DOI: 10.6977/IJoSI.202109_6(5).0006

A Text Analytics Approach to Study Python Questions Posted on Stack Overflow

Lee Yong Meng, Soo Yin Yi, Gan Keng Hoon* and Nur-Hana Samsudin
School of Computer Sciences, Universiti Sains Malaysia, Pulau Pinang, MALAYSIA
* Corresponding author E-mail: khgan@usm.my
(Received July 13, 2021; Final version received 1 September 2021; Accepted 9 September 2021.)

Abstract

Stack Overflow (SO) is one of the largest discussion platforms for programmers to communicate their ideas and thoughts related to various topics like software development and data analysis. Many programmers are actively contributing to this platform and discuss about Python programming language. To better study the topics related to Python questions posted on the platform, a text analytics approach incorporating text preprocessing steps and Latent Dirichlet Allocation (LDA) topic modelling algorithm is proposed. The two main objectives of this study are: to discover and compare the topics of the questions about Python programming language posted on SO from 2008 to 2016, and to analyze questions about Python programming language with high votes posted on SO from 2008 to 2016 using topic modelling technique with a suitable number of topics. From the study, we find that the topics of the Python questions posted on Stack Overflow have gradually shifted towards those related to data modelling and analysis from 2008 to 2016. Furthermore, the study also shows that a suitable number of topics using the topic modelling technique yield a high coherence score concerning the topic model in use, which is important to extract more meaningful topics from the collection of Python questions.

Keywords: Stack Overflow, text processing, text analytics, topic modelling.

1. Introduction

Stack Overflow (SO) is one of the largest open-source software platforms for programmers to ask and discuss programming questions. This platform includes voting, badging and user reputation systems to ensure that the questions and answers posted on the platform are meaningful or relevant to its users. Therefore, SO ecosystem encourages many programmers to not only help each other solve their programming questions voluntarily, but also to showcase their ability in programming problem solving and seeking a better job (Xu et al., 2020). Nevertheless, with its rise in popularity, issues such as duplication of questions (Wang et al., 2020) and the quality of the answers in response to the questions on the platform (Meldrum et al., 2020) greatly affect the browsing experience by programmers when searching for answers through this platform.

Many programming questions have been posted on SO since its official launch in 2008. These include questions related to different programming languages such as C language, Python, Java, and R, to name a few. Specifically, Python and R are the two programming languages most highly associated with the questions related to data analysis posted on the platform. This is reasonable because there are many existing libraries and packages useful for data analysis in both Python and R. This kind of information, which can be extracted using text analytics approaches, can serve for various usages. For example, it can be used by the programming language development team to identify the aspects of the language that are most relevant to these topics so that they can work on improving the language in terms of syntax, features, and even documentations. Besides, it can also be used as a guideline for the programming language course team to identify the important topics to be covered in the content of their courses to meet the requirements of the learners.

Several recent studies have been conducted to analyze the questions and answers (Q&A) about computer programming and software development posted on SO. These include several works performed to extract and classify the topics of discussion on SO related to mobile application development for different platforms, such as Android, iOS, and Window Phones (Ahmad et al., 2019; Beyer et al., 2020; Fontão, et al., 2018). In these studies, the analyses are performed without splitting those discussions according to years to discover the trend or to compare the change of the topics discussed over several years. Furthermore, it is also important to identify the important topics from the questions of a selected programming language on the platform that are most relevant to the users. In this context, a relevant question means any question posted on SO that receives many votes from users who find it helpful for them through the voting system implemented on this platform.

In this study, a text analytics approach involving text preprocessing steps and topic modelling algorithm will be used to analyze the questions related to Python programming language posted on SO. The two main objectives of this study are listed below:

✓ To discover and analyze the topics of the questions about Python programming language posted on SO from 2008 to 2016 to identify and compare the topics being discussed in each year.

✓ To analyze questions about Python programming language with high votes posted on SO from 2008 to 2016 using topic modelling technique with a suitable number of topics.

This paper is structured as follows: Section 2 discusses the related works, including the application of topic modelling algorithms on SO and other forums. Section 3 describes the proposed solution to address the two main objectives of this study, including the dataset description and preprocessing steps, the text preprocessing steps, and the topic modelling approach. Then, Section 4 presents the analysis of findings from the proposed solution. Finally, Section 5 discusses the work done and some limitations of this study and Section 6 summarizes the key points discussed throughout this paper.

2. Related Works

As a large open-source software platform, the Q&A posted on SO contains a fruitful source of information that can be studied and analyzed to understand the topics being discussed by programmers from time to time. Text preprocessing techniques and topic modelling algorithm such as Latent Dirichlet Allocation (LDA) is used in several recent works to study the textual data extracted from the Q&A available on SO for various purposes. For example, LDA algorithm is utilized by researchers to extract the topics of the questions related to various programming languages for further analysis. The study by Ali and Linstead (2020) focuses on several programming languages such as Python, JavaScript, C++, and R to word cloud discover the topics related to these programming languages that have been exhausted for 10 years. Topic exhaustion is a term describing the occurrence where the number of questions related to a topic posted on SO decreases and it takes a longer waiting time to obtain an answer for those questions over the years. The study by Chakraborty et al. (2021) is conducted to identify difficult topics for questions related to new programming languages such as Go, Swift and Rust posted on SO. It is found that topics related to “data” and “data structure” are difficult topics regardless of programming languages. Another study is conducted using the LDA algorithm by Marçal et al. (2020) to identify skill gaps between college and workspace by analyzing topics of the questions related to Computer Science posted on SO.

Topic modelling techniques are also used by researchers to analyze the topics of discussions on different Q&A websites. Stack Exchange, being a network of multiple online Q&A websites in a vast variety of fields (including SO – the main Q&A website under Stack Exchange dedicated for programmers), is often the choice of many researchers to analyze the trends of popular topics being discussed among communities in different fields. For example, the threads from Data Science Stack Exchange (and Reddit) are analyzed using the LDA algorithm by Karbasian and Johri (2020) to identify not only the important Data Science topics and useful examples relevant for teaching Data Science courses, but also various topics related to professional developments. On the other hand, the LDA algorithm is used by Tamla et al. (2019) to identify the thoughts and needs of serious games (SG) developers from their discussions on GameDev Stack Exchange.

On top of that, the LDA algorithm is also commonly used for topic extraction of different fields in several other online forums and social media networks. A combination of Twitter and Reddit datasets are used by Curiskis et al. (2020) to extract the topics being discussed by users in online social networks (OSNs) using the LDA topic modelling algorithm. Besides, the LDA topic modelling algorithm is used to analyze the content specific to eating disorder on Reddit (Moessner et al., 2018) and to extract patient knowledge through their narratives on a patient forum (Dirkson, et al., 2019), respectively. Another application of the LDA topic modelling algorithm is presented by Jaworska and Nanda (2018) to examine the change of topics over time in the large corpus of corporate social responsibility (CSR) reports from the oil sector. The analysis shows that the popular topics of the CSR reports have shifted from the topics related to "climate change" to those related to "human rights".

The works done by researchers using the topic modelling approaches have opened the door for countless possibilities for future studies to analyze texts from different Q&A websites and forums. In this study, the LDA topic modelling algorithm is used to analyze Python questions posted on SO in two ways: first, to identify and compare the topics discussed by programmers on the platform for different years; and second, to extract the topics of the Python questions with high votes on the platform.

3. Proposed Solution

To achieve the two objectives in this study, we propose a text analytics solution that utilizes text processing techniques and the LDA topic modelling algorithm to study and identify the topics of the Python questions posted on SO from 2008 to 2016.

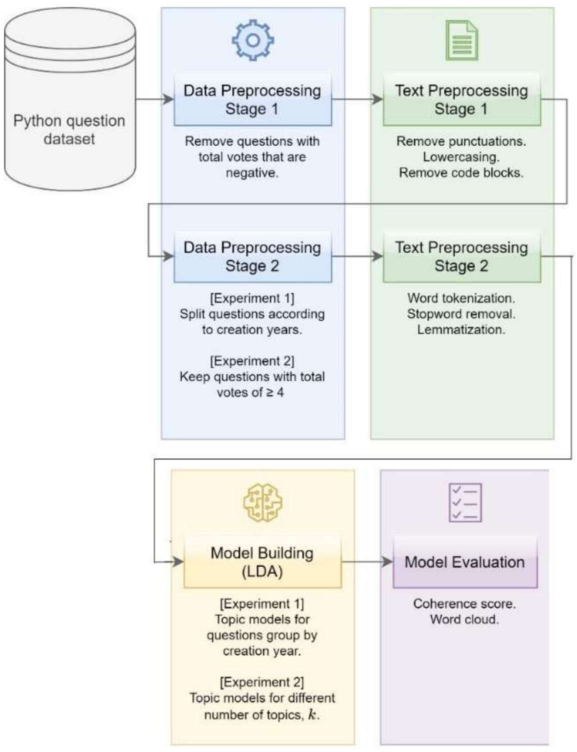

Python question dataset
Data Preprocessing Stage 1
Remove questions with total votes that are negative.
Text Preprocessing Stage 1
Remove punctuations. Lowercasing. Remove code blocks.
Data Preprocessing Stage 2
[Experiment 1] Split questions according to creation years.
[Experiment 2] Keep questions with total votes of ≥ 4
Text Preprocessing Stage 2
Word tokenization. Stopword removal. Lemmatization.
Model Building (LDA)
[Experiment 1] Topic models for questions group by creation year.
[Experiment 2] Topic models for different number of topics, k.
Model Evaluation
Coherence score. Word cloud.

Fig. 1. Overall framework of the proposed solution.

There are two different experiment sets designed to analyze these Python questions. In the first experiment set, the selected Python questions are split into different groups according to the creation year of these questions on the platform. Then, the topic models are built for the respective group of questions to extract the topics of Python questions created in each year. Finally, a comparison is made between topics extracted from different groups of Python questions to investigate any changes in topics of Python questions posted on SO from 2008 to 2016. In the second experiment set, the set of Python questions receiving at least 4 votes from SO users from 2008 to 2016 are studied to identify the relevant topics of Python questions among SO users. In this experiment set, different numbers of topics, k, are tested to identify the optimal k to build a topic model, which covers different topics of the Python questions receiving high votes on the platform.

This study is conducted using the Python programming language. Fig. 1 shows the overall framework of the proposed solution. The description and preprocessing steps of the dataset used, the text preprocessing steps, the topic modelling techniques used for model building, and the model evaluation used in the proposed solution are more thoroughly discussed next in the following subsections.

3.1 Dataset Description and Preprocessing

The dataset used in this study is the Python questions dataset (Overflow, 2019) which can be retrieved from the Kaggle website. This dataset consists of three CSV files, namely: "Answers.csv", "Questions.csv" and "Tags.csv". For this purpose, our work focuses on "Questions.csv", the CSV file that stores 607,282 Python questions posted on SO from August 2008 to October 2016. Table 1. Data description of the Python questions dataset.

summarizes the details and descriptions of each attribute in the Python questions dataset.

Table 1. Data description of the Python questions dataset.

Attribute name	Attribute type	Attribute description
Id	Categorical	The unique identifier of Python questions posted on SO.
OwnerUserId	Categorical	The unique identifier of SO users who post the Python questions.
CreationDate	Date and time	The recorded date and time at which the Python questions are posted SO.
Score	Numerical	Total votes received by the Python questions on SO.
Title	Text	A user-generated title for the Python questions on SO.
Body	Text	Description of the Python questions containing normal text and code blocks, written in HTML script.

Since SO is an English-only Q&A website dedicated to the global community, the texts used in the Python questions dataset are very unlikely to contain non-English words that are not understood by SO users. In addition, SO users can choose to downvote any questions which are not posted in the English language. Therefore, no special care is taken to filter out Python questions not written in the English language from the dataset before the analysis.

In this study, the main source of textual data used for the analysis comes from the "Body" attribute. This is because the "Body" attribute contains more detailed descriptions than the "Title" attribute about the Python questions posted by SO users. There are some questions with poorly defined titles that do not describe the questions well, such as "Introducing Python" and "Most possible pairs". Furthermore, the descriptions of the questions provided are usually complete English sentences which can better represent the frequency of each word being used in the written English language. The data preprocessing steps, which are crucial to ensure the quality and validity of the data used to build the topic models, are conducted at two different stages. The first stage is conducted once on the original Python questions dataset to reduce the amount of data and select relevant features to be used for both experiment sets. On the other hand, the data preprocessing steps during the second stage are conducted specifically for each experiment sets by using different attributes.

a) Data Preprocessing: Stage 1

During this stage, the "Id" and "OwnerUserId" attributes are removed from the dataset because these attributes do not help in our study to analyze Python questions posted on SO. Next, Python questions receiving negative scores (i.e., more downvotes than upvotes) by SO users are also removed from the dataset. Then, to prepare the data for the first experiment set, a new attribute "CreationYear" is derived from the "CreationDate" attribute from the dataset. This attribute stores only the year when the Python questions are posted on SO to enable the grouping of each question into its respective year group. Finally, the selected attributes of the Python questions dataset for further analysis are "Score", "Body" and "CreationYear".

b) Data Preprocessing: Stage 2

For the first experiment set, the "CreationYear" attribute is used to split the Python questions into different groups according to the creation year of each question. On the other hand, for the second experiment set, Python questions with "Score" below 4 are filtered out. After performing this step, the number of Python questions used for further analysis has reduced to only 74,195 questions, which is sufficient for the analysis in this study.

3.2 Text Preprocessing

a) Text Preprocessing: Stage 1

During this stage, several text preprocessing steps are used to prepare the textual data for both experiment sets, including the removal of punctuations, lowercasing the texts, and the removal of code blocks. The first two steps: removing punctuations and lowercasing the texts are crucial to removing the unnecessary parts from the text which can introduce noise to the analysis of the Python questions.

Furthermore, special treatment is required to preprocess the descriptions of the Python questions stored in the "Body" attribute due to several reasons. First, the descriptions of the Python questions are written in HTML script, for instance, each paragraph is enclosed within "<p>" and "</p>" tag pairs. Second, including source code of Python or any programming language in the descriptions of the Python questions (enclosed within "<code>" and "</code>" HTML tag pairs) will certainly affect the result of the analysis in this study. Therefore, steps are taken to first remove the source code from the descriptions. Then, the descriptions of the Python questions are transformed from HTML script to normal English text without HTML tags. This whole process is performed using the BeautifulSoup library from the bs4 module in Python.

b) Text Preprocessing: Stage 2

The text preprocessing steps performed during this stage are a series of steps within a text normalization pipeline. In a text normalization pipeline, all the texts are converted from human-readable texts, including slang words and informal texts, into their corresponding machine-readable forms (Rahate and Chandak, 2019). The steps included in this pipeline are word tokenization, stop words removal and lemmatization. In natural language processing (NLP), tokenization is defined as the task to split a stream of characters into words and punctuations. More specifically, there are two types of tokenization methods used in NLP, which are word tokenization and sentences tokenization. Word tokenization is used to separate words via unique space character whereas sentences tokenization is used to perform tokenization based on sentence boundaries and one of the examples is punctuations. In this study, word tokenization is the method performed to split the words in the description of each Python question into individual units of words or tokens. In Python, word tokenization is performed using "simple_preprocess" utility functions implemented in the Gensim library.

Another text preprocessing step is stop word removal. In NLP, stops words or noise words are the words that contain little information that is not required in the analysis process (Kaur and Buttar, 2018). Therefore, stop words are often removed from the text corpus to improve the efficiency with little influence on the results of NLP tasks. Stop words are usually the most common words in a language. Some stop words in the English language are "I", "am", "is", "are", "this" and "that". In this study, the stop words removed from the list of tokenized words are English stop words listed in the NLTK library in Python.

Finally, lemmatization is the technique used to complete the text normalization pipeline. Lemmatization is performed to convert the tokenized words into their base form or dictionary form. For example, the words "use", "used", "uses" and "using" are all conjugated verbs that are derived from and will be converted to their base form "use" after lemmatization. One benefit of performing lemmatization is that it helps reduce the impact of inflection on English words, such as treating derived words in a text corpus as different words, to the results generated by the NLP models. The lemmatization in Python is performed with the core English language model using the SpaCy library, a popular NLP library along with NLTK.

3.3 Model Building

Topic modelling is one of the applications in text analytics used for studying and identifying the underlying key topics of texts and documents, which are often referred to as the combination of different topics (Curiskis et al., 2020). In simple terms, the goal of a topic modelling task is to extract different "topics" hidden within the given texts and documents through a topic model. A topic model is a generative model driven by the probability framework to help identify such topics. In general, a topic is associated with different words and phrases from the texts and documents that tend to occur together. In other words, similar words or phrases tend to be grouped within the same topic.

One of the popular algorithms used in topic modelling is the Latent Dirichlet Allocation (LDA) model. LDA works by assuming that there is a mixture of different topics within the texts and documents (Alghamdi and Alfalqi, 2015). There is a probability distributed over each topic, measuring the likelihood that a word appears in each topic. LDA algorithm then assigns these words to the topic based on the probability that these words appear in the corresponding topics. In the end, the list of the most probable words in each topic indicates the context of the topics. In this study, the LDA model is built using the Gensim library in Python.

3.4 Model Evaluation

Due to its unsupervised nature, a topic modelling task is often used for exploratory analysis. The dataset is not split into training and test dataset during the topic model building process.

Therefore, the challenge in evaluating the model performance of a topic modelling task is similar to those unsupervised learning tasks, in which, there is no ground truth label used for evaluating the performance of a topic model. This is different from another text analytics task called topic classification, which is supervised learning tasks.

In this study, two approaches are used to evaluate the topics generated for each topic model, namely: the inspection using word cloud and coherence score. Evaluation by inspecting the word cloud of each topic is treated as an informal approach, whereas the coherence score, which relates to the computation of the similarity of words within each topic, is a formal approach to evaluate the performance of a topic model.

a) Word Cloud

Word cloud is a visual representation that captures and displays a list of words from a document, which means a "bag of words", and their corresponding frequencies. Visually speaking, the more frequent a word appears in the document, the bigger the size of the word in the word cloud. In this context, we treat each topic as a document, in which, the frequency of each word in the word cloud is simply the probability that the word appears in that topic. With this convention, we can visually study the list of most common words that appear in topics generated by the topic models. In Python, word clouds can be generated using the "wordcloud" library.

The advantages of using word clouds to visually represent the topics generated are, it is intuitive and engaging. Humans are visual creatures, in such, there is a region in the human brain specialized for processing visual elements. Therefore, the information delivered through a word cloud can be easily perceived by humans in general. However, the application of word cloud to evaluate the performance of a topic model could be subjective. This means different people might perceive the word cloud differently due to the differences in expertise and cognitive ability among different people. Therefore, word cloud is used as an informal approach to evaluate the performance of a topic model.

b) Coherence Score

Another approach that can be used to evaluate the performance of a topic model is by measuring the coherence score of a topic. Coherence score, which is also referred to as the topic coherence measures, is obtained by computing the similarities of most probable words in each topic semantically (Röder et al., 2015). A high coherence score implies that there is a high degree of similarities among words within the same topic, and thus the topic is said to be more coherent. Therefore, the words are more associated with each other, which implies that the topic is relevant and not merely because the same words appear to be the high scoring words across different topics.

The coherence score for one topic can be calculated using the following formula, Eq. 1:

$$score = \sum_{i<j} sim(v_i, v_j) \tag{1}$$

where vi and vj are two words in the selected topic such that i and j are both integer values not more than the total number of words in the topic, and sim(vi,vj) is the similarity function used to calculate the word similarity between vi and vj. After obtaining the coherence score for each topic, the coherence score for the topic model is calculated by taking the average value of the coherence scores for all topics generated by the topic model.

A model performance evaluated using numerical measures is often perceived as a more formal approach. Therefore, it is used for our purpose of evaluating the performance of the topic models in our proposed solution. In Python, the coherence score can be calculated by calling the "get_coherence" method of an instance of "CoherenceModel" object in the Gensim library. The "coherence" and "topn" parameters are set to their default values, "c_v" and "20" respectively.

4. Analysis of Findings

a) Experiment 1: Comparing the topics of Python questions in different years

The number of topics, k must be specified before building topic models using LDA algorithms. In this case, we choose $k = 5$ for the first experiment set to identify the 5 topics hidden within the descriptions of Python questions for different years. These coherence scores of the topic models from 2008 to 2016 are plotted in a bar chart in 錯誤! 找不到參照來源。.

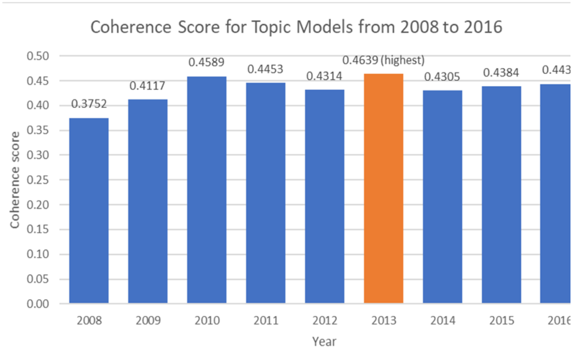


Fig. 2. Bar chart showing the coherence scores for topic models from 2008 to 2016.

From Fig. 2, the coherent score of the topics extracted from the Python questions in 2008 is the lowest among the scores of those in other years. This might be because the data only contains the Python questions posted in SO from August to December 2008. Therefore, the coherence score of the topics in 2018 might be susceptible to undesirable behaviour such as noise and scarcity in the textual data. The word clouds for each topic of Python questions posted on three selected years: 2008 (the first year), 2013 (the year with the highest coherence score) and 2016 (the last year), are shown in Fig. 3, 4 and 5 respectively.

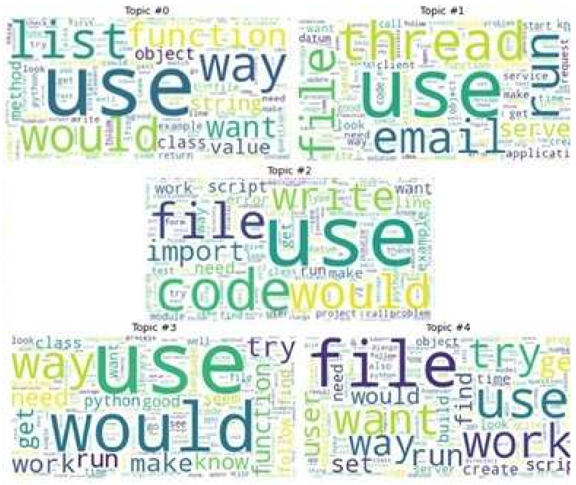


Fig. 3. Word clouds for topics extracted from Python questions in 2008.

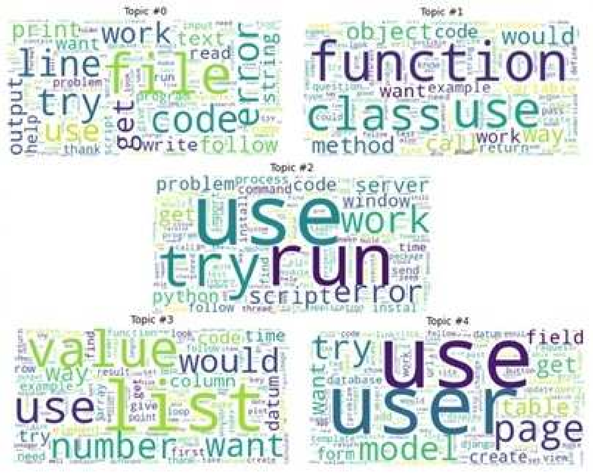


Fig. 4. Word clouds for topics extracted from Python questions in 2013.

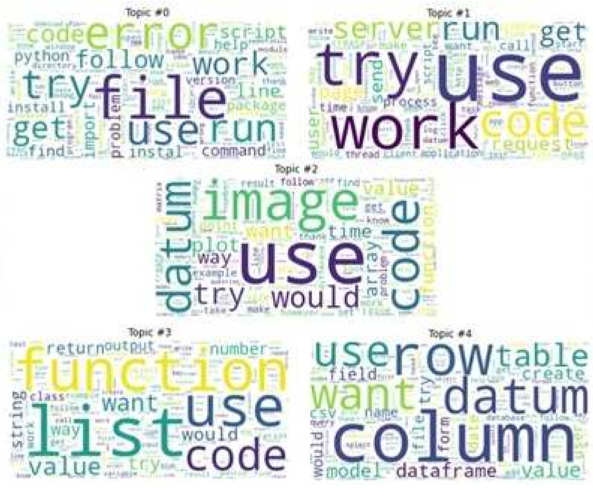


Fig. 5. Word clouds for topics extracted from Python questions in 2016.

By visually inspecting each word cloud generated from the extracted topics within the same year, the same words might appear multiple times in different topics extracted from the topic model. For example, in 2008 (see 錯誤! 找不到參照來源。), the words “file” and “way” appear as words with high probability score in three of the extracted topics. In 2016 (see Fig. 5), two of the words: “try” and “code”, are also common words in at least three extracted topics in that year. This implies that the topics extracted from the Python questions posted on SO within one year are quite similar to each other, even though the list of high scoring words in each topic might differ slightly from one topic to another.

Comparing the topics extracted from the Python questions across different years, there is a gradual shift towards the keywords such as “table”, “datum”, “row” and “column” from 2008 to 2016. These words are not among the high score words in any topics extracted in 2008. This shift might be due to the emergence of Python as an important programming language mainly used by software engineers or data scientists to perform their daily tasks involving transforming and preprocessing data stored in table format. On the other hand, words such as “write” and “import” only appear as high score words in one of the topics in 2008. This could imply that the questions related to importing libraries and writing files have been resolved in earlier days. Users might have already gathered enough information from earlier questions to solve similar problems without having to post new questions to the platform. Therefore, it can be said that there is no one-to-one correspondence between the topics extracted from one year to another because some topics in the past might not stay relevant today and they might eventually be replaced by newer topics in later years.

Finally, the word “use” seems to have a high probability score in all the topics extracted from Python questions for any given year. Therefore, the word “use” might be one of the domain-specific stop words which should be removed from the texts and documents. Table 2. List of high scoring words for each topic extracted from Python questions in 2008, 2013 and 2016.

summarizes the topics extracted from Python questions in 2008, 2013 and 2016, their most probable labels and the corresponding words in each topic. Note that the word “use” is removed from the lists in the table because it appears as one of the six words with the highest probability score in all topics from different years.

b) Experiment 2: Showing the topic for question with high scores

First, a baseline topic model is first built by using the LDA algorithm, with the number of topics, k= 5 specified. The topic model with 5 topics is chosen as the baseline model because we have also set the number of topics, k = 5 for all topic models in the first experiment set. The results from the first experiment set show that topic models with number of topics, k = 5 can generate good results in this study. However, it is also important to test different number of topics which yields the best model performance measured by the coherence score. Therefore, a hyperparameter tuning procedure is performed to search for the optimal value of k from a list of numbers from 2 to 10. The coherence scores for each topic model with the different number of topics are plotted in a bar chart in Fig. 6.

Table 2. List of high scoring words for each topic extracted from Python questions in 2008, 2013 and 2016.

Year	Topic	Label	Words
2008	#0	Python syntax	List, way, would, function, want, string
	#1	Server application	Thread, run, file, email, server, application
	#2	File I/O, import	Code, file, would, write, import, script
	#3	Other	Would, way, try, work, make, need
	#4	File application	File, work, want, try, way. run
2013	#0	File application	File, try, code, error, line, get
	#1	Functional, object-oriented	Function, class, object, would, way, call
	#2	Server application	Run, try, work, error, script, server
	#3	List, Array	List, value, number, would, want, way
	#4	Data model, table form	User, page, try, model, table, get
2016	#0	File application	File, error, try, run, work, get
	#1	Server application	Work, try, code, run, server, get
	#2	Plotting data	Image, code, datum, try, would, plot
	#3	Python Syntax	List, function, code, value, want, try
	#4	Data model, table form	Column, row, datum, want, table, value

Figure 6 shows that k = 8 yields the best topic model with the highest coherence score of 0.4482. Therefore, another topic model is built by using the LDA algorithm with the number of topics, k = 8 specified. The word clouds for each topic of Python questions with a high number of upvotes for topic models with 5 and 8 topics are shown in Fig. 7 and Fig. 8 respectively.

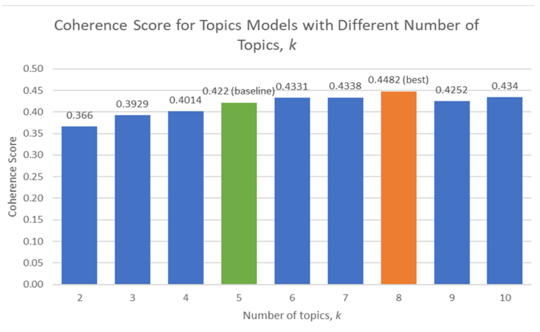


Fig. 6. Bar chart showing the coherence scores for topic models from 2008 to 2016.

By comparing the two sets of word clouds in Fig. 7 and 8 respectively, it can be observed that some of the topics relevant to the Python programmers are not significant in the baseline topic model. For example, topics related to "import" and "column" are not significant in extracted topics of the baseline topic model. These two words do not have their topics. Instead, they can be found in some other topics extracted using the same model. On the other hand, the words "import" and "column" are among the highest-scoring words in Topic #4 in Topic #5 respectively, extracted using the best topic model. The associated words in Topic #4 include: "file", "instal" and "package", indicating that this topic is related to the installation and importing of Python packages. Whereas in Topic #5, the associated high scoring words include: "model", "datum", "row" and "table", which might be a topic related to the data analysis: data model or data stored in table forms.

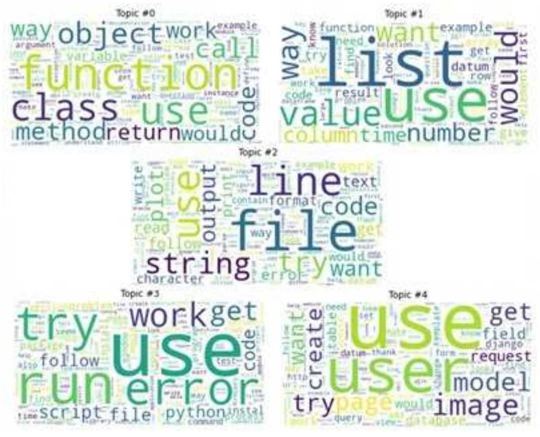


Fig. 7. Word clouds for topics extracted from Python questions with a high score using the best topic model – topic model with 5 topics.

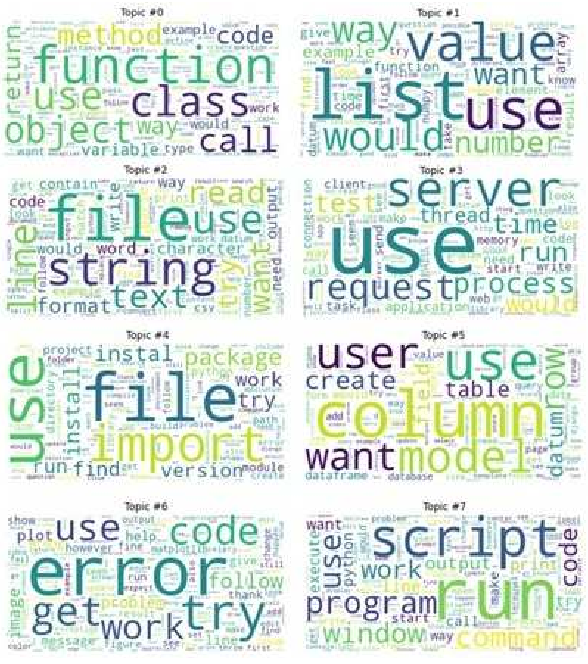


Fig. 8. Word clouds for topics extracted from Python questions with a high score using baseline topic model – topic model with 8 topics.

It is also observed that the high scoring words within the topics extracted using the best topic model (k = 8, which has a higher coherence score) are more associated with each other, as compared to those extracted using the baseline topic model (k = 5, with a lower coherence score). The list of topics extracted from both topic models, the most probable labels and the corresponding words of each topic are summarized in Table 3. Again, the word "use" that appear in all the topics is removed from the list of words in this table.

5. Discussions

Based on the experiment sets, it is shown that topic modelling is useful in extracting the topics from the description of the Python questions posted on SO. The topics are also manually labelled according to the high scoring words within each topic. In this section, we will discuss some improvements which can be performed on the study when extracting the information from the description of the Python questions with our topic modelling approach.

Table 3. List of topics extracted from the baseline and the best topic models, the most probable labels and the corresponding words for each topic.

Model	Topic	Label	Words
Base-line	#0	Functional, object-oriented	function, class, object, call, method, code
	#1	List, array	List, value, would, way, want, number
	#2	Text, string	File, line, string, try, code, plot
	#3	Python script	Run, try, error, work, get, script
	#4	Others	User, image, model, try, get, page
Best	#0	Functional, object-oriented	Function, class, object, call, method, return
	#1	List, array	List, value, would, way, number, want
	#2	File I/O	File, string, line, text, read, want
	#3	Server application	Server, request, process, run, time, test
	#4	Package import and installation	File, import, package, instal, try, version
	#5	Data model, table form	Column, model, user, want, row, create
	#6	Handling error	Error, try, get, code, work, follow
	#7	Python script and command	Run, script, program, command, window, work

First, during the text preprocessing steps, only the most common English stop words are removed. That means, the stop word removal only handles the most frequent words in general English language such as “a”, “the”, “I”, “me”, “by” and “was”. There is no additional step performed to collect the domain-specific stop words before performing stop word removal. Therefore, the experiment results might be influenced by these words to a certain degree. For example, the word “use” that appears as the high scoring words in every topic might be a domain-specific stop word.

Second, the hyperparameter tuning process in the second experiment set only involves changing the number of topics to a limited range of values (from 2 to 10) to obtain the best topic model. With this, there is a high chance that some better topic models (which might yield even higher topic coherence score using the same dataset) are missed out. However, by adding more hyperparameters, the computational resources required to complete the hyperparameter tuning process will increase exponentially.

The future works include efforts to collect domain-specific stop words and exclude them from the analysis of the Python questions posted on SO. Besides, a more thorough hyperparameter tuning process can be performed over a wider range of number of topics (say up to 50 topics), or by including more hyperparameters to the process to search for the model settings that yield the best topic model to the dataset. On top of that, this solution can be adapted to perform text analytics on the questions of other programming languages posted on SO, such as R, C++ and Java.

6. Conclusion

In this study, we apply topic modelling, a text analytics approach to study the Python questions posted on SO from 2008 to 2016. Specifically, we study the description of these questions because the description contains more semantic information than the title of these questions. Due to the unstructured nature of textual data, we perform a series of text preprocessing steps on the descriptions of the Python questions such as removing punctuations and changing the texts into lowercase, transforming the HTML script to normal text, tokenization, stop word removal and lemmatization. Then, two experiment sets are performed on the preprocessed texts. First, the questions are grouped into years and then a topic model is built for each group using the LDA algorithm. The extracted topics are then compared across different years to identify the trend and changing topics of questions over years. Second, the questions with a score of at least 4 are used to build another topic model to identify the topics that cover these questions. In both experiment sets, the evaluation criteria used are the inspection through the word clouds (informal approach) and the computation of the coherence score of each topic model (formal approach).

Through the results obtained from the first experiment set, it is observed that there is a gradual shift to the topics of the Python questions posted on SO from 2008 to 2016. The topic about the data model and table becomes more prominent over the years. For example, there is one topic with keywords such as “table”, “datum”, “row” and “column” generated by the topic model in 2016. These keywords are not significant in any topics generated by the topic models from earlier years. At the same time, the topic with keywords related to file input and output such as “code”, "file", "import" and "script" become less significant over the years.

On the other hand, the results obtained from the second experiment set shows that a suitable number of topics to the topic model built using the LDA algorithm yields extraction of more meaningful topics from Python questions with high votes posted on SO. In this study, the topic model with k = 8 yields the highest coherence score. The topics related to Python package installation and importing, and data models are more prominent in this topic model as compared to the baseline model in this experiment set (k = 5) with a lower coherence score. Therefore, the topic model with the right number of topics that yields a higher coherence score, the topic model is more effective in extracting relevant topics from texts and documents.

Several limitations of this study are also identified and discussed. First, the stop word removal process does not include domain-specific stop words. Second, the hyperparameter tuning process in the second experiment set only involves changing the number of topics to a limited range of values from 2 to 10 due to limited computational resources. Therefore, the future works of this study include collection of domain-specific stop words and exclude these stop words from the analysis and conducting a more thorough hyperparameter tuning process to identify the best topic model to extract information from Python questions posted on SO.

References

Ahmad, A., Feng, C., Li, K., Asim, S. M., & Sun, T. (2019). Toward empirically investigating non-functional requirements of iOS developers on stack overflow. IEEE Access, 7, 61145–61169.

Alghamdi, R., & Alfalqi, K. (2015). A survey of topic modeling in text mining. Int. J. Adv. Comput. Sci. Appl.(IJACSA), 6.

Ali, R. H., & Linstead, E. (2020). Modeling Topic Exhaustion for Programming Languages on StackOverflow. SEKE, (pp. 400–405).

Beyer, S., Macho, C., Di Penta, M., & Pinzger, M. (2020). What kind of questions do developers ask on Stack Overflow? A comparison of automated approaches to classify posts into question categories. Empirical Software Engineering, 25, 2258–2301.

Chakraborty, P., Shahriyar, R., Iqbal, A., & Uddin, G. (2021). How do developers discuss and support new programming languages in technical Q&A site? An empirical study of Go, Swift, and Rust in Stack Overflow. Information and Software Technology, 137, 106603.

Curiskis, S. A., Drake, B., Osborn, T. R., & Kennedy, P. J. (2020). An evaluation of document clustering and topic modelling in two online social networks: Twitter and Reddit. Information Processing & Management, 57, 102034.

Dirkson, A. R., Verberne, S., Kraaij, W., Jorge, A. M., Campos, R., Jatowt, A., & Bhatia, S. (2019). Narrative detection in online patient communities. Proceedings of Text2Story—Second Workshop on Narrative Extraction From Texts co-located with 41th European Conference on Information Retrieval (ECIR 2019), (pp. 21–28).

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

Jaworska, S., & Nanda, A. (2018). Doing well by talking good: A topic modelling-assisted discourse study of corporate social responsibility. Applied Linguistics, 39, 373–399.

Karbasian, H., & Johri, A. (2020). Insights for curriculum development: Identifying emerging data science topics through analysis of Q&A communities. Proceedings of the 51st ACM Technical Symposium on Computer Science Education, (pp. 192–198).

Kaur, J., & Buttar, P. K. (2018). A systematic review on stopword removal algorithms. International Journal on Future Revolution in Computer Science & Communication Engineering, 4, 207–210.

Marçal, I., Garcia, R. E., Eler, D., & Correia, R. C. (2020). A Strategy to Enhance Computer Science Teaching Material Using Topic Modelling: Towards Overcoming The Gap Between College And Workplace Skills. Proceedings of the 51st ACM Technical Symposium on Computer Science Education, (pp. 366–371).

Meldrum, S., Licorish, S. A., Owen, C. A., & Savarimuthu, B. T. (2020). Understanding stack overflow code quality: A recommendation of caution. Science of Computer Programming, 199, 102516.

Moessner, M., Feldhege, J., Wolf, M., & Bauer, S. (2018). Analyzing big data in social media: Text and network analyses of an eating disorder forum. International Journal of Eating Disorders, 51, 656–667.

Overflow, S. (2019, 10). Python Questions from Stack Overflow. Python Questions from Stack Overflow. Retrieved from https://www.kaggle.com/stackoverflow/pythonquestions

Rahate, P. M., & Chandak, M. (2019). Text Normalization and Its Role in Speech Synthesis. International Journal of Engineering and Advanced Technology Special Issue, 8, 115–122. doi:10.35940/ijeat.e1029.0785s319

Röder, M., Both, A., & Hinneburg, A. (2015). Exploring the space of topic coherence measures. Proceedings of the eighth ACM international conference on Web search and data mining, (pp. 399–408).

Tamla, P., Böhm, T., Nawroth, C., Hemmje, M., & Fuchs, M. (2019). What Do Serious Games Developers Search Online? A Study of GameDev StackExchange. CERC, (pp. 131–142).

Wang, L., Zhang, L., & Jiang, J. (2020). Duplicate question detection with deep learning in stack overflow. IEEE Access, 8, 25964–25975.

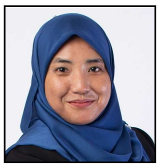
Xu, L., Nian, T., & Cabral, L. (2020). What makes geeks tick? a study of stack overflow careers. Management Science, 66, 587–604.

AUTHOR BIOGRAPHIES

Lee Yong Meng is currently pursuing his Master's degree in Data Science and Analytics from the School of Computer Sciences, Universiti Sains Malaysia (USM). He received his B.App.Sc. degree in mathematical modelling from the School of Mathematical Sciences, USM in 2016. He is also a Graduate Technologist with the Malaysia Board of Technologists (MBOT).

Soo Yin Yi is a postgrad student who is currently pursuing Master Data Science and Analysis at Universiti Sains Malaysia. He is currently working as a Data Analyst in Keysight Tecnologies. He also holds a Bachelor Degree in Logistics from Universiti Utara Malaysia.

Gan Keng Hoon is a senior lecturer in School of Computer Sciences, Universiti Sains Malaysia. She received her Ph. D. degree from Universiti of Malaya (UM) in 2013. She is current the Program Manager of Research Ecosystem and Innovation at the School of Compter Sciences. Her domains of specialization include information retrieval, structured retrieval, structured document representation and query optimization. She has initiated a research platform SIIR (Semantics in Information Retrieval @ ir.cs.usm.my) which is a research initiative related to semantically enhanced information retrieval, and its related applications.



Nur-Hana Samsudin was born in Kuala Lumpur, Malaysia. She received her Ph.D. degree in Computer Science from University of Birmingham, United Kingdom in 2017. Currently she is a Senior Lecturer at Universiti Sains Malaysia in Penang Malaysia. She currently holds one patent and one copyright besides producing research paper since her Master studies. Her interest covers in Natural Language Processing, Speech Processing, sustainable under-resourced language studies and polyglot speech synthesis.