



**FEDERAL UNIVERSITY OF FRONTEIRA SUL
CAMPUS OF CHAPECÓ
COURSE OF COMPUTER SCIENCE**

ANDREW MALTA SILVA

**AN EXPLORATORY ANALYSIS USING TOPIC MODELING
TRACKING EVOLUTION AND LOYALTY FROM STACK OVERFLOW USERS'
INTERESTS**

**CHAPECÓ
2021**

ANDREW MALTA SILVA

**AN EXPLORATORY ANALYSIS USING TOPIC MODELING
TRACKING EVOLUTION AND LOYALTY FROM STACK OVERFLOW USERS'
INTERESTS**

Final undergraduate work submitted as requirement to
obtain a Bachelor's degree in Computer Science from the
Federal University of Fronteira Sul.
Advisor: Denio Duarte

**CHAPECÓ
2021**

Silva, Andrew Malta

An exploratory analysis using topic modeling: Tracking evolution
and loyalty from Stack Overflow users' interests / Andrew Malta Silva.
– 2021.

67 pp.: il.

Advisor: Denio Duarte.

Final undergraduate work – Federal University of Fronteira Sul,
course of Computer Science, Chapecó, SC, 2021.

1. Stack Overflow. 2. Topic modeling. 3. Users' interests. 4. Topic
evolution. 5. Topic loyalty. I. Duarte, Denio, advisor. II. Federal
University of Fronteira Sul. III. Title.

ANDREW MALTA SILVA

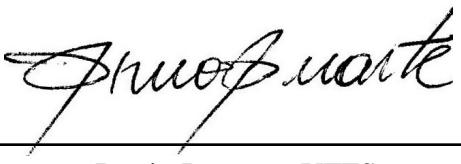
**AN EXPLORATORY ANALYSIS USING TOPIC MODELING
TRACKING EVOLUTION AND LOYALTY FROM STACK OVERFLOW USERS'
INTERESTS**

Final undergraduate work submitted as requirement to obtain a Bachelor's degree in Computer Science from the Federal University of Fronteira Sul.

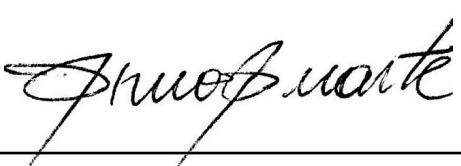
Advisor: Denio Duarte

This final undergraduate work was defended and approved by the examination committee on:
19/05/2021.

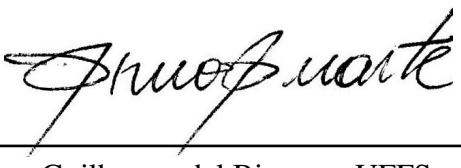
EXAMINATION COMMITTEE



Denio Duarte – UFFS



Fernando Bevilacqua – UFFS



Guilherme dal Bianco – UFFS

ACKNOWLEDGMENTS

As this work marks the end of a stage in my life and the beginning of another one, there are several people to thank. First of all, I am very grateful to my parents, Margarete and Vanderlei Silva, for giving me the necessary conditions and education to focus on my studies. Without them, everything would be harder and I thank them for all their support.

I also thank my professors and examination committee members, Fernando Bevilacqua and Guilherme dal Bianco. Their clear and straightforward perceptions and advisements allowed me to better understand some facets of this work, making it better.

Of course, I also thank my professor and advisor, Denio Duarte, who introduced me to this area of study and helped me in this research process. Also, he taught me a lot of things, from concepts on topic modeling to scientific methodology. Our weekly advising meetings were critical for me to keep the progress and the motivation.

And finally, but no less important, I thank Estela Vilas Boas, my awesome and lovely girlfriend, who was there to support me all the time. She motivated me when I was discouraged and celebrated with me when things went right. No matter the challenge I needed to face, she was there to show me the way. She is the lady of my life.

Because of these mentioned people, I am finally closing this stage and starting to fly higher flights. You will never be forgotten.

“Torture the data, and it will confess to anything.”

Ronald Coase

ABSTRACT

The web presents many platforms for sharing knowledge among users, such as newsletters, blogs, social networks, and communities. Among them, Stack Overflow is a popular question and answer (Q&A) community allowing users to share and acquire knowledge on computer programming topics. If explored, Stack Overflow posts present information that can provide many insights into the shared knowledge. Some works have explored these posts and generated relevant information, but the databases they analyzed are outdated now. Besides, many of them did not consider posts' authorship and publish dates in their analyses, which may provide useful temporal and user-centric insights. Therefore, this work made several experiments to infer Stack Overflow topics and analyzed them employing proposed metrics to measure the relative popularity of the topics and their drift. Then, these metrics were applied to analyze topics' popularity across temporal and authorship information, tracking the popularity evolution and the drift of these topics globally and for each individual user. The methodology employed topic modeling, natural language processing, statistical techniques, and validating experiments to accomplish these promising results, which were made web-accessible for free. Finally, the addressed methodology was shown to be effective on performed analyses, which successfully inferred Stack Overflow topics and tracked their general and user-centric popularity evolution.

Keywords: Stack Overflow. Topic modeling. Users' interests. Topic evolution. Topic loyalty.

LIST OF FIGURES

Figure 1 – Random question from Stack Overflow	23
Figure 2 – Accepted answer from Figure 1 question	23
Figure 3 – Example of applying tokenization	26
Figure 4 – Example of applying stopword removal	26
Figure 5 – Example of applying lemmatization	27
Figure 6 – Example of applying stemming	27
Figure 7 – Example of finding bi-grams	28
Figure 8 – Illustration of LDA intuitions	31
Figure 9 – Example of a distribution over topics	32
Figure 10 – LDA probabilistic graphical model	35
Figure 11 – Perplexity as a function of the number of topics on NIPS dataset	40
Figure 12 – Histogram of developers per topic plotted by Wand, Lo, and Jiang (2013) . .	41
Figure 13 – Methodology applied by Barua, Thomas, and Hassan (2014) in a flowchart .	41
Figure 14 – Top 5 increasing and decreasing trends in Stack Overflow by Barua, Thomas, and Hassan (2014)	43
Figure 15 – An overview of the addressed methodology in a flowchart	45
Figure 16 – Random post content subjected to the cleaning sub-step	47
Figure 17 – Random post content subjected to the enrichment sub-step	47
Figure 18 – 3D chart of Stack Overflow C_v coherence by number of topics K and iterations i	49
Figure 19 – General trend topics in Stack Overflow	53
Figure 20 – General topic popularity evolution by month in Stack Overflow	54
Figure 21 – General topic popularity drift in Stack Overflow	55
Figure 22 – Trend topics for user 1,289,716	56
Figure 23 – Topic popularity evolution for user 1,289,716	57
Figure 24 – Topic popularity drift for user 1,289,716	58

LIST OF TABLES

Table 1 – Simplified scheme of the Stack Overflow Posts	21
Table 2 – Trend topics from Stack Overflow by Barua, Thomas, and Hassan (2014) . .	43
Table 3 – Corpus definition	48

CONTENTS

1	INTRODUCTION	15
1.1	DOCUMENT ORGANIZATION	18
2	STACK OVERFLOW	21
2.1	SCHEME	21
2.2	DISCUSSION	24
3	NATURAL LANGUAGE PROCESSING	25
3.1	TOKENIZATION	25
3.2	STOPWORD REMOVAL	26
3.3	LEMMATIZATION	26
3.3.1	Stemming	27
3.4	N-GRAMS	28
3.5	DISCUSSION	28
4	TOPIC MODELING	31
4.1	TOPIC MODEL	32
4.1.1	Generative process	33
4.1.2	Inference process	33
4.1.3	Labeling process	33
4.2	LATENT DIRICHLET ALLOCATION	34
4.3	METRICS	36
4.3.1	Sliding and context windows	36
4.3.2	PMI and NPMI	37
4.3.3	C_v coherence metric	37
4.4	DISCUSSION	38
5	RELATED WORK	39
5.1	TOPIC MODELS	39
5.2	EXPLORATORY ANALYSES	40
5.3	DISCUSSION	44
6	EXPERIMENTS	45
6.1	EXTRACTION	45
6.2	PRE-PROCESSING	46
6.3	TOPIC MODELING	48
6.3.1	Corpus building	48
6.3.2	LDA inference	49
6.3.3	Labeling	50
6.4	DISCUSSION	50
7	RESULTS	51
7.1	METRICS	51

7.2	GENERAL ANALYSES	52
7.3	USER-CENTRIC ANALYSES	54
7.4	DISCUSSION	56
8	CONCLUSIONS	59
	REFERENCES	61
	APPENDIX A – STACK OVERFLOW TOPICS AND TOP WORDS .	65
	APPENDIX B – STACK OVERFLOW TOPICS AND TREND TOPIC POPULARITY MEASURES	67

1 INTRODUCTION

The Web is a huge source of information and knowledge for many different purposes. Anyone can access blogs, communities, newsletters, social networks, scientific repositories, streaming services, and many other websites. People can have fun, learn, and interact with several contents and people around the world. The more time passes, the more people engage web interactions and the more information is available.

Besides, sometimes people use the Web to seek solutions for specific issues they are dealing with. There are many ways to seek these solutions, but question and answer (Q&A) communities have exactly the purpose of solving user issues. Q&A communities allow users to ask for help with their issues, creating questions that can be answered by other users. Hence, the more users ask and answer, the more knowledge the Q&A community presents.

Some Q&A communities feature posts (general term for questions and answers) on a wide topic variety. For example, Quora¹ and Yahoo Answers² allow users to contribute on almost any topic. Differently, there are Q&A communities opting to delimit the topics they address. Consequently, they decrease topic variety but shared knowledge becomes more specialized.

Following the concept of topic-delimited Q&A communities, Stack Overflow is focused on featuring posts on computer programming topics. With over 50 million posts, Stack Overflow is a platform where professional and enthusiast software developers can concisely share and acquire knowledge in this field. They are able to seek solved questions, create new ones based on their own issues, and answer other questions.

Stack Overflow allows users to evaluate the quality of posts, creating a kind of quality control of shared knowledge. Posts with high scores provide reputation and badges for users who created them, motivating users to make significant contributions (CAVUSOGLU; LI; HUANG, 2015). Besides, users are also motivated to edit other posts to fix typos, add more information, and improve English clearness. These features make users continuously improve the quality of Stack Overflow posts.

As Stack Overflow presents plenty of useful information, several works use the posts to perform exploratory analyses, identify patterns, and get insights. Some of them employ topic modeling techniques to discover discussed topics in the posts, which is useful for many kinds of analyses. For example, Barua, Thomas, and Hassan (2014) have discovered 40 topics from Stack Overflow posts from 2008 August to 2010 August, analyzing and ranking them according to their relative popularity.

Posts also present other analyzable information, such as authors and creation dates. Analyzing them may provide relevant results for user-centric purposes, such as many kinds of recommender systems. For example, Wand, Lo, and Jiang (2013) have investigated users' behaviors across topics they write about, identifying their preference as questioners and answerers.

¹ <https://www.quora.com/>

² <https://answers.yahoo.com/>

Besides, Barua, Thomas, and Hassan (2014) have analyzed how Stack Overflow topics evolve and how they are affected by users' interests.

As Stack Overflow data is a rich source of text information, it can be explored employing these topic modeling techniques. Therefore, this work aims to make an exploratory analysis from Stack Overflow questions and answers to investigate some facets. However, this kind of study has some challenges to face. As Stack Overflow questions and answers are user-created, their topics are inevitably based on users' interests. Consequently, when users create posts related to what they are using, studying, or working with, they make Stack Overflow discussed topics evolve over time. Therefore, posts are mixtures of topics that indirectly represent users' interests.

Naturally, users have different interests that change depending on many circumstances, such as starting a new job, learning a new technology, or finding new hobbies. Some users are loyal to some topics they are very interested in, whereas other ones drift their interests constantly. Identifying how users' interests evolve may provide some information on their loyalty to topics they are interested in.

However, making these analyses is a challenging task because of many factors. Stack Overflow database presents a large volume of questions, answers, and much more information, which makes the posts' organization and analyzing a hard task. Any algorithm performing this task needs to be built optimizing memory and disk usage.

Furthermore, as Stack Overflow posts are user-created documents, analyzing them demands special attention. Several semantic, syntactical, and morphological information is hidden among the words, which is composed of natural language (BIRD; KLEIN; LOPER, 2009). Humans can easily read natural language, being able to understand meanings, interpret contexts, and identify discussed topics. However, doing that on a large scale from a collection of thousands or million documents is an impracticable task for humans.

Building algorithms capable to optimally make large-scale analyses is a solution for this problem. However, processing the natural language present in a collection of documents is a challenging task for an algorithm to do. Fortunately, rather than interpret documents just as arrays of characters, techniques on natural language processing can be employed to analyze documents considering grammar, morphology, and many other factors (BIRD; KLEIN; LOPER, 2009).

Although topic modeling presents many techniques to discover the topics from a collection, most of them are defined by statistical and probabilistic methods with hyperparameters that directly affect the results, such as the number of topics (BLEI; NG; JORDAN, 2003; BLEI, 2012; STEYVERS; GRIFFITHS, 2007). Therefore, employing metrics to measure their effectiveness and interpretability is important to have satisfactory results in this process (FITELSON, 2003; WALLACH et al., 2009; CHANG et al., 2009; ALETRAS; STEVENSON, 2013; RÖDER; BOTH; HINNEBURG, 2015).

Among others, these techniques were applied by the aforementioned works (WANG;

LO; JIANG, 2013; BARUA; THOMAS; HASSAN, 2014), but result availability is another factor to highlight. Although these works have explored Stack Overflow posts and obtained conclusions, their results are not available to be explored by other people. For example, other researchers could analyze additional aspects from the results, whereas some job recruiters could be interested in analyzing user preferred topics to find candidates.

Therefore, the main contribution of this work is providing an exploratory analysis from Stack Overflow questions and answers, which aims to track evolution and loyalty from users' interests across discussed topics in Stack Overflow. Hence, the specific contributions are defined in the following.

1. Defining the best number of topics to summarize Stack Overflow posts;
2. Discovering and labeling the topics discussed in Stack Overflow;
3. Proposing a metric for measuring the popularity of a topic across a set of posts;
4. Proposing a metric for measuring the drift of a user in a topic;
5. Identifying the trend topics in Stack Overflow by analyzing the general popularity of the topics across all posts;
6. Tracking the popularity evolution of the topics across all posts;
7. Identifying the trend topics each user has by analyzing the popularity one presents in the topics one is interested in;
8. Tracking the popularity evolution each user presents in the topics one is interested in;
9. Analyzing the loyalty each user presents in the topics one is interested in;
10. Making all results web-accessible for being analyzed by other researchers.

Topic modeling is an unsupervised learning technique, and so tuning the hyperparameters is essential to get good models (WALLACH et al., 2009; BLEI, 2012). The number of topics is a determinant factor to results quality; inferring a few topics makes them general and coarser-grained, but inferring many topics makes them detailed and finer-grained (BARUA; THOMAS; HASSAN, 2014).

Defining the best number of topics is essential to ensure the quality of the discovered topics. After conducting several experiments employing LDA technique to accomplish these objectives, the best distribution over topics was identified and labeled. The results presented 30 well-defined topics, which was very promising.

Also, as analyzing the popularity of the topics is an important part of this work, the Topic Popularity metric was proposed to perform this measurement, which was necessary to accomplish other objectives. This proposed metric was applied across all Stack Overflow posts,

providing the general topic popularity of all topics and giving a comprehensive view of their behaviors and trends.

Topics' popularity evolution was tracked by computing the proposed metric for each monthly time slice in Stack Overflow history. As analyzing patterns of increasing and decreasing values is a frequent study in machine learning (BISHOP, 2006; PEDREGOSA et al., 2011), this study provided much information for understanding how Stack Overflow topics evolve.

However, as aforementioned, users' interests have much impact on Stack Overflow posts. Thus, assuming posts present these interests hidden among the words, analyzing the topics related to posts created by a user may define the interests this user has. Therefore, the topics' popularity was separately computed for each user, including the trend topics and their popularity evolution. This user-centric data allows individual relevant analyses for each user, providing insightful information to analyze.

Following a different line, users' interests evolve differently for each topic they are interested in. Understanding how users keep loyal or drift from their interest topics may be useful for many purposes. For example, job recruiters seeking developers with specific talents can analyze Stack Overflow users' loyalty in topics related to required expertise. Therefore, based on the statistic standard deviation, the Topic Popularity Drift metric was proposed to measure the drift of the popularity evolution of a topic.

The Topic Popularity Drift metric was applied globally and for each user, generating several measures to analyze. Although this metric succeed in computing the variation of a set of topic popularity measures and provided interesting results, it has shown to be not really useful for analyzing the loyalty in a topic. Therefore, the requirements for topic loyalty traits must be better analyzed for another metric proposing in the future.

Finally, as analyzing this data would be useless if the results do not reach the people who need them, they were made web-accessible on the Internet Archive³ website. Thus, other researchers are allowed to take the generated results to validate and explore them with other approaches and for many purposes.

1.1 DOCUMENT ORGANIZATION

As understanding the data is essential to process them properly, chapter 2 presents more details about Stack Overflow, focusing on presenting the attributes that compose questions and answers. Then, chapter 3 presents some techniques in Natural Language Processing, which aim to interpret and analyze texts.

Thereafter, chapter 4 briefly presents main concepts on topic modeling and some metrics. Besides, as understanding works presenting similar objectives is essential to guide the methodology addressed in this work, chapter 5 presents some related work divided into topic model approaches and exploratory analyses from Stack Overflow.

³ <https://archive.org/details/staty>

Considering all presented content, chapter 6 details the methodology addressed in this work, presenting the techniques, steps, and experiments employed to accomplish the objectives. Then, chapter 7 presents the obtained results with charts and tables, providing enough information to make some conclusions in chapter 8.

2 STACK OVERFLOW

Released in 2008, Stack Exchange¹ is a network of Q&A communities, each one covering a different set of discussed topics. The largest one is Stack Overflow², which features questions and answers covering a wide range of topics in computer programming since 2008. The user base consists of professional and enthusiast programmers that seek solutions to issues they face and give solutions to other ones.

With more than 13 million users (registered and unregistered) and more than 50 million posts, Stack Overflow is the largest online community for software developers. Allowing them to learn, share knowledge, and build careers in a dynamical platform, Stack Overflow became a huge source of knowledge in this field.

The knowledge Stack Overflow provides is contained inside posts, that is, created questions and answers. The responsible ones to create and assess these posts are users, which help each other to find, improve, and evaluate solutions. Therefore, as Stack Overflow posts and users have been highlighted as the main objects of study of this work, the following sections briefly present their scheme and discuss their importance.

2.1 SCHEME

Stack Overflow stores data in a relational database composed of some tables and relationships. Posts is the table that stores every post, including questions and answers. There are 23 attributes in this table, each one with its own purposes. However, only 8 of them present meaningful information according to what this work intends to do, which are presented in table 1.

Table 1 – Simplified scheme of the Stack Overflow Posts

Attribute	Type
<code>Id</code>	<code>int</code>
<code>PostTypeId</code>	<code>tinyint</code>
<code>ParentId</code>	<code>int</code>
<code>CreationDate</code>	<code>datetime</code>
<code>Body</code>	<code>nvarchar(max)</code>
<code>OwnerUserId</code>	<code>int</code>
<code>Title</code>	<code>nvarchar(250)</code>
<code>Tags</code>	<code>nvarchar(250)</code>

Source – Stack Overflow Data Explorer (<https://bit.ly/3ep7Tdm>)

Besides questions and answers, there are six other types of posts in Stack Overflow. Although they are the minority, filtering them is essential to keep only the user-created posts.

¹ <https://stackexchange.com/>

² <https://stackoverflow.com/>

The attribute `PostTypeId` stores the type of post, where 1 and 2 represent questions and answers, respectively. Therefore, identifying and separating them from the other posts is an easy task.

Whenever a user creates a post, this post automatically stores the identifier of the author in the `OwnerId` attribute. Although other users may edit any post to fix grammar mistakes, typos, or provide further information, this attribute consistently stores only the post author. Therefore, considering the objectives of this work, the `OwnerId` attribute is useful in to identify posts authorship, which is represented by numeric identifiers.

Another information also automatically stored whenever a post is created is the creation date. This information is essential to keep posts in chronological order, making it possible to analyzing posts according to their temporal information. For example, Stack Overflow features annual surveys³ analyzing the data generated in that year and providing some interesting insights. Therefore, the `CreationDate` attribute is essential for this work.

When users look at a question, the first thing they see is the question's title, which is a short sentence that summarizes what the question is about. Well written titles make users quickly understand the topics a question addresses, increasing the probability they are answered. Therefore, the `Title` attribute may be helpful for discovering question topics, although answers have no titles.

Besides titles, questions also have tags helping to identify their subjects and improve the search engine results. Each tag is defined by a word or a short set of words split by hyphens that briefly define a topic addressed in the question. However, although tags give many tips about topics of questions, they are user-defined and may present some inconsistencies (BARUA; THOMAS; HASSAN, 2014).

For example, a user may create a question assigning the tag *iphone-api* and another user, assigning *iphone-sdk*. Even if the questions are about similar topics and present similar contexts, their assigned tags are differently written and, therefore, a search based on only one of those tags will result in only one of those questions (BARUA; THOMAS; HASSAN, 2014). Therefore, the `Tags` attribute can not define questions topics by itself, but still providing useful tips for that.

Although only questions have title and tags, every post has a unique identifier stored in the `Id` attribute. The relationship between questions and answers occurs by storing the question identifier in its respective answers. Therefore, every answer is associated with a question by storing the question `Id` in the `ParentId` attribute, making it easy to find a question given an answer.

Finally, whereas titles the tags give just brief ideas and tips about topics of a post, the `Body` attribute presents the complete and detailed content. In this attribute, the author writes everything about the faced issue or proposed solution. As the `Body` is stored as rendered HTML, a post may present sentences, paragraphs, topic structures, images, references, URLs, and code snippets. These features make posts enough detailed and understandable to other programmers who want to contribute and learn from them.

³ <https://insights.stackoverflow.com/survey/2019>

Figure 1 – Random question from Stack Overflow

Title

Asked 4 years ago Active today Viewed 134k times

Score 122

Body

I have installed the GPU version of tensorflow on an Ubuntu 14.04.
I am on a GPU server where tensorflow can access the available GPUs.
I want to run tensorflow on the CPUs.
Normally I can use `env CUDA_VISIBLE_DEVICES=0` to run on GPU no. 0.
How can I pick between the CPUs instead?
I am not interested in rewriting my code with `with tf.device("/cpu:0"):`

Tags

share improve this question follow

edited Feb 7 '18 at 13:45
dsalaj 1,343 ● 2 ● 14 ● 31

CreationDate

asked Jun 6 '16 at 14:41
Alexander R. Johansen 1,866 ● 3 ● 13 ● 20

Owner

Source – Stack Overflow website (<https://bit.ly/3d45hQG>)

Figure 1 presents a snapshot of a random question picked from Stack Overflow website, where the Title, Tags, CreationDate, Body, and Score attributes are highlighted in blue rectangles. Instead of exhibiting the OwnerId as a number, the Owner rectangle in the figure presents some information about the owner: its name (Alexander R. Johansen), reputation (1.866), and badges (3 gold, 13 silver, and 20 bronze). Finally, the Body attribute has six sentences and two short code snippets that give more details about the issue.

Figure 2 – Accepted answer from Figure 1 question

7 Answers

Body

Score 112

CreationDate

answered Jun 6 '16 at 15:11
Ivan Aksamentov - Drop 11.8k ● 3 ● 27 ● 59

Owner

You can apply `device_count` parameter per `tf.Session`:

```
config = tf.ConfigProto()
        device_count = {'GPU': 0}
    )
sess = tf.Session(config=config)
```

See also protobuf config file:
tensorflow/core/framework/config.proto

share improve this answer follow

Source – Stack Overflow website (<https://bit.ly/3d45hQG>)

Figure 2 presents a snapshot of the accepted answer from the question in figure 1. Similarly to figure 1, the `CreationDate`, `Body`, `Score` attributes and the owner information are tagged in blue rectangles. Observe that there are a larger code snippet presenting the code solution for the faced issue.

2.2 DISCUSSION

This chapter briefly presented some Stack Overflow information, focusing on the users, questions, and answers. Each Stack Overflow post is composed of many attributes, but especially 8 of them were presented because their importance to objectives of this work. Therefore, understanding them properly is the first step to identify what is need to analyze them.

Furthermore, the Internet Archive⁴ website provides a complete data dump from all user-contributed content in the Stack Exchange network, which is generated and updated by Stack Exchange Community⁵. Therefore, the information presented in this chapter may be downloaded as a XML file⁶.

However, Stack Overflow posts are composed of freely written English texts, which means the posts have to be processed to be machine-understandable. Therefore, the next chapter briefly presents some Natural Language Processing techniques able to analyze posts in many forms, helping to algorithmically process them.

⁴ <https://archive.org>

⁵ <https://archive.org/details/stackexchange>

⁶ <https://archive.org/download/stackexchange/stackoverflow.com-Posts.7z>

3 NATURAL LANGUAGE PROCESSING

Human communication is especially based on natural languages allowing them to daily speak, write, and read. Natural languages present many complex and variable sets of rules that define and organize them, such as grammar, syntax, and morphology. However, differently from artificial languages, such as programming languages and mathematical notations, natural languages naturally evolve as they are passed from generation to generation (BIRD; KLEIN; LOPER, 2009).

Nevertheless, natural languages usually present sets of rules defining and organizing them, such as grammar, syntax, morphology, and ideograms (BIRD; KLEIN; LOPER, 2009). These rules follow the language evolution, adapting themselves over time to better represent the language. Therefore, understanding these rules allows learning a natural language.

However, making an algorithm capable of understanding natural languages is a challenging task. An algorithm usually interprets sentences just as meaningless arrays of characters. Therefore, Natural Language Processing (or NLP for short) comes for that purpose: provide techniques capable to perform any kind of computer manipulation of natural language. NLP features techniques with many uses and for many purposes, such as speech recognition, natural language understanding, and natural language generation (BIRD; KLEIN; LOPER, 2009).

As manipulating texts algorithmically has always been a challenging task, NLP gets increasingly relevant and employed both in industry and academia. Fields such as human-computer interaction, business information analysis, artificial intelligence, and topic modeling employ NLP as an essential part of their processes (BIRD; KLEIN; LOPER, 2009).

Therefore, as Stack Overflow posts are freely written English documents, NLP is necessary to manipulate them properly. Among others, techniques such as tokenization, stopword removal, lemmatization/stemming, and *n*-grams are useful to clean, enrich, and understand a collection of documents, each one explained and discussed in the following sections.

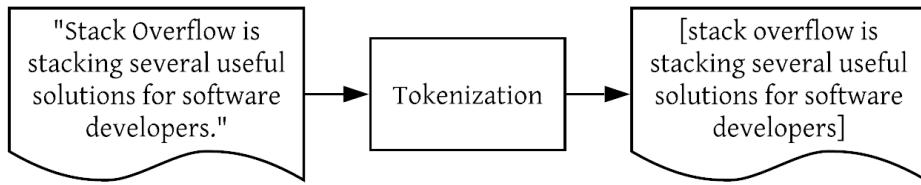
3.1 TOKENIZATION

When processing a text with NLP, identifying the tokens comprising it is essential, which are basic units of text representing words, digits, and punctuation marks (MANNING; SCHÜTZE, 1999). This process, named as tokenization, divides documents into lists of tokens, allowing identifying and analyzing each basic unit of text apart.

Depending on the case, each token presents different importance. Punctuation marks may be useful when identifying the text structure and grammatical forms (MANNING; SCHÜTZE, 1999), such as subordinate clauses or affirmative and interrogative forms. Differently, punctuation marks can be ignored when only word meanings matter.

Figure 3 presents an example of applying tokenization to a sentence using gensim library for Python (ŘEHŮŘEK; SOJKA, 2010). Note that punctuation marks were ignored, and all

Figure 3 – Example of applying tokenization



letters were cast to lowercase to prevent case differentiation (*e.g.*, “the”, “The”, and “THE”). Hence, any further processing can easily analyze each token apart.

3.2 STOPWORD REMOVAL

All words present syntactic and semantic importance, contributing to improving coherence and cohesion. However, when aiming to identify word semantics employing a word-by-word match, there are some words that rarely contribute with useful information (MANNING; SCHÜTZE, 1999).

These words are called stopwords and, as they “[...] usually have little lexical content, and their presence in a text fails to distinguish it from other texts” (BIRD; KLEIN; LOPER, 2009, p. 60), filter them out reduces time-consuming in further processing. For example, words such as “the”, “is”, “this”, and “a” are considered stopwords.

Figure 4 – Example of applying stopword removal

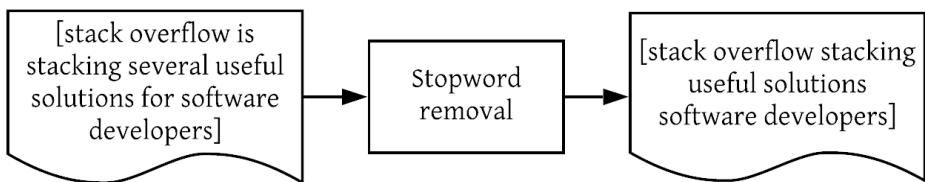


Figure 4 presents an example of removing stopwords from the already tokenized sentence in Figure 3, also using gensim library (ŘEHŮŘEK; SOJKA, 2010). Observe that the number of words was reduced from twelve to seven, reducing time-consuming in possible further processing. Even with fewer words, it still possible to understand the sentence subject.

3.3 LEMMATIZATION

Morphology is a linguist field that studies word formation and classification (MANNING; SCHÜTZE, 1999). Words have various forms and inflections that change part of their meanings and differ them from each other. However, every word has a lemma, that is, a canonical, dictionary, or citation form that summarize the main meaning of this word (BIRD; KLEIN; LOPER, 2009).

For example, the words “go”, “goes”, “gone”, “going”, and “went” have the same lemma: “go”. The word “go” succeeds on summarizing the meaning of its inflected forms. Therefore, finding word lemmas allows grouping words by their meanings, reducing the number of unique words and associating their semantics.

Besides, the morphology group the words with similar syntactic behavior into classes, which are commonly called as part-of-speech (MANNING; SCHÜTZE, 1999). Nouns, verbs, adjectives, and adverbs are common English part-of-speech, each one presenting different inflection behaviors. For example, verbs can inflect to indicate different tenses, whereas nouns can inflect to indicate singular and plural forms. Therefore, identifying the part-of-speech of a word is essential to understand these inflection behaviors and, consequently, find lemmas.

Figure 5 – Example of applying lemmatization

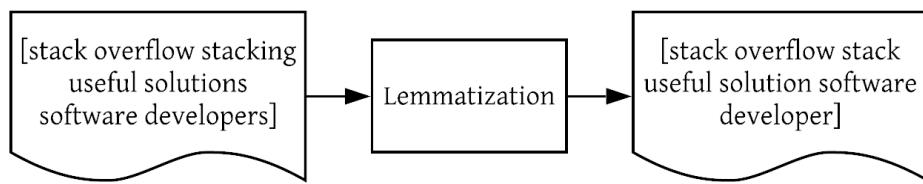


Figure 5 presents an example of applying WordNet Lemmatizer to the sentence cleaned in Figure 3 and Figure 4 using Natural Language Tool Kit (or NLTK for short) for Python (BIRD; KLEIN; LOPER, 2009). Observe that all inflected words were accurately reduced to their respective lemmas, because the process uses recognition of part-of-speech.

3.3.1 Stemming

When performance is a problem to face, stemming techniques are good substitutes for lemmatization. Stemming is a crude heuristic process that works by removing attached suffixes and prefixes from words, letter by letter (JIVANI et al., 2011). Unlike lemmatization, stemming techniques do not depend on properly identifying part-of-speech and associating word meanings, which give them a better performance (JIVANI et al., 2011). However, stemming techniques find stems (not lemmas), giving results that are not always real words.

Figure 6 – Example of applying stemming

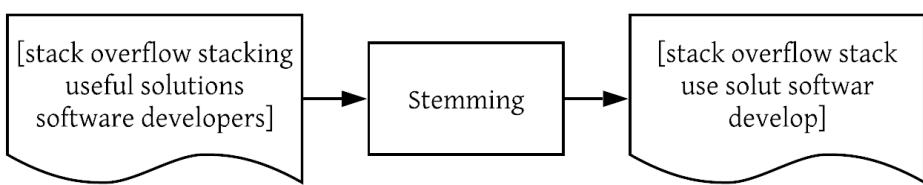


Figure 6 presents an example of applying stemming to the sentence pre-processed in figures 3 and 4, also employing NLTK (BIRD; KLEIN; LOPER, 2009). Observe that all inflected words were reduced, but not accurately because “useful”, “solutions”, and “software”

were reduced to “use”, “solut”, and “softwar”, respectively. In that case, the correct meaning of them was partially lost, making them hard to understand.

3.4 N-GRAMS

The possible meanings of a sentence may change according to the employed syntax, where word order is a relevant factor. Words have their own meanings, but there are terms composed of multiple words that may represent different meanings from their original ones. For example, the word “new” has many meanings related to something that does not exist before, but “new york” is related to a USA city or state. Therefore, identifying these multi-word terms is essential to capture their real meanings.

The n -grams technique consists of applying probabilistic methods to understand patterns on word order (MIKOLOV et al., 2013). It aims to find sequences of n words that frequently appear in the collection, where sequences of two words are called bi-gram, sequences of three words are called tri-grams, and so on.

Figure 7 – Example of finding bi-grams

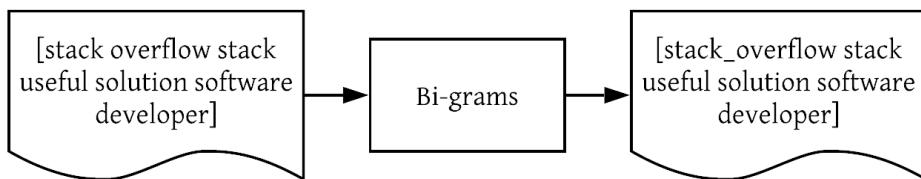


Figure 7 presents an example of finding bi-grams from the lemmatized sentence on Figure 5. Only one bi-gram can be observed in this example: “stack_overflow”. The applied technique verified that “stack” and “overflow” words appear together many times according to other documents, uniting the words with an underline. Hence, the algorithm understands “stack_overflow” as a different concept from separate occurrences of “stack” and “overflow” words.

3.5 DISCUSSION

This chapter presented and discussed some NLP techniques that are usually applied in a pre-processing step to clean documents and keep only useful information. Tokenization identifies all tokens, whereas stopword removal reduces the collection size and avoids processing meaningless words. Besides, lemmatization reduces all words to their respective lemmas, associating the meanings of related words. Finally, computing n -grams can help to identify multi-word terms, improving semantic representations. Applying these techniques prepares the collection for any succeeding process.

The choice between lemmatization and stemming needs to accord to the field of application. “When there is an urgent need in the efficiency of English document clustering, Porter

stemmer is a better choice, when there is a need in precision or in cluster label generation, lemmatization might be a good choice” (HAN et al., 2012, p. 119). Therefore, this work opts to employ lemmatization and focus on the precision and quality of the results, even if it is more time-consuming.

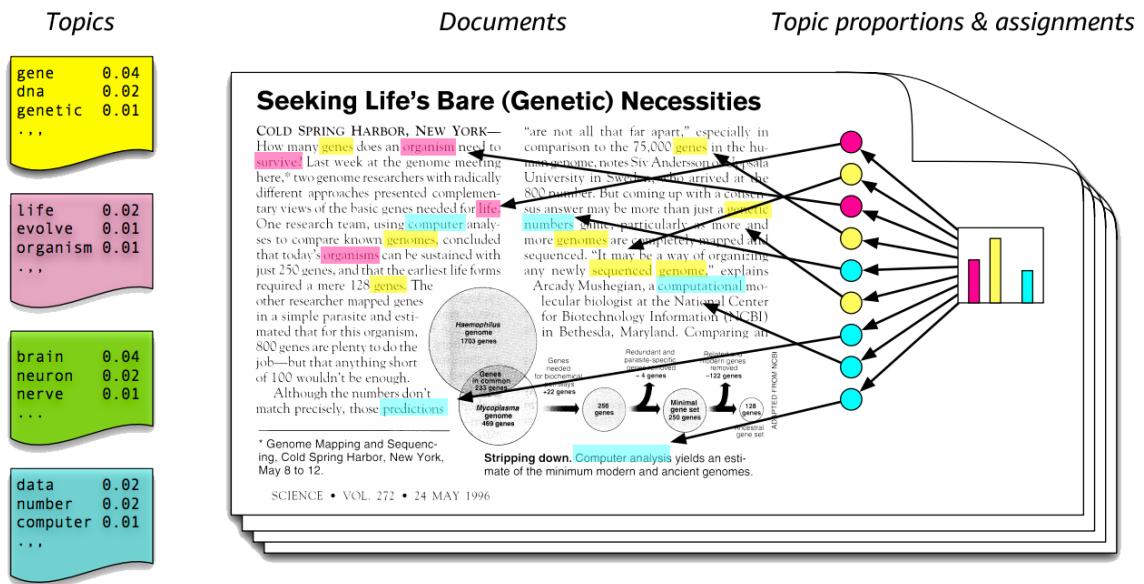
Considering that many objectives of this work address tracking the topics present in Stack Overflow user-created posts, understanding how to discover these topics is essential. Therefore, the next chapter focuses on briefly presenting some main concepts on probabilistic topic modeling and useful metrics.

4 TOPIC MODELING

The Web is an increasing source of documents that may be found in the form of news, blogs, articles, books, social networks, Q&A communities, among others. These documents are composed of user-created content based on natural languages, which provide contexts and information for human cognition (STEYVERS; GRIFFITHS, 2007).

When reading a document, a human can identify some topics that summarize it. The intuition behind this process is that documents are mixtures of topics, each topic defined by a set of words that look semantically related and frequently appear together (BLEI, 2012). Words commonly associated with a specific topic may be annotated in the document, making it easy to identify them. Naturally, some of these words may appear more frequently than others, even if they are related to the same topic (BLEI, 2012).

Figure 8 – Illustration of LDA intuitions



For example, Figure 8 presents an article entitled “Seeking Life’s Bare (Genetic) Necessities”, which is about applying data analysis to determine the number of genes an organism needs to survive the species evolution. Remark this article has some highlighted words, where each color represents a different topic. Words highlighted in blue, such as “computer” and “prediction”, are about data analysis; words highlighted in pink, such as “life” and “organism”, are about evolutionary biology; and words highlighted in yellow, such as “genes” and “sequenced”. are about genetics.

The topics present in the document in Figure 8 are detailed to the left, each one with its top-3 words. Note that each word is followed by a real number that represents its weight, *i.e.*, the probability of the presence of this word in its topic (BLEI, 2012). Also, the figure shows

a histogram to the right, which presents a brief visualization of the proportions of each topic associated to the document.

However, as text data are found in large collections with thousands or even millions of unstructured and unlabeled documents, reading an entire collection and identifying its topics may be impracticable if done by humans. Therefore, in order to do that efficiently, topic modeling gives “a suite of algorithms that aim to discover and annotate large archives of documents with thematic information” (BLEI, 2012, p. 1).

4.1 TOPIC MODEL

To generalize this concept, topic modeling defines a topic model as a distribution over topics, where each topic is a probability distribution over words (BLEI, 2012). In other words, a topic is defined by a set of weighted words, where each weight is the probability of its respective word to be found in documents (like in Figure 8). The words in a topic model are defined by a fixed vocabulary and the sum of word probabilities in a topic must be 1 (STEYVERS; GRIFFITHS, 2007).

Figure 9 – Example of a distribution over topics

Topic 247	Topic 5	Topic 43	Topic 56
DRUGS .069	RED .202	MIND .081	DOCTOR .074
DRUG .060	BLUE .099	THOUGHT .066	DR. .063
MEDICINE .027	GREEN .096	REMEMBER .064	PATIENT .061
EFFECTS .026	YELLOW .073	MEMORY .037	HOSPITAL .049
BODY .023	WHITE .048	THINKING .030	CARE .046
MEDICINES .019	COLOR .048	PROFESSOR .028	MEDICAL .042
PAIN .016	BRIGHT .030	FELT .025	NURSE .031
PERSON .016	COLORS .029	REMEMBERED .022	PATIENTS .029
MARIJUANA .014	ORANGE .027	THOUGHTS .020	DOCTORS .028
LABEL .012	BROWN .027	FORGOTTEN .020	HEALTH .025
ALCOHOL .012	PINK .017	MOMENT .020	MEDICINE .017
DANGEROUS .011	LOOK .017	THINK .019	NURSING .017
ABUSE .009	BLACK .016	THING .016	DENTAL .015
EFFECT .009	PURPLE .015	WONDER .014	NURSES .013
KNOWN .008	CROSS .011	FORGET .012	PHYSICIAN .012
PILLS .008	COLORED .009	RECALL .012	HOSPITALS .011

Source – (STEYVERS; GRIFFITHS, 2007)

For example, Figure 9 presents four topics obtained from the TASA collection (LANDAUER; FOLTZ; LAHAM, 1998), which has over 37,000 text passages from educational materials (STEYVERS; GRIFFITHS, 2007). Remark that the topics are identified by numbers and show only their respective top-16 words. Each word is followed by its probability, where, for example, the word “red” has a probability of 20.2% to be found in documents associated to topic 5.

4.1.1 Generative process

Given a topic model, it is possible to use a generative process to pick words from its distribution over topics and create new documents (BLEI, 2012; STEYVERS; GRIFFITHS, 2007). This generative process can be generalized by a simple probabilistic procedure that operates in the two-stage process below:

- a) Randomly choose a set of one or more topics;
- b) For each word in the document:
 - Randomly choose a topic from the chosen topics;
 - Randomly choose a word from that topic.

For example, if creating a 100-words document from some topics in Figure 9, the steps to follow are: (a) choose the new document topics (e.g., topics 247 and 5) and (b) pick words from these topics based on the probabilities of the words until the size of the document is 100 (e.g., “red drugs bright dangerous red color effects...”). Remark that “drugs” and “red” tends to be the most frequent words in the new document because of their probabilities.

4.1.2 Inference process

Instead of generating new documents according to a given topic model, some works need to automatically discover the distribution over topics from a given collection. In that case, topic modeling also presents techniques that use a given collection of documents to do the reverse process: infer the topic model that probably generated the collection (BLEI, 2012; STEYVERS; GRIFFITHS, 2007).

This inference aims to discover the hidden patterns in documents and words to build a topic model (BLEI, 2012). There are several inference techniques, which usually work making use of probabilistic and statistical methods (BLEI, 2012). As identifying the topic structure from a large collection may provide many insights for human cognition (STEYVERS; GRIFFITHS, 2007), this kind of inference is addressed in several works.

4.1.3 Labeling process

The topic modeling inference usually results in topics identified by numbers that differentiate each other, but devoid of useful meaning about their content. For example, the topics presented in Figures 8 and 9 have no labels, but colors and numbers identifying them. There are some studies addressing automatic topic labeling (ALLAHYARI; KOCHUT, 2015), but it is possible to label topics by manually analyzing their distributions over words.

For example, the four topics presented in Figure 9 may be labeled as “drug use” (topic 247), “colors” (topic 5), “memory and mind” (topic 43), and “doctor visits” (topic

56) (STEYVERS; GRIFFITHS, 2007). Thus, the topics look more comprehensible for human cognition and give a better notion about the content presented in documents.

4.2 LATENT DIRICHLET ALLOCATION

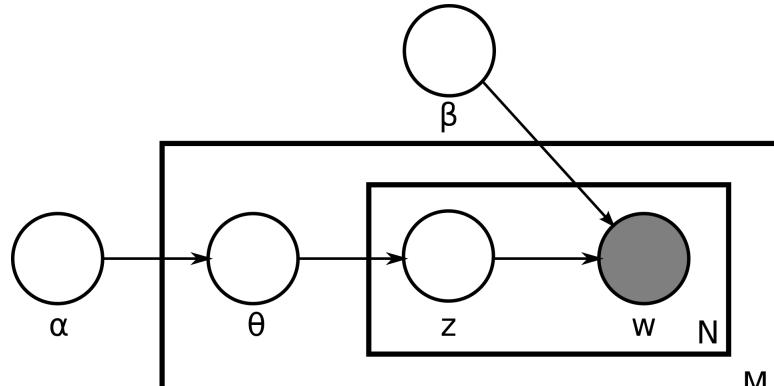
Latent Dirichlet Allocation (LDA for short) is a popular topic model that is considered the simplest one (BLEI, 2012). LDA is an unsupervised learning approach to discover and associate latent variables hidden in observed variables, independent of what the variables represent (BLEI; NG; JORDAN, 2003). In the case of topics and documents, the latent variables are the topics and the observed variables are the documents (BLEI, 2012).

LDA is described as a generative probabilistic model that generates documents using the Dirichlet distribution to randomly choose per-document topic distributions and per-topic word distributions (BLEI; NG; JORDAN, 2003). As Dirichlet distribution is a family of continuous and multivariate probability distributions that are parameterized by a vector of positive real numbers (MINKA, 2000), its properties facilitate LDA inference (BLEI; NG; JORDAN, 2003). LDA definitions and representations are based on formal notation and terminology, which are presented in the following.

- D is the corpus, *i.e.*, the collection of documents, where d is a document
- M is the number of documents in the corpus
- N is the number of words in a given document, where a document d has N_d words
- K is the number of topics
- θ_d is the topic distribution for document d
- z_{dn} is the topic for the n -th word in document d
- w_{dn} is a specific word.
- α is the parameter of the Dirichlet prior on the per-document topic distributions
- β is the parameter of the Dirichlet prior on the per-topic word distribution

Figure 10 presents LDA probabilistic graphical model in a plate notation using LDA formal notation and terminology (BLEI; NG; JORDAN, 2003). The boxes represent iterations of specific elements, whereas circles (or plates) represent LDA variables. The outer box iterates the M documents and apply the parameter α to Dirichlet distribution in order to compute the θ_d for each document d . Then, the inner box iterates the N_d words from each iterated document d and apply the parameter β to Dirichlet distribution in order to use the topic assignment z_{dn} to the word w_{dn} .

Figure 10 – LDA probabilistic graphical model



Source – (BLEI; NG; JORDAN, 2003; BLEI, 2012)

These formal definitions of LDA serve as a guide to define how LDA inference process works. Dirichlet is a convenient distribution because “[...] it is in the exponential family, has finite dimensional sufficient statistics, and is conjugate to the multinomial distribution” (BLEI; NG; JORDAN, 2003, p. 996), which facilitate the inference process (BLEI; NG; JORDAN, 2003).

LDA inference analyzes documents to identify word occurrence patterns and use them to build topics (BLEI, 2012). Although this inference is considered intractable because of its complexity, LDA does that by applying approximation methods, such as Laplace approximation, variational Bayes approximation, and Markov chain Monte Carlo (BLEI; NG; JORDAN, 2003).

When inferring the distribution over topics that probably generated a given collection of documents, LDA requires a corpus where each document is in a bag-of-words format, that is, a text representation that describes the occurrence of words within a document (BLEI; NG; JORDAN, 2003). This representation is simple: each document is represented by a list with the number of times each unique word appears in the document. Hence, this format assumes that the documents have a fixed dictionary defining all unique words the collection is composed of (BLEI; NG; JORDAN, 2003), making it necessary to compute the dictionary before performing the inference.

Besides, some works compute the term frequency-inverse document frequency (TF-IDF for short), a numerical statistic that calculates how important a word is to a document in a collection (RAMOS et al., 2003). TF-IDF is defined as the product of two statistics: term frequency and inverse document frequency, which successfully weight words importance (RAMOS et al., 2003). Instead of presenting the number of occurrences for each word (bag-of-words), TF-IDF calculates a weight representing the importance of each word in a document (RAMOS et al., 2003).

As TF-IDF results can be represented in the bag-of-words format, it is possible to directly use TF-IDF as input to LDA inference. Indeed, TF-IDF is considered the most frequently applied weighting method in recommender systems and topic modeling works (BEEL et al.,

2016). Therefore, when doing a LDA inference, it is necessary to compute the dictionary and the bags of words (or TF-IDF) from the given collection (BLEI; NG; JORDAN, 2003).

However, the inference does not guarantee a high-quality result. Applying TF-IDF and NLP techniques may improve the inferred model, but in order to detect these improvements, it is necessary to apply metrics. Therefore, the next section presents some metrics capable to evaluate a model quality and help to improve it.

4.3 METRICS

Topic modeling inferring techniques have no prior knowledge about the collection topics before inferring them and, therefore, are considered unsupervised techniques (LAU; NEWMAN; BALDWIN, 2014). For that reason, evaluating the quality of an inferred model is a challenging task, because there are no known labels to guide an accuracy or precision analysis (CHANG et al., 2009; LAU; NEWMAN; BALDWIN, 2014).

In that case, it is possible to use human evaluation by analyzing some aspects from the results, but performing that on a large scale may become expensive and impracticable, further to susceptible to potential human mistakes. Therefore, automated evaluation metrics may be very useful for that purpose (RÖDER; BOTH; HINNEBURG, 2015).

Coherence metrics are considered relatively close to human perception (CHANG et al., 2009; LAU; NEWMAN; BALDWIN, 2014), which make them good choices. They work by analyzing the semantic similarity and associativity of words and topics from a given collection and its inferred topics (CHANG et al., 2009). There are several coherence metrics and several methods that they are based on, some of them are presented next.

4.3.1 Sliding and context windows

There are many coherence metrics that make use of *sliding window* and *context window*. Assuming that consecutive words may have some kind of association, both sliding and context windows techniques consist of splitting a given set of words into subsets that may provide useful information to the coherence metric computation (RÖDER; BOTH; HINNEBURG, 2015).

A *sliding window* is a subset of N consecutive words that “slides” over a given set of words, word by word (RÖDER; BOTH; HINNEBURG, 2015; CHANG et al., 2009). For example, if creating a *sliding window* of size 2 from the set of words $D = \{w_1, w_2, w_3, w_4\}$, the results would be $SW = \{w_1, w_2\} \rightarrow \{w_2, w_3\} \rightarrow \{w_3, w_4\}$.

A *context window* is a subset of N consecutive words before and after a given word from a set of words (RÖDER; BOTH; HINNEBURG, 2015; CHANG et al., 2009). For example, if creating a *context window* of size 1 given the word w_3 from the set of words $D = \{w_1, w_2, w_3, w_4\}$, the results would be $CW = \{w_2, w_4\}$.

4.3.2 PMI and NPMI

Pointwise Mutual Information (PMI for short) is a method capable measuring the associativity between two words (RÖDER; BOTH; HINNEBURG, 2015; ALETRAS; STEVENSON, 2013; CHANG et al., 2009). Some works reveal that coherence metrics using PMI-based methods tend to give results relatively similar to human rating (RÖDER; BOTH; HINNEBURG, 2015; CHANG et al., 2009). Formally, PMI is defined as the equation in the following (RÖDER; BOTH; HINNEBURG, 2015):

$$PMI(w_i, w_j) = \log \left(\frac{P(w_i, w_j) + \epsilon}{P(w_i)(w_j)} \right) \quad (4.1)$$

Assuming that w_i and w_j are given words, $P(w_i, w_j)$ is the frequency both words are found in the same window (either sliding or context), and $P(w_i)$ and $P(w_j)$ are the frequency each word is found separately (RÖDER; BOTH; HINNEBURG, 2015; ALETRAS; STEVENSON, 2013).

PMI results provide useful information about word semantics and there are several PMI-based methods. Normalized Pointwise Mutual Information (NPMI for short) is a variation that normalizes PMI results to $[-1, 1]$ intervals (RÖDER; BOTH; HINNEBURG, 2015; ALETRAS; STEVENSON, 2013). NPMI is formally defined as the equation in the following (RÖDER; BOTH; HINNEBURG, 2015):

$$NPMI(w_i, w_j) = \frac{PMI(w_i, w_j)}{-\log(P(w_i, w_j))} = \frac{\log \left(\frac{P(w_i, w_j) + \epsilon}{P(w_i)(w_j)} \right)}{-\log(P(w_i, w_j))} \quad (4.2)$$

Assuming that w_i and w_j are given words, a result next to -1 represents no co-occurrence of the two given words, whereas a result next to 1 represents complete co-occurrence. A 0 result represents the given words are independent of each other. This NPMI property facilitates the results interpretability by both humans and coherence metrics that applies NPMI (RÖDER; BOTH; HINNEBURG, 2015).

4.3.3 C_v coherence metric

C_v is a coherence metric that, considering a given word from a given topic, calculates the co-occurrence of this word against all words from its topic (RÖDER; BOTH; HINNEBURG, 2015). This calculation applies a variation of NPMI over a *sliding window* of size 110 and results in a set of vectors, one for each word. Formally, C_v is defined as the equation in the following (SYED; SPRUIT, 2017; RÖDER; BOTH; HINNEBURG, 2015):

$$\vec{v}(W') = \left\{ \sum_{w_i \in W'} NPMI(w_i, w_j)^\gamma \right\}_{j=1, \dots, |W|} \quad (4.3)$$

Experiments show that C_v has a better performance in comparison to other coherence metrics, such as C_{UCI} , C_{NPMI} , C_A , C_P (see (RÖDER; BOTH; HINNEBURG, 2015) for further details). Therefore, C_v is a good choice when checking the representability of a given distribution over topics according to its original collection.

4.4 DISCUSSION

This chapter briefly presented topic modeling general concepts and techniques. As discovering topics from Stack Overflow posts is an essential part of this work, LDA may be used for that. Hence, it is also necessary to compute the dictionary, bags of words, and TF-IDF from the collection. The *gensim* library¹ for Python provides many classes and methods to work with topic modeling, including the aforementioned ones.

Besides, the quality of LDA inference depends on the collection quality and results of other applied techniques. For example, NLP methods improve the collection quality in many aspects, whereas several of the aforementioned techniques have hyperparameters that also affect the final result. Therefore, C_v coherence metric is essential to help to improve topics representability.

Finally, it is necessary to understand how other works apply the aforementioned concepts and techniques. Therefore, the next chapter presents a set of works presenting related objectives to this work in different aspects, which can base the addressed methodology.

¹ <https://radimrehurek.com/gensim/>

5 RELATED WORK

Identifying latent topics from a given collection of documents may provide insightful information. Several works have applied topic modeling to several kinds of applications, such as collaborative filtering, information retrieval, authorship identification, recommender systems, and opinion extraction (YANG et al., 2019).

However, many conventional approaches usually do not consider extra features hidden in documents metadata, such as authorship and creation/publish date (XU; SHI; QIAO; ZHU; JUNG, et al., 2014; XU; SHI; QIAO; ZHU; ZHANG, et al., 2014). Analyzing these extra features may help the development of personalized and user-centric applications (YANG et al., 2019). Therefore, the following sections present some topic model approaches and exploratory analyses that address this kind of study.

5.1 TOPIC MODELS

Analyzing authorship in topic modeling processes may provide useful information about authors' interests. Rosen-Zvi *et al.* (2004) address the authorship problem by proposing the Author-Topic (AT) model, an LDA-style generative model that includes authorship information. AT model works by associating authors to multinomial distributions over topics that proportionally define the topics each author is related to.

However, the authors' interests are not static. Time is a significant factor, because authors' interests may drift and evolve over time. The Topics over Time (TOT) model, proposed by Wang and McCallum (2006), is an LDA-style model capable of tracking the topic evolution over time, similarly to the Dynamic Topic Model (DTM) proposed by Blei and Lafferty (2006). However, although both models address topic evolution tracking, they do not consider authorship information.

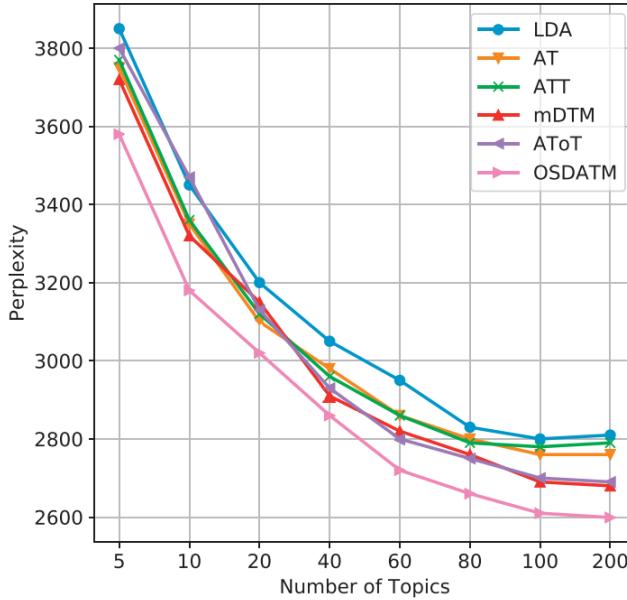
There are some works that address the authors' interests evolution tracking. Tang and Zhang (2010) propose the Author-Time-Topic (ATT) model, which combines AT and TOT models to estimate topic distributions over words and timestamps and their associations with authors. Besides, Temporal-Author-Topic (TAT) model, proposed by Ali Daud (2012), makes the same combination, but using annual timestamps rather than dynamic timestamps.

Furthermore, Xu *et al.* (2014) propose the Author-Topic over Time (AToT) model, which also combines AT and TOT models, but applying different inference methods. AToT model also has a distributed variation called Distributed Author-Topic over Time (D-AToT), which make use of multiple processors and improve the model performance (XU; SHI; QIAO; ZHU; ZHANG, et al., 2014).

Recently, Yang *et al.* (2019) propose the Ordering-sensitive and Semantic-aware Dynamic Author Topic Model (OSDATM), a novel model for authors' interests evolution tracking. Different from the other aforementioned models, OSDATM is not LDA-style. Instead of using

the bag-of-words assumption, OSDATM is sensitive to word ordering and claims to learn better topics than state-of-the-art topic models (YANG et al., 2019).

Figure 11 – Perplexity as a function of the number of topics on NIPS dataset



Source – Adapted from (YANG et al., 2019)

Figure 11 presents the performance of the previously presented approaches, that is, LDA, AT, ATT, AToT, and OSDATM. The performance was measured using the Neural Information Processing Systems (NIPS) collection, which represents the proceedings of the top machine learning conferences in the world; and the perplexity metric, which measures how a model deals with unseen documents, where the lower result, the better the model (WALLACH et al., 2009; NEWMAN et al., 2010).

Furthermore, remark that, OSDATM presents the best evaluations no matter the number of topics. On the other hand, LDA is the worst one. Besides, observe that the performances of all models in Figure 11 tend to stabilize when they reach 100 topics.

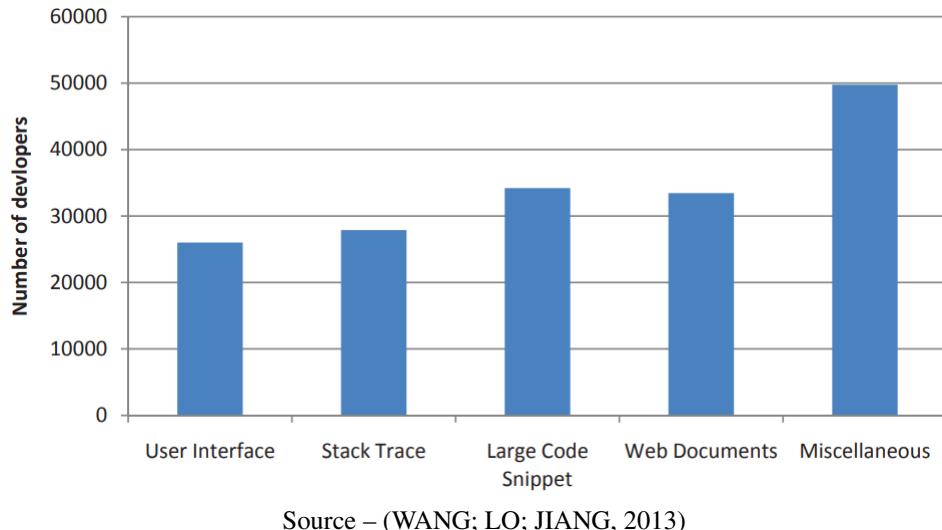
5.2 EXPLORATORY ANALYSES

Topic models are proposed to be employed in other studies. Among others, an exploratory analysis analyzes a collection of data aiming to discover insightful information from it. In topic modeling, the discovered information is given as distributions over topics, which can be analyzed for many purposes. Therefore, as this work is an exploratory analysis, understanding other ones that present related objectives is essential to guide the methodology addressed in this work.

Wand, Lo, and Jiang (2013) present an exploratory analysis of user interactions in Stack Overflow. The authors investigate users' behaviors, analyzing their bias as questioners and answerers, and employs topic modeling to assign topics to questions from Stack Overflow. They

extracted 63,863 questions from Stack Overflow, including their contents, authors, and answers. Thereafter, the extracted collection was subjected to a processing step that, among other results, separated all questions from their code snippets in a sub-collection.

Figure 12 – Histogram of developers per topic plotted by Wand, Lo, and Jiang (2013)



Then, a text mining step was applied, where a pre-processing step performed tokenization, stopword removal, and stemming, and a topic modeling step inferred 5 topics from both normal texts and code snippets using LDA (WANG; LO; JIANG, 2013). Figure 12 shows a histogram presenting the 5 inferred topics and the number of developers related to each topic. Observe that the topics have broad labels and the number of developers for each topic is very similar, except for *Miscellaneous* topic.

However, although this aforementioned work has analyzed the relationship between topics and developers, it does not consider the time factor. Therefore, Barua, Thomas, and Hassan (2014) present an exploratory analysis addressing research questions strongly related to this work. Among others, this work aims to discover the main topics in Stack Overflow and how developers' interests change over time.

Figure 13 – Methodology applied by Barua, Thomas, and Hassan (2014) in a flowchart

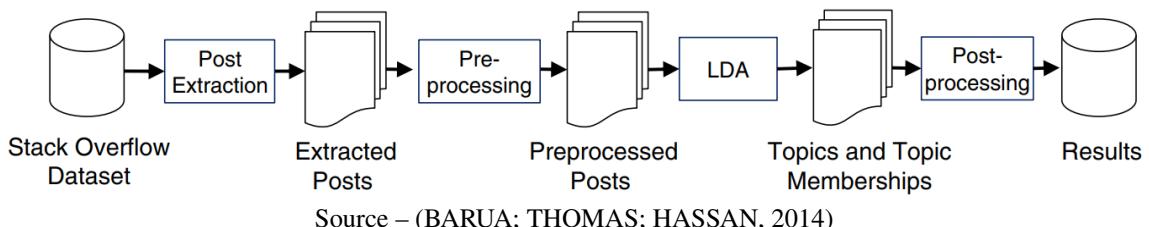


Figure 13 shows a flowchart presenting an overview of the methodology applied by Barua, Thomas, and Hassan (2014). They extracted 3,474,987 questions and answers from

Stack Overflow, each one separated into its own document. Also, every post includes the body, timestamp, type (question or answer), user-specified tags, and the question-answer relationships.

The pre-processing step, unlike Wand, Lo, and Jiang (2013), discarded all code snippets, because most source code is similar on syntax and keywords, which represent noise into further processing (BARUA; THOMAS; HASSAN, 2014). Then, all HTML tags were removed, maintaining only the text. Finally, tokenization, stopword removal, and stemming also have been applied.

The topic modeling process concatenated bi-grams into the pre-processed collection and employed the LDA implementation provided by Mallet (MCCALLUM, 2002). After empirical experimentation, the author set the number of topic to 40, each one manually labeled based on the 4-top words (BARUA; THOMAS; HASSAN, 2014).

Besides, as every document is associated with all topics in distinct proportions, $\gamma = 0.1$ was defined as a threshold to determine if a topic is or not related to a document (BARUA; THOMAS; HASSAN, 2014). In other words, a document is associated with a topic if this association is greater than or equal to 10%.

Finally, the post-processing step applied some metrics to answer the proposed research questions. Among them, it is possible to highlight two metrics: *share* and *impact* metrics. The *share* metric measures the relative popularity of a topic across all documents. Considering D the collection and $\theta(d_i, z_k)$ the weighted topic membership for document d_i and a given topic z_k , *share* metric is defined by the equation 5.1.

$$\text{share}(z_k) = \frac{1}{|D|} \sum_{\theta(d_i, z_k) \geq \gamma}^{d_i \in D} \theta(d_i, z_k) \quad (5.1)$$

The *impact* metric is a variation of *share* metric that “[...] measures the relative proportion of posts related to that topic compared to the other topics in that particular month” (BARUA; THOMAS; HASSAN, 2014, p. 9). Although the original work has considered time slices defined as months, it is possible to apply *impact* metric to any time slice. Indeed, *share* and *impact* metrics are almost the same, but *impact* splits the documents according to the time slices they belong to. Considering m a given month (or time slice), *impact* metric is defined by the Equation 5.2.

$$\text{impact}(z_k, m) = \frac{1}{|D(m)|} \sum_{\theta(d_i, z_k) \geq \gamma}^{d_i \in D(m)} \theta(d_i, z_k) \quad (5.2)$$

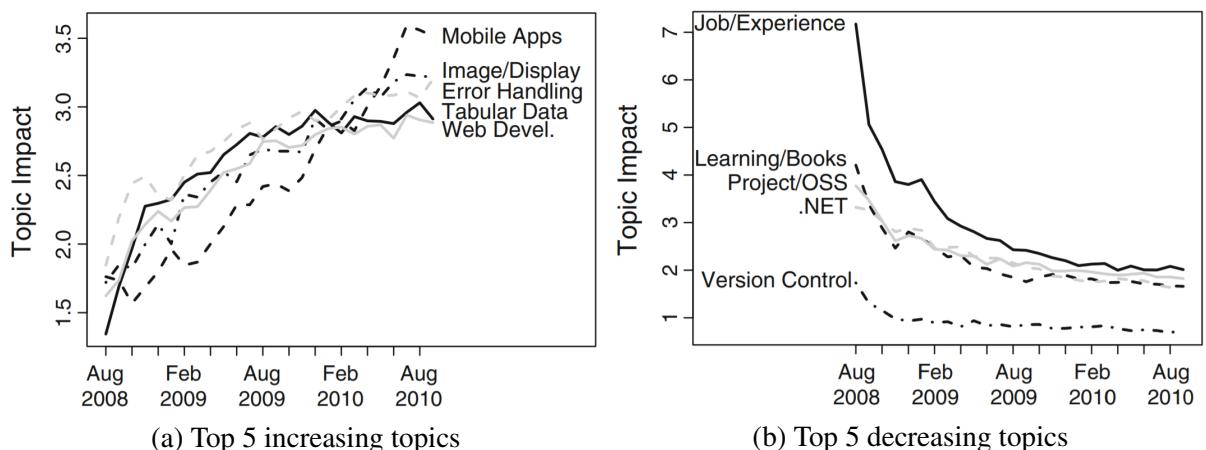
By applying *share* metric to all 40 topics and ordering them in descending order, Barua, Thomas, and Hassan (2014) discovered the main trend topics in Stack Overflow from 2008 August to 2010 August. Table 2 presents the top 5 trend topics and their respective *shares* and trend patterns. Observe that *share* metric generates proportion-like measures, where the sum of all *shares* must be 1 (or 100%).

Table 2 – Trend topics from Stack Overflow by Barua, Thomas, and Hassan (2014)

Topic label	share	Trend	Trend line
Coding Style/Practice	4.5%	↓	
Problem/Solution	4.5%	↑	
QA and Links	3.8%	↓	
Function	3.4%	↑	
Job/Experience	3.3%	↓	

Source – (BARUA; THOMAS; HASSAN, 2014)

Figure 14 – Top 5 increasing and decreasing trends in Stack Overflow by Barua, Thomas, and Hassan (2014)



Besides, they also applied *impact* metric for each time slice (*i.e.*, monthly), computing every topic evolution and trend patterns. Figure 14 presents two charts, where 14a shows the top 5 increasing topics and 14b shows the top 5 decreasing topics from 2008 August to 2010 August.

According to these and other results, Barua, Thomas, and Hassan (2014) claim that web-related discussions are the most popular ones in Stack Overflow, such as web development, web design, and web service. Furthermore, mobile development discussions also are very common and tends to increase.

5.3 DISCUSSION

This chapter presented some works that have objectives related to ours. Several of them are topic models proposed to deal with authorship and temporal information, whereas other works are applications that employ, among others, topic modeling and NLP techniques to track users' interests and topic evolution from Stack Overflow.

Most of the proposed models applied perplexity to evaluate and compare their results against other models. However, although inferred topics must be meaningful for humans, perplexity and other traditional metrics are not correlated to human interpretation (CHANG et al., 2009). Instead, coherence metrics are strongly correlated to human interpretation and are more recommended than perplexity to evaluate topic modeling results (CHANG et al., 2009).

Although some topic model approaches that could be useful for the objectives of this work have been presented, the discussed exploratory analyses have applied LDA as the topic model algorithm. A possible reason is that LDA is available in several frameworks and several programming languages, whereas most presented topic models approaches are hard to find and complex to implement. Furthermore, considering LDA output provides topics and their memberships for each document, it is easy to algorithmically compute authorship and temporal relations if the collection has this information.

Besides, the discussed exploratory analyses also provide important methodological information. Wand, Lo, and Jiang (2013) have subjected only questions to topic modeling process, including normal text and code snippets. Differently, Barua, Thomas, and Hassan (2014) have extracted both questions and answers, further to discarding all code snippets claiming that they represent noise to posterior processes.

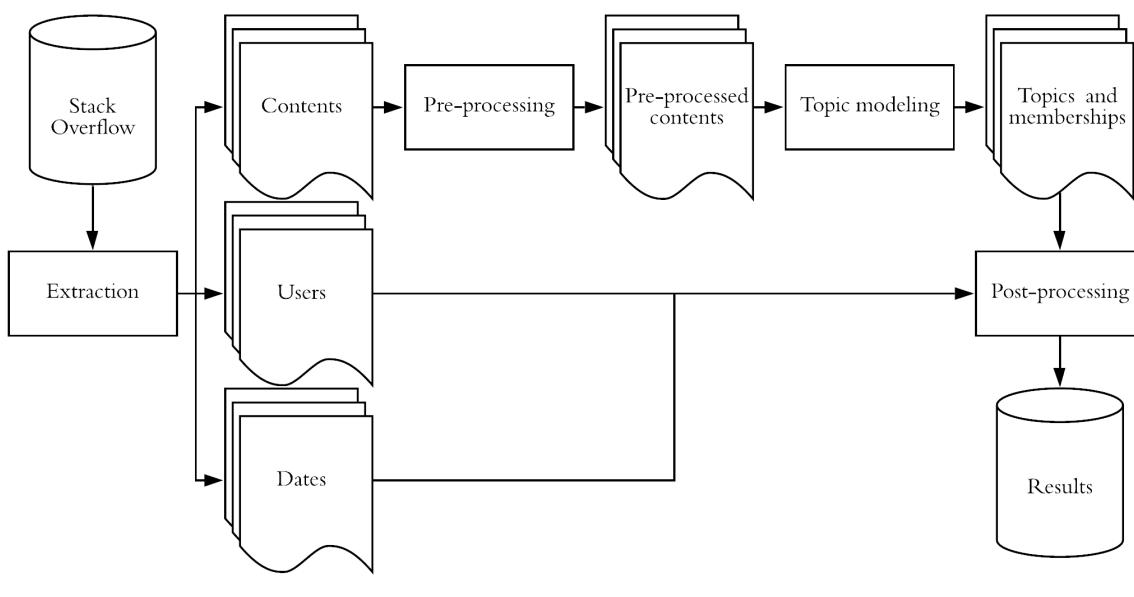
Furthermore, both works have employed a pre-processing step before the topic modeling step, which applied tokenization, stopword removal, and stemming. However, Barua, Thomas, and Hassan (2014) have concatenated bi-grams to each document. Also, they inferred 40 topics much more detailed and insightful than the 5 topic inferred by Wand, Lo, and Jiang (2013).

Moreover, the *share* and *impact* metrics have been remarkable when computing the trend topics and the changes over time for each topic and time slice. It is possible to adapt both metrics to make them consider authorship information, which would be important to accomplish the objectives of this work.

6 EXPERIMENTS

As introduced in chapter 1, this work aims to perform an exploratory analysis from Stack Overflow posts. The purpose of this study is to analyze the discussed topics across authors and publishing dates to track topic popularity evolution, understand their trends, and identify how topic popularity drifted over time. With these data, it is possible to get general insights on Stack overflow topics and user-centric insights for each user.

Figure 15 – An overview of the addressed methodology in a flowchart



Source – from authors

Considering these purposes, Figure 15 presents an overview of the employed methodology in a flowchart. First of all, extracting posts from Stack Overflow database is an essential step. Thereafter, the extracted collection needs to be subjected to a pre-processing step, which cleans and enriches the extracted posts. Then, the topic modeling step performs experiments applying several techniques to discover the distribution over topics and topic memberships for each post. Finally, the discovered topics and temporal-authorship information are analyzed in the post-processing step to compute metrics and generate charts.

6.1 EXTRACTION

First, the `Posts.xml`¹ file was downloaded from Stack Overflow data dump. This data dump has other XML files, one for each Stack Overflow data entity (*e.g.*, `Users.xml`, `Tags.xml`, `Comments.xml`, etc), but `Posts.xml` already has all the necessary data: all non-deleted posts since 2008, including authorship and temporal information. The downloaded posts were last updated in September 2020, including **50,337,841** posts.

¹ <https://archive.org/download/stackexchange/stackoverflow.com-Posts.7z>

After downloading the posts, they were filtered according to two requirements. First, every extracted post must be a question or answer, which can be verified by checking the `PostTypeId` attribute, where 1 is for questions and 2 for answers. Second, as posts may present some missing information, every extracted post must have non-null values in the `OwnerUserId`, `CreationDate`, and `Body` attributes, because they are essential for posterior steps. The posts violating any one of these requirements were removed from the collection, resulting in **739,023** removed posts and in a collection of **49,598,818** posts.

After filtering the posts, the necessary information was extracted and stored into `.txt` files. For each post, the user who created it, the publishing date, and the content itself was stored. The post author was extracted from the `OwnerUserId` attribute, which is an identifying number. Then, the creation date was extracted by accessing the `CreationDate` attribute, but hours, minutes, and seconds are discarded by applying a `yyyy-mm-dd` format.

The content of a post is primarily composed of the `Body` attribute. However, as `Title` and `Tags` attributes also contain useful information about the topics related to a post, they were concatenated to the content. In the case of answers, which have no titles or tags, the `ParentId` attribute can be used to identify the parent question by its `Id` and concatenate its `Tags` to the content of its respective answers.

All these data were stored into three different files: `users.txt`, `dates.txt`, and `contents.txt`, where the n -th line in each file represents the user, date, and content belonging to the n post. As authorship and temporal information are only employed in the post-processing step, storing the content separated from them improves the performance when analyzing only the post contents.

6.2 PRE-PROCESSING

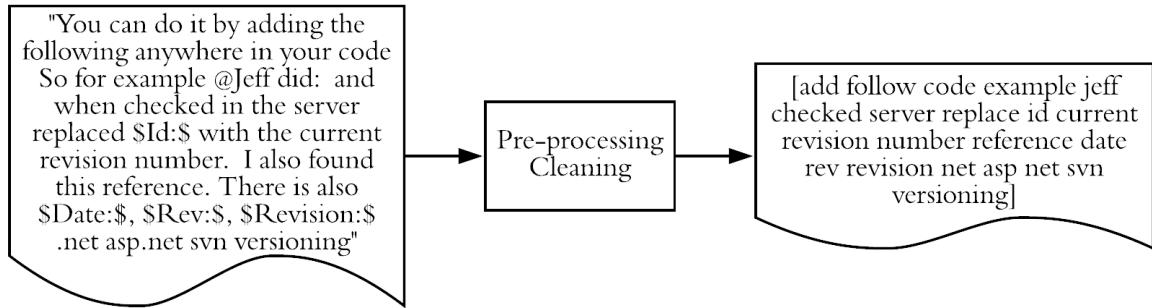
After extracting Stack Overflow posts, their contents were subjected to a pre-processing step divided into two sub-steps: cleaning and enrichment. First, posts were cleaned by discarding code snippets (surrounded by `<code>` tags), removing HTML tags (such as `<p>` and ``), and performing tokenization, stopword removal, and lemmatization. For HTML manipulation, the Beautiful Soup² library for Python was employed, where NLP techniques were obtained from the NLTK³ library.

Although stemming presents good results with better performance than lemmatization, employing lemmatization can maximize the quality of the results, even if impacting the performance. This performance loss occurs because capturing the part-of-speech of each word in the collection is a costly task, but doing that helps to pre-process the data properly and provide better results in posterior steps.

² <https://pypi.org/project/beautifulsoup4/>

³ <https://www.nltk.org/>

Figure 16 – Random post content subjected to the cleaning sub-step



Source – from authors

Figure 16 presents an example of applying the cleaning sub-step to a random post. Observe that all words were cast to lower case and all stopwords and punctuation marks were removed. Also, inflected words were reduced to their lemmas, such as “adding”, “following”, and “replaced”. Besides, as lemmatization considers part-of-speech, “versioning” was not mistakenly reduced to “version” because this word is a noun.

After cleaning the posts, they were subjected to an enrichment sub-step adding information to refine the results. Indeed, computing n -grams can improve text categorization processes, but can also be a very costly task. Fortunately, computing bi-grams are sufficient to substantially raise the quality of a collection (TAN; WANG; LEE, 2002).

Therefore, the enrichment sub-step consists of computing and concatenating the bi-grams of each post to its content. In other words, this sub-step resulted in a collection using both uni-grams and bi-grams. The final result of the pre-processing step was stored into the `pre-processed-contents.txt` file, which follows the same rules presented in the extraction step: the n -th line in this file represents the pre-processed content belonging to the n post.

Figure 17 – Random post content subjected to the enrichment sub-step



Source – from authors

Figure 17 presents an example of applying the enrichment sub-step to the same random post in Figure 16. Although there are no many changes, observe that the sub-step successfully generated the “asp_net” bi-gram. Separated, the words “asp” and “net” may present many

different meanings according to the sentence, but as a bi-gram, they are clearly related to ASP.NET, an open-source web framework.

6.3 TOPIC MODELING

After pre-processing the collection, the topic modeling step was executed. This step consists of performing experiments applying the pre-processed contents to topic modeling techniques to infer the distribution over topics and generate the topic memberships for each post. This step also has three sub-steps: corpus building, LDA inference, and labeling. These sub-steps were implemented employing tomotopy⁴ library for Python.

6.3.1 Corpus building

First of all, the corpus must be defined to perform the LDA inferences. Although the pre-processing step removes and summarizes post contents to improve the topic modeling results, there is the possibility that short content posts have all words removed. Therefore, these empty posts were ignored when building the corpus, resulting in **25,214** ignored posts and **49,573,604** used posts.

Besides, the corpus dictionary defines which words the topic model inference considers when inferring topics. Filtering the words in the dictionary may help to improve the quality of the inferred topics. Discarding words that appear in very few posts or that appear in too many posts may help to keep only the relevant information. However, there are no filtering parameters that are appropriate to all cases as they produce different results depending on the collection.

Therefore, the collection was filtered by many different parameters to define appropriate filtering. Words with a smaller document frequency than 200 were removed from the dictionary, whereas the top 20 words were also removed. Table 3 presents the statistics about the words and vocabularies in the corpus, including the removed ones.

Table 3 – Corpus definition

Words	Vocabs	Used vocabs
1,523,460,528	5,836,775	59,478

Finally, the collection was transformed into a corpus able to be applied in a topic model inference. As LDA was chosen to perform the topic inference, the corpus must be in the bag-of-words format. Therefore, the TF-IDF was employed to transform and prepare the corpus for LDA inference.

⁴ <https://bab2min.github.io/tomotopy/>

6.3.2 LDA inference

The inference sub-step consists of performing multiple experiments with LDA inference employing different hyperparameters and measuring the C_v coherence of each inferred topic model. The interest hyperparameters are the number of topics K and the number of iterations i . Considering that the higher C_v coherence the better the model, these experiments are useful to find the best set of K and i .

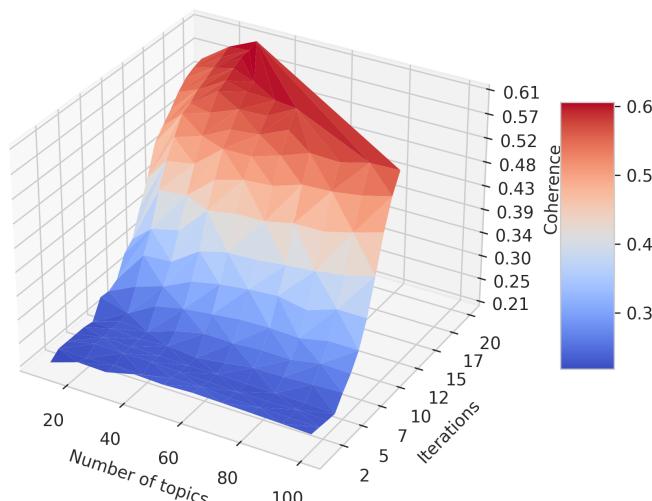
The number of iterations i defines how many times the model iterates the entire collection to infer the topics. As more iterations do not necessarily result in better models, it is important to consider them in the experiments. However, as more iterations result in larger LDA models, the number of iterations i also impacts the performance as computational resources are finite.

Besides, K is very determinant to define the general behavior of the inferred topics. Larger values of K produce more detailed and finer-grained topics, whereas smaller values of K produce more general and coarser-grained topics (BARUA; THOMAS; HASSAN, 2014). Furthermore, the collection size also is determinant of the granularity of the K inferred topics. Consequently, there is no value of K that is most appropriate for all collections, because each case may present different behaviors. Therefore, the experiments are essential to define the best number of topics K to summarize Stack overflow data.

Finally, several experiments were performed, each one with a hyperparameter combination coming from $K = \{10, 20, 30, \dots, 100\}$ and $i = \{1, 2, 3, \dots, 20\}$. After performing each experiment, the C_v coherence was computed to evaluate the topic model quality. Every hyperparameters set and respective coherence was stored into the `experiments.csv` file, allowing further analyses.

Figure 18 – 3D chart of Stack Overflow C_v coherence by number of topics K and iterations i

Best experiment: iterations=20 topics=30 coherence=0.6140



Source – from authors

Figure 18 presents a triangulated 3D chart with the results of these experiments. Apparently, as K and i increased, the C_v coherence also increased. Also, all number of topics presented low coherence with less than 12 iterations and the best experiment presented approximately 0.6140 of coherence for 30 topics and 20 iterations. This result means that the best number of topics for this collection and these experiments is 30 topics. Appendix A shows the inferred topics and their respective top words.

6.3.3 Labeling

The last sub-step for topic modeling is labeling the inferred topics. There is no method that can be considered the most appropriate for topic labeling. Several topic modeling studies label the topics by analyzing the top-10 words of each topic manually and empirically, whereas other studies prefer to ask human labeling.

In this work, the labeling process was performed by searching the top-10 words of each topic on Google and Scholar Google. As Stack Overflow features questions and answers on computer programming topics, this labeling method was efficient because many top-words were easily found in documentations. The search results were empirically analyzed to understand their common factors and define a label.

However, as some search results presented ambiguities or no common factors, the results were improved by removing the less relevant words from the search query. Appendix A shows the labels assigned to each topic according to their respective top-10 words. Concluding this step, the labeled topics and their respective top-10 words were stored into the `labeled-topics.csv`.

6.4 DISCUSSION

The performed steps in this chapter were aimed at preparing posts' content and inferring the distribution over topics from them. Several experiments were conducted to define the topics, where the C_v coherence was the main factor for that. Therefore, considering the C_v measures and the computational resources this work had available, the best number of topics is 30.

As the number of iterations reached just 20 because of the limited computational power, performing these experiments with more iterations could improve even more the quality. However, the resulted distribution over topics and the label assignments generated very promising topics. They are distinct from each other and appear to be more specific than broad. Also, as C_v coherence is relatively close to human interpretation of topics (CHANG et al., 2009; LAU; NEWMAN; BALDWIN, 2014), these results are solid and well-founded even with 20 iterations.

7 RESULTS

After concluding all experiments and finding Stack Overflow topics, the post-processing step was performed to analyze the results and compute metrics. This step consists of using authorship and temporal information, the inferred topics, and the topic memberships to make exploratory analyses. These analyses are divided into two categories: general analyses and user-centric analyses. For each one, several metrics were computed and made available in the Internet Archive¹ website, allowing anyone on the Web to analyze the generated results.

7.1 METRICS

First, it is necessary to understand the metrics employed to make these analyses. Proposed by Barua, Thomas, and Hassan (2014), the *share* (see Equation 5.1) and *impact* (see Equation 5.2) metrics can measure the relative popularity of a topic across all documents D and across the documents $D(m)$ belonging to a given month m , respectively. Indeed, both metrics are very similar; the difference between them is that the *share* metric analyzes the entire collection, whereas *impact* analyzes a sub-collection composed of the posts in a given month.

However, as these metrics do not consider authorship information, they was adapted to become appropriate for this work. Rather than computing the popularity of a topic across a predefined set of documents, it is possible to compute the same metric across any set of documents. Hence, considering SD a given set of documents, z_k a given topic, and $\theta(d_i, z_k)$ the weighted topic membership for document d_i and topic z_k , the Topic Popularity TP is the proposed metric defined by the equation 7.1.

$$TP(SD, z_k) = \frac{1}{|SD|} \sum_{\substack{d_i \in SD \\ \theta(d_i, z_k) \geq \gamma}} \theta(d_i, z_k) \quad (7.1)$$

Given a topic z_k and a set of documents SD obtained from the original collection D , the topic popularity $TP(SD, z_k)$ is the average of the topic memberships $\theta(d_i, z_k)$ summation for each document d_i belonging to the set of documents SD . However, note that these $\theta(d_i, z_k)$ are filtered according to a γ threshold, which was defined as 10%. This feature means that topics with less than 10% of $\theta(d_i, z_k)$ are not associated with a document d_i . This filtering is required because LDA models associate all topics to all documents, even if $\theta(d_i, z_k)$ value is very low. In the case of a document d_i presenting no topics above γ , the highest $\theta(d_i, z_k)$ is keep and any other $\theta(d_i, z_k)$ equals to highest is also keep.

As variation form from *share* and *impact* metrics, the Topic Popularity is a more generic metric can compute the popularity of a topic relative to any set of documents. This feature allows analyzing sub-collections according to any kind of condition, such as a sub-collection that includes only posts created by a given user in a given month.

¹ <https://archive.org/details/staty>

Furthermore, as one of the purposes of this work is identifying how users are loyal or drift the topics they are interested in, it is necessary to compute a metric capable to measure the drifting or variance of a topic. Therefore, the statistical standard deviation can be used for this task, because this metric can measure the variance and dispersion of any set of values. Hence, considering P a given set of Topic Popularity measures, \bar{P} their mean value, and p_i the i -th given Topic Popularity, the Topic Popularity Drift TPD is defined by the equation 7.2.

$$TPD(P) = \sqrt{\frac{1}{|P|} \sum_{p_i \in P} (p_i - \bar{P})^2} \quad (7.2)$$

Considering P a given set of previously computed Topic Popularity measures for topic z_k , $TPD(P)$ is the standard deviation of P , which is defined as the squared root of the P variance. This variance computes the average of the squared distance every topic popularity p_i is from the mean \bar{P} . As the closer a computed measure is to zero the less the topic z_k drifts, this metric can be helpful to analyze topic loyalty.

Rather than proposing a specific metric for users and time slices, Topic Popularity Drift is proposed as a generic metric able to be applied to any set of Topic Popularity measures. For example, if P is computed from posts in a particular time slice, $TPD(P)$ is the Topic Popularity Drift of topic z_k only in this time slice. Besides, as Topic Popularity results in values between 0 and 1, the results of $TDP(P)$ also follow this rule.

7.2 GENERAL ANALYSES

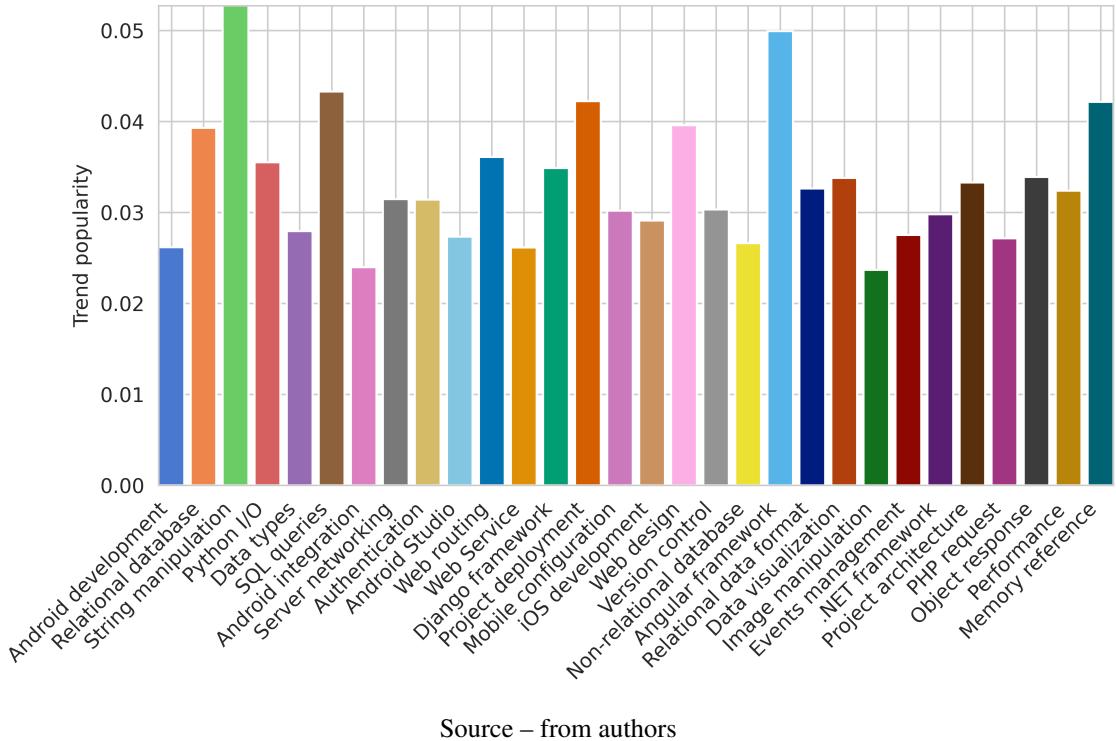
The general analyses does not consider authorship information and, consequently, aim to provide insights related to all Stack Overflow posts. The three employed analyses compute metrics to understand the trend topics, their evolution, and their drift by Topic Popularity measures. These general analyses resulted in **4,421** metric computations, which were stored into CSV files with general suffix in their names.

The first general analysis consists of, for each topic, computing the Topic Popularity across the entire collection, without considering temporal or authorship information. The result of this analysis is the general popularity of each topic across the whole Stack Overflow history, identifying the trend topics, which are stored into the `general-trend.csv` file.

Figure 19 presents a bar chart containing the computed Topic Popularity measures across all posts. Observe that, although these measures do not vary too much, the topics *String manipulation* (0.052), *Angular Framework* (0.049), *SQL queries* (0.043), *Project deployment* (0.042), and *Memory reference* (0.042) are the top-5 global trend topics in Stack Overflow.

After that, the monthly popularity evolution of each topic across all posts was analyzed. This analysis provided useful information on how Stack Overflow discussed topics change over time, such as their tendencies of increasing and decreasing. In this case, the `dates.txt` file was necessary to split the posts into sub-collections according to monthly time slices. Then, for each

Figure 19 – General trend topics in Stack Overflow



Source – from authors

topic, the Topic Popularity was computed across every sub-collections, generating a set of Topic Popularity measures over time slices, which are stored into the `general-popularity.csv` file.

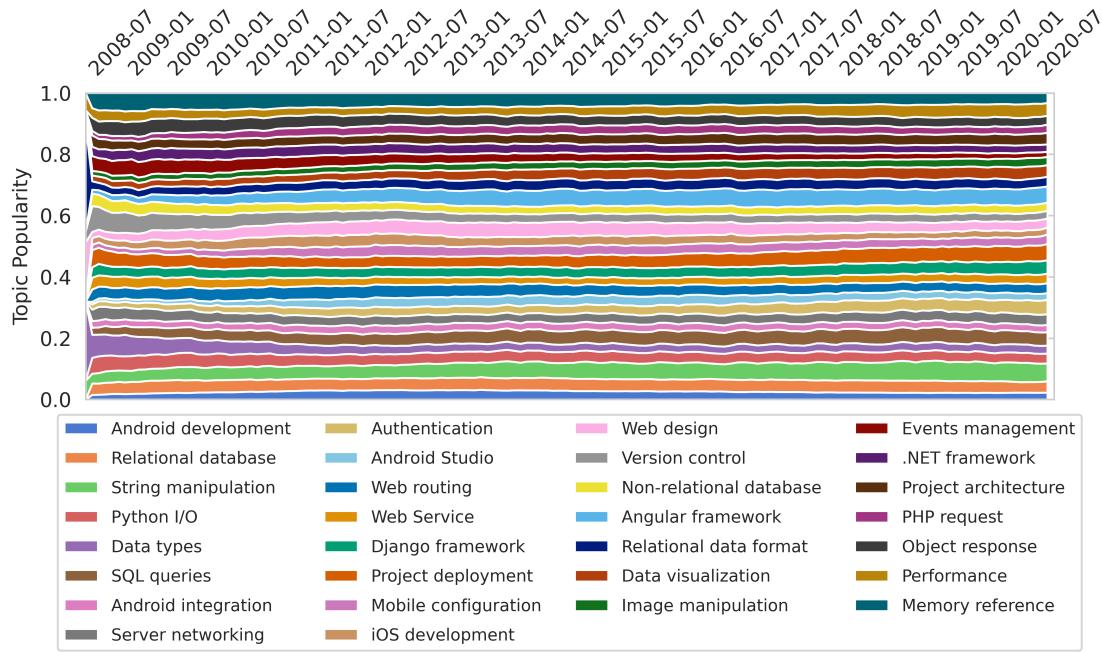
Appendix B presents the popularity evolution each topic had. Observe that each topic presented a distinct evolution line, giving a general idea of their trends of increasing, decreasing, or keep constant. Also, as the number of posts in the first Stack Overflow months was too low, an extreme increasing or decreasing behavior is clearly visible in these first months. For example, the Relational data format topic presented a very sudden popularity decreasing, whereas Events management presented a very sudden popularity decreasing.

To better understand Stack Overflow topic evolution, Figure 20 presents a stacked area chart with the monthly popularity evolution of each topic across all posts, where date labels are grouped in semesters to avoid labels overlapping. This chart works like a pie chart, but with temporal information, which means that the summation of all topics' popularity in a month is 1. This chart very is useful for understanding how the topics' popularity change over time, allowing comparing their evolution trends.

Another interesting data to analyze is how these topics drifted on their own evolution lines. Observe that the Topic Popularity measures presented in Figure 20 do not vary too much, which is a trait also present in Figure 19. However, Appendix B shows that several topics present sudden variation in first months.

Therefore, the last general analysis employs the Topic Popularity Drift metric to measure the variability each topic presents over the monthly time slices, which are stored into the

Figure 20 – General topic popularity evolution by month in Stack Overflow



Source – from authors

general-drift.csv file. Computing this metric requires the set of Topic Popularity measures of each topic, resulting in standard deviation measures, where the lower the measure the more constant the topic is. As this measure is a statistic one, it has a lot of possible uses in different variability studies.

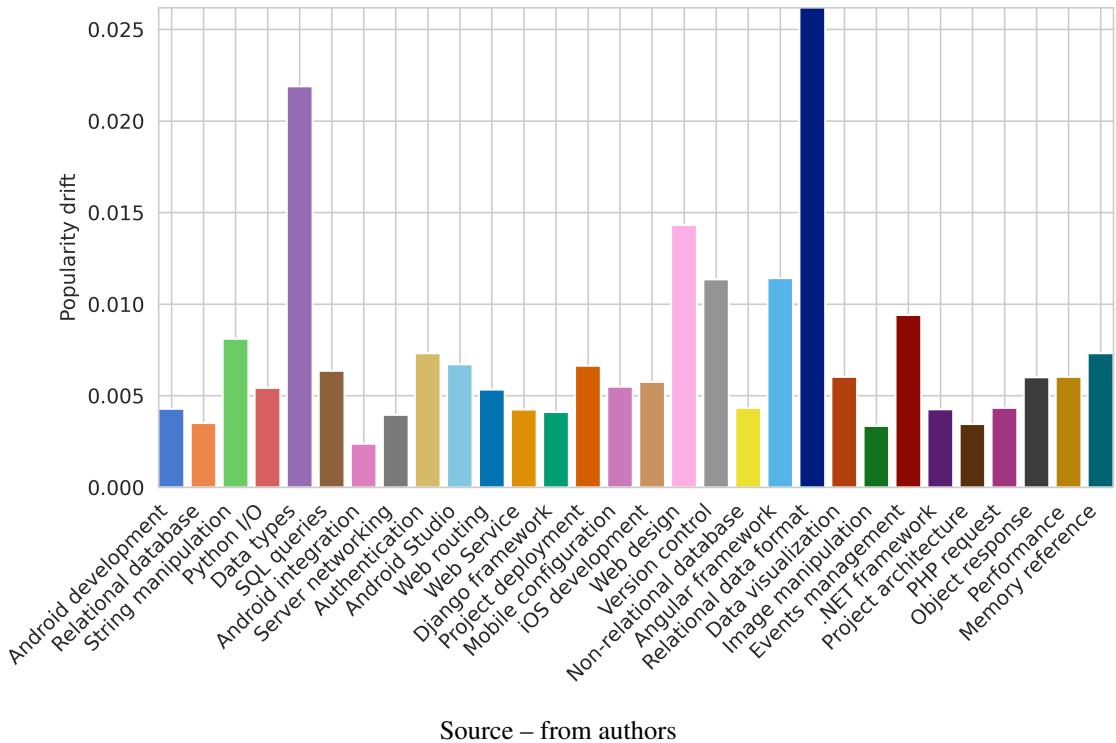
Figure 21 presents a bar chart containing the Topic Popularity Drift measures for each Stack Overflow topic. Observe that the most variable topics are “Relational data format” (0.0261) and “Data types” (0.0218), which, according to Figure 20, start their history in Stack overflow with high popularity, but lessen these popularity over time. On the other hand, “Android integration” (0.0023) and “Image manipulation” (0.0033) are the most constant topics, although they present lower trend popularity.

7.3 USER-CENTRIC ANALYSES

After the general analyses, user-centric analyses were performed. They consist of computing the same metrics employed in the general analyses, but, rather than applying them to all posts, these metrics were separately computed for each user stored in the `user.txt` file. To do that, the collection was split into sub-collections by their authors, each one with their own set of trend topics, topic popularity evolution, and popularity drift, allowing getting several kinds of user-centric insights.

Considering all employed filtering methods, the final number of users in the collection

Figure 21 – General topic popularity drift in Stack Overflow



Source – from authors

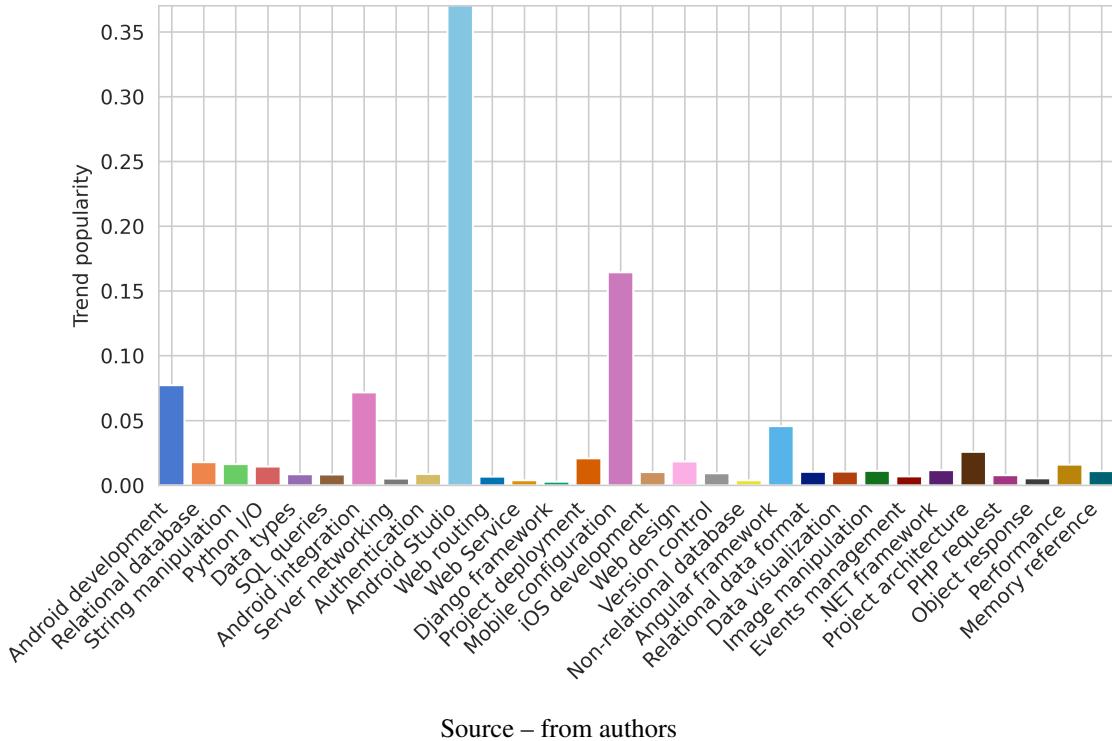
was 4,943,206. As every analysis was singularly made for each user, the number of computed metrics on the user-centric analyses was 135,841,897. All these computed metrics were stored similarly as general analyses were: into the `user-trends.csv`, `user-popularity.csv`, and `user-drift.csv` files.

Figure 22 presents the trend topics for user id 1,289,716, which was randomly chosen among the users with at least one year of contribution in Stack Overflow. Note that the top-5 trend topics of this user are clearly visible: *Android Studio* (0.36), *Mobile configuration* (0.16), *Android development* (0.08), *Android integration* (0.07), and *Angular Framework* (0.04). With this information, it is possible to consider this user as a great contributor in topics related to Android development in different frameworks, such as Android Studio and Angular. However, although the user preferences are clearly defined, this user still interested in other topics, as the figure shows.

Figure 23 presents the topic popularity evolution of the same user. The chart clearly shows the preference of this user to *Android Studio* topic, which was addressed by several years. Besides, the other trend topics also are addressed by many months, even if presenting lower popularity. Considering the presented data of this user, it is possible to suggest that this user has solid knowledge in Android development frameworks, especially Android Studio, and was very committed to contributing to this field on Stack Overflow from 2012 March to 2015 July.

Finally, Figure 24 presents the topic popularity drift. Although this user presented preference to Android-related topics, the chart shows that the trend topics for this user have the

Figure 22 – Trend topics for user 1,289,716



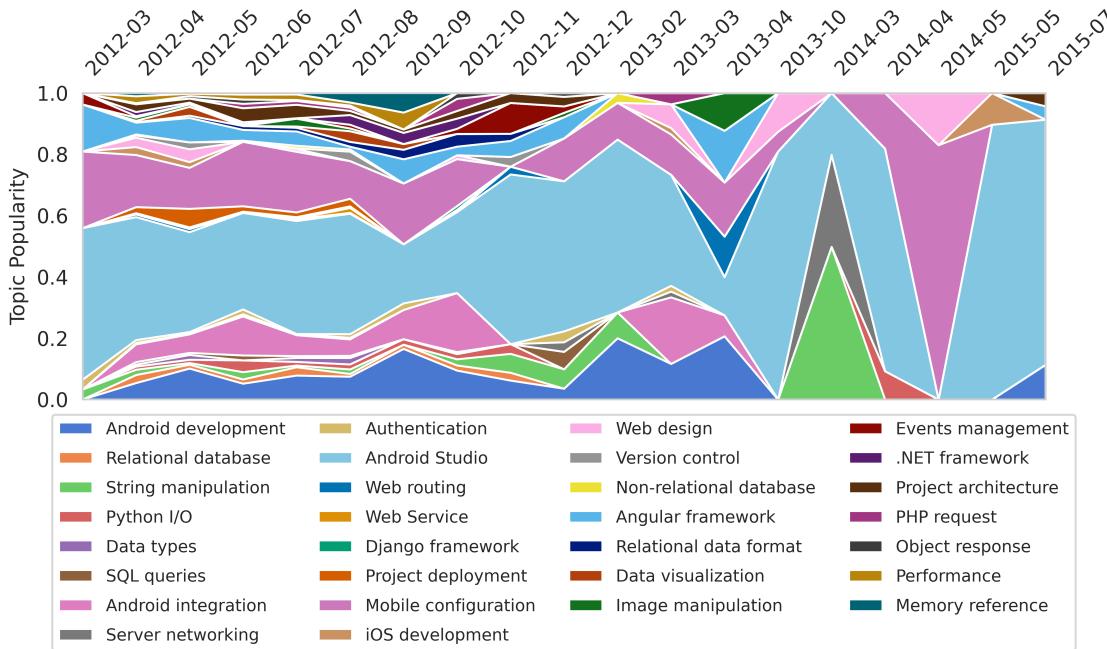
highest drifts among all other topics. Also, the less popular topics for this user presents lower drift as their absolute variations are lower than the trend topics.

7.4 DISCUSSION

This chapter presented the metrics proposed and employed for analyzing Stack Overflow questions and answers across the inferred topics. The Topic Popularity Drift metric was proposed for measuring how a set of popularity measures drifts over time, which allows understanding the loyalty of a user on this topic. However, the user-centric results showed that basing this metric on statistic standard deviation often makes the less popular topics presents lower drift than trend topics. The reason for this behavior is that the monthly absolute variation of topics with high popularity tends to be higher as their topic popularity dominance is greater.

For example, suppose that user U wrote some posts where topic A has 0.6 of popularity, topic B 0.3, and topic C 0.1. Then, topic A increases its popularity to 0.8, topic B decreases to 0.2, and topic C to 0. The absolute drift for topic A is 0.2 and the Topic Popularity Drift 0.1, whereas they are 0.1 and 0.05 for both topics A and B. As topic A is the dominant one and increased its popularity, this user is loyal to topic A. However, as the Topic Popularity Drift is greater for topic A, it suggests this topic is the one that drifts more. Therefore, although the Topic Popularity Drift metric can provide useful information for analyzing topic popularity variation, considering topic dominance in this loyalty analysis could improve the results.

Figure 23 – Topic popularity evolution for user 1,289,716



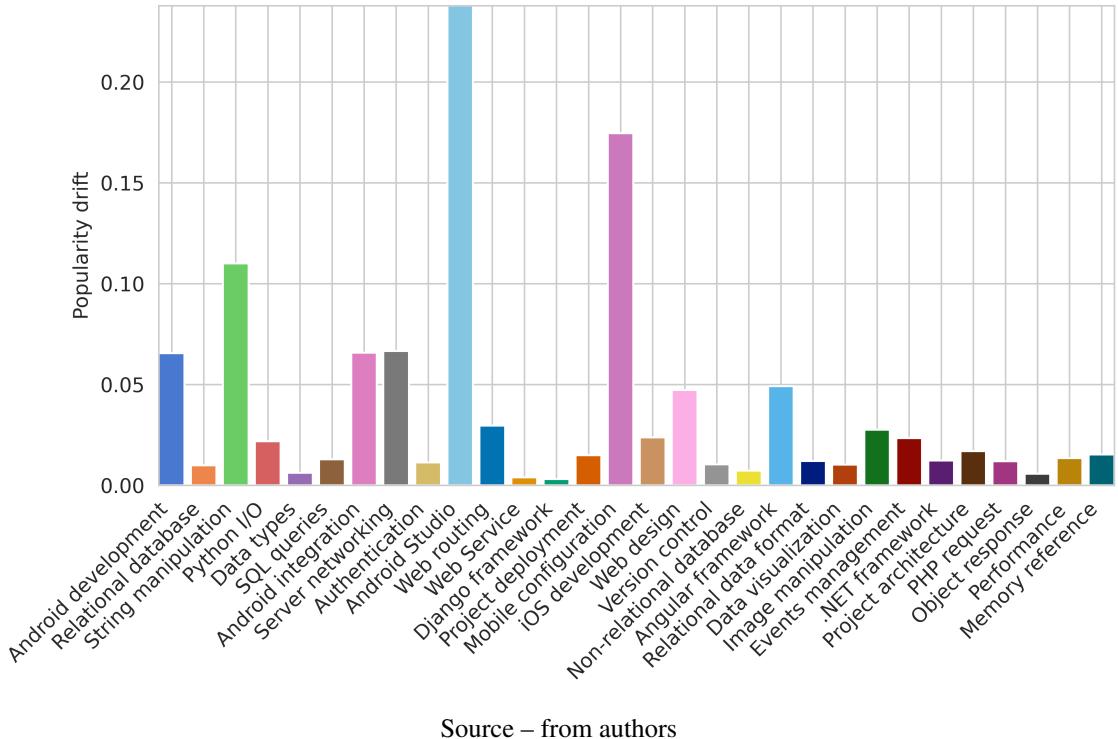
Source – from authors

On the other hand, the Topic Popularity metric was very versatile in the made analyses. This metric has successfully computed the relative topic popularity in several different sub-collections of documents, allowing analyzing the posts written by individual users in many different time slices. This result suggests that the Topic Popularity metric is able to consider any other information in the analyses; just split the collection by this information and Topic Popularity can handle the job.

Besides, understanding the presented charts shows that there is a lot of knowledge hidden in these metric calculations. The general analyses can be explored in many other aspects, such as specifying some time slice intervals to analyze their topic trends or trying to predict topics' popularity in the future. Also, the user-centric metrics allow deep and individual analyses of the interest topics each user has, which could be helpful for many purposes. That is the reason the most important contribution of this work is to make all these results accessible to any person with an internet connection, so much more exploratory analyses can be performed.

Although this work does not explore deeply the computed results, it provides updated data on Stack Overflow topics' current state. As Appendix B presents topic's popularity and their trends, the top-3 topics *String manipulation*, *Angular framework*, and *SQL queries* still in an increasing trend, whereas *Project architecture* is stagnant and *Memory reference* keeps decreasing over time. However, other topics presented interesting increasing trends, such as *Data visualization* and *Image manipulation*, which suggests that Computer Vision and Data Science fields still becoming more popular.

Figure 24 – Topic popularity drift for user 1,289,716



Source – from authors

However, these results are not limited to what this work presented, because they can be much more explored. For example, the Stack Overflow team can use this data to better understand the user-generated content in the last months in order to analyze the current trend. If any topic becomes too dominant over others, Stack Overflow moderation could note this behavior by analyzing the topic's popularity evolution, which could be helpful in some decision makings.

Besides, this work provided exclusive data for each Stack Overflow user, which was never been done before. Each user contains sufficient data for complete and deep analyses on their interest topics, allowing personal and user-centric analyses on topic popularity trends, drift, and evolution. The potential of this data for tracking user interests was shown to be great and the using possibilities are very numerous. For example, the Stack Overflow team can use this data to better understand the user groups according to their interests in order the provide specific tools for them.

8 CONCLUSIONS

This work proposed a methodology to discover and analyze the trend topics in Stack Overflow, a popular Q&A community on computer programming. Experiments were done to define the best distribution over topics for Stack Overflow posts employing LDA. Also, metrics were proposed to measure the popularity and the drift of each topic across post and authorship-temporal information, generating several data that was made available in the Internet Archive¹ website for anyone who wants to validate and explore the results.

As the number of topics is an important factor in topic modeling studies, this work contributed to finding the best number of topics that summarize Stack Overflow questions and answers. As C_v coherence was employed to evaluate the experimented topic models, the resultant topics are relative close to human interpretation of topics. Also the assigned labels address several computer programming topics and are more specific than broad, suggesting that the discovered topics successfully summarize Stack Overflow shared knowledge.

Since the topics were discovered, they become essential to compute the metrics that make this work significant. Every computed metric was important to build the results, providing a rich data to explore. Besides, as this data was made web-accessible for free, this work also contributed to the knowledge democratizing. Hence, any researcher can employ this data to provide Stack Overflow insights for improving the available features and tool.

However, the uses for this data are also beyond the Stack Overflow website. For example, an interactive dashboard with this data can help job recruiters finding users who are interested in specific topics and check which of them has the required expertise for a specific demand. Therefore, another possible future work is building a website where these results can be interactively analyzed by anyone.

The provided contributions can be significantly explored in future works and there are a lot of potential applications for the generated results. This work can be the basis for more deep studies on Stack Overflow topics as the results allow both general and user-centric analyses. Also, including other Stack Overflow metadata in the posts may provide more data to analyze, such as analyzing which topics each post tag is related to.

Besides, validating the proposed metrics and proposing other metrics can help to improve the exploratory analyses in the topic modeling field. Although the Topic Popularity Drift successfully measures the variation and dispersion of a set of topics' popularity, it is not consistent for topic loyalty analysis. However, many metrics can be employed for this task, such as R^2 metric.

Finally, although this work analyzed Stack Overflow questions and answers, the addressed methodology can be employed in other text resources, such as web portals, blogs, forums, newsletters, scientific repositories, and much more. Comparing the use of this methodology in other resources can validate it and provide insightful results on other contexts.

¹ <https://archive.org/details/staty>

REFERENCES

- ALETRAS, Nikolaos; STEVENSON, Mark. Evaluating topic coherence using distributional semantics. In: PROCEEDINGS of the 10th International Conference on Computational Semantics (IWCS 2013)–Long Papers. [S.l.: s.n.], 2013. p. 13–22.
- ALLAHYARI, Mehdi; KOCHUT, Krys. Automatic topic labeling using ontology-based topic models. In: IEEE. 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA). [S.l.: s.n.], 2015. p. 259–264.
- BARUA, Anton; THOMAS, Stephen W; HASSAN, Ahmed E. What are developers talking about? an analysis of topics and trends in stack overflow. **Empirical Software Engineering**, Springer, v. 19, n. 3, p. 619–654, 2014.
- BEEL, Joeran et al. paper recommender systems: a literature survey. **International Journal on Digital Libraries**, Springer, v. 17, n. 4, p. 305–338, 2016.
- BIRD, Steven; KLEIN, Ewan; LOPER, Edward. **Natural language processing with Python: analyzing text with the natural language toolkit**. [S.l.]: "O'Reilly Media, Inc.", 2009.
- BISHOP, Christopher M. **Pattern recognition and machine learning**. [S.l.]: springer, 2006.
- BLEI, David M. Probabilistic topic models. **Communications of the ACM**, ACM New York, NY, USA, v. 55, n. 4, p. 77–84, 2012.
- BLEI, David M; LAFFERTY, John D. Dynamic topic models. In: PROCEEDINGS of the 23rd international conference on Machine learning. [S.l.: s.n.], 2006. p. 113–120.
- BLEI, David M; NG, Andrew Y; JORDAN, Michael I. Latent dirichlet allocation. **Journal of machine Learning research**, v. 3, Jan, p. 993–1022, 2003.
- CAVUSOGLU, Huseyin; LI, Zhuolun; HUANG, Ke-Wei. Can gamification motivate voluntary contributions? The case of StackOverflow Q&A community. In: PROCEEDINGS of the 18th ACM conference companion on computer supported cooperative work & social computing. [S.l.: s.n.], 2015. p. 171–174.
- CHANG, Jonathan et al. Reading tea leaves: How humans interpret topic models. In: PROCEEDINGS of the Twenty-third Advances in neural information processing systems. [S.l.: s.n.], 2009. p. 288–296.
- DAUD, Ali. Using time topic modeling for semantics-based dynamic research interest finding. **Knowledge-Based Systems**, Elsevier, v. 26, p. 154–163, 2012.
- FITELSON, Branden. A probabilistic theory of coherence. **Analysis**, JSTOR, v. 63, n. 3, p. 194–199, 2003.
- HAN, Pu et al. The influence of word normalization in English document clustering. In: IEEE. 2012 ieee international conference on computer science and automation engineering (csae). [S.l.: s.n.], 2012. v. 2, p. 116–120.

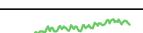
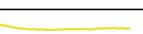
- JIVANI, Anjali Ganesh et al. A comparative study of stemming algorithms. **Int. J. Comp. Tech. Appl.**, v. 2, n. 6, p. 1930–1938, 2011.
- LANDAUER, Thomas K; FOLTZ, Peter W; LAHAM, Darrell. An introduction to latent semantic analysis. **Discourse processes**, Taylor & Francis, v. 25, n. 2-3, p. 259–284, 1998.
- LAU, Jey Han; NEWMAN, David; BALDWIN, Timothy. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In: PROCEEDINGS of the 14th Conference of the European Chapter of the Association for Computational Linguistics. [S.l.: s.n.], 2014. p. 530–539.
- MANNING, Christopher D; SCHÜTZE, Hinrich. **Foundations of statistical natural language processing**. [S.l.]: MIT press, 1999.
- MCCALLUM, Andrew Kachites. Mallet: A machine learning for language toolkit. <http://mallet.cs.umass.edu>, 2002.
- MIKOLOV, Tomas et al. Distributed representations of words and phrases and their compositionality. In: ADVANCES in neural information processing systems. [S.l.: s.n.], 2013. p. 3111–3119.
- MINKA, Thomas. **Estimating a Dirichlet distribution**. [S.l.]: Technical report, MIT, 2000.
- NEWMAN, David et al. Automatic evaluation of topic coherence. In: HUMAN language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics. [S.l.: s.n.], 2010. p. 100–108.
- PEDREGOSA, Fabian et al. Scikit-learn: Machine learning in Python. **the Journal of machine Learning research**, JMLR.org, v. 12, p. 2825–2830, 2011.
- RAMOS, Juan et al. Using tf-idf to determine word relevance in document queries. In: NEW JERSEY, USA. PROCEEDINGS of the first instructional conference on machine learning. [S.l.: s.n.], 2003. v. 242, p. 133–142.
- ŘEHŮŘEK, Radim; SOJKA, Petr. Software Framework for Topic Modelling with Large Corpora. English. In: PROCEEDINGS OF THE LREC 2010 WORKSHOP ON NEW CHALLENGES FOR NLP FRAMEWORKS. Valletta, Malta: ELRA, May 2010. p. 45–50. <http://is.muni.cz/publication/884893/en>.
- RÖDER, Michael; BOTH, Andreas; HINNEBURG, Alexander. Exploring the Space of Topic Coherence Measures. In: PROCEEDINGS of the Eighth ACM International Conference on Web Search and Data Mining. Shanghai, China: Association for Computing Machinery, 2015. p. 399–408. ISBN 9781450333177. DOI: [10.1145/2684822.2685324](https://doi.org/10.1145/2684822.2685324).
- ROSEN-ZVI, Michal et al. The author-topic model for authors and documents. **arXiv preprint arXiv:1207.4169**, 2004.
- STEYVERS, Mark; GRIFFITHS, Tom. Probabilistic topic models. **Handbook of latent semantic analysis**, v. 427, n. 7, p. 424–440, 2007.

- SYED, Shaheen; SPRUIT, Marco. Full-text or abstract? Examining topic coherence scores using latent dirichlet allocation. In: IEEE. 2017 IEEE International conference on data science and advanced analytics (DSAA). [S.l.: s.n.], 2017. p. 165–174.
- TAN, Chade-Meng; WANG, Yuan-Fang; LEE, Chan-Do. The use of bigrams to enhance text categorization. **Information processing & management**, Elsevier, v. 38, n. 4, p. 529–546, 2002.
- TANG, Jie; ZHANG, Jing. Modeling the evolution of associated data. **Data & Knowledge Engineering**, Elsevier, v. 69, n. 9, p. 965–978, 2010.
- WALLACH, Hanna M et al. Evaluation methods for topic models. In: PROCEEDINGS of the 26th annual international conference on machine learning. [S.l.: s.n.], 2009. p. 1105–1112.
- WANG, Shaowei; LO, David; JIANG, Lingxiao. An empirical study on developer interactions in stackoverflow. In: PROCEEDINGS of the 28th Annual ACM Symposium on Applied Computing. [S.l.: s.n.], 2013. p. 1019–1024.
- WANG, Xuerui; MCCALLUM, Andrew. Topics over time: a non-Markov continuous-time model of topical trends. In: PROCEEDINGS of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. [S.l.: s.n.], 2006. p. 424–433.
- XU, Shuo; SHI, Qingwei; QIAO, Xiaodong; ZHU, Lijun; JUNG, Hanmin, et al. Author-Topic over Time (AToT): a dynamic users' interest model. In: MOBILE, ubiquitous, and intelligent computing. [S.l.]: Springer, 2014. p. 239–245.
- XU, Shuo; SHI, Qingwei; QIAO, Xiaodong; ZHU, Lijun; ZHANG, Han, et al. A dynamic users' interest discovery model with distributed inference algorithm. **International Journal of Distributed Sensor Networks**, SAGE Publications Sage UK: London, England, v. 10, n. 4, p. 280892, 2014.
- YANG, Min et al. Discovering author interest evolution in order-sensitive and Semantic-aware topic modeling. **Information Sciences**, Elsevier, v. 486, p. 271–286, 2019.

APPENDIX A – STACK OVERFLOW TOPICS AND TOP WORDS

Id	Label	Top LDA words
0	Android development	<i>button click android view page jquery select javascript item image</i>
1	Relational database	<i>table sql query column database row mysql server select id</i>
2	String manipulation	<i>string python array list number character match regex output line</i>
3	Python I/O	<i>line string command python script character run output program write</i>
4	Data types	<i>time type number case model example object performance question point</i>
5	SQL queries	<i>array table sql query column string list return row php</i>
6	Android integration	<i>android message app send server application time run thread process</i>
7	Server networking	<i>server run test thread service client application spring connection message</i>
8	Authentication	<i>user api request google http token post facebook access url</i>
9	Android Studio	<i>android xml view layout list activity item text fragment json</i>
10	Web routing	<i>javascript html jquery page http browser load chrome script event</i>
11	Web Service	<i>server web http net service user application request com page</i>
12	Django framework	<i>python test run script django ruby command ruby_rail module line</i>
13	Project deployment	<i>project run build server version application http window web service</i>
14	Mobile configuration	<i>android app google application device com ios window http user</i>
15	iOS development	<i>ios image swift xcode iphone objective app google map window</i>
16	Web design	<i>image css html text element set jquery javascript color div</i>
17	Version control	<i>git test object type branch version commit new case repository</i>
18	Non-relational database	<i>object key type database mongodb return exception case model instance</i>
19	Angular framework	<i>jquery html javascript css button angular component click page element</i>
20	Relational data format	<i>date php time sql query mysql database table server format</i>
21	Data visualization	<i>array number point loop line element excel plot example python</i>
22	Image manipulation	<i>image line text element python set size example html css</i>
23	Events management	<i>event net view control property object controller thread javascript set</i>
24	.NET framework	<i>net user form asp page mvc controller view model session</i>
25	Project architecture	<i>project php folder run http directory version com path server</i>
26	PHP request	<i>php user io post product page api json http form</i>
27	Object response	<i>net type object json asp property parameter xml template return</i>
28	Performance	<i>time number python memory column result loop row list size</i>
29	Memory reference	<i>object variable type pointer return array reference memory program case</i>

APPENDIX B – STACK OVERFLOW TOPICS AND TREND TOPIC POPULARITY MEASURES

Id	Label	Topic Popularity	Trend	Evolution line
0	Android development	0.0261 / 2.61%	–	
1	Relational database	0.0392 / 3.92%	–	
2	String manipulation	0.0527 / 5.27%	↑↑	
3	Python I/O	0.0354 / 3.54%	–	
4	Data types	0.0279 / 2.79%	↓↓	
5	SQL queries	0.0432 / 4.32%	↑↑	
6	Android integration	0.0239 / 2.39%	–	
7	Server networking	0.0314 / 3.14%	–	
8	Authentication	0.0313 / 3.13%	↑↑	
9	Android Studio	0.0273 / 2.73%	↑↑	
10	Web routing	0.0360 / 3.60%	↓↓	
11	Web Service	0.0261 / 2.61%	–	
12	Django framework	0.0348 / 3.48%	–	
13	Project deployment	0.0422 / 4.22%	–	
14	Mobile configuration	0.0301 / 3.01%	↑↑	
15	iOS development	0.0290 / 2.90%	↑↑	
16	Web design	0.0395 / 3.95%	–	
17	Version control	0.0303 / 3.03%	↓↓	
18	Non-relational database	0.0266 / 2.66%	–	
19	Angular framework	0.0498 / 4.98%	↑↑	
20	Relational data format	0.0326 / 3.26%	–	
21	Data visualization	0.0337 / 3.37%	↑↑	
22	Image manipulation	0.0236 / 2.36%	↑↑	
23	Events management	0.0275 / 2.75%	↓↓	
24	.NET framework	0.0297 / 2.97%	↓↓	
25	Project architecture	0.0332 / 3.32%	–	
26	PHP request	0.0271 / 2.71%	↑↑	
27	Object response	0.0338 / 3.38%	↓↓	
28	Performance	0.0323 / 3.23%	↑↑	
29	Memory reference	0.0421 / 4.21%	↓↓	