Avrupa Bilim ve Teknoloji Dergisi
Sayı 19, S. 354-365, Ağustos 2020

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
No. 19, pp. 354-365, August 2020

Research Article

What Java Developers have talked about? An empirical study on Stack Overflow

Ahmet Samet Şahin [1,2,*], Nilgün Güler Bayazıt [2]

[1] TUBITAK Informatics And Information Security Research Center, Kocaeli, Turkey (ORCID: 0000-0002-8315-1482)
[2] Mathematical Engineering Department, Yıldız Technical University, ˙Istanbul, Turkey ORCID: 0000-0003-0221-294X)

(İlk Geliş Tarihi 12 Mart 2020 ve Kabul Tarihi 7 Haziran 2020)

(**DOI:** 10.31590/ejosat.702949)

ATIF/REFERENCE: Şahin, A. S. & Güler Bayazıt, N. (2020). What Java Developers have talked about? An empirical study on Stack Overflow. *Avrupa Bilim ve Teknoloji Dergisi*, (19), 354-365.

Abstract
Java has been a widely used programming language for a long time in various fields. Java and its libraries have been frequently updated for various reasons including bugs, change requests, performance and usability requirements and so on. In this paper, we examine how these changes affect the use of Java and analyze trends in its usage. As a data source, we use the Stack Overflow public dataset which is the largest online Q&A site about software technologies. We firstly employ a practical approach to extract the Java-related posts from the Stack Overflow dataset using cosine similarity and compare it with previous works. We then apply Latent Dirichlet Allocation (LDA) to the corpus for topic modelling. We divided the data set into two-year periods to obtain consistent clusters. After obtaining main topics, we examine topics and keywords on a two-year basis. Finally, unlikely previous works, we manually classify topics into two as "domain-specific" and "development environment" and investigate tendencies of these classes to change in both the short term and the long term.

Keywords: Topic Modelling, Latent Dirichlet Allocation, Java, Stack Overflow

Java Geliştiricileri Stack Overflow Üzerinde Ne Hakkında Konuşuyor? Deneysel Bir Çalışma

Öz

Java, çeşitli alanlarda uzun zamandır yaygın olarak kullanılan bir programlama dilidir. Java ve kütüphaneleri; hatalar, değişiklik istekleri, performans ve kullanılabilirlik gereksinimleri vb. gibi çeşitli sebeplerle sık sık güncellenmektedir. Bu makalede, bu değişikliklerin Java kullanımını nasıl etkilediği ve kullanımındaki eğilimler analiz edilmiştir. Veri kaynağı olarak, yazılım teknolojileri alanındaki en büyük çevrimiçi soru-cevap sitesi olan Stack Overflow genel veri kümesi kullanılmıştır. İlk olarak, Stack Overflow veri kümesinde Java ile ilgili sorulmuş tüm soru-cevap gönderilerini bulmak için kosinüs benzerliği kullanan pratik bir yaklaşım önerilmiş ve bu yaklaşım daha önceki yaklaşımlarla karşılaştırılmıştır. Daha sonra, oluşturulan derlem üzerinde konu modelleme yapmak için Gizli Dirichlet Ayırımı yöntemi kullanılarak ana konular belirlenmiş; ana konular ve anahtar kelimeler yıllara göre incelenmiştir. Son olarak, konular "alana özgü" ve "geliştirme ortamıyla ilgili" konular olarak sınıflandırıp, bu sınıfların hem kısa vadede hem de uzun vadede değişme eğilimleri araştırılmıştır.

Anahtar Kelimeler: Konu Modelleme, Gizli Dirichlet Ayırımı, Java, Stack Overflow

[*] Sorumlu Yazar: Ahmet Samet Şahin, TUBITAK Informatics and Information Security Research Center, Kocaeli, Turkey, ORCID: ORCID: 0000-0002-8315-1482, samet.sahin@tubitak.gov.tr

1. Introduction

Java is an object-oriented, free, open-source and platform-independent programming language. Java also has rich open source libraries**,** community support and documentation support. The first version of Java, called Java 1, was released on 23 January 1996. From that day it has been used to develop desktop and mobile applications. It has been used in embedded applications, web applications, enterprise applications, scientific applications, financial services applications, and so on. Within this period some needs, platforms, standards, and designs have changed, so Java and its libraries has been updated in compliance with these requirements. Thus, it is of interest to examine frequently used keywords, concepts, libraries, and tools in Java and their changes over time.

In this work, we make use of the Stack Overflow dataset to intelligently carry out this examination. Stack Overflow provides a questions and answers dataset in the XML format. It has been selected as a data source because it is comprehensive, worldwide, and public. It contains about 17 million questions and 23 million answers on multiple topics.

We extract Java-related questions from the Stack Overflow dataset by using the cosine similarity metric. After preprocessing the dataset, we partition it into two-year periods and use LDA to perform topic modelling for each period. Based on the findings we carry out various analyses. Our empirical work examines three main research questions.

1) What has been discussed?

We extract seven main topics, 25 related keywords, and their distributions using LDA for each two-year period.

2) What has been discussed about each topic in separate periods?

We compare distributions of keywords in topics within two-year time periods. From these results, we draw trend lines for specific technologies and concepts for each topic. We also indicate the popularity of some topics as increasing, decreasing and constant.

3) Over the years, which type of topic is prone to change? Which type of topic is steady?

The main contributions of our study are:

1) We conduct a large empirical study on Java related discussions to identify the main topics and analyze trends. Unlike the previous studies on comparison of Java technologies, for trends analysis we thoroughly examine the words and their distributions in the topics over time.

2) We propose methods using the cosine similarity measure for two purposes:

a) In order to find tags related to Java on Stack Overflow.

b) In order to determine the tendencies of the topics to change over time.

Our methodology is actually not specific to Java, so it can be, in principle, adapted and used for analysing different topics. Java is just a case-study.

The remainder of this paper is organized as follows. In section 2, we discuss the related works. The material and methodology are explained in the third section. In section 4 is presented the empirical results and discussion. Finally, concluded remarks are given in Section 5.

2. Related Works

Various empirical studies analyze trends, challenges, and user behaviors in software technologies, using LDA for topic modelling to the Stack Overflow dataset.

While in some studies all questions, answers, and comments on Stack Overflow were used as a dataset, in some studies the scope was narrowed. Barua et al. (2014) conducted the largest study of general trends in software development. Zou et al. (2015) examined non-functional requirements. Zou et al., (2017) extended Zou et al. (2015). Allamanis and Sutton (2013) compared the most common technologies in view of linguistics and semantics. S. Wang et al. (2013) extracted 63869 unique questions from 100,000 randomly selected questions to model statistically behavior of users.

There are studies on specific research technologies such as mobile development (Linares-Vásquez et al., 2013; Fontão et al., 2018), libraries of IOS and Android (W. Wang & Godfrey, 2013), security in development (Yang et al., 2016), Android testing (Villanes et al., 2017), concurrent programming (Ahmed & Bagherzadeh, 2018), and front-end technologies in web development (Bajaj, 2012). In these studies, the dataset is filtered. In order to create the corpus, related tags were selected manually to create a dataset from questions with these tags and their answers. Some studies propose metrics to expand the dataset (Rosen & Shihab, 2016; Yang et al., 2016). They selected specified tags and candidate tags manually and applied metrics to candidate tags. Yang et al. (2016) generated a metric from a number of posts with specified tags and a number of posts with candidate tags and they applied this metric to candidate tags. Rosen and Shihab (2016), suggested a method composing of two metrics named TRT (tag relevance threshold) and TST (tag significance threshold).

There are several research questions for popular topics, challenges, trends changing over time and unanswered questions.

In order to determine popular topics, Rosen and Shihab (2016) benefited from an average count of views for questions. Ahmed and Bagherzadeh (2018) added an average count of questions marked as favorite by users, and an average count of score of questions as well. Yang et al. (2016) used the average number of comments in addition to these three parameters. Bajaj (2012) applied a topic model to the dataset containing the most visited 2000 questions which have accepted an answer for each topic after extracting the topics.

Several approaches have been proposed to meet challenges that developers face. Zou et al. (2017) proposed a metric using ensemble averaging from the accepted answer rate, the duration of the accepted answer, and the number of answers given to the question. Rosen & Shihab (2016) benefited from the time taken for the approved answer to the questions and the proportion of approved answers to the questions. Yang et al. (2016) used an approach consisting of the average approved response time, the average number of answers and the average number of views. Villanes et al. (2017) filtered questions containing “what,” “why,” and “how" in their title. Ahmed & Bagherzadeh (2018) developed an approach from the percentage of questions with no accepted answers and the time taken for the approved answer to the questions. Bajaj (2012) suggested a method consisting of the number of answers, the number of comments, the number marked as favorite, as well as positive and negative votes (accumulated post score) to score questions. They examined the first 50 questions with the highest score and analyzed the challenges manually.

There are other research questions. Barua et al. (2014) researched the relationship of the top five triggered questions and their answers to detect closely-coupled topics. Zou et al. (2017) used Spearman’s rank correlation coefficient method to analyze the number of questions about non-functional requirements depending on the time for different languages and technologies. Allamanis & Sutton (2013) examined the relationship between keywords and programming languages, technologies and concepts. They also analyzed the frequency of type markers (class, variable, and method names) for each topic. They investigated similar languages using cosine similarity. They narrowed the scope of the research, analyzed topics with a verb content and reconciled the verb roots with the question types (how to, why, way of). From these findings, they analyzed the specific intended use of programming languages. In addition, they compared weekends with weekdays regarding the number of questions. S. Wang et al. (2013) demonstrated distributions of members asking and answering. They also examined users’ questioning and answering habits. They indicated single labels and multiple labels for each document. They also analyzed distributions of developers for each topic. Linares-Vásquez et al. (2013) examined whether the top contributors to mobile-related questions changed for each separate platform (Android, IOS) or not. Rosen & Shihab (2016) analyzed question types; they selected random questions from each platform at certain rates and classified questions manually as “what”, “why” and “how”. Fontão et al. (2018) investigated whether the number of questions on the release date increased or not. They also examined questions including "what" and "how". W. Wang & Godfrey (2013) used 250 Android and 55 IOS open-source projects in addition to the Stack Overflow dataset to investigate libraries used in the development of IOS and Android. They compared the number of posts from libraries with the number of calls from libraries in projects by type.Villanes et al. (2017) used the number of the questions and answers belonging to test tools and test libraries to show trends in test tools and strategies. Ahmed & Bagherzadeh (2018) used a *p* number to analyze the correlation between popular and difficult topics. They also showed popular topics in a hierarchical way and visualized topics and their proportions.

As a contribution to LDA, Yang et al. (2016) tuned LDA with a genetic algorithm. A silhouette coefficient was used to detect the optimum topic number (Fontão et al., 2018 ; Villanes et al., 2017). Zou et al. (2015) received support from four Ph.D. students to verify extracted topics manually. For verification, they used part of the dataset.

For comparison of Java Technologies, Counsell et al. (2003) investigated the changes in the classes for 52 Java libraries during a period of three years. Masovic et al. (2012) examined architecture of Java web components. B.A. & Bhosale (2017) compared Java IDEs. Taboada et al. (2013) shared practices and experiences in high performance computing in Java. Weifeng & Keji (2010) studied on the evolution of java imaging technology.

The study closest to our study in terms of researching trends of specific technologies is the one conducted by Barua et al. (2014). However, we have a scope narrowed down to Java posts. While Barua et al. (2014) just compared existent technologies with each other over time (programming languages, tools), we also included emerging and obsolete technologies in this comparison. Unlike research questions in the previous studies, we have also introduced a new research question that aims to analyse tendencies of the topics to change over time. We have suggested two approaches based on cosine similarity that allows us to automatically filter the dataset and determine the tendencies of the topics to change.

3. Material and Methods

3.1. Stack Overflow Dataset

We downloaded the “Posts” dataset from [2] as XML format. The fields of our used dataset are as given in Table 1. We firstly form a corpus from those posts having Java tags. Then we employ a practical approach based on cosine similarity to extract other posts from the Stack Overflow dataset that do not have Java tags but are related to Java and use them to augment the corpus. Note that, there can be Java-related posts without “Java” tags since users may tag posts deficiently. For example, if a post has a "Spring" tag without "Java" tag, it is needs to be added to the corpus. For this reason, in previous studies, tags like “Spring” were mostly selected manually and posts with such tags were selected to create the corpus.

[2] https://archive.org/download/stackexchange/stackoverflow.com-Posts.7z (last updated 2 December 2019)

Hence, the "relation score" assigned to each post to automate the extraction process may be expressed as

$$Cos(\theta)_{similarity} = \frac{\vec{A}\,\vec{B}}{\|\vec{A}\|\,\|\vec{B}\|} \quad (1)$$

Table 1. Descriptions of Post table fields

Field	Description
Id	*denotes id of the post.*
PostTypeId	*classifies posts as question and answer. This value is 1 if post is a question, and 2 for an answer.*
ParentId	*denotes id of the question to which the answer belongs. For questions this value is null.*
CreationDate	*denotes creation date of the post.*
Body	*denotes content of the post.*
Tags	*denotes tags to summarize the question. Each question has min 1 tag, max 5 tags. For answers this value is null.*
Title	*denotes title of the question. For answers this value is null.*

where $\vec{A}$ is the vector of frequencies of the most common keywords occurring in documents having a candidate Java related tag and $\vec{B}$ is the vector of frequencies of the same keywords occurring in documents having the same candidate Java related tag and the tag "Java". As the 'relation score' approaches 1, the similarity of the vectors increases.

The cosine similarity expresses the similarity between the estimates of the probability mass distributions $P(a|X)$ and $P(a|\text{"Java"}, X)$ where the former is the probability mass distribution of most commonly occurring keywords in documents containing a candidate Java related tag and the latter is the probability mass distribution of the same keywords in documents containing the same candidate Java related tag and the tag "Java". Kullback Leibner distance is another measure that could alternatively be used for quantifying the dissimilarity of these distributions.

The steps of our approach (relation score) are:

1) Select the most prominent tags on the subject. For our study we select the "Java" tag.
2) Find the 50 tags that most frequently occur together with the most prominent tag "Java". These are called candidate Java related tags.
3) For each candidate tag, two vectors are created. The 25 tags (keywords) that most frequently occur together with the candidate tag are first determined to yield set A. The 25 tags that most frequently occur together with both the candidate tag and the "Java" tag to yield set B. The components of the first vector are the frequencies of joint occurrences of keywords set $A \cup B$ along with the candidate tag. The components of the second vector are the frequencies of joint occurrences of keywords set $A \cup B$ along with the candidate tag and the "Java" tag.
4) The cosine similarity for the two vectors is calculated. If this value is greater than 0.98, this candidate tag is a subset of Java.

Example: Let's assume the candidate tag X is "Spring". There are 162277 questions with the "Spring" tag and 97213 questions with both the "Spring" and "Java" tags. Table 2 lists the 25 tags that most frequently occur together with "Spring" along with their joint occurrence frequencies (probabilities) as well as the 25 tags that most frequently occur together with both "Spring" and "Java" along with their joint occurrence frequencies. Note that the two sets of 25 tags happen to be the same in this case.

The similarity of the distributions $P(a|\text{"Spring"})$ and $P(a|\text{"}Java\text{"},\text{"Spring"})$ can be assessed by applying the cosine similarity measure to the two (25 dimensional) vectors $[0.01416\ 0.2023\ \ldots\ 0.02876]^T$ and $[0.0171\ 0.02218\ \ldots\ 0.03003]^T$ formed by taking the two column of frequencies excluding the first and last rows in Table 2. Cosine similarity measure is calculated as 0.99726.

In the Table 3, relation score (cosine similarity measure) calculated for all the candidate Java related tags are presented and tags that exceed the 0.98 threshold are marked in bold.

We have made some deductions from our findings. While tags used in other technologies in addition to Java have a low relation score (string, regex, xml, json, mysql, collection, class, multithreading, rest, sql), tags used commonly together with Java have a high relation score (android, eclipse, intellij-idea, netbeans, selenium-webdriver). Tags used only together with Java have a relation score that is very high (swing, spring, jpa, junit, jar).

The thresholds in our approach were determined experimentally. We have tested 25, 50 and 100, respectively, for the vector dimensions and decided to use 25, because the results were similar for all three values. We have also applied this approach to the

"python" keyword. We have found that the optimum threshold value for determining python related tags (python-specific tags) is 0.98. In summary, our approach is not specific only to java.

The corpus contains 1,606,056 posts containing a "Java" tag. In addition to posts containing these prominent tags, we add posts with tags related to Java to augment the corpus. Altogether, the corpus contains 1,833,526 posts.

Table 2. The most common 25 tags and their frequencies (probabilities) when the label Spring is present and when the labels Spring and Java are both present (t denotes tag label).

Tag	**Freq (t | "Spring")**	**Freq (t | "Java", "Spring")**
Xml	2281 (0.01406)	1662 (0.0171)
Mysql	3283 (0.02023)	2156 (0.02218)
Junit	2603 (0.01604)	1714 (0.01763)
Spring-data-jpa	5594 (0.03447)	3038 (0.03125)
Hibernate	19646 (0.12106)	12703 (0.13067)
Spring-batch	2989 (0.01842)	1458 (0.015)
Maven	6934 (0.04273)	4747 (0.04883)
Spring-data	4461 (0.02749)	2266 (0.02331)
Thymeleaf	2504 (0.01543)	1201 (0.01235)
Spring-mvc	30395 (0.1873)	18676 (0.19211)
Mongodb	2399 (0.01478)	1384 (0.01424)
Rest	5895 (0.03633)	3531 (0.03632)
Spring-boot	28957 (0.17844)	16254 (0.1672)
Json	3577 (0.02204)	2262 (0.02327)
Spring-integration	2738 (0.01687)	1311 (0.01349)
Tomcat	4257 (0.02623)	2784 (0.02864)
Spring-security	11656 (0.07183)	5790 (0.05956)
Annotations	1957 (0.01206)	1275 (0.01312)
Java-ee	2244 (0.01383)	1380 (0.0142)
Eclipse	2112 (0.01301)	1337 (0.01375)
Javascript	2105 (0.01297)	916 (0.00942)
Jpa	7924 (0.04883)	4942 (0.05084)
Servlets	1788 (0.01102)	1278 (0.01315)
Jsp	4667 (0.02876)	2919 (0.03003)

In the literature, two methods have been proposed to determine subject-related posts from Stackoverflow. In search of posts related to the subject "mobile", Rosen and Shihab (2016) manually specifies an initial set of keywords related to "mobile" and expands the set by using all the tags of all the posts that contain any keyword. Then two heuristic tests are applied to filter the tags of the expanded set and in turn the posts that contain them. The first test checks that the frequency of cooccurrence of the tag with at least one keyword is large compared to the frequency of its occurrence. The second test checks that the frequency of cooccurrence of the tag with at least one keyword is large compared to the largest of the cooccurence frequencies of the tags in the expanded set with at least one keyword.

The approach of Yang et al. (2016) is similar, but instead of using subject related keywords, determines the tag set by directly using the subject word ("security" in this case). Two tests are also employed to filter out any tag with a frequency of cooccurrence with the word "security" that is not comparatively large against either the frequency of occurrence of the tag or the frequency of occurrence of the word "security". Both Yang et al. (2016) and Rosen & Shihab (2016) further expand the set of posts by including the answers to the question posts.

Compared to Rosen & Shihab (2016), our approach holds the advantage of not relying on a manually generated keyword list. Furthermore, the approaches in Yang et al. (2016) and Rosen & Shihab (2016) needs manually generated candidate tags. Whereas, the whole process in our approach is automated.

3.2. Implementation of Latent Dirichlet Allocation

After the questions with tags related to Java and their answers are selected (according to section 3.1) from the Stack Overflow Posts.xml dataset, they are classified by two-year periods. The dataset is then preprocessed to obtain a valid corpus. For preprocessing, firstly we remove code snippets, URLs, HTML tags (, , <li>,) and stop words ("is," "the," numbers, and punctuation marks). In addition to this, we remove words that cannot belong to any cluster ("exception", "problem", "error",

"question", "answer"). Secondly, in order to find the root of each word, we have applied the Wordnet Lemmatizater stemming to the word.

Table 3. Candidate tags and their relation score. Those tags with a relation score exceeding the threshold value of 0.98 are marked in bold.

Tag	Relation Score	Tag	Relation Score
Android	0.91800	**Java-ee**	0.99323
Spring	0.99726	Sql	0.55167
Swing	0.99998	Generics	0.31758
Eclipse	0.96095	**Junit**	0.98823
Hibernate	0.99692	**Android-studio**	0.99586
Arrays	0.32594	Web-services	0.66812
Maven	0.97752	**Java-8**	0.99859
Multithreading	0.41479	Netbeans	0.86006
Spring-boot	0.99679	Sockets	0.63662
Xml	0.63278	Html	0.59164
Json	0.41125	**Jar**	0.98314
Spring-mvc	0.99780	Intellij-idea	0.88636
String	0.56121	Algorithm	0.75241
Mysql	0.31781	User-interface	0.58542
Jsp	0.95232	Database	0.68938
Jpa	0.99795	File	0.46484
Arraylist	0.97022	Exception	0.55020
Servlets	0.99781	Class	0.42812
Tomcat	0.98436	Selenium-webdriver	0.94912
Regex	0.34952	Performance	0.52684
Javafx	0.99433	Methods	0.70835
Jdbc	0.99732	Collections	0.57566
Javascript	0.67834	Gradle	0.94089
Selenium	0.79642	Oracle	0.59169
Rest	0.71364	Oop	0.53435

LDA is an unsupervised clustering method for text mining in which each document consists of distributions of the topics, each topic consists of distributions of the keywords (Blei et al., 2003). As an implementation of LDA, we have used the MALLET version 2.0.8. For optimum configuration, we use the default values $\alpha = 50/K$ and $\beta = 0.01$ for MALLET's hyperparameters (Biggers et al., 2014) where K denotes the number of topics. We ran MALLET for 500 Gibbs sampling iterations (Barua et al. ,2014). In order to compare changes in the topics over the years, we aim to obtain similar topics for different time periods. For this reason, the number of topics was set as 7 experimentally.

4. Results and Discussion

4.1. Research Question-1

"What has been discussed?"

In this section, we aim to cluster Java-related posts by two-year periods. Therefore, we find prominent topics and related keywords for each two-year period. As described above, we set the number of topics to seven experimentally to cluster similar topics in each period. For topic modelling, as mentioned in section 3, we use LDA Gibbs sampling implementation. Table 4 shows seven main topics and the first 25 related keywords. It shows that main topics are unchanged. These are ***JVM operations, GUI programming, web development, database operations, Java basics (basic commands and data types), object-oriented programming and build tools*.** Topic names were assigned based on the common themes of the topics' keywords manually. It's clear that, while keywords and their proportions in some topics are more stable, some are more changeable. We investigate this in detail in the results of research questions 2 and 3.

4.2. Research Question-2

"What has been discussed about each topic in separate periods?"

In this section, for each topic, we aim to find whether the popularity of some specific technologies and keywords is increasing or not. Table 4 shows main topics and related keywords. For this question, we select some outstanding keywords (similar technologies, emerging technologies, and obsolete technologies) from Table 4, and then draw trend lines. The trend lines displayed in Figure 1 are based on the proportion of the keyword in the topic within each two-year period.

From Figure 1, we have deduced the following tendencies for certain topics:

Lambda Expression: With Java version 8, lambda expressions have been added for functional programming. It is shown that Lambda expressions are talked a lot after release of Java 8.

Performance: In addition to above trend line, we observe that while "performance", "cpu", "ram", "heap" keywords have been talked less recently, "parallel", "cluster" keywords have been more popular. We interpret that distributed and parallel programming has been used to handle too many requests quickly by servers.

Web Testing Automation: Since the software is getting more and more complex, the need to automate software tests has emerged. Selenium is generally preffered for web-testing automation and selenium is compatible with Java unit testing libraries (JUnit, TestNG).

Database: Mysql and Oracle are still widely used as relational database management system. With the spread of NoSQL databases, MongoDB has been popular as well as other programming languages. As database connection library JDBC is rarely used in new projects, due to Jpa and Spring Data is generally selected due to ease of use. Also, both use JDBC in the background.

Web Frameworks: New Spring technologies have been facilitated in network and database operations. Such as: Spring data, Spring mvc Spring boot. Also, with the development of microservice architecture Rest Api has been prevalent. On the other hand, Servlets and JSP have been considered obsolete technologies and no longer preffered for the new projects.

IDEs: Desktop applications have mostly been replaced by web and mobile applications. For Android user interface Android Studio and for server-side development Intellij Idea and Eclipse are still common. Since NetBeans is commonly used to develop desktop user interface, use of it has been decresed. Also, due to support of HTML and CSS, JavaFX has been popular than Swing.

Build Tools: Since Ant is not functional as Gradle and Maven, Ant is a not preffered build tool for new projects. Since Both Maven and Gradle are comprehensive build tools, both are used. Due to flexibility and simple syntax, gradle has been popular in recent.

4.3. Research Question-3

"Over the years, which type of topic is prone to change? Which type of topic is steady?"

In this section, we investigate the types of topic that have undergone changes. Firstly, we manually partition topics into two as "domain-specific" and "development environment". Development environment topics are related to infrastructure of projects (commands, language version, IDEs, build tools, design patterns, project management tools). From the first research question, we place "Java Basics", "Object Oriented Programming", and "Build Tools" topics into the "development environment" group. Domain-specific topics are related to components of projects (database connection, server, client, concurrent programming, user interface design). From the first research question, we place "Database Operations", "Web Development", "JVM Operations", and "GUI Programming" into the "domain-specific" group. We assign a vector to each topic in a two-year period by using the distribution of its keywords over that period.

To determine short term changes in the discussion items of each topic, we benefit from cosine similarity to model similarity of the vectors in two consecutive two-year periods. If the cosine similarity approaches 1, we conclude that the topic is stable between the two consecutive year periods. The results are tabulated in Table 5. We also repeat the procedure for the determining the long-term changes in a topic by applying the cosine similarity to vectors formed from keywords of that topic in the first and last time periods of our study, 2012-2014 and 2018-2020. The results are tabulated in Table 6.

According to Table 5 and Table 6, it can be seen that, in both the long and short terms, trends in the domain-specific topics change faster than trends in the development environment topics. In addition, Table 6 shows that trends in web development programming have changed faster in the long term.

The reason for this is that while keywords used in the domain-specific topics change, the context of keywords used in development environment topics change. For example, if we make a deduction from the answer to the second research question, with the change of the Java version, basic concepts (class, method, interface, static) in object-oriented programming have not changed. However, we see that these concepts have updates in their functions and properties. Conversely, "Spring" technologies have become more widespread recently and the development of desktop GUI programing has decreased. Therefore, keywords in questions about these topics have changed.

5. Conclusions

In this paper, we conducted a large study on Java posts on Stack Overflow. In section 3, we suggested a practical approach based on the cosine similarity metric to find Java-related posts on Stack Overflow. After preprocessing the dataset, we have partitioned it according to period. By applying LDA to the corpus, we have extracted the following seven main topics: JVM operations, GUI programming, web development, database operations, Java basics (basic commands and data types), object-oriented programming and build tools.

After finding the main topics, we have detailed them and their related keywords in section 4. We have then presented the trend lines of some of the frequently used keywords from topic distributions by year in section 4. It is clearly seen that the trends of "Spring technologies", "parallel and concurrent programming", "object relational mapping" and "lambda expression" have been on the rise. Conversely, “desktop GUI programming" and "performance" related posts have decreased. Also, it is obvious that the technologies like Spring Data, JPA, Rest, Gradle not only ease developer workload but also offer maintainability and more clean codes. Thus, it can be popular quickly. As a result of all these changes, it is conceivable that post subjects change in accordance with trends.

In the last section, we classified topics as development environment topics and domain-based topics. We compared the tendency of these classes to change using cosine similarity. Based on our findings, we can conclude that the interest in the domain-specific topics has changed faster than the interest in the development environment topics in both the short term and the long term. In addition, we have determined that while the keywords themselves have changed in domain-based topics, the contexts of keywords have undergone change in development environment topics.

As future work, we plan to incorporate other data sources in addition to the Stack Overflow dataset to extend the analysis. We also plan to compare popular programming languages.

Table 4. Main topics and related first 25 keywords

2012-2014 (1.period)	**(JVM)** thread time server memory connection run client message application data process running task call read program start block performance jvm stream send system queue case **(WEB)** spring page request server web application service xml bean url servlet client json jsp form html response http tag controller session browser app access data **(GUI)** button image activity text component add set click show change view display android layout event screen game app window create panel swing size frame gui **(DATA)** data database table query key hibernate entity column map row node list sql field set object transaction update create record store jpa mysql insert model **(BUILD)** file project run jar eclipse application version test library path folder class command directory build source maven package running android add window program dependency tomcat **(OOP)** class method object type call variable instance return create reference constructor interface case field called parameter static implement implementation change test function access list null **(BASIC)** string array number line loop element list character time input output return print read byte check program date case text word end match bit convert
2014-2016 (2.period)	**(JVM)** thread time memory run process task call case block data stream running read program start operation application job performance jvm system queue size stack wait **(WEB)** server spring application client request service page web message url bean send connection tomcat servlet response controller configuration app jsp access http session html browser **(GUI)** button image activity click text add view app set change show layout android display component game screen create fragment window event color method size frame **(DATA)** data list database table query object key element map entity column json hibernate field row node xml add set create sql property item model store **(BUILD)** file project run jar version android eclipse library folder path build application command test maven dependency directory add class package running app studio source window **(OOP)** class method object type call variable instance return create constructor case reference test parameter interface called static field implement function change implementation access null argument **(BASIC)** string array number loop line input output character program print time return read file element text check word byte date statement end case integer convert
2016-2018 (3.period)	**(JVM)** thread time test run memory case process task date call running start stream block job read application operation queue data jvm performance wait message event **(WEB)** spring server application request service client web message page url send response boot spring_boot api controller app configuration rest tomcat connection http access post servlet **(GUI)** button image activity app click view android text show add change set layout fragment display screen item game create window color component method time device **(DATA)** data database table query entity column row hibernate field sql update record create jpa transaction key set save mysql insert model document list spring jdbc **(BUILD)** file project run version jar android build dependency library folder command maven path eclipse application directory studio add app android_studio running package module gradle line **(OOP)** class method object type call variable instance create return field parameter interface constructor case reference called implement implementation annotation function static change add bean null **(BASIC)** string array number list element loop line input output return print program character check read key case add word text method function index statement time
2018-2020 (4.period)	**(JVM)** thread time message process memory run task call data application case connection read event stream running job operation start transaction queue block cache multiple wait **(WEB)** spring server application request service boot spring_boot client api response url web configuration rest app send controller page tomcat http access post call connection login **(GUI)** app button image activity android view click show text change set add display fragment layout item device page screen create open time game inside color **(DATA)** data database table query json entity object field column date time hibernate row sql create class format model record update list key jpa save set **(BUILD)** file project run version jar dependency build android library folder command maven path application test running eclipse directory module studio package gradle add android_studio line **(OOP)** class method object type test call instance variable create return case parameter interface constructor annotation bean implementation called reference field implement function static add change **(BASIC)** string array list number element loop line input output return print character program check method case function key add time index map statement variable word

Figure 1. Trend lines of some specific technologies and keywords

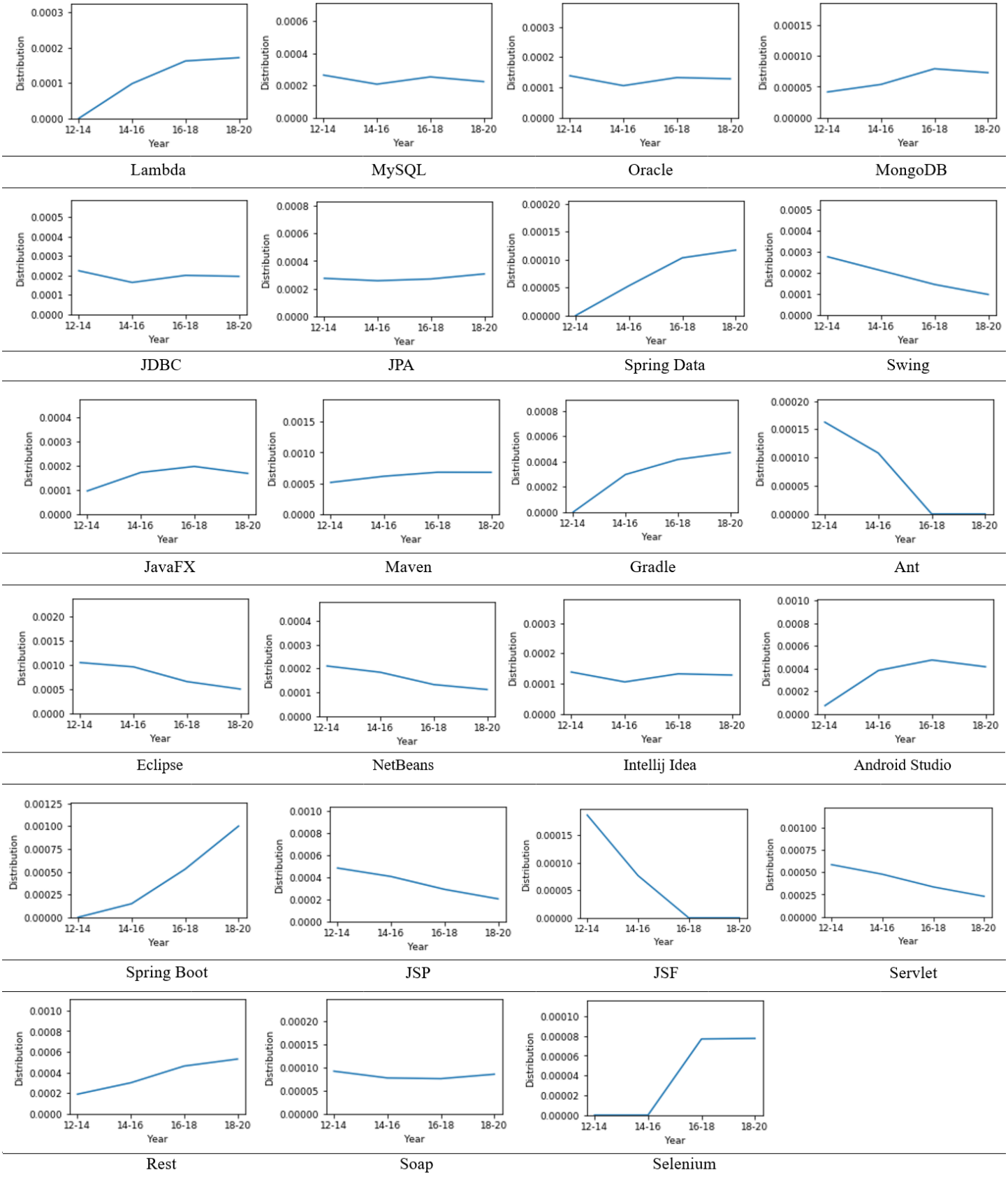

Table 5. Cosine similarity of consecutive periods and their average for each topic.

Topic	Class	1.Period-2.Period	2.Period-3.Period	3.Period-4.Period	Average
JVM Operations	Domain-Specific	0.894	0.874	0.825	0.864
Database Operations	Domain-Specific	0.890	0.833	0.914	0.879
GUI Programming	Domain-Specific	0.984	0.978	0.973	0.978
Web Development	Domain-Specific	0.898	0.970	0.965	0.944
Build Tools	Development Environment	0.983	0.992	0.993	0.989
Java Basics	Development Environment	0.973	0.940	0.988	0.967
Object Oriented Programming	Development Environment	0.994	0.990	0.981	0.988

Table 6. Cosine similarity of topics from firt period to last period

Topic	Class	(2012-2014) -(2018-2020)
JVM Operations	Domain-Specific	0.887
Database Operations	Domain-Specific	0.866
GUI Programming	Domain-Specific	0.890
Web Development	Domain-Specific	0.784
Build Tools	Development Environment	0.964
Java Basics	Development Environment	0.944
Object Oriented Programming	Development Environment	0.976

References

Ahmed, S., & Bagherzadeh, M. (2018). What do concurrency developers ask about?: A large-scale study using stack overflow. *International Symposium on Empirical Software Engineering and Measurement*, *October 2018*. https://doi.org/10.1145/3239235.3239524

Allamanis, M., & Sutton, C. (2013). Why, when, and what: Analyzing stack overflow questions by topic, type, and code. *IEEE International Working Conference on Mining Software Repositories*, *Table I*, 53–56. https://doi.org/10.1109/MSR.2013.6624004

B.A., P. J., & Bhosale, K. A. (2017). Research Paper on Java Interactional Development Environment Programming Tool. *Iarjset*, *4*(4), 121–124. https://doi.org/10.17148/iarjset/nciarcse.2017.35

Bajaj, K. (2012). *Mining Stack Overflow for Questions Asked by Web Developers*. *December*.

Barua, A., Thomas, S. W., & Hassan, A. E. (2014). What are developers talking about? An analysis of topics and trends in Stack Overflow. In *Empirical Software Engineering* (Vol. 19, Issue 3). https://doi.org/10.1007/s10664-012-9231-y

Biggers, L. R., Bocovich, C., Capshaw, R., Eddy, B. P., Etzkorn, L. H., & Kraft, N. A. (2014). Configuring latent Dirichlet allocation based feature location. *Empirical Software Engineering*, *19*(3), 465–500. https://doi.org/10.1007/s10664-012-9224-x

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, *3*(4–5), 993–1022. https://doi.org/10.1016/b978-0-12-411519-4.00006-9

Counsell, S., Hassoun, Y., Johnson, R., Mannock, K., & Mendes, E. (2003). Trends in Java Code Changes: The Key to Identification of Refactorings? *Proceedings of the 2Nd International Conference on Principles and Practice of Programming in Java*, 45–48. http://dl.acm.org/citation.cfm?id=957289.957305

Fontão, A., Ábia, B., Wiese, I., Estácio, B., Quinta, M., Santos, R. P. dos, & Dias-Neto, A. C. (2018). Supporting governance of mobile application developers from mining and analyzing technical questions in stack overflow. *Journal of Software Engineering Research and Development*, *6*(1), 1–34. https://doi.org/10.1186/s40411-018-0052-6

Linares-Vásquez, M., Dit, B., & Poshyvanyk, D. (2013). An exploratory analysis of mobile development issues using Stack Overflow. *IEEE International Working Conference on Mining Software Repositories*, 93–96. https://doi.org/10.1109/MSR.2013.6624014

Masovic, S., Saracevic, M., Kamberovic, H., & Kudumovic, M. (2012). Java technology in the design and implementation of web applications. *Technics Technologies Education Management*, *7*(2), 504–512.

Rosen, C., & Shihab, E. (2016). What are mobile developers asking about? A large scale study using stack overflow. In *Empirical Software Engineering* (Vol. 21, Issue 3). Empirical Software Engineering. https://doi.org/10.1007/s10664-015-9379-3

Taboada, G. L., Ramos, S., Expósito, R. R., Touriño, J., & Doallo, R. (2013). Java in the high performance computing arena: Research, practice and experience. *Science of Computer Programming*, *78*(5), 425–444. https://doi.org/10.1016/j.scico.2011.06.002

Villanes, I. K., Ascate, S. M., Gomes, J., & Dias-Neto, A. C. (2017). What are Software Engineers asking about Android Testing on Stack Overflow? *ACM International Conference Proceeding Series*, 104–113. https://doi.org/10.1145/3131151.3131157

Wang, S., Lo, D., & Jiang, L. (2013). An Empirical Study on Developer Interactions in StackOverflow Categories and Subject Descriptors. *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 1019–1024.

Wang, W., & Godfrey, M. W. (2013). Detecting API usage obstacles: A study of iOS and android developer questions. *IEEE International Working Conference on Mining Software Repositories*, 61–64. https://doi.org/10.1109/MSR.2013.6624006

Weifeng, M. A., & Keji, M. (2010). Research on java imaging technology and its programming framework. *Lecture Notes in Electrical Engineering*, *72 LNEE*(Table 1), 61–68. https://doi.org/10.1007/978-3-642-14350-2_8

Yang, X. L., Lo, D., Xia, X., Wan, Z. Y., & Sun, J. L. (2016). What Security Questions Do Developers Ask? A Large-Scale Study of Stack Overflow Posts. *Journal of Computer Science and Technology*, *31*(5), 910–924. https://doi.org/10.1007/s11390-016-1672-0

Zou, J., Xu, L., Guo, W., Yan, M., Yang, D., & Zhang, X. (2015). Which non-functional requirements do developers focus on? An empirical study on stack overflow using topic analysis. *IEEE International Working Conference on Mining Software Repositories*, *2015-Augus*, 446–449. https://doi.org/10.1109/MSR.2015.60

Zou, J., Xu, L., Yang, M., Zhang, X., & Yang, D. (2017). Towards comprehending the non-functional requirements through Developers' eyes: An exploration of Stack Overflow using topic analysis. *Information and Software Technology*, *84*, 19–32. https://doi.org/10.1016/j.infsof.2016.12.003