School of Computer Sciences
Universiti Sains Malaysia

CDS522 Text and Speech Analytics
Semester 1, 2020/2021

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

Lee Yong Meng (P-COM0012/20)
Soo Yin Yi (P-COM0123/19)

6 January 2021

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

Yong Meng Lee[a], Yin Yi Soo[b]
School of Computer Sciences
Universiti Sains Malaysia
Penang, Malaysia.
[a]yongmeng@student.usm.my, [b]sooyinyi@student.usm.my

***Abstract*— Stack Overflow is one of the largest discussion platforms for programmers with different technical backgrounds to discuss and communicate their ideas and thoughts related to various topics, including but not limited to software development and data analysis. Many programmers are actively contributing to this platform and discuss about Python programming language, which is one of the most popular programming languages used for data analysis. To better determine what are the topics of the Python questionsn posted on the platform, the dataset of Python questions posted on Stack Overflow from 2008 to 2016 has been retrieved from Kaggle and NLP analysis have been done. This report will answer the topic trends for questions related to Python programming language from two different perspectives: the topics of the Python questions for each year from 2008 to 2016, and the topics of the Python questions with high scores.**

Keywords—Stack Overflow, text processing, topic modelling, LDA, coherence scores

I. Introduction

Stack Overflow is one of the largest open source software platforms where programmers ask and discuss about the programming questions. Stack Overflow does not only provide the platform for programmers to ask and answer the question, but also includes a system of votes, badges and user reputation to ensure that the questions and answers are meaningful or relevant to the platform users. This ecosystem have encourages many programmer to helps each other for free to solve their questions and prove their ability in programming problem solving to seek for a better job [1]. However, the popularity of Stack overflow have also surfaces several issue to the developer such as duplication of questions [2] and security issue [3].

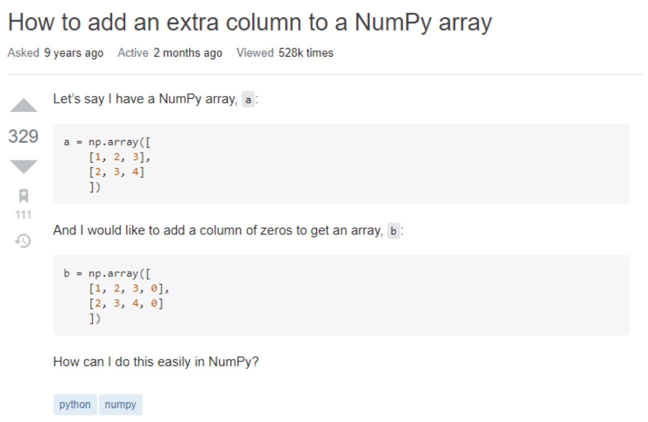
There are many questions posted on Stack Overflow, which are related to many different programming languages including C language, Python, Java, and R, to name a few. In terms of data analysis, both Python and R are commonly used. This is because both consists of many pre-built packages that are useful in data analysis and can be use directly when doing data analysis project.

In this study, a more exploratory method will be used to study the topics of Python questions posted on Stack Overflow. As mentioned in the previous part, the user can review the questions and answers and vote for a higher score if the answer is useful for them. To examine which kind of questions tend to receive higher score and explore the hidden trend of the topics related to the Python questions posted on the platform, the Python questions dataset [4] have been retrieved from Kaggle website.

II. Problem Statement

The first problem is to study the Python questions posted on Stack Overflow to discover the topics hidden behind these questions, and then to identify the trend of these topics over the years from 2008 to 2016. The topics that are discussed in Stack Overflow is constantly changing. According to a research done in Canada, they have concluded that the topic discussed in Stack overflow have been changing according to several trends such as embracing open source software, towards mobile development and better documentation method [5]. However, their research focus on all the language and in this report will be focus on only Python programming language.

The second problem in this study is to examine the popular topics of the Python questions posted on Stack Overflow and receive high scores from 2008 to 2016. As mentioned, Stack Overflow is an open source platform for any users not only to openly discuss their thoughts and ideas, but also to contribute to the platform by reviewing the questions posted and vote for those that are relevant to the users. Therefore, it is important to identify what are the topics of the Python questions posted on the platform that are most relevant to the users.



How to add an extra column to a NumPy array

Asked 9 years ago Active 2 months ago Viewed 528k times

329

111

Let's say I have a NumPy array, a:

```
a = np.array([
    [1, 2, 3],
    [2, 3, 4]
    ])
```

And I would like to add a column of zeros to get an array, b:

```
b = np.array([
    [1, 2, 3, 0],
    [2, 3, 4, 0]
    ])
```

How can I do this easily in NumPy?

python numpy

Fig. 1. An example of Python question posted on Stack Overflow. [6]

The information obtained from this study 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 the documentations. Besides, it can also be used as a guideline for 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.

III. Proposed Solution

To address the problems that have been discussed in the previous section, we propose a solution which utilizes topic modelling technique to study and identify the topics among the Python questions posted on Stack Overflow. We have designed two different experiment sets to study the Python questions posted on Stack Overflow. In the first experiment set, the Python questions dataset are split into different groups according to the year these questions are created on the platform. Then, the topic models are built for each group to enable comparison between the topics of the Python questions created for different years. This comparison enables the trends of common topics of Python questions being asked across different years to be studied.

In the second experiment set, the size of the Python questions dataset is reduced to only 10% of its original number of records, showing only the questions with score (or upvotes received) of 4 and above. The purpose of this experiment set is to study and identify the topics of the Python questions posted on the platform across different years. In this experiment set, only one topic model is built, using the reduced Python question dataset.

Before building the topic models for the Python questions, data preprocessing is a crucial step to ensure the quality and validity of the data used to build the topic models. Steps taken to preprocess the Python questions dataset include the removal of all the Python questions with negative scores and attributes from the original dataset which are not relevant to our studies such as "Id" and "OwnerUserId". Since textual data is used for the study, the text preprocessing techniques such as tokenization and stop word removal are performed to transform the description of the Python questions into the form that can be transformed into computer-readable text corpus. For the topic modelling task, the statistical techniques named Latent Dirichlet Allocation (LDA) is used. Finally, we employ both informal and formal approaches to analyse the topics obtained from the model, which are word cloud and coherence score respectively. Fig. 2 shows the pipeline of the proposed solution.

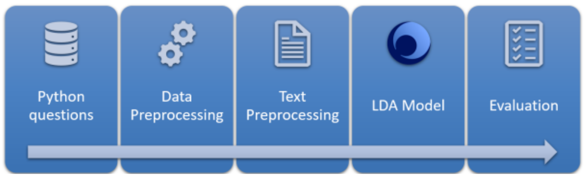


Fig. 2. The pipeline of the proposed solution.

The text preprocessing techniques, the LDA algorithm used in the topic modelling, and the evaluation criteria for the proposed solution are discussed next.

A. *Text Preprocessing*

Before the NLP process been start, the text preprocessing process must be performed to turn the words into the forms that is more useable and convenience in building the NLP model. This process is called as Text Normalization which will convert all the data from human readable language into a form that is machine readable [7]. Two text normalization process will be used in this project which are the Tokenization and Stop words removal.

In NLP, tokenization can be defined as the task to split a stream of characters into words [8]. There are basically two tokenization in the NLP process which are the word tokenization and the sentences tokenization. Word tokenization used to separate words via unique space character where sentences tokenization used to perform tokenization based on sentence boundaries and one of the examples is punctuations.

In this study, the tokenization process is used to determine the key words or token word that are useful in the NLP analysis. From the dataset, the special words that are probably be used to split the sentences will be the words such as "I", "me", "by" and "was" since they provide less information and the words after it might be useful in analysis. As an example, is the data item "In Python monkey patching generally works by overwriting a class or functions signature…", the sentences after "by" mentioned about overwriting the class or signature and this answer will be useful in the NLP process.

Another NLP preprocessing method is stop word removal. Stops words or noise words are the words that contain a little information that is not required in the analysis process [9]. Since the stop words contain little or no information for the data, it shall be removed to improve the efficiency of the NLP process. In this dataset, some symbol such as "<p>" or "</p>" have occurs to shows the distance between sentences. However, this symbol has no information in NLP process, and this is one of the stop words that have to be removed. Finally, lemmatization is also performed to convert the words to their base form (or dictionary form).

B. *Topic Modelling*

Topic modelling is one of the applications in text analytics used for studying and identifying the underlying key topics of texts and documents in an organized and summarized manner [10]. In its simplest term, 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 together 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 [11]. There is a probability distributed over each topic, measuring the likelihood of 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 package [12].

C. *Evaluation Criteria*

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 when building a topic model. Therefore, the challenge in evaluating the model performance of a topic modelling task is similar to those unsupervised learning tasks. Unlike topic classification, a supervised learning tasks in text analytics, there is no ground truth label used for evaluating the performance of a topic model.

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.

1) *Word Cloud*

Word cloud is a visual representation that captures and displays list of words from a document, which literally 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.

The advantages of using word clouds to visually represent the topics generated are, it is intuitive and engaging. Human is a visual creature, 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 human 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.

2) *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 [13]. The higher the degree of similarities among words within the same topic, the topic is said to be more coherent. That means, 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. 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.

IV. Analysis of Findings

In both experiment sets, the Python questions that are used to build the models are those with nonnegative scores only. Therefore, the number of Python questions used for generating the topic models are reduced from 607,282 to 570,972. Besides, the selected attribute used for the topic modelling is the column named "Body", which keeps the question description in textual form. However, the descriptions are stored as HTML script which include the code blocks enclosed with HTML tag pairs <pre><code></code></pre> in the original data. Therefore, steps are taken to remove the code blocks from the HTML script and then transform the complete HTML script to normal text. The Python library used for this purpose is the BeautifulSoup library from the bs4 module.

A. *Experiment 1: Comparing the topics of Python questions across 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 Python question descriptions for each year. Table I lists the coherence score of the topic models from 2008 to 2016.

TABLE I. List of coherence scores for topic models from 2008 to 2016.

Year	Coherence score (rounded to 4 decimal places)
2008	0.3752
2009	0.4117
2010	0.4589
2011	0.4453
2012	0.4314
2013	**0.4639** ***(highest)***
2014	0.4305
2015	0.4384
2016	0.4437

From Table I, 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 Stack Overflow since August 2008. Therefore, the coherence score of the topics in 2018 might be susceptible to undesirable behavior 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.

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 topic model. For example, in 2008 (Fig. 4), the words "file" and "way" appear as words with high probability score in three of the extracted topics. In 2016 (Fig. 5), the words "try" and "code" are also common words in at least three extracted topic in that year. This implies that the topics extracted from the Python questions posted on Stack Overflow 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.

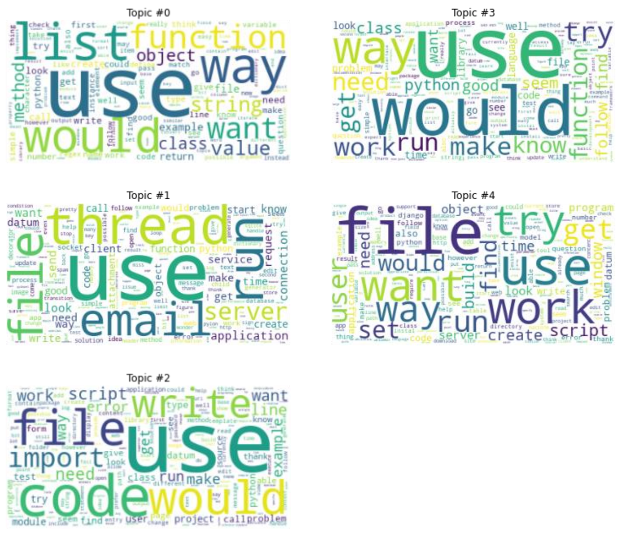


Fig. 3. Word clouds for topics extracted from Python questions in 2008.

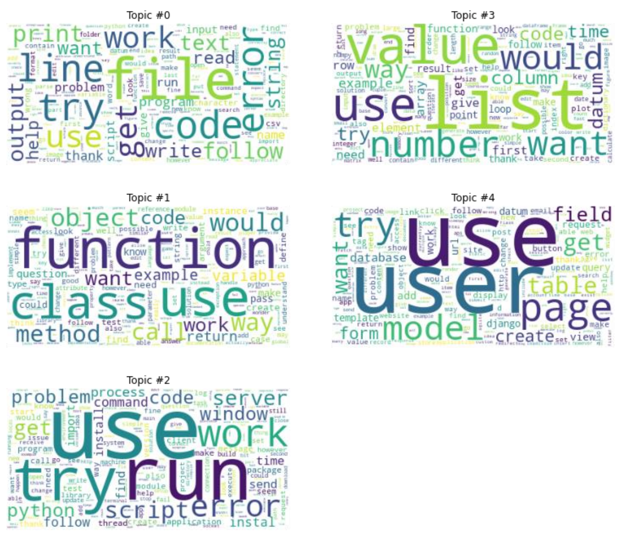


Fig. 4. Word clouds for topics extracted from Python questions in 2013.

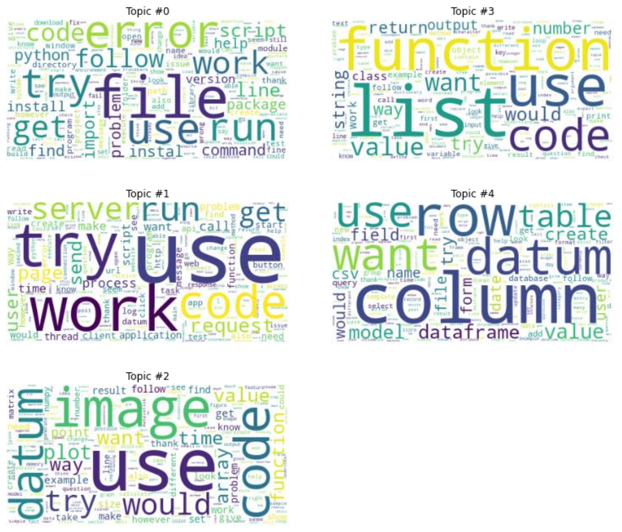


Fig. 5. Word clouds for topics extracted from Python questions in 2016.

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 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 II 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.

TABLE II. 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

B. *Experiment 2: Showing the topic for question with high scores.*

For the second experiment set, the size of the Python questions dataset is further reduced from 570,972 questions in the first experiment set to only 74,195 questions. These questions are the Python questions with score of at least 4 upvotes as when the dataset is collected. The data and text preprocessing steps are similar as the first experiment set.

Next, a baseline topic model is first built by using LDA algorithm, with the number of topics, $k = 5$ specified. The coherence score of the topic model is 0.4220. This is followed by a hyperparameter tuning procedure to identify the optimal value of k from list of numbers from 2 to 10 which yields the best model performance measured by the coherence score. The coherence score for each topic model with different number of topics from 2 to 10 are listed in Table III.

From Table III, it 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 LDA algorithm with the number of topics, $k = 8$ specified. The word clouds for each topic of Python questions with high number of upvotes for topic models with 5 and 8 topics are shown in Fig. 6 and 7 respectively.

TABLE III. COHERENCE SCORE FOR TOPIC MODELS ON HIGH SCORE PYTHON QUESTIONS WITH DIFFERENT NUMBER OF TOPICS.

Number of topics, *k*	Coherence score (rounded to 4 decimal places)
2	0.3660
3	0.3929
4	0.4014
5	0.4220 *(baseline)*
6	0.4331
7	0.4338
8	**0.4482 *(highest)***
9	0.4252
10	0.4340

By comparing the two sets of word clouds in Fig. 6 and 7 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 own 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.

The outcome of the evaluation using both informal (word cloud) and formal (coherence score) approaches correspond to each other. This means, the high scoring words within the topics extracted using the best topic model (which has a higher coherence score) are more associated with each other, as compared to the those extracted using the baseline topic model (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 is summarized in Table IV. Again, the word "use" that appear in all the topics is removed from the list of words in this table.

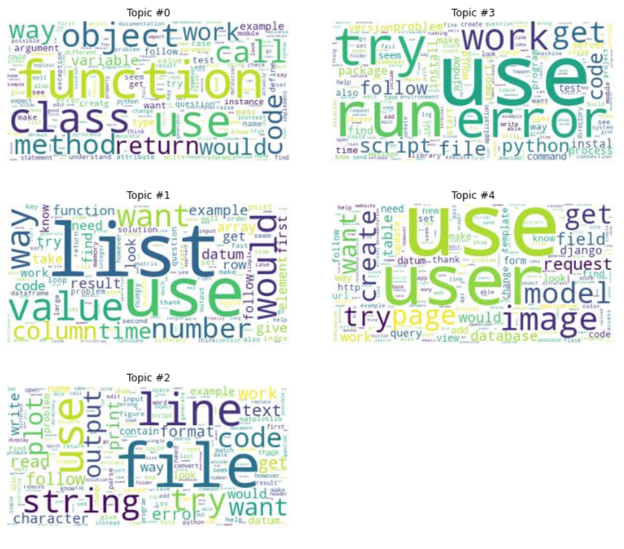


Fig. 6. Word clouds for topics extracted from Python questions with high score using the best topic model – topic model with 5 topics.

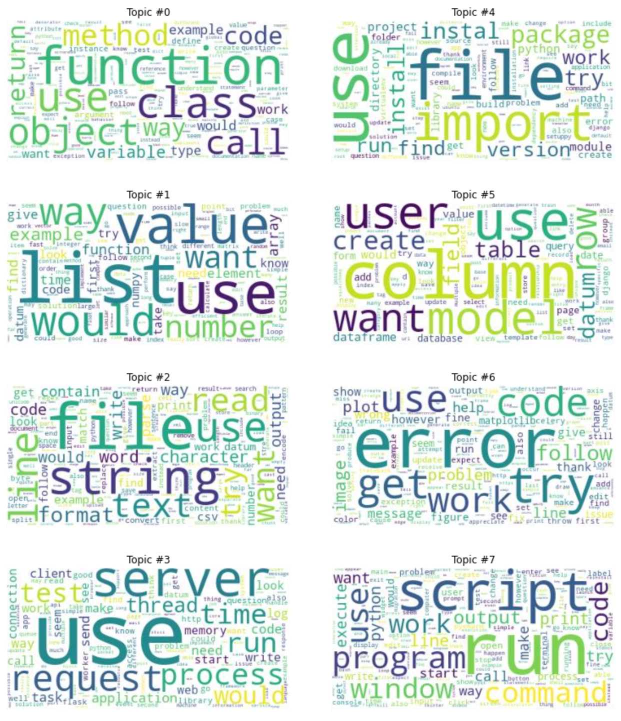


Fig. 7. Word clouds for topics extracted from Python questions with high score using baseline topic model – topic model with 8 topics.

TABLE IV. LIST OF TOPIC EXTRACTED FROM THE BASELINE AND BEST TOPIC MODELS, THE MOST PROBABLE LABELS AND THE CORRESPONDING WORDS OF EACH TOPIC.

Model	Topic	Label	Words
Baseline	#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

V. 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 Stack Overflow. The topics are also labelled according to the high scoring words within each topic. In this section, we will discuss about some improvements which can be perform on the study when extracting the information from the description of the Python questions with our topic modelling approach.

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 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 Stack Overflow. 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 Stack Overflow, such as R.

VI. Conclusion

In this study, we apply topic modelling, a text analytics approach to study the Python questions posted on Stack Overflow from 2008 to 2016. Specifically, we study the description of these questions because the description contains more semantic information than the question title. This is useful in the discovery of the topics hidden within the 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, tokenization, transforming the HTML script to normal text, 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 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 score 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 coherence score of each topic model (formal approach).

There is a gradual shift to the topics of the Python questions posted on Stack Overflow from 2008 to 2016. The topic about the data model and table becomes more prominent over the years, whereas topics related to file input and output becomes less significant over the years. It is also shown that a suitable number of topics to the topic model built using LDA algorithm yields extraction of more meaningful topics. This corresponds to the observation that, for the topic model with the right number of topics that yields higher coherence score, the topic model is more effective in extracting relevant topics from texts and documents.

References

[1] L. Xu, T. Nian, and L. Cabral, “What Makes Geeks Tick? A Study of Stack Overflow Careers,” *Management Science*, vol. 66, no. 2, pp. 587–604, 2020.

[2] Y. Zhang, D. Lo, X. Xia, and J.-L. Sun, “Multi-Factor Duplicate Question Detection in Stack Overflow,” *Journal of Computer Science and Technology*, vol. 30, no. 5, pp. 981–997, 2015.

[3] F. Fischer, K. Bottinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, and S. Fahl, “Stack Overflow Considered Harmful? The Impact of Copy&Paste on Android Application Security,” *2017 IEEE Symposium on Security and Privacy (SP)*, 2017..

[4] S. Overflow, "Python Questions from Stack Overflow," Stack Overflow, [Online]. Available: https://www.kaggle.com/stackoverflow/pythonquestions. [Accessed 12 9 2020].

[5] Anton Barua, Stephen W. Thomas, Ahmed E. Hassan, "What are developers talking about? An analysis of topics and trends in Stack Overflow," Springer Science, Kingston, 2012.

[6] "How to add an extra column to a NumPy array," Stack Overflow, [Online]. Available: https://stackoverflow.com/questions/8486294/how-to-add-an-extra-column-to-a-numpy-array?rq=1. [Accessed 12 9 2020].

[7] S. Kapadia, "Building Blocks: Text Pre-Processing," Towards Data Science, 3 Mar 2019. [Online]. Available: https://towardsdatascience.com/building-blocks-text-pre-processing-641cae8ba3bf. [Accessed 15 Dec 2020].

[8] B. Habert, G. Adda, M. Adda-Decker, P. Boula de Mareuil, S. Ferrari, O. Ferret, G. Illouz, P. Paroubek, "Towards Tokenization Evaluation," Orsay Cedex, 1998.

[9] J. Kaur and P. K. Buttar, “A Systematic Review on Stopword Removal Algorithms,” *International Journal on Future Revolution in Computer Science & Communicatin Engineering*, vol. 4, no. 4, pp. 207–210, Apr. 2018.

[10] Z. Tong and H. Zhang, “A Text Mining Research Based on LDA Topic Modelling,” *Computer Science & Information Technology (CS & IT)*, 2016.

[11] R. Alghamdi and K. Alfalqi, “A Survey of Topic Modeling in Text Mining,” *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 1, 2015.

[12] “Gensim: topic modelling for humans,” *Radim Å˜ ehÅ¯Ä™ek: Machine learning consulting*, 04-Nov-2020. [Online]. Available: https://radimrehurek.com/gensim/index.html. [Accessed: 22-Dec-2020].

[13] M. Röder, A. Both, and A. Hinneburg, “Exploring the Space of Topic Coherence Measures,” *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, 2015.

Appendix

This program is written in Python programming language using an Anaconda Jupyter Notebook environment. Note that some part of this program might not run as intended if the required Python libraries and packages are not installed on the local machine.

- The important/sample source code of this program is included at the end of this report.
- The complete source code is saved as “cds522_project_code.ipynb” and submitted together with this report.
- Alternatively, for viewing purpose, the HTML version of the Jupyter Notebook is saved as “cds522_project_code.html” and submitted together with this report.

A. Important Source Code

```
### LOAD DATA
# ---
file = "dataset/Questions.csv"
df = pd.read_csv(file, engine='python') # specify engine, for Jupyter Notebook


### DATA PREPROCESSING
# ---
# Remove rows with negative score
# Create new attribute: Year
# Remove columns: ['Id', 'OwnerUserId', 'Title', 'CreationDate']
# ...


### TEXT PREPROCESSING
# ---
body_processed = []

# Remove punctuation and lowercase
# Remove Code Blocks and Transform Question Descriptions from HTML to Normal Text
# ...

df_nonneg['BodyProcessed'] = body_processed

# Create new data frame
df_nonneg_save = df_nonneg[['Score', 'CreationYear', 'BodyProcessed']]

# Backup processed dataset
df_nonneg_save.to_csv('output/Questions_processed.csv')
df_nonneg_2 = pd.read_csv('output/Questions_processed.csv')


### EXPERIMENT SET 2
# ---
df_nonneg_score = df_nonneg_2[df_nonneg_2['Score'] >= 4]

# 1. Word tokenization
data_words_score = list(sent_to_words(df_nonneg_score['BodyProcessed']))

# 2. Bigram
bigram = gensim.models.Phrases(data_words_score, min_count=5, threshold=100) # higher
threshold fewer phrases.
bigram_mod = gensim.models.phrases.Phraser(bigram)

# 3. Stopword removal and lemmatization
data_words_nostops = remove_stopwords(data_words_score)
data_words_bigrams = make_bigrams(data_words_nostops, bigram_mod)
nlp = spacy.load("en_core_web_sm", disable=['parser', 'ner'])
data_lemmatized_score = lemmatization(data_words_bigrams, allowed_postags=['NOUN', 'ADJ',
'VERB', 'ADV'])


### LDA MODEL
# ===
# Helper function
def create_lda_model(corpus, id2word, num_topics=10):
    return LdaMulticore(
        corpus=corpus, id2word=id2word, num_topics=num_topics,
        random_state=100, chunksize=100, passes=10, per_word_topics=True)

# Create Dictionary: id2word
id2word = corpora.Dictionary(data_lemmatized_score)

# Create Corpus
texts = data_lemmatized_score
corpus = [id2word.doc2bow(text) for text in texts]

# Build LDA model for question with high score
lda_model_score = create_lda_model(corpus, id2word, num_topics=5)

# Print the Keyword in each topic
pprint(lda_model_score.print_topics())


### WORD CLOUD
# ---
def show_word_cloud_by_topic(model, num_topic=None, num_words=200):
    # Generate word cloud for each topic
      # ...

show_word_cloud_by_topic(lda_model_score)


### COHERENCE VALUES
# ---
# This function is called during hyperparameter tuning.
def compute_coherence_value_by_model(model, data_lemmatized, id2word):
    coherence_model_lda = CoherenceModel(model=model, texts=data_lemmatized,
dictionary=id2word, coherence='c_v')

    return coherence_model_lda.get_coherence()
```