Mining Patterns from Stack Overflow Questions

Team: Indraneel Mane, Kartik Lal, Vivek Kandregula and Ativeer Patni *Professor*: David Brady

Table of Contents

Contents

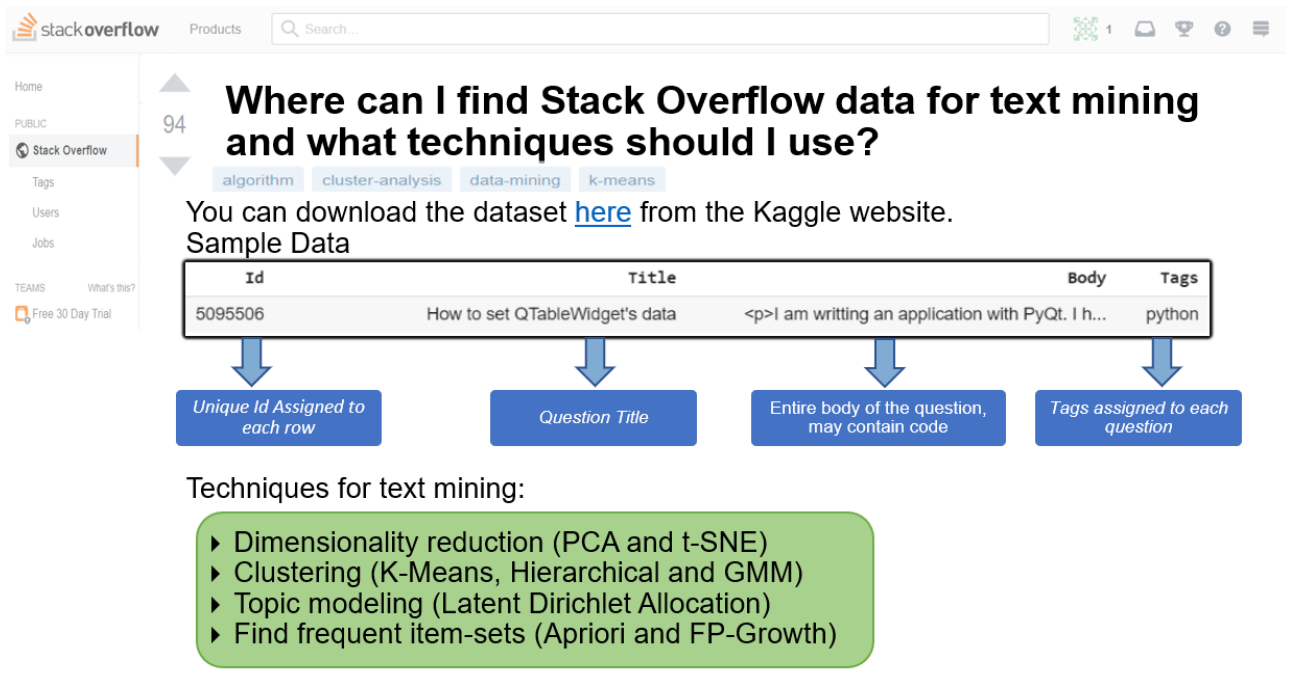
Business Objective:

- Understanding underlying patterns in the Stack Overflow questions will help the organization understand the technology requirements of their user base. This will help them hire technical experts who can answer users' questions
- Whenever a user posts a question, he/she assigns tags (metadata) alongside. Currently, no superset of all possible tags exist, hence allowing users to enter free text. This makes it hard for the organization to analyse the data, and hinders the scope of predictive modelling. Mining patterns and topics will enable creation of a tags-superset

Project Goals:

- Stack Exchange is the largest open community for developers to learn and share their knowledge
- The project goal is to mine patterns based on the question title and the body using various unsupervised machine learning techniques. The patterns a means of connecting experts with questions they can answer and can also be used to help users identify questions that are relevant to them

Data Description and Methodology:



Where can I find Stack Overflow data for text mining and what techniques should I use?

You can download the dataset here from the Kaggle website.

Sample Data

Id	Title	Body	Tags
5095506	How to set QTableWidget's data	<p>I am writting an application with PyQt. I h...	python

Unique Id Assigned to each row

Question Title

Entire body of the question, may contain code

Tags assigned to each question

Techniques for text mining:

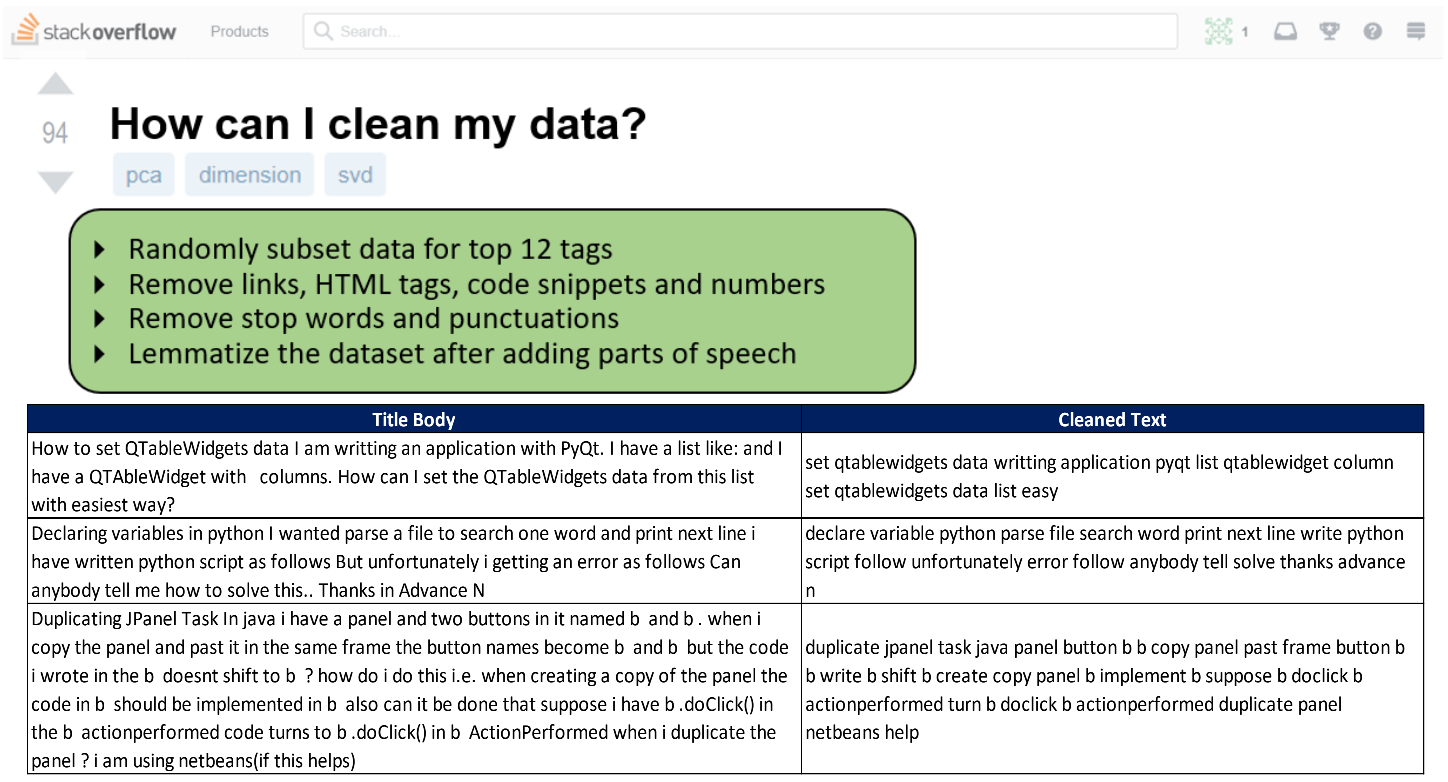
- Dimensionality reduction (PCA and t-SNE)
- Clustering (K-Means, Hierarchical and GMM)
- Topic modeling (Latent Dirichlet Allocation)
- Find frequent item-sets (Apriori and FP-Growth)

- The data consists of Stack Overflow questions mainly related to math and programming, with ~**6M records** and 4 attributes
- Attributes: Unique *ID*, question *Title*, the *Body* of the question and the *Tags* associated with the question
- Out of the 6M records, 50K questions were sampled at random to allow for faster computations

Techniques used for text mining:

- **Dimensionality reduction**
  - Dimensionality reduction an important step in text mining, as it solves the problem of inefficient computation and improves the performance of clustering techniques
  - In the project, two dimensionality reduction methods: PCA and t-SNE
- **Clustering**
  - Text clustering uses machine learning and natural language processing to understand and categorize unstructured, textual data
  - Three clustering methods were used in the project: K-Means clustering, Hierarchical clustering, and Gaussian mixture model clustering
- **Topic Modelling**
  - Topic modelling can be described as a method for finding a group of words from a collection of documents that best represent the information in the collection
  - Latent Dirichlet Allocation was used to perform topic modelling
- **Finding frequent Item sets**
  - Frequent item sets refer to set of words that often appear together in individual documents
  - Two methods were used to generate frequent item sets: Apriori and FP-Growth

Data Cleaning:



How can I clean my data?

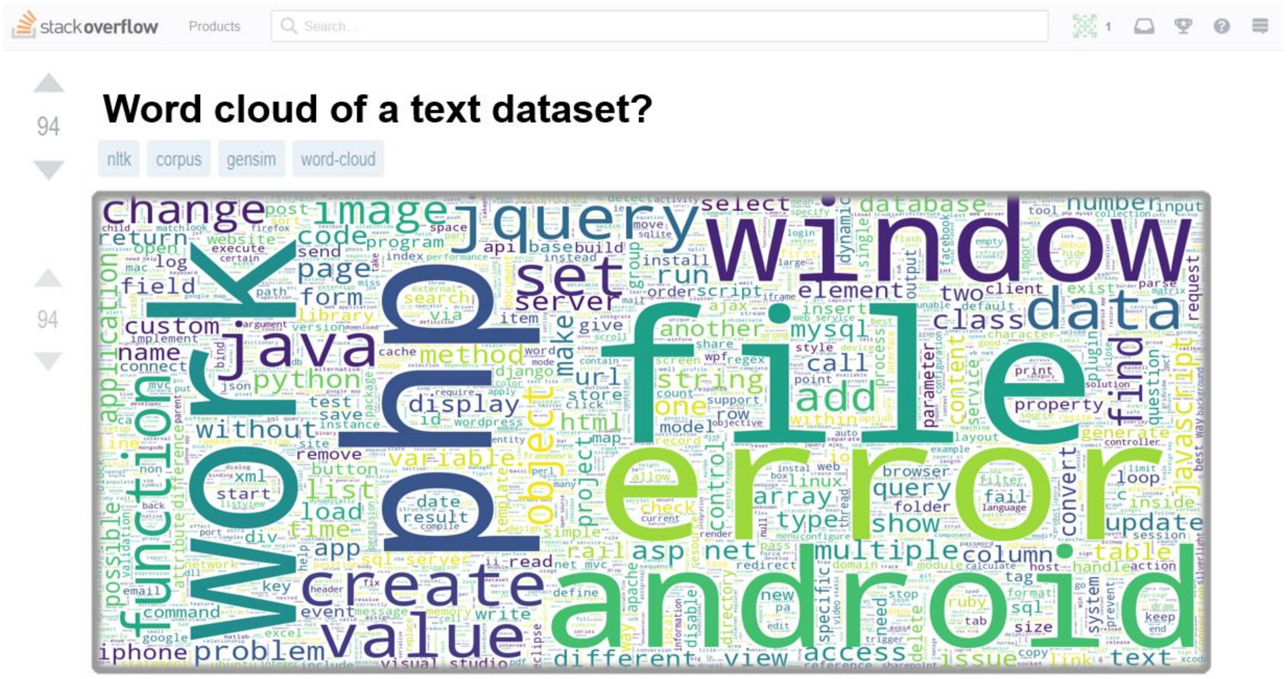
- Randomly subset data for top 12 tags
- Remove links, HTML tags, code snippets and numbers
- Remove stop words and punctuations
- Lemmatize the dataset after adding parts of speech

Title Body	Cleaned Text
How to set QTableWidgets data I am writting an application with PyQt. I have a list like: and I have a QTAbleWidget with columns. How can I set the QTableWidgets data from this list with easiest way?	set qtablewidgets data writting application pyqt list qtablewidget column set qtablewidgets data list easy
Declaring variables in python I wanted parse a file to search one word and print next line i have written python script as follows But unfortunately i getting an error as follows Can anybody tell me how to solve this.. Thanks in Advance N	declare variable python parse file search word print next line write python script follow unfortunately error follow anybody tell solve thanks advance n
Duplicating JPanel Task In java i have a panel and two buttons in it named b and b . when i copy the panel and past it in the same frame the button names become b and b but the code i wrote in the b doesnt shift to b ? how do i do this i.e. when creating a copy of the panel the code in b should be implemented in b also can it be done that suppose i have b .doClick() in the b actionperformed code turns to b .doClick() in b ActionPerformed when i duplicate the panel ? i am using netbeans(if this helps)	duplicate jpanel task java panel button b b copy panel past frame button b b write b shift b create copy panel b implement b suppose b doclick b actionperformed turn b doclick b actionperformed duplicate panel netbeans help

The following was done to clean the data:

- Selected the top 12 tags in the data and randomly subset the data to 50,000 questions pertaining to the top tags
- The feature variables were cleaned following the below steps,
    - The first pre-processing step was to transform the questions data into lower case. This prevented having multiple copies of the same words
    - The next step is removal of punctuations, as it doesn't add any extra information while treating text data. Therefore, all instances of it were removed to reduce the size of the training data
    - Removed stop words from the text data. For this purpose, an additional list of stop words was created and added to a predefined stop word library
    - The final step was removal of links, HTML tags, code snippets and numbers
- Tokenization of the text data
    - Performed Tokenization on the data, that is, dividing the text into a sequence of words or sentences
- Parts of speech tagging was performed before Lemmatizing the text data to ensure the correct root. For example, "running" and "ran" will output "run" if POS tags are assigned before lemmatization
- There are two approaches to reduce the text into inflectional forms: Stemming and Lemmatization
    - Stemming refers to the removal of suffices, like "ing", "ly", "s", etc. by a simple rule-based approach
    - Lemmatization refers to converting the word into its root word, rather than just stripping the suffices
    - Lemmatization (post POS tagging) has been preferred as it produces better results

Highest frequency words in the dataset:



Data visualizations (like charts, graphs, infographics, and more) give businesses a valuable way to communicate important information at a glance, but what for text-based dataset?

There are two approaches to find the most frequent words in the dataset:

- Counting occurrence of every word and representing them in a tabular form and arranging them in descending order
- Using word clouds

Word Cloud:

- Word clouds (also known as text clouds or tag clouds) work in a simple way: the more a specific word appears in a source of textual data, the bigger and bolder it appears in the word cloud

Word Cloud Functionality:

- For an intuitive visualization format that highlight important textual data points, using a word cloud can immediately convey important information about term frequencies.
- A word cloud is a collection, or cluster, of words depicted in different sizes. The bigger and bolder the word appears, the more often it's mentioned within a given text and the more important it is.

Word Cloud on Stack Overflow dataset:

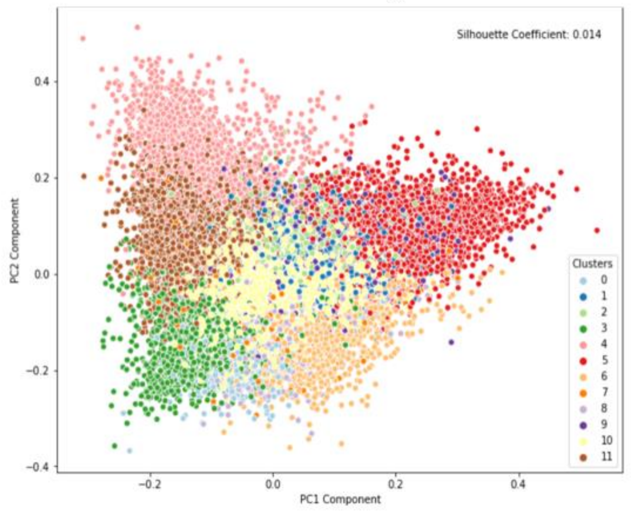
By creating a word cloud of our current dataset, we get the above output. Quite clearly, *words like php, work, file, error, and android* stand out and we can make out that most of the questions are centred around php or errors people usually get in myriad topics

Text Vectorization and Clustering:

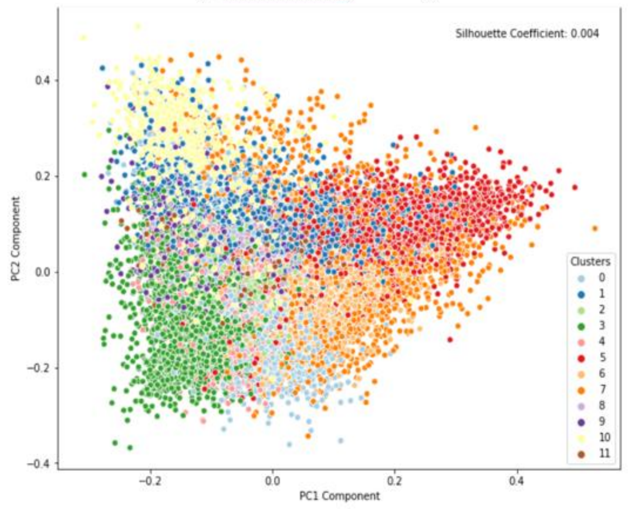
Clustering Using PCA

Dimensions were reduced from 1000 to 2

GMM using PCA



Fuzzy KMeans using PCA



- *TF- IDF* Vectorizer was used to convert the text to document-term matrix
- *TF* stands for 'term frequency' which is frequency of word in the document. *IDF* stands for 'inverse document frequency' which gives us how common or rare a word is in entire document set
- The following formula shows how to calculate both the terms –

$$TFIDF\ score\ for\ term\ i\ in\ document\ j = TF(i,j) * IDF(i)$$

$$where$$

$$IDF = Inverse\ Document\ Frequency$$

$$TF = Term\ Frequency$$

$$TF(i,j) = \frac{\text{Term i frequency in document j}}{\text{Total words in document j}}$$

$$IDF(i) = \log_2\left(\frac{\text{Total documents}}{\text{documents with term i}}\right)$$

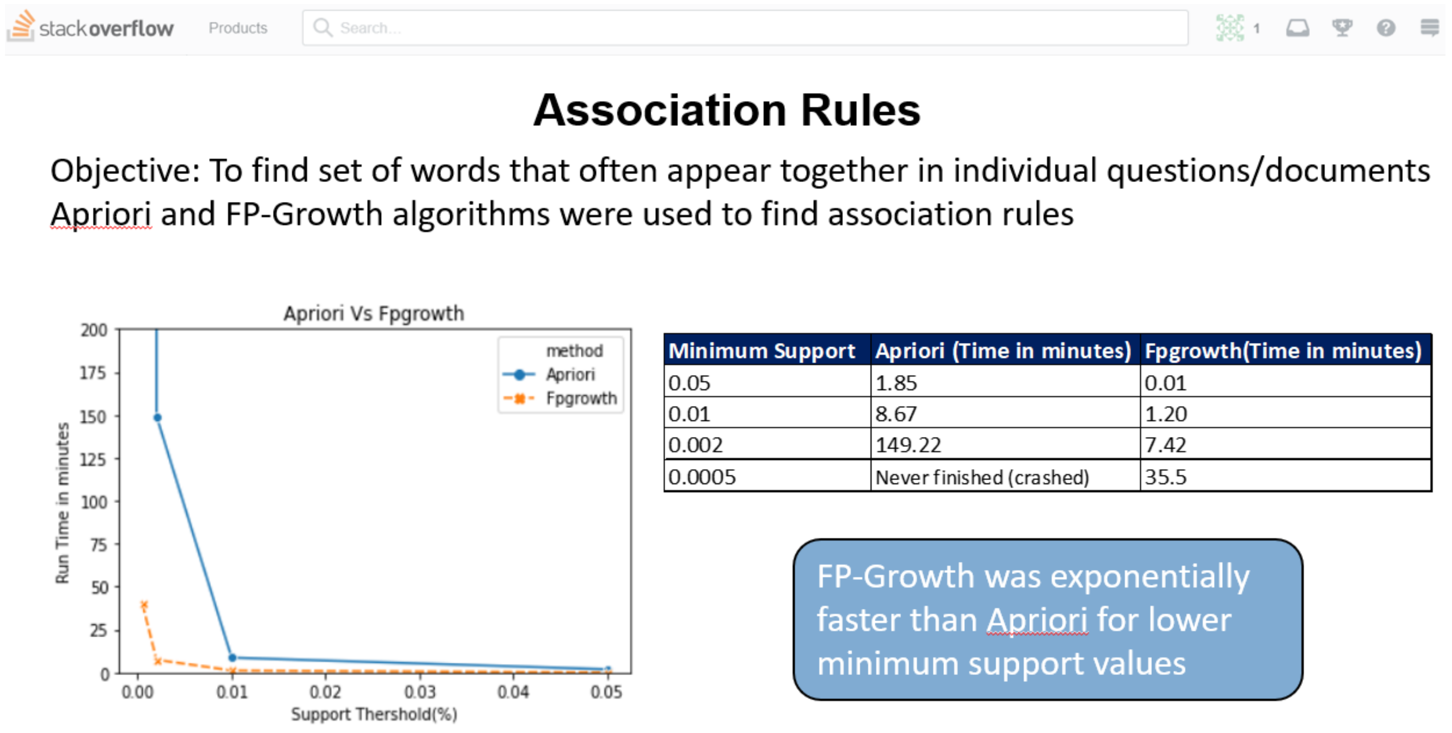
$$and$$

$$t = Term$$

$$j = Document$$

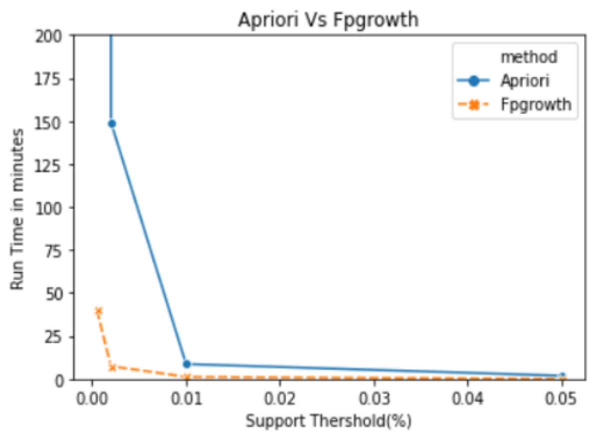
- Different clustering algorithms like Gaussian Mixture Models (GMM), KMeans, Fuzzy KMeans and Hierarchical Clustering were implemented
- Overlapping results were obtained using KMeans and hierarchical clustering. To overcome overlapping Fuzzy KMeans and GMM were used
- GMM had highest silhouette coefficient value (0.014) compared to Fuzzy KMeans (0.004)

Apriori v/s FP_Growth:



Association Rules

Objective: To find set of words that often appear together in individual questions/documents
Apriori and FP-Growth algorithms were used to find association rules



Minimum Support	Apriori (Time in minutes)	Fpgrowth(Time in minutes)
0.05	1.85	0.01
0.01	8.67	1.20
0.002	149.22	7.42
0.0005	Never finished (crashed)	35.5

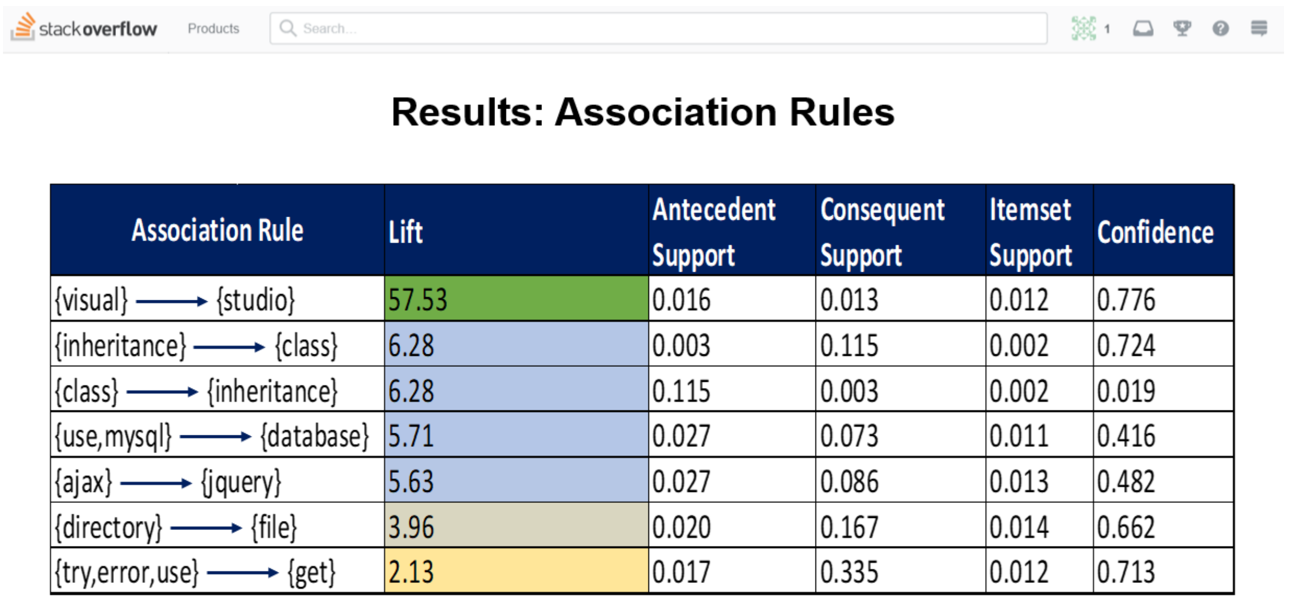
FP-Growth was exponentially faster than Apriori for lower minimum support values

- The intuition of our clustering criterion is that each cluster is identified by some common words, called frequent item sets, for the documents in the cluster
- Apriori and FP-Growth are algorithms for frequent itemset mining and generating association rules
- Apriori proceeds by identifying the frequent individual items in the data and extending them to larger and larger item sets if the item sets appear sufficiently in the data
- The FP-Growth Algorithm is an efficient and scalable method for mining the complete set of frequent patterns by pattern fragment growth, using an extended prefix-tree structure for storing compressed and crucial information about frequent patterns named frequent-pattern tree (FP-tree)
- Difference between Apriori and FP-Growth can be found in the table below:

Apriori algorithm	**FP-Growth algorithm**
Array based algorithm	Tree-based algorithm
Multiple database scans to generate candidate sets	Requires only two database scans
Generally slower than FP-growth	Faster than Apriori algorithm
Uses breadth-first search	Uses depth-first search

- From the graph above (above slide), Fp-growth takes exponentially less time than Apriori as the minimum support is decreased
- For the minimum support value of 0.0005 the Fp-growth took 35 minutes to run while for Apriori it ran for more than 5 hours and crashed
- For higher values of support both the methods take approximately the same time

Association Rules:


stackoverflow
Products
Search...

Results: Association Rules

Association Rule	Lift	Antecedent Support	Consequent Support	Itemset Support	Confidence
$\{visual\} \longrightarrow \{studio\}$	57.53	0.016	0.013	0.012	0.776
$\{inheritance\} \longrightarrow \{class\}$	6.28	0.003	0.115	0.002	0.724
$\{class\} \longrightarrow \{inheritance\}$	6.28	0.115	0.003	0.002	0.019
$\{use,mysql\} \longrightarrow \{database\}$	5.71	0.027	0.073	0.011	0.416
$\{ajax\} \longrightarrow \{jquery\}$	5.63	0.027	0.086	0.013	0.482
$\{directory\} \longrightarrow \{file\}$	3.96	0.020	0.167	0.014	0.662
$\{try,error,use\} \longrightarrow \{get\}$	2.13	0.017	0.335	0.012	0.713

Here are the formulae used to calculate the confidence and lift:

Support, $s(X \to Y) = \frac{\sigma(X \cup Y)}{N}$ Confidence, $c(X \to Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$

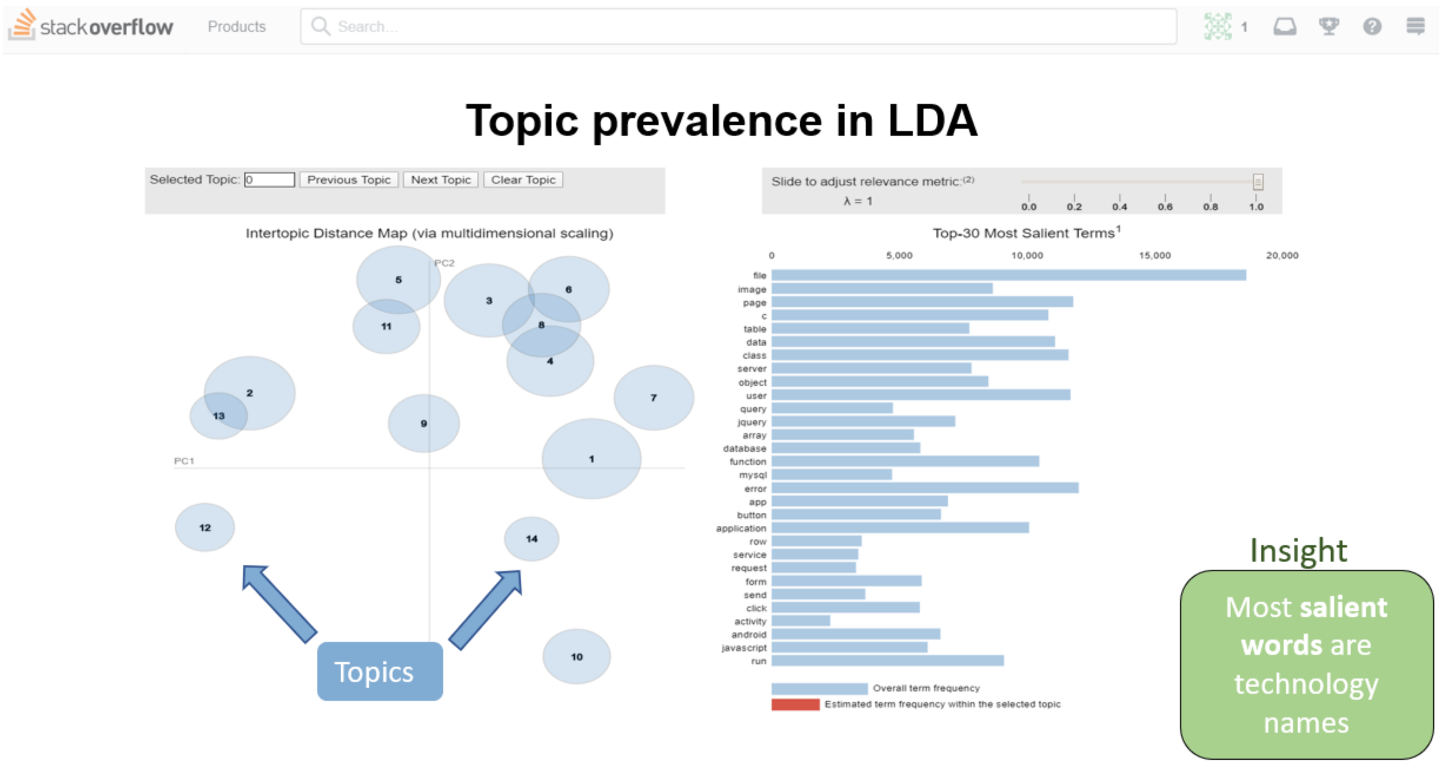
Lift or Interest factor, $Lift = I(X,Y) = \frac{c(X \to Y)}{\sigma(Y)} = \frac{\sigma(X \cup Y)}{\sigma(X)\sigma(Y)}$

If rules have lift of 1, it implies that probability of occurrence of antecedent is independent of consequent. Lift of greater than 1 implies that both have positive relationship, where strength of relationship is defined by the magnitude of the value. Lift of smaller than 1 implies that presence of one item has negative influence on the other

Here are some of the association rules we generated using Apriori and Fp-growth.

- The first association rule {visual} -> {studio} has a very high lift (*57*) which shows that both have a very strong positive relationship. If visual occurs in the question there is very high chance that studio will also appear, which makes sense as visual studio is an integrated development environment from Microsoft.
- The next two association rules show some interesting results. We see that both {inheritance} -> {class} and {class} -> {inheritance} have same itemset support and lift but different confidence
  - {inheritance} -> {class} has a higher confidence which tells us that having inheritance makes it more likely to have class associated with it as inheritance is mostly associated with classes
  - On the other hand, {class} -> {inheritance} has a very low confidence compared to {inheritance} -> {class}, which means that class is heavily associated with other words than inheritance
- Moving on, we see that association rule {use, mysql} -> {database} has a lift greater than one which shows they have positive relationship. Which is quite obvious as question having {use,mysql} would tend to have {database}
- The last rule {try, error, use} -> {get} has a high lift and confidence. This can be related to some common questions asked on stackoverflow like – *"I am trying to use (…) function but I get (…) type of error?"*

Topic Modelling:



Topic modeling is an unsupervised machine learning technique that can scan a set of documents, detecting word and phrase patterns within them, and automatically clustering word groups and similar expressions that best characterize a set of documents.

There are multiple Topic modelling algorithms such as LDA, LSA and lda2Vec

LDA has been preferred for this project as:

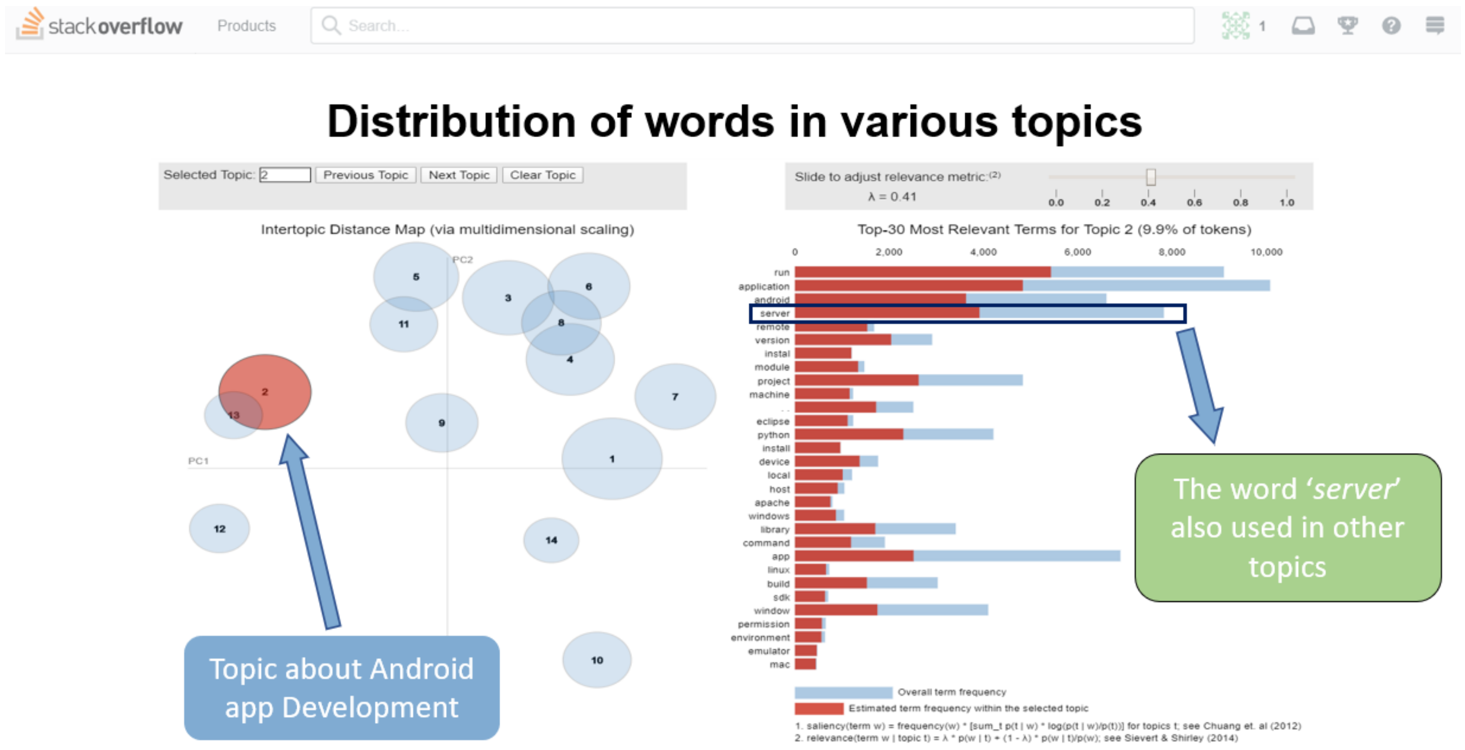
- The output is intuitive, modelling topics as weighted lists (or probability distribution) of words is a simple approximation yet a very intuitive approach in terms of interpretability
- Good performance on unseen documents if the structure of the text data is like the trained data
- It is fast as the Stack Overflow questions (documents) are not too long and the vocabulary size isn't too big, given the data subset

LDA (short for Latent Dirichlet Allocation) is an unsupervised machine-learning model that takes documents as input and converts the bag-of-words document into a (sparse) vector, where each dimension corresponds to a topic, and topics are learned to capture statistical relations of words. The model also says in what percentage each document talks about each topic.

The image above shows a probability mass function over all the possible words in the corpus of Stack Overflow questions for each individual topic

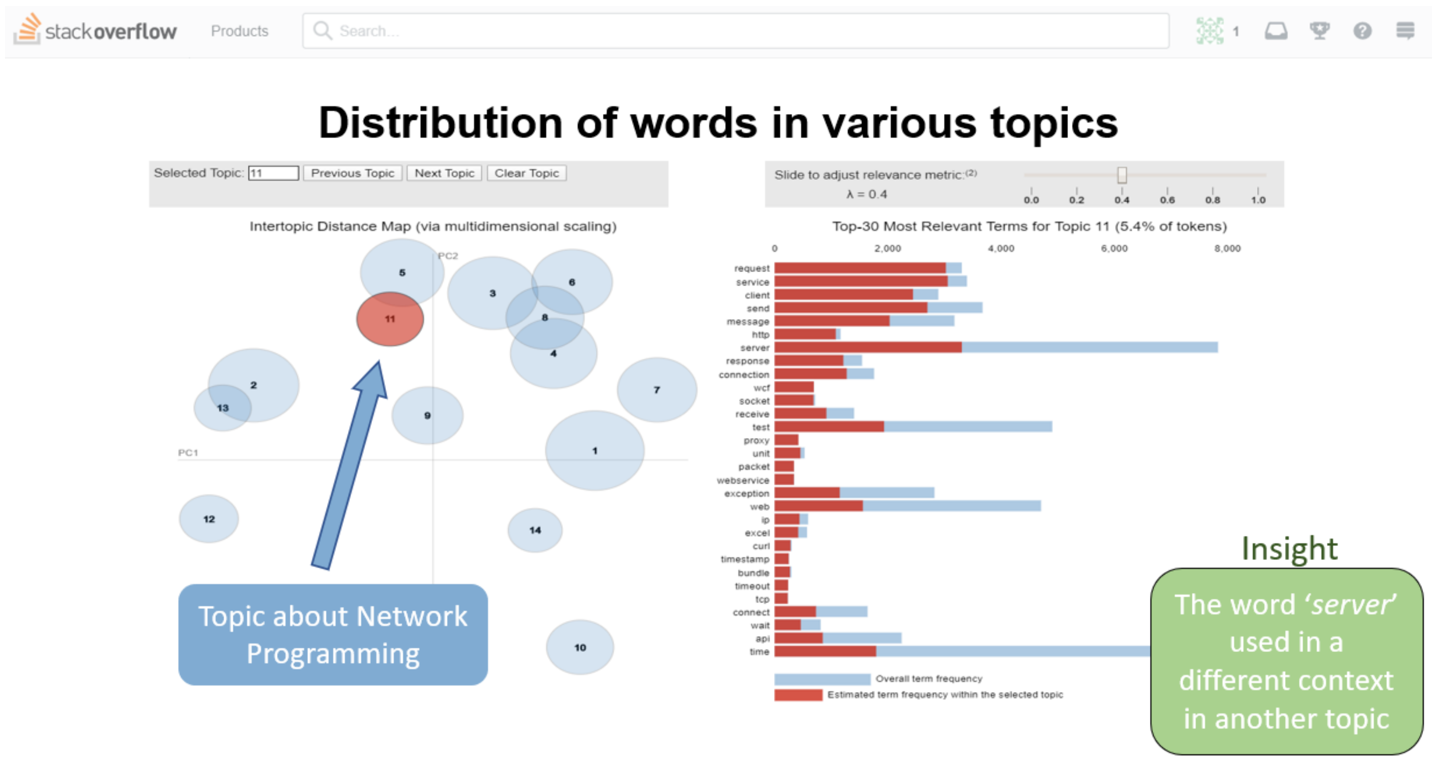
- The area of each circle is the topic prevalence
- The distances between circles (topics) are approximated measure of similarity between topics
- Some clusters overlap, which is due to similar words being used in different contexts
- Most salient words are names of technologies, alluding to the contents of topics

Topic Interpretation:



- **Understanding the topic:**
  - Each topic (or circle) is a probability distribution of all words in the vocabulary
  - The theme of the topic can be inferred from the relevant terms for the topic (stacked bar plot on the right in the image above)
  - For example, topic 2 (in the image above) is about Android application development due to the presence of relevant words like *android, application, server, version etc.*
  - The frequency of each word is split into the term frequency in the selected topic and the overall term frequency
  - For example, the frequency of the word *Server* is split almost equally in the selected topic and in the others, thus meaning that words in a topic are not exclusive but are also part of other topics
- **Tuning the relevance metric λ:**
  - The relevance metrics helps in interpretability of topics
  - Intuitively, the objective of tuning λ is to strike a balance between choosing only jargons or words that are more "layman-oriented terms"
  - If $\lambda=0$, then only the words that are exclusive to the topic (or jargons) would be outputted. Although, the interpretability would decrease as the theme would be hard to infer without context words. In the example above, words like *android, app, sdk, application* would be lost, which are words which specify android app development
  - On the other hand, $\lambda=1$ would output relevant words according to the overall topic frequency across all documents. Thus, $\lambda \epsilon [0.3,0.7]$ is ideal. $\lambda=0.4$ was chosen for the example

<u>Topic Interpretation and Insights:</u>

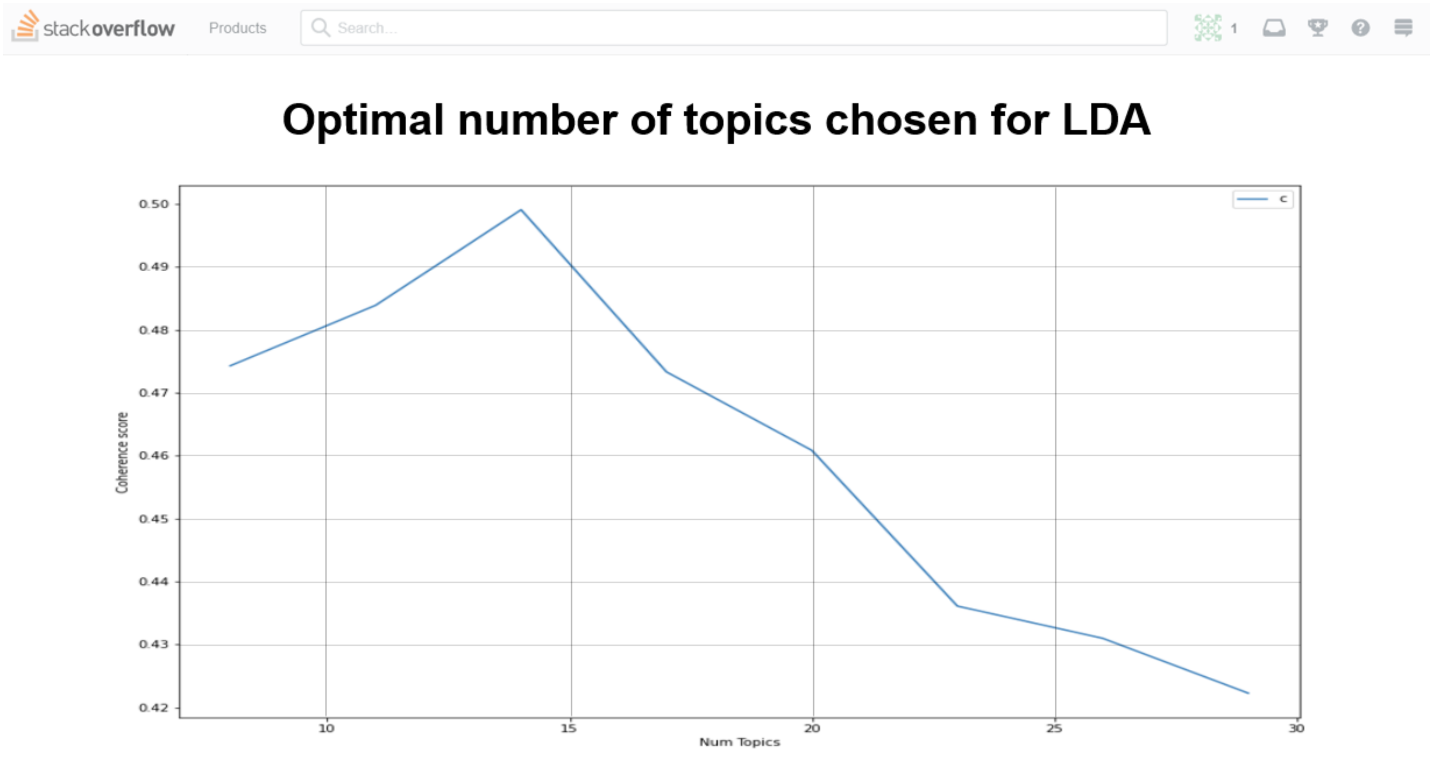


The image above illustrates an example where the same word can be present across multiple topics and have different contextual meaning

- <u>**Same word, different context:**</u>
  - It was found that the word *server* was also highly relevant in topic 11
  - Topic 11 (in the image above) is about network programming due to the presence of relevant words like *http, request, server, response, connection, tcp, socket etc.*
  - *Server* plays a vital role in network programming where it acts as the necessary middleware for *connections* to be established, client *requests* to come through to *sockets* and pass on the *responses* via *TCP* (Transmission Control Protocol)
    - The *italic* words above are very frequent use cases of *server* in the given context
  - In topic 2, *server* was being used in the context of android application development, whereas in topic 11, it is being used in the context of network programming

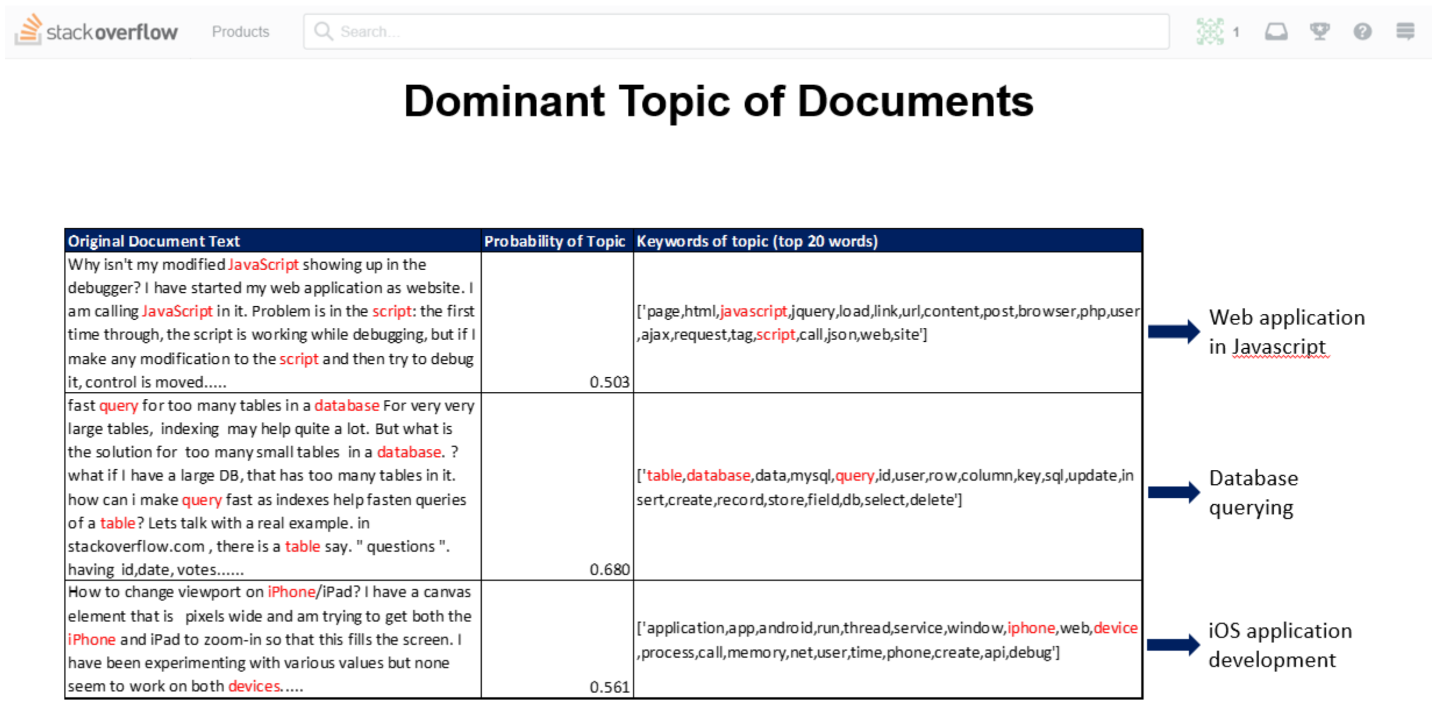
Also, as the topics of android development and network programming are loosely related to each other, their semantic distance is not very high.

Number of topics for LDA:



- A set of statements or facts is said to be coherent, if they support each other. Thus, a coherent fact set can be interpreted in a context that covers all or most of the facts
- Topic *Coherence* is a measure used to evaluate topic models. It scores a single topic by measuring the degree of semantic similarity between high scoring words in the topic. These measurements help distinguish between topics that are semantically interpretable topics and topics that are artifacts of statistical inference
- Each such generated topic consists of words, and the topic coherence is applied to the top N words from the topic. It is defined as the average / median of the pairwise word-similarity scores of the words in the topic
- A good model will generate coherent topics, i.e., topics with high topic coherence scores. Good topics are topics that can be described by a short label; therefore, this is what the topic coherence measure should capture
- Topic coherence is a good measure for tuning the hyperparameter- *Number of topics*
- Using *C_v* method of topic coherence retrieves the co-occurrence counts for the given words using a sliding window over the text
- LDA models were trained for different values of *No_topics* and coherence scores were calculated for each model
- The optimal number of topics was found to be 14, i.e. the point of inflection in the plot
- The number 14 is used in the topic models generated above, where 14 different topics are created
- Choosing right number of topics is imperative. If too many topics are generated, then there would be heavy overlap between topics, and if very few topics are generated, there would not be enough clusters capturing the various themes/patterns in data

Dominant Topic of Documents:


stackoverflow
Products
Search...

Dominant Topic of Documents

Original Document Text	Probability of Topic	Keywords of topic (top 20 words)
Why isn't my modified JavaScript showing up in the debugger? I have started my web application as website. I am calling JavaScript in it. Problem is in the script: the first time through, the script is working while debugging, but if I make any modification to the script and then try to debug it, control is moved.....	0.503	['page,html,javascript,jquery,load,link,url,content,post,browser,php,user,ajax,request,tag,script,call,json,web,site']
fast query for too many tables in a database For very very large tables, indexing may help quite a lot. But what is the solution for too many small tables in a database. ? what if I have a large DB, that has too many tables in it. how can i make query fast as indexes help fasten queries of a table? Lets talk with a real example. in stackoverflow.com , there is a table say. " questions ". having id,date, votes......	0.680	['table,database,data,mysql,query,id,user,row,column,key,sql,update,insert,create,record,store,field,db,select,delete']
How to change viewport on iPhone/iPad? I have a canvas element that is pixels wide and am trying to get both the iPhone and iPad to zoom-in so that this fills the screen. I have been experimenting with various values but none seem to work on both devices.....	0.561	['application,app,android,run,thread,service,window,iphone,web,device,process,call,memory,net,user,time,phone,create,api,debug']

- Once LDA has been implemented on the pre-processed data, the next step is to identify the dominant topic of every document.
- The dominant topic for each document/question with its probability along with the original question text was obtained. The above image shows a sample of the same
- Based on the above table, the keywords obtained show that most of keywords generated are part of the document
- Hence, using the above techniques we can assign topic to every document

Conclusion and Future Scope:

- To assign tags to unlabelled data we can perform clustering techniques such as KMeans, GMM, etc but too many overlapping results were obtained
- Thus, different techniques such as Apriori, FP-growth, LDA and topic modelling using LDA were used
- Visually, topic modelling gives the best result where most of the topics assigned to documents are correctly identified
- However, cleaning the dataset is the most important step otherwise it assigns stop-words or commonly used words in English as the most prominent topic in many documents
- Besides engineering more sophisticated features, future work should focus on optimizing the runtime of the models, as they are currently too slow to be used in practice
- Having a better approximation of conversion of text to a document-term matrix can significantly improve the clustering results
- Deep learning techniques (self-supervised) like Word2Vec and Doc2Vec can also enhance data mining
- Further, one might want to investigate whether co-occurrence statistics of tags could further improve the model, as there are a lot of tag pairs such as inheritance and class, visual and studio that co-occur very often