varying levels of experience.

Latent Dirichlet Allocation. LDA has been previously employed to analyze the trends present in software engineering data. Hindle et al. (2009) apply LDA to the commit log messages in a version control system in an effort to determine what topics are being worked on by developers at any given time, and to see how development trends are changing. Neuhaus and Zimmerman (2010) apply LDA on a well-known vulnerability database to find the trends in specific security vulnerabilities over time. Thomas et al. (2010) apply LDA to analyze software evolution, and later propose a variant of LDA that can better detect topic trends in source code (2011). Our approach is different from these three aforementioned approaches in that we apply LDA in a totally different domain, that of a developer Q&A website. LDA also helps us to overcome the issues with tags mentioned in Section 2.3. In addition, we introduce techniques to find the question/answer relationship between pairs of topics and determine the impact of individual technologies based on their underlying topics.

7 Conclusion and Future Work

In this paper, we proposed a methodology to discover and quantify the topics and trends in Stack Overflow, a popular Q&A website with millions of active users. Our methodology is based on LDA, a widely-applied statistical topic model, which discovers *topics* from the textual content of Stack Overflow. We define various metrics to quantify the topics and their changes over time, which allows us to gain insight into the discussions in Stack Overflow.

Our analysis provides an approximation of the wants and needs of the contemporary developer. We have found, for example, that mobile application development is on the rise, faster than web development. Both Android and iPhone development is much more prevalent than Blackberry development. The PHP scripting language is becoming very popular, much more so than, say, Perl. Java is a constant player in the field of programming languages and APIs, whereas the .NET framework is decreasing slightly. Git has surpassed SVN in the VCS popularity contest, and MySQL is the hottest DBMS of the last few years.

Tool developers can use the results of our analysis to fine-tune their tools, or decide which technologies to support. For example, the increasing popularity of the

Git suggests that Git should be supported by modern development environments. Additionally, the recent popularity gains of Android suggests that developers need better tool support for developing applications for Android.

Our analysis can be used by the Stack Overflow team to better understand the content generated by its users. Knowing what topics are present, and which are popular at any given time, could help in the moderation of the website. For example, if 90% of the current posts are about only two topics, then Stack Overflow can create spin-off Q&A websites specifically for these topics.

We can extend this work in many avenues. We plan to perform topic analysis of the questions and answers separately, so that we might find different topics from the overall analysis. We also plan to apply LDA to a smaller time interval of posts (e.g., 1–3 months), which may help reveal topics that emerge only during these months but later disappear. We plan to experiment with other values of K, in an effort to find finer-grained topics, which may reveal additional insight. Our technique can be augmented with statistical models proposed by other researchers (Shah and Pomerantz 2010) to determine the quality of an answer. Our methodology can also be applied to other developer resources, such as web portals, blogs, and forums; we can cross-reference these resources with Stack Overflow to determine whether similar trends hold in those mediums.

8 Acknowledgements

We would like the thank the anonymous reviewers for their valuable feedback.

A Topic Listings

Table 1 The 40 topics discovered by LDA. In addition to the top words in the topic as discovered by LDA, we provide a manually-created topic name to briefly describe each topic.

Topic Name	Top LDA words
Coding Style/Practice	*code case design differ implement approach write depend exampl*
Problem/Solution	*work problem code solut fine issu solv work_fine wrong idea*
QA and Links	*question answer post comment articl edit link exampl find read*
Function	*function type variabl valu paramet call pass return argument*
Job/Experience	*develop work time peopl softwar year dai compani product job*
OO Programming	*class method object call implement instanc interfac creat static*
UI Development	*control view event button click set user tab creat add*
File Operation	*file directori path folder creat read upload open write copi*
Website Design/CSS	*css browser element div style html firefox flash work javascript*
String Manipulation	*string charact encod format space convert word quot input text*
Learning	*languag program learn book c++ c# java c write read*
Code Snippet	*time call code loop second execut block check set befor*
Database Platform	*databas tabl sql data server insert db sql_server record updat*
Numerical Operation	*valu number arrai integ bit result int calcul time oper*
Web Development	*form javascript jqueri ajax valid function submit load post call*
Compiling	*compil command librari c c++ linux code program rail rubi*
Image/Display	*imag color draw point size screen anim set background displai*
SQL	*queri tabl sql result row join select return column procedur*
Networking	*server connect client request http send host network respons port*
Authentication/Security	*user session site secur access password account authent login cooki*
Object-Rel. Mapping	*object properti model set entiti valu field map attribut creat*
Performance/Optimization	*perform time optim faster larg memori size speed slow cach*
Text-related	*text document tag html content templat search link pdf editor*
Windows/VS Tips	*window instal visual studio applic visual_studio dll debug set*
Data Structure/Alg.	*list item element sort arrai node order iter collect tree*
Mobile App Development	*app applic iphon devic video android api develop plai phone*
Project/Open Source	*project sourc build tool code open product sourc_code open_sourc*
.NET Framework	*version net support framework c# assembl compon featur librari*
Scripting Language	*script php python modul packag perl instal django import function*
Tabular Data-related	*select row column valu tabl field cell displai excel box*
Thread/Process	*thread process messag lock email send task queue call wait*
Web Service/Application	*servic web applic mvc app web_servic wcf site web_applic client*
Memory/Pointer	*memori pointer object alloc arrai stack address refer copi buffer*
Error/Exception	*error log messag code throw fail warn error_messag catch debug*
Regular Expression	*url match express regex regular rout regular_express rule pattern*
Java	*java eclips plugin applic jar spring configur class servlet maven*
Cryptography	*data kei store hash structur encrypt dictionari valu read gener*
XML	*xml cach serial pars namespac json xml_file attribut element*
Version Control	*chang repositori branch svn commit version git merg work control*
Testing	*test unit code unit_test write mock work case test_code fail*

Table 2 Topic share and trends. We show the ***share*** metric (Equation 1), the percentage of posts containing the topic that which questions (Q) and answers (A), the trend (using the Cox Stuart trend test with α=0.05) and the trendline of the topic's impact over time.

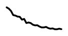
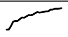
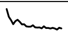
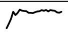
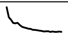
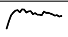
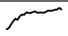
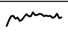
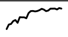
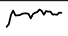
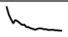
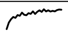
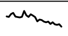
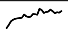
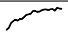
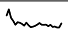
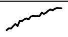
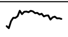
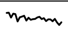
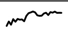
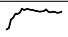
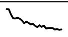
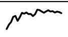
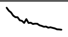
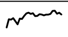
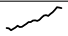
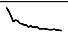
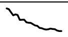
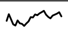
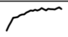
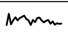
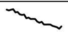
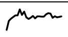
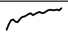
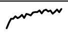
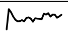
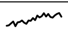
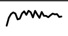
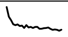
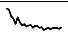
Topic Name	***share*** (%)	Q (%)	A (%)	Trend	Trendline
Coding Style/Practice	4.5	14.3	85.7	⇓	
Problem/Solution	4.5	36.5	63.5	⇑	
QA and Links	3.8	17.3	82.7	⇓	
Function	3.4	19.7	80.3	⇑	
Job/Experience	3.3	17.6	82.4	⇓	
OO Programming	3.2	21.7	78.3	⇓	
UI Development	3.1	36.9	63.1	⇑	
File Operation	2.8	31.4	68.6	–	
Website Design/CSS	2.6	30.5	69.5	⇑	
String Manipulation	2.5	23.0	77.0	–	
Learning	2.5	19.8	80.2	⇓	
Code Snippet	2.4	22.8	77.2	⇑	
Database Platform	2.4	32.4	67.6	⇓	
Numerical Operation	2.4	24.2	75.8	⇑	
Web Development	2.3	31.8	68.2	⇑	
Compiling	2.3	24.5	75.5	⇓	
Image/Display	2.3	39.1	60.9	⇑	
SQL	2.2	30.1	69.9	–	
Networking	2.2	30.1	69.9	⇓	
Authentication/Security	2.1	28.2	71.8	⇑	
Object-Rel. Mapping	2.1	32.7	67.3	–	
Performance/Optimization	2.1	18.9	81.1	⇓	
Text-related	2.1	32.9	67.1	⇑	
Windows/VS Tips	2.1	32.7	67.3	⇓	
Data Structure/Alg.	2.0	25.2	74.8	⇑	
Mobile App Development	1.9	41.1	58.9	⇑	
Project/Open Source	1.9	23.6	76.4	⇓	
.NET Framework	1.9	22.5	77.5	⇓	
Scripting Language	1.8	27.9	72.1	⇑	
Tabular Data-related	1.8	45.4	54.6	⇑	
Thread/Process	1.6	25.3	74.7	⇓	
Web Service/Application	1.6	36.3	63.7	⇓	
Memory/Pointer	1.6	17.5	82.5	–	
Error/Exception	1.6	40.7	59.3	⇑	
Regular Expression	1.6	23.3	76.7	⇑	
Java	1.5	28.8	71.2	–	
Cryptography	1.3	24.6	75.4	⇑	
XML	1.1	27.0	73.0	–	
Version Control	1.1	24.1	75.9	⇓	
Testing	0.9	21.5	78.5	⇓	

Table 3 The top Stack Overflow tags related to each topic. For each topic, tags are ordered by the ***tag_score*** metric in descending order.

Topic Name	Top Stack Overflow Tags
Coding Style/Practice	c# java .net c++ php javascript python
Problem/Solution	c# php javascript jquery javaasp.net iphone c++
QA and Links	c# php java .net c++ javascript asp.net
Function	c# c++ javascript php java .net c
Job/Experience	c# php java .net asp.net sqlcareer-development mysql
OO Programming	c# java .net c++ objective-c php python
UI Development	c# iphone wpf asp.net jquery javascript .net
File Operation	c# php java asp.net .net python c++
Website Design/CSS	css jquery javascript html flash internet-explorer asp.net
String Manipulation	c# php java regex string python
Learning	c# c++ java c python books .net
Code Snippet	c# java php javascript .net jquery c++
Database Platform	sql-server sql mysql database c#php sql-server-2005 oracle
Numerical Operation	c# php java c++ c arrays python
Web Development	jquery javascript ajax asp.net php html c#
Compiling	c++ c linux ruby-on-rails ruby java gcc
Image/Display	iphone c# android image java javascript wpf
SQL	sql mysql sql-server c# php tsql linq
Networking	c# java php asp.net .net sockets http
Authentication/Security	php asp.net c# security ruby-on-railsjava authentication .net
Object-Rel. Mapping	c# net java nhibernate ruby-on-rails asp.net-mvc
Performance/Optimization	c# performance java .net c++ php algorithm
Text-related	html php c# javascript asp.net java jquery
Windows/VS Tips	c# .net windows visual-studio c++ visual-studio-2008 asp.net
Data Structure/Alg.	c# java python php jquery c++
Mobile App Development	iphone android objective-c iphone-sdk java c# xcode
Project/Open Source	c# .net java visual-studio asp.netc++ visual-studio-2008 msbuild
.NET Framework	c# .net asp.net delphi java vb.net reporting-services
Scripting Language	php python perl django mysqljavascript apache linux
Tabular Data-related	c# asp.net jquery javascript iphone excel php
Thread/Process	multithreading c# java .net c++ email php
Web Service/Application	c# asp.net wcf .net web-services asp.net-mvc java
Memory/Pointer	c++ c c# iphone objective-c java memory-management
Error/Exception	c# java php asp.net c++ .net python
Regular Expression	regex php c# javascript asp.net mod-rewrite java
Java	java eclipse spring maven-2 tomcatjsp hibernate netbeans
Cryptography	c# java php .net python iphone encryption
XML	xml c# java xslt .net php
Version Control	svn git version-control mercurial tortoisesvn branch merge
Testing	unit-testing c# testing java .net ruby-on-rails tdd

References

Adamic LA, Zhang J, Bakshy E, Ackerman MS (2008) Knowledge sharing and Yahoo Answers: Everyone knows something. In: Proceedings of the 17th International Conference on World Wide Web, pp 665–674

Apache Subversion (2012) URL http://subversion.apache.org/

Bajracharya S, Lopes C (2012) Analyzing and mining a code search engine usage log. Empirical Software Engineering 17:424–466

Barnard K, Duygulu P, Forsyth D, De Freitas N, Blei DM, Jordan MI (2003) Matching words and pictures. The Journal of Machine Learning Research 3:1107–1135

Becher M, Freiling FC, Hoffmann J, Holz T, Uellenbeck S, Wolf C (2011) Mobile security catching up? Revealing the nuts and bolts of the security of mobile devices. In: IEEE Symposium on Security and Privacy, pp 96–111

Blei DM, Lafferty J (2009) Topic Models. Text Mining: Theory and Applications. Taylor and Francis, London, UK

Blei DM, Ng AY, Jordan MI (2003) Latent Dirichlet allocation. Journal of Machine Learning Research 3:993–1022

Cox D, Stuart A (1955) Some quick sign tests for trend in location and dispersion. Biometrika 42(1/2):80–95

Díaz-Herrera JL (2005) Computing & Information Sciences: The discipline, careers, and future directions. In: ACM Southeast Regional Conference

Dugan RF (2004) Performance lies my professor told me: The case for teaching Software Performance Engineering to undergraduates. In: Proceedings of the 4th International Workshop on software and Performance, pp 37–48

Evans Data Corporation (2011) Software development platforms - 2011 rankings. URL http://www.evansdata.com/reports/viewRelease_download.php?reportID=19

Gamma E, Helm R, Johnson R, Vlissides J (1995) Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Boston, MA

Geman S, Geman D (1984) Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Transactions on Pattern Analysis and Machine Intelligence pp 721–741

Git SCM (2012) URL http://git-scm.com/

Google Play (2012) URL https://play.google.com/about/features

Grant S, Cordy JR (2010) Estimating the optimal number of latent concepts in source code analysis. In: Proceedings of the 10th International Working Conference on Source Code Analysis and Manipulation, pp 65–74

Griffiths TL, Steyvers M (2004) Finding scientific topics. Proceedings of the National Academy of Sciences 101:5228–5235

Griffiths TL, Steyvers M, Tenenbaum JB (2007) Topics in semantic representation. Psychological Review 114(2):211–244

Gyöngyi Z, Koutrika G, Pedersen J, Garcia-Molina H (2008) Questioning Yahoo! Answers. In: Proceedings of the 1st Workshop on Question Answering on the Web

Hall D, Jurafsky D, Manning CD (2008) Studying the history of ideas using topic models. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp 363–371

Hassan AE (2008) The road ahead for mining software repositories. In: Frontiers of Software Maintenance, pp 48–57

Heymann P, Garcia-Molina H (2006) Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10, Stanford InfoLab, URL http://ilpubs.stanford.edu:8090/775/

Hindle A, Godfrey MW, Holt RC (2009) What's hot and what's not: Windowed developer topic analysis. In: Proceedings of the 25th International Conference on Software Maintenance, pp 339–348

jQuery (2012) URL http://docs.jquery.com/How_jQuery_Works

Kuhn A, Ducasse S, Girba T (2007) Semantic clustering: Identifying topics in source code. Information and Software Technology 49(3):230–243

Linstead E, Lopes C, Baldi P (2008) An application of latent Dirichlet allocation to analyzing software evolution. In: Proceedings of the 7th International Conference on Machine Learning and Applications, pp 813–818

Mamykina L, Manoim B, Mittal M, Hripcsak G, Hartmann B (2011) Design lessons from the fastest Q&A site in the west. In: Proceedings of the Conference on Human Factors in Computing Systems, pp 2857–2866
Manning CD, Raghavan P, Schtze H (2008) Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA
McCallum A (2002) MALLET: A machine learning for language toolkit. URL http://mallet.cs.umass.edu
McGraw G (2002) Building secure software: Better than protecting bad software. IEEE Software 19(6):57–58
McIntosh S, Adams B, Nguyen TH, Kamei Y, Hassan AE (2011) An empirical study of build maintenance effort. In: Proceedings of the 33rd International Conference on Software Engineering, pp 141–150
Mei Q, Shen X, Zhai C (2007) Automatic labeling of multinomial topic models. In: Proceedings of the 13th International Conference on Knowledge Discovery and Data Mining, pp 490–499
Microsoft Developer Network (2012) URL http://msdn.microsoft.com/en-us/
Microsoft SQL Server (2012) URL http://www.microsoft.com/sqlserver/en/us/default.aspx
Microsoft Visual Studio (2012) URL http://msdn.microsoft.com/en-us/vstudio
MySQL (2012) URL http://www.mysql.com/
Neuhaus S, Zimmermann T (2010) Security trend analysis with CVE topic models. In: Proceedings of the 21st International Symposium on Software Reliability Engineering, pp 111–120
Nielsen Company (2012) The mobile media report: State of the media. URL http://www.nielsen.com/content/dam/corporate/us/en/reports-downloads/2011-Reports/state-of-mobile-Q3-2011.pdf
Oracle Java (2012) URL http://www.java.com/en/
OSS Watch (2012) Essential tools for running a community-led project. URL http://www.oss-watch.ac.uk/resources/communitytools.xml
Pagano D, Maalej W (2012) How do open source communities blog? Empirical Software Engineering pp 1–35
Perforce (2012) URL http://www.perforce.com/
Porter MF (1997) An algorithm for suffix stripping. In: Readings in Information Retrieval, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 313–316
Pressman RS (2005) Software Engineering: A Practitioner's Approach. McGraw-Hill
Shah C, Pomerantz J (2010) Evaluating and predicting answer quality in community QA. In: Proceedings of the 33rd International Conference on Research and Development in Information Retrieval, pp 411–418
Stack Overflow (2012a) URL http://www.stackoverflow.com
Stack Overflow (2012b) Stack Overflow Creative Commons License data dump. URL http://blog.stackoverflow.com/2009/06/stack-overflow-creative-commons-data-dump/
Tan C, Wang Y, Lee C (2002) The use of bigrams to enhance text categorization. Information Processing and Management 38:529–546
Thomas SW (2012a) URL http://research.cs.queensu.ca/∼sthomas/
Thomas SW (2012b) Mining software repositories with topic models. Tech. Rep. 2012-586, School of Computing, Queen's University
Thomas SW, Adams B, Hassan AE, Blostein D (2010) Validating the use of topic models for software evolution. In: Proceedings of the 10th International Working Conference on Source Code Analysis and Manipulation, pp 55–64
Thomas SW, Adams B, Hassan AE, Blostein D (2011) Modeling the evolution of topics in source code histories. In: Proceedings of the 8th Working Conference on Mining Software Repositories, pp 173–182
Treude C, Barzilay O, Storey M (2011) How do programmers ask and answer questions on the web? In: Proceedings of the 33rd International Conference on Software Engineering, pp 804–807
Wallach HM, Murray I, Salakhutdinov R, Mimno D (2009) Evaluation methods for topic models. In: Proceedings of the 26th International Conference on Machine Learning, pp 1105–1112
Yahoo! Answers (2012) URL http://answers.yahoo.com