Mapping functionality scripts to documents using Latent Dirichlet Allocation

Hardik Prabhu

Introduction

- We have been provided a corpus of functionality scripts. We wish to understand the functionality tested by the scripts.
- We approach the task by first extracting different documents corresponding to the different functionalities.
- We will then devise a method to map the scripts to the documents using Latent Dirichlet Allocation (LDA), which is a probabilistic topic modeling technique.

Terminology

- Corpus: A collection of documents.
- Document: A set of words.
- Word: The most basic entity.
- Vocabulary: Set of all the words present across the entire corpus.

Latent Dirichlet Allocation(LDA)

- LDA is a three-level hierarchical bayesian model, in which each document of a collection is modeled as a finite mixture over an underlying set of topics.
- The documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words (vocabulary).
- The model assumes that the number of topics are specified before any data is generated.
- The probabilistic topic modelling method is based on the “bag-of-words” assumption, i.e. the order of words in a document can be neglected.

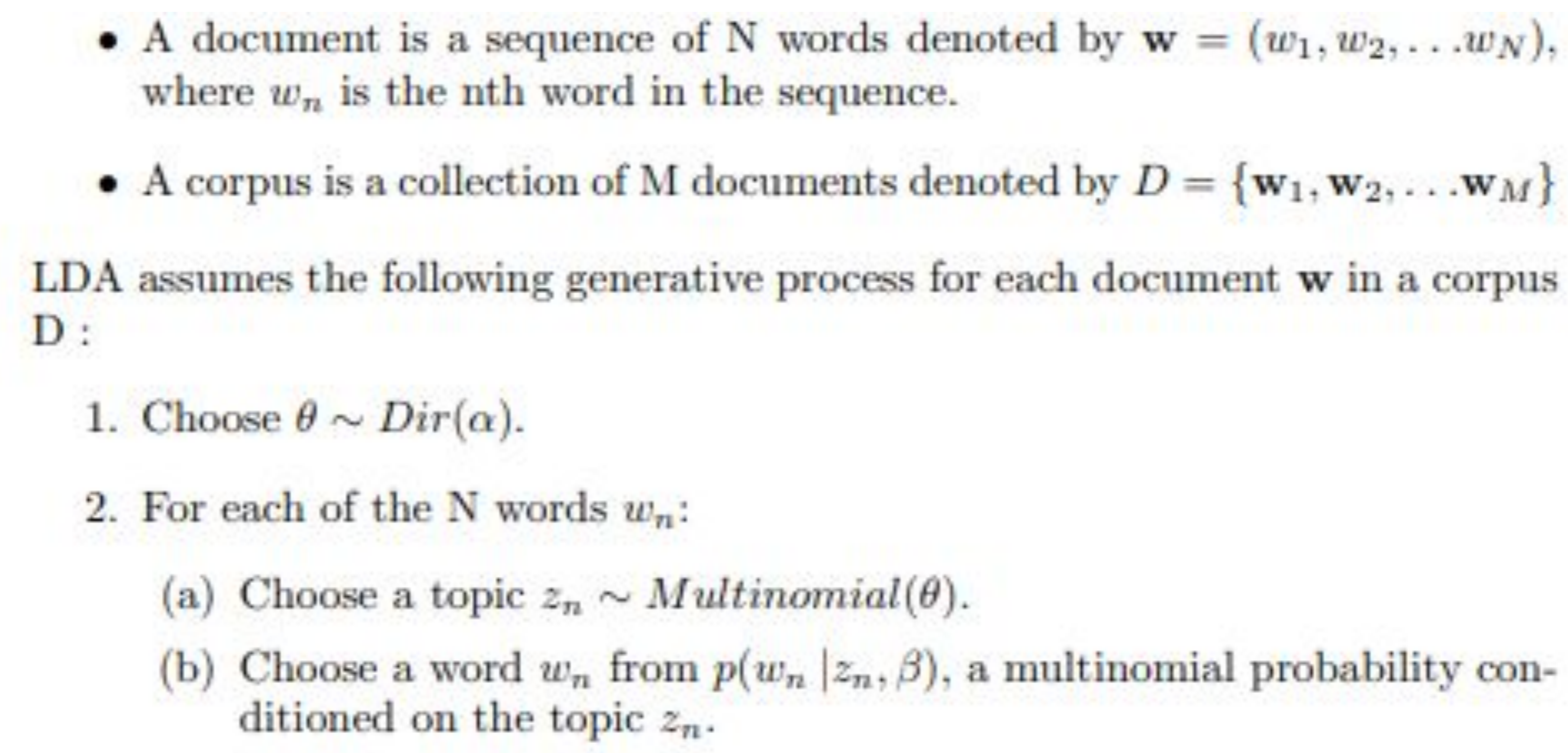
Why LDA?

- A document can be a part of multiple topics, kind of like in soft clustering in which each data point belongs to more than one cluster.
- LDA exploits the sparsity of the structure of a document. That is, the documents are mostly made up of a select few topics which themselves have few words occurring with higher probabilities.
- We can look at how the topics are distributed and see what the major themes are, both within documents and the topics. We can pick out documents with similar themes in the corpus.

LDA- The generative model

- Imagine writing a document consisting of 30 percent food and 70 percent science. By article, we mean a bag of words, where the order of words isn’t important.
- To generate each word in the document, the model will :
  1. First pick a topic according to a multinomial distribution over topics. In our case the distribution over topic will be, **{Food:0.3 , Science:0.7, Rest of the topics:0}.**
  2. Based on the selected topic, we will have a multinomial distribution over the vocabulary. The model will then select a word from the vocabulary based on that distribution. For example, the distribution for food topic maybe something like, **{BANANA:0.3, Fruits:0.3, Dairy:0.1, Chef:0.1, Cricket:0, electrons:0,Cuisine:0.2, Carbon:0}**
- In real use case, the topics may not be clearly distinguishable, the topics are abstract, and are only characterized by the multinomial distribution over the vocabulary.

LDA- The generative model

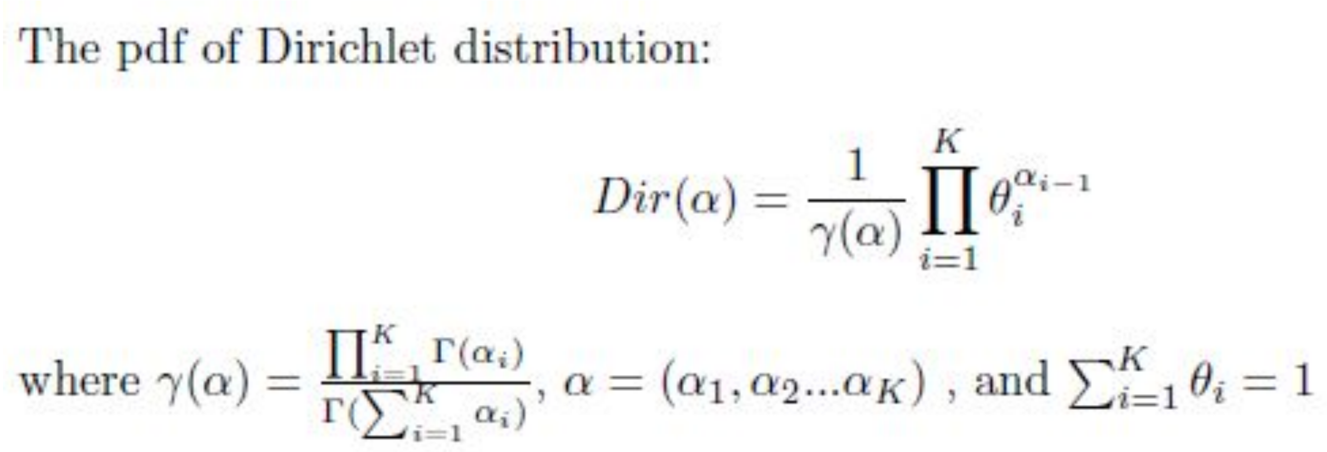


- A document is a sequence of N words denoted by $\mathbf{w} = (w_1, w_2, \ldots w_N)$, where w_n is the nth word in the sequence.
- A corpus is a collection of M documents denoted by $D = \{\mathbf{w}_1, \mathbf{w}_2, \ldots \mathbf{w}_M\}$

LDA assumes the following generative process for each document $\mathbf{w}$ in a corpus D :

1. Choose $\theta \sim Dir(\alpha)$.
2. For each of the N words w_n:
   (a) Choose a topic $z_n \sim Multinomial(\theta)$.
   (b) Choose a word w_n from $p(w_n | z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

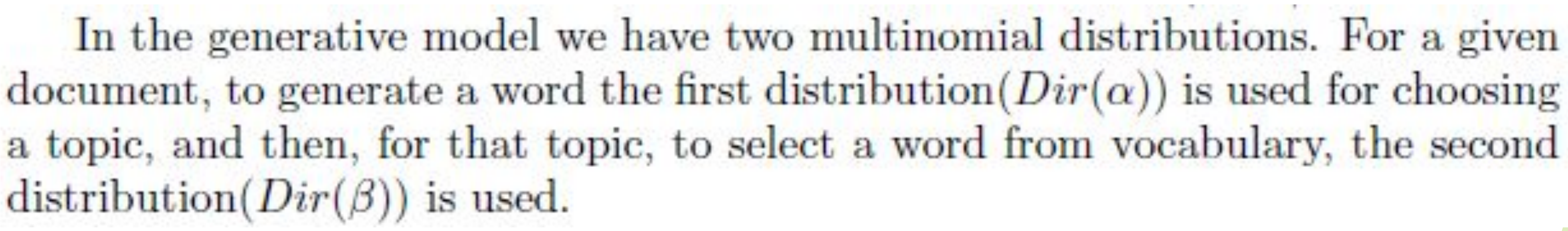
Dirichlet Distribution



The pdf of Dirichlet distribution:

$$Dir(\alpha) = \frac{1}{\gamma(\alpha)} \prod_{i=1}^{K} \theta_i^{\alpha_i - 1}$$

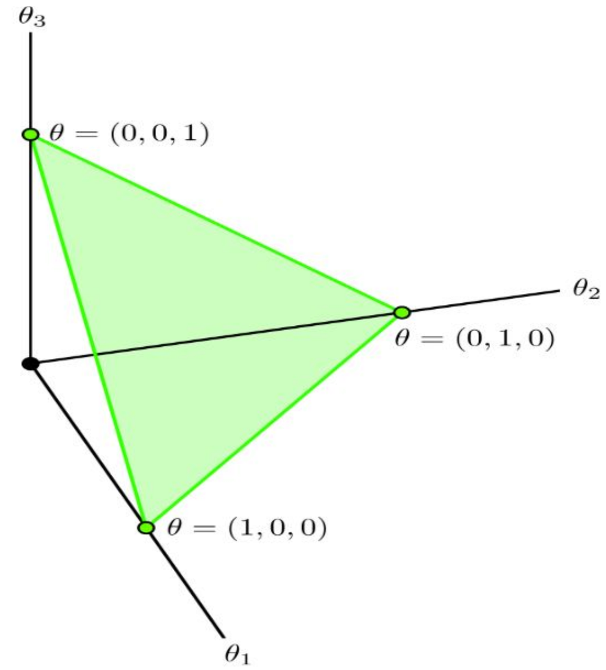
where $\gamma(\alpha) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{K} \alpha_i)}$, $\alpha = (\alpha_1, \alpha_2 ... \alpha_K)$, and $\sum_{i=1}^{K} \theta_i = 1$

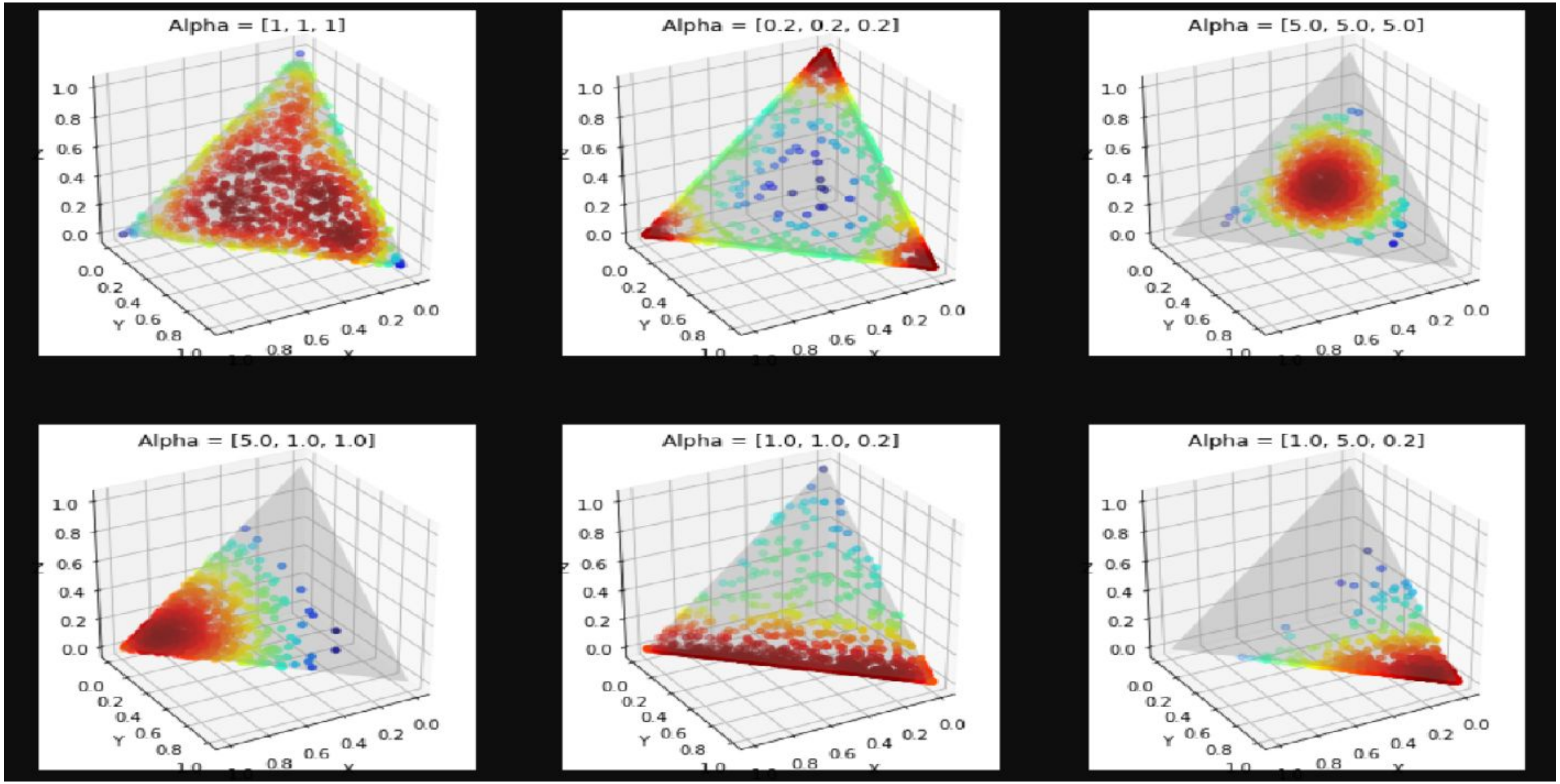


In the generative model we have two multinomial distributions. For a given document, to generate a word the first distribution($Dir(\alpha)$) is used for choosing a topic, and then, for that topic, to select a word from vocabulary, the second distribution($Dir(\beta)$) is used.

Example

- Consider a topic model over only 3 topics.
- For the 3 topics, we have the 2-dimensional simplex which contains all the points (x,y,z) such that x+y+z = 1.
- Each of these points represents a multinomial distribution over topics.
- Each coordinate represents the probability of choosing a particular topic.





- For symmetric Dirichlet distributions, a low alpha value places more weight on having each document composed of only a few dominant topics (whereas a high value will return many more relatively dominant topics).

- Using such a generative model for a collection of documents, LDA tries to backtrack from the documents to find a set of topics that are likely to have generated the collection.

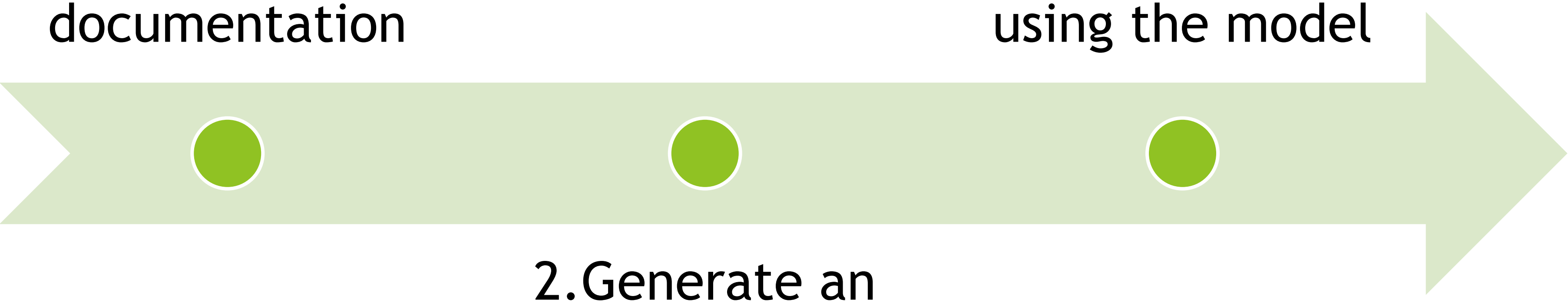
- Given the corpus of the documents, the model finds the parameters (Dirchlet priors) that best maximize the joint probability of the documents.

- This is done by using the EM algorithm.

Mapping scripts to documents

In this section, we demonstrate the entire mapping procedure by mapping payment scripts to payment documents.

Mapping scripts to documents



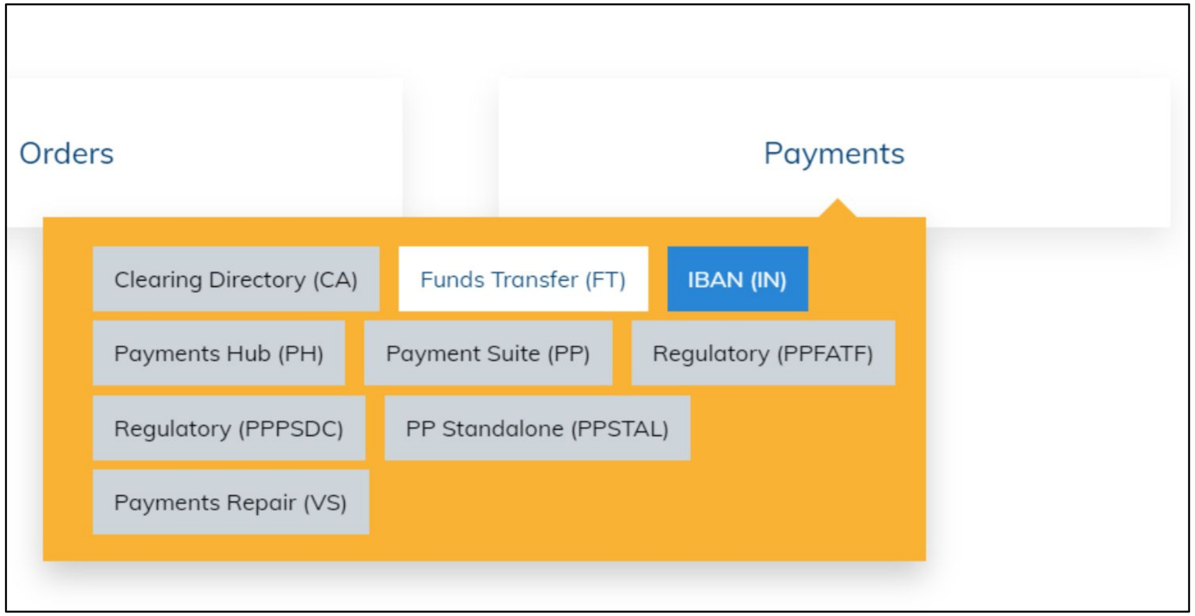
1. Extract documents from documentation

2. Generate an LDA model by training over the scripts

3. Map the documents to the scripts by using the model

1. Extracting documents

- We will attempt at mapping documents under payment documentation to payment scripts using LDA.
- The documents corresponding to different functionality are extracted as follows:
  1. Under the payments family, we select the options we have access to, which are namely, IBAN and Funds Transfer.



2. Under IBAN we have 6 documents regarding different functionality. For each of the documents, it has 3 sections containing texts under the headings “**Introduction**", “**Configuration**", “**Working with**". All the texts from the 3 sections are appended together to form each document.

- NOTE: Funds Transfer only contains "**Introduction**". Therefore, the document Funds transfer" only contains the introduction.

2.Training model on the scripts

- Sticking to the language of text collections, we refer to the entities such as “documents", and “corpus", as:

  Corpus: PAYMENTS scripts.

  Documents: Scripts in the folder. (not to be confused with the extracted documents)

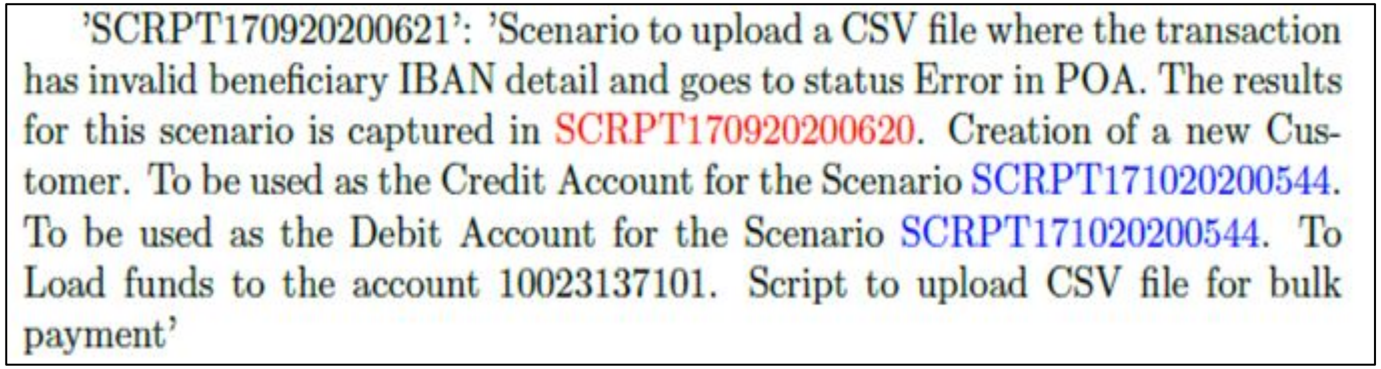
- The LDA model is constructed in python using the gensim library.

Preprocessing

- We cannot directly train over the documents(scripts), we must first convert the scripts into the suitable format.

1. **Reduction:** Each Script is a JSON file. We reduce the scripts to only contain the descriptions, field names and applicationid which are present in “sstObjectList” and "ssaObject".

2. **Catching direct reference**: There are scripts that directly refer to another in their description. These Scripts are dependent on each other. It is reasonable to combine these scripts to form a single document to represent them.



'SCRPT170920200621': 'Scenario to upload a CSV file where the transaction has invalid beneficiary IBAN detail and goes to status Error in POA. The results for this scenario is captured in SCRPT170920200620. Creation of a new Customer. To be used as the Credit Account for the Scenario SCRPT171020200544. To be used as the Debit Account for the Scenario SCRPT171020200544. To Load funds to the account 10023137101. Script to upload CSV file for bulk payment'

Preprocessing

3. **Bag of words format:** The documents are then converted to a bag of words, that is, a list containing words.

4. **Removing Stopwords:** The words that occur commonly in English literature contributes to noise since these occur in abundance. The topics have a multinomial distribution over the entire vocabulary. Therefore, it's important to reduce the size of the vocabulary to useful words. The words "create", "script“ and "scenario", also occur frequently across all the documents. These words were also removed.

Preprocessing

5. **Catching context using n-gram:** The model works on the principle of exchangeability. The downside to this is that the occurrence of collection of words in order(phrases) is considered the same as the occurrence of individual words throughout the document. Some valuable contextual information is lost this way. For practicality, it is sufficient to consider bi-grams and tri-grams.

- Gensim provides a method to identify the phrases and automatically adds underscore between the words.

Example: "New York" would be converted to "New_York", if it occurs frequently in the corpus.

Preprocessing

6. **Lemmatization:** To reduce the vocabulary even further, we remove inflectional endings of a word to return the base form of the word.

- After following the preprocessing step, we are ready to input the scripts for training the model.

Model Inputs

- The two main inputs to the LDA topic model are the dictionary (id2word) and the corpus.

1. **Dictionary:** After the prepossessing, we get the corpus in a list of list of words format. Gensim provides a method to create a dictionary by passing the corpus as the input. It creates an unique id for each word in the corpus.
   - Gensim also allows us to filter extreme words out of the dictionary.
   - We filtered the words which occurred in more than 60 percent of documents and, the words having a total count below 5. Our model will simply ignore these words when found in the documents.

Model Inputs

2. **Documents as vectors:** The model takes the corpus as the training input, but before feeding the corpus, the documents are to be converted to a vector form. For each word occurring in the document, a tuple of integers is associated. The tuple is obtained by denoting the first entry by the unique word id of the word, and the second entry as the count of total occurrence of the word in the given document. This way a document is converted into a vector containing tuples (word id, count). Gensim provides a method for the conversion.

Model Hyperparameters

- To create a model, along with the inputs, we require the following hyperparameters.
  1. The number of topics. The model assumes the specied number of topics beforehand.(k)
  2. The hyperparameters concerning the EM algorithm.
  3. Parameters for Dirichlet distributions(Dir(alpha) and Dir(eta)).

Parameters for Dirichlet distributions

- The parameters for Dirichlet priors can be set to “auto”. The model learns about “alpha”(document-topic) and “eta”(topic-word) automatically while training
- If symmetric parameters are desired, set “alpha”, ”eta” to symmetric values.

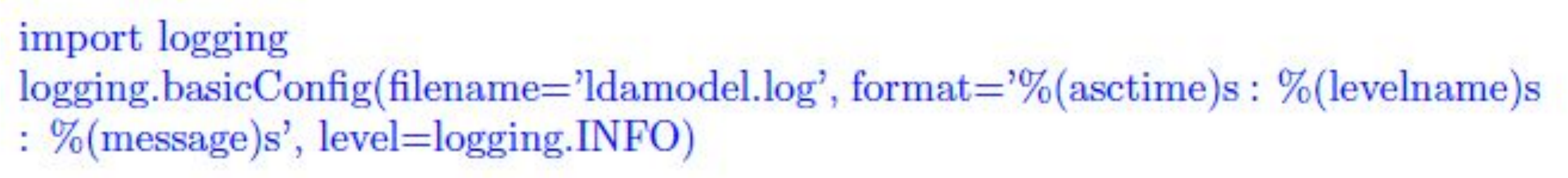
Parameters concerning the EM algorithm.

1. **Passes:** Number of passes through the corpus during training.
2. **Iterations:** puts a limit on how many times LDA will execute the E-Step for each document.
3. **Chunksize:** Number of documents to be used in each training chunk.
4. **update_every:** Number of chunks to process prior to moving onto the Mstep of EM.
5. **evaluate every:** For calculating perplexity after the number of updates. By setting it to "None" improves the runtime.

- We will keep the default values of all of the parameters, except the number of passes and iterations.
- The number of passes and iterations should be high enough. The algorithm will converge after a point and higher values will increase the running time significantly.

Parameters concerning the EM algorithm.

- By enabling logging, we found that the model converges for iteration=1000, around passes=120. After each pass, we get a topic diff score in the logs. The score measures the differences between each consecutive pass. The code for logging is given below.
- The number of topics don’t impact the convergence significantly. By enabling logging for different values of k we found similar results.



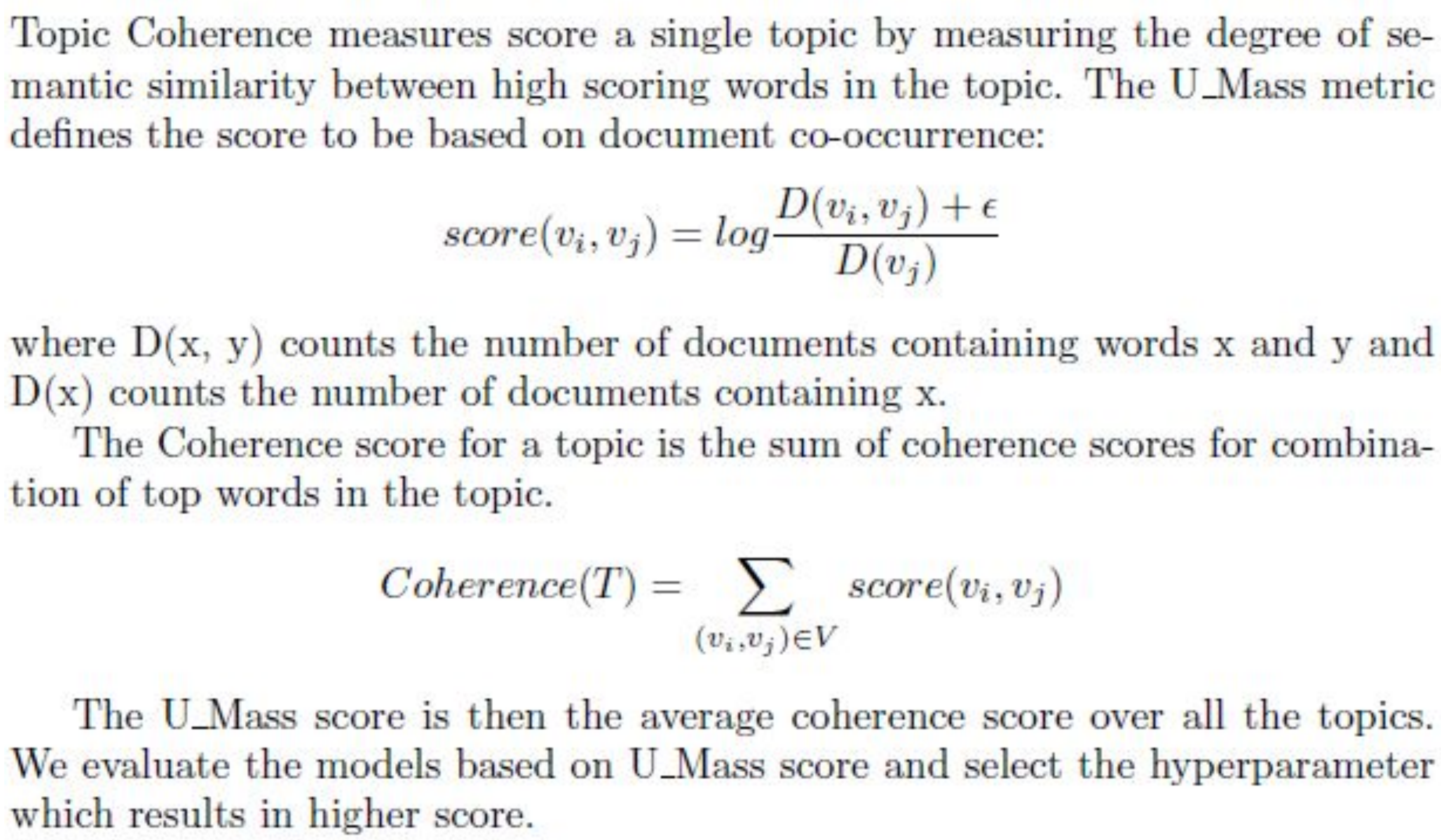
```
import logging
logging.basicConfig(filename=’ldamodel.log’, format=’%(asctime)s : %(levelname)s
: %(message)s’, level=logging.INFO)
```

The number of topics(k)

- To tune the value for k, we require a measure to evaluate the model. By generating models for different k values, we can select the parameters with a higher evaluation score.

- **Coherence Measure**: 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. An example of a coherent fact set is “the game is a team sport”, “the game is played with a ball”, “the game demands great physical efforts”. {game, sport, ball, team} is a coherent set in the context of sports. **U_mass** score is one such measure which measures the aggregate coherence across all topics.

U_Mass score



Topic Coherence measures score a single topic by measuring the degree of semantic similarity between high scoring words in the topic. The U_Mass metric defines the score to be based on document co-occurrence:

$$score(v_i, v_j) = log\frac{D(v_i, v_j) + \epsilon}{D(v_j)}$$

where D(x, y) counts the number of documents containing words x and y and D(x) counts the number of documents containing x.

The Coherence score for a topic is the sum of coherence scores for combination of top words in the topic.

$$Coherence(T) = \sum_{(v_i, v_j) \in V} score(v_i, v_j)$$

The U_Mass score is then the average coherence score over all the topics. We evaluate the models based on U_Mass score and select the hyperparameter which results in higher score.

Tuning the parameters

- First, we set a high value for number of iteration, iterations=1000.

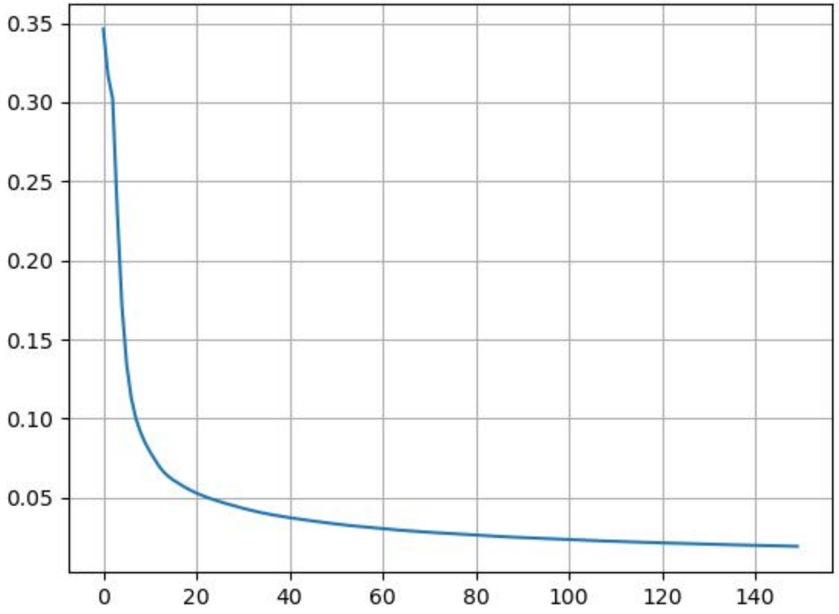
- For setting the number of passess, we create a topic model with arbitrary number of topics, for example, k=20.

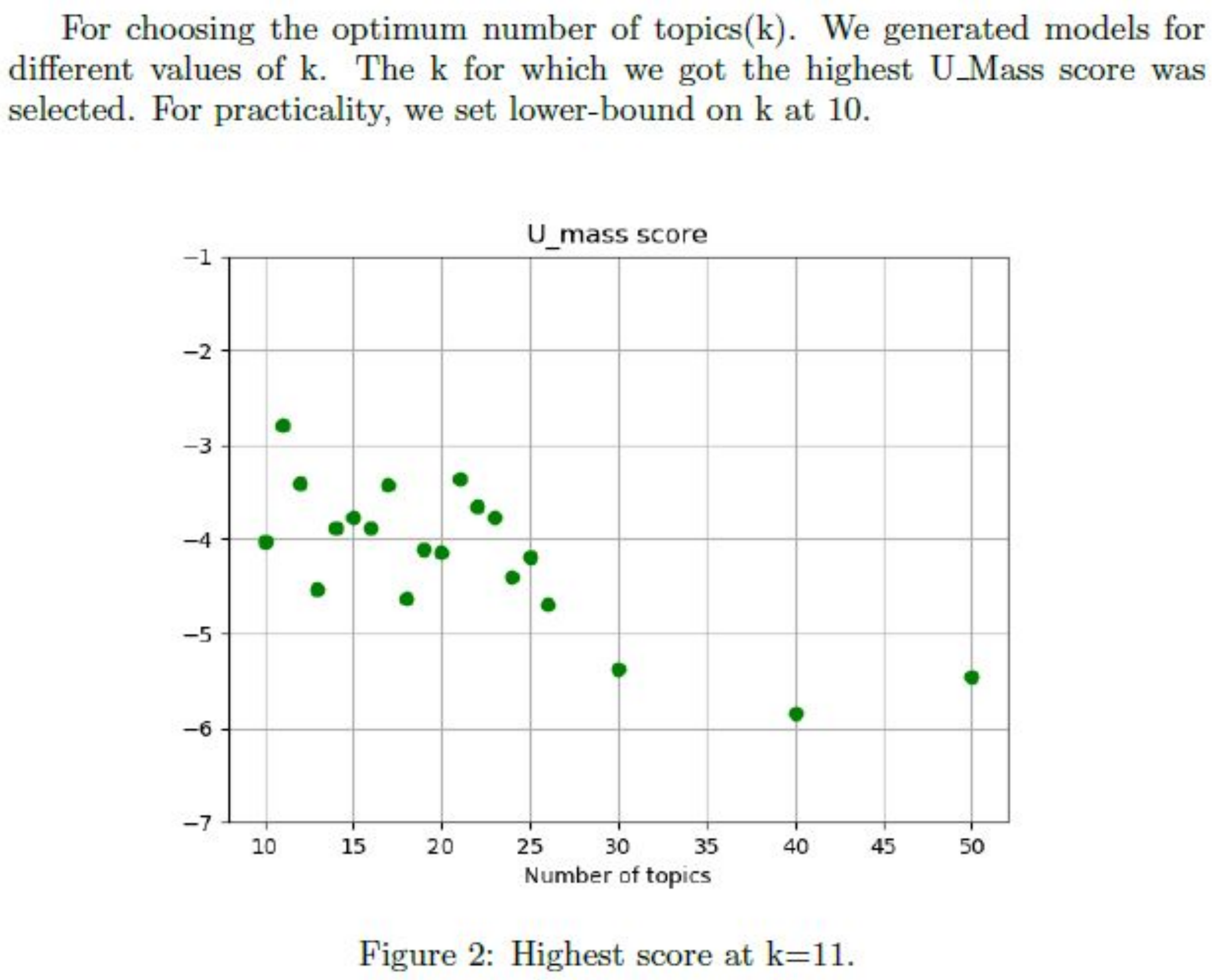
- Then we assign an arbitrary value for number of passes, passes=150.

```
lda = gensim.models.LdaModel(bag, id2word=dictionary, passes=150, num_topics=20, random_state=42, iterations=1000,alpha="auto",
eta="auto", eval_every=0)
```

Tuning the parameters

- After each pass, we get a topic diff score in the logs. The score measures the differences between each consecutive pass.
- By creating a plot for topic diff score, we can see that the model converges for number of passess =120.

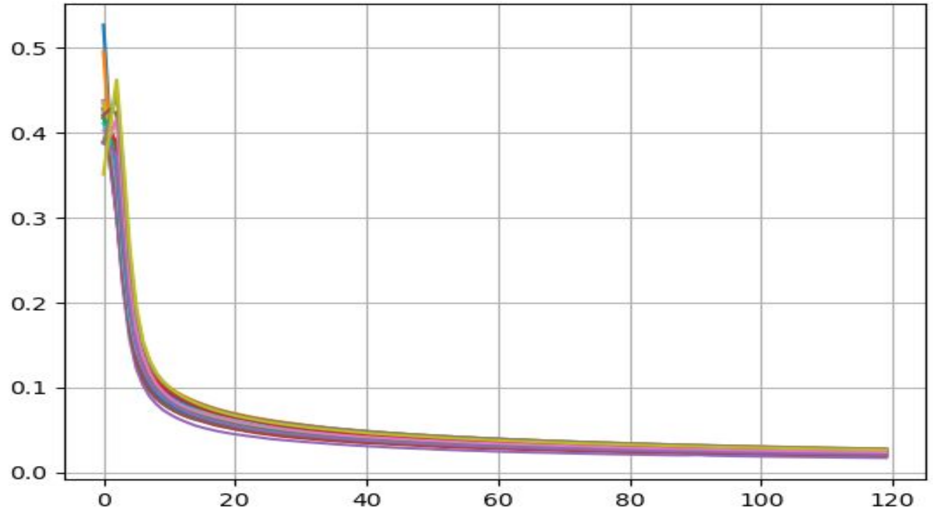




For choosing the optimum number of topics(k). We generated models for different values of k. The k for which we got the highest U_Mass score was selected. For practicality, we set lower-bound on k at 10.

Figure 2: Highest score at k=11.

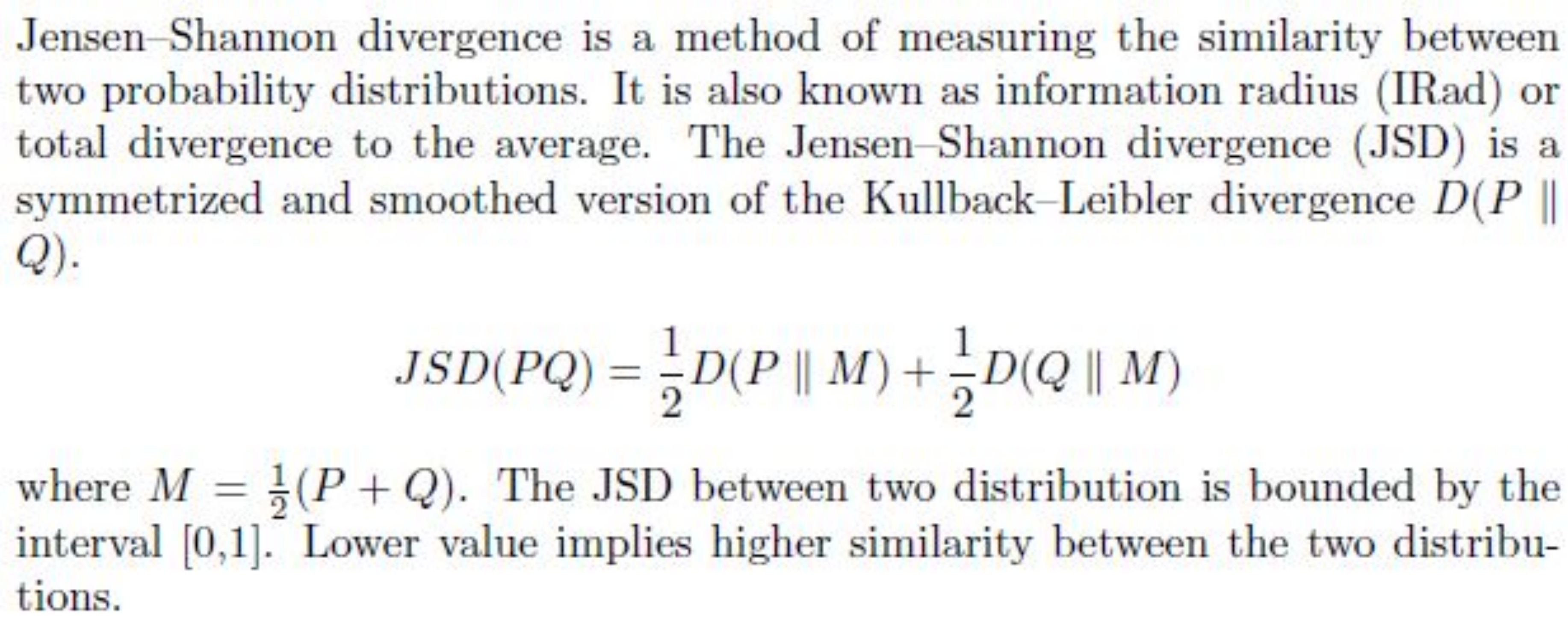
- To check whether, the topic models actually converged for passes =120 for different number of topics, we can use the log file to plot the topic diff over different values for number of topics.
- The convergence is not affected significantly with different number of topics.
- So, we will stick to first generating a model for an arbitrary number of topics to get the value for number of passes and then we will tune the number of topics using the umass scores.



3. Mapping documents to the script

- We will attempt to map the documents which were extracted based on different functionality to the scripts by using the topic model which was trained on the scripts. The scripts on which the model was trained, have a multinomial distribution over the latent topics.
- It is reasonable to assume that similar scripts have similar distributions. For each document, by using the trained model, we can get the multinomial distribution over the topics.
- The scripts which have higher similarity for distributions are mapped to the documents.

Similarity Measure



Jensen–Shannon divergence is a method of measuring the similarity between two probability distributions. It is also known as information radius (IRad) or total divergence to the average. The Jensen–Shannon divergence (JSD) is a symmetrized and smoothed version of the Kullback–Leibler divergence $D(P \parallel Q)$.

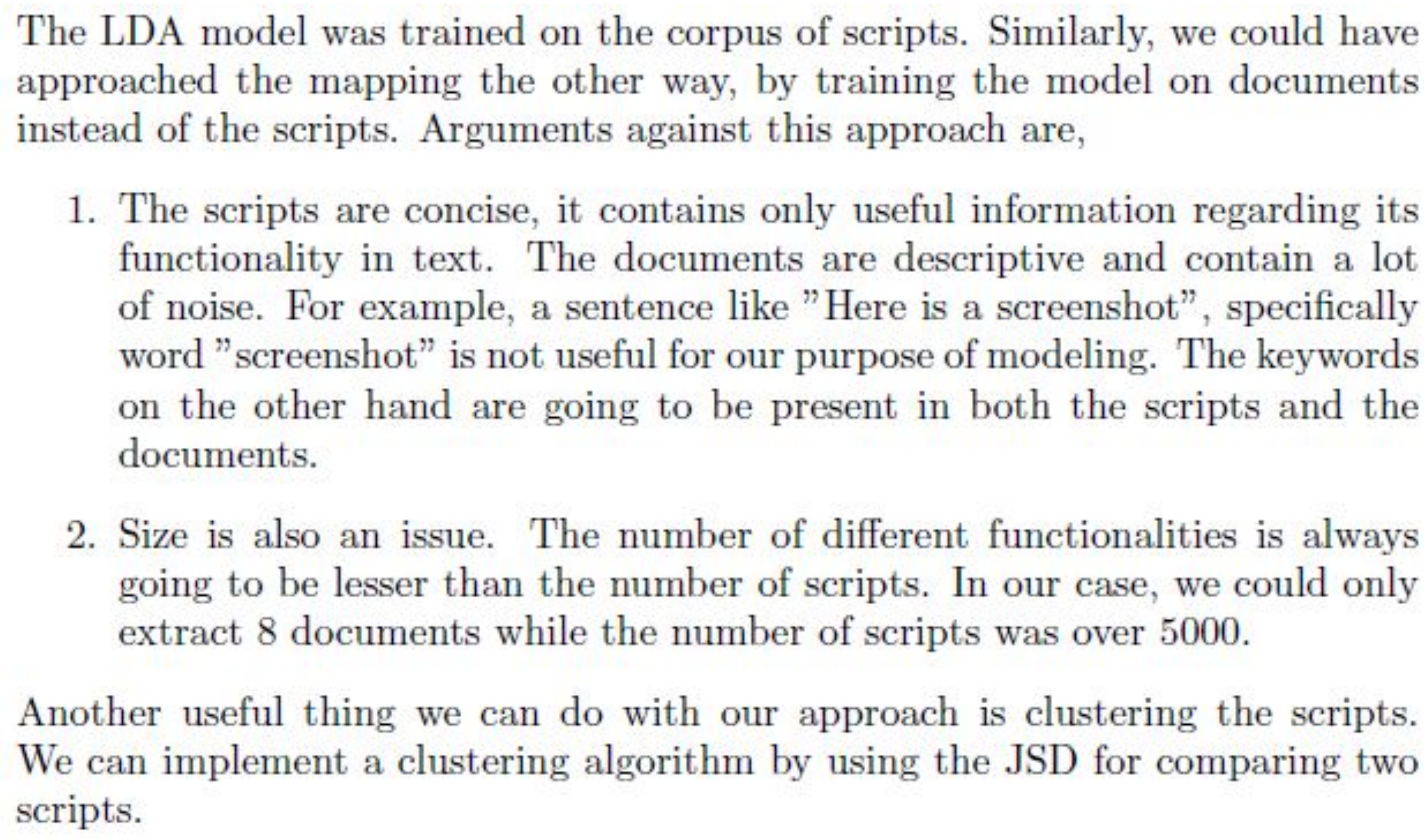
$$JSD(PQ) = \frac{1}{2}D(P \parallel M) + \frac{1}{2}D(Q \parallel M)$$

where $M = \frac{1}{2}(P+Q)$. The JSD between two distribution is bounded by the interval [0,1]. Lower value implies higher similarity between the two distributions.

Mapping Implementation

1. For each document and script, by using the topic model, we obtain the multinomial distributions over the topics.
2. We calculate the JSD between the documents and the scripts.
3. We set a threshold value. The scripts and the documents which have JSD value below the threshold are mapped with each other.

Discussion



The LDA model was trained on the corpus of scripts. Similarly, we could have approached the mapping the other way, by training the model on documents instead of the scripts. Arguments against this approach are,

1. The scripts are concise, it contains only useful information regarding its functionality in text. The documents are descriptive and contain a lot of noise. For example, a sentence like ”Here is a screenshot”, specifically word ”screenshot” is not useful for our purpose of modeling. The keywords on the other hand are going to be present in both the scripts and the documents.

2. Size is also an issue. The number of different functionalities is always going to be lesser than the number of scripts. In our case, we could only extract 8 documents while the number of scripts was over 5000.

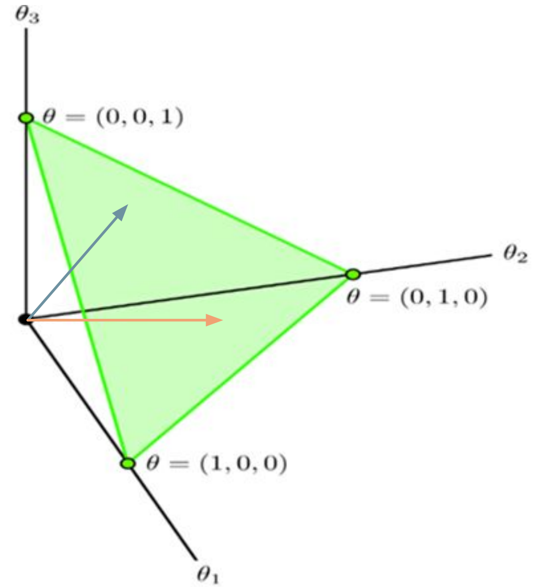
Another useful thing we can do with our approach is clustering the scripts. We can implement a clustering algorithm by using the JSD for comparing two scripts.

Discussion

Cosine Similarity:

Cosine similarity is another approach for measuring similarity. For each document\script, we have distribution over the topics. If we consider the distributions as vectors with dimensions equal to number of topics. We can measure the cosine of the angles between two vectors. The closer the script is to the document, the cosine approaches 1. Similarly, as we did with JSD, we can set a threshold value and the scripts are mapped to the documents for cosine value above the set threshold.

- Figure for distributions over 3 topics.



Discussion

Top words for documents:

The scripts are mapped to the documents based on the similarity between the topic distributions. By using the topic distribution of the document, we can then get the probability for a word to be generated by the document.

For this, we assume documents to be of length 1 word, and then by using the generative model, we get the joint probability of the document(word) and topic distribution by summing over all the topics. We then create a list of words in decreasing order of probability (word given topic distribution) to represent the top-words corresponding to mapping each document.

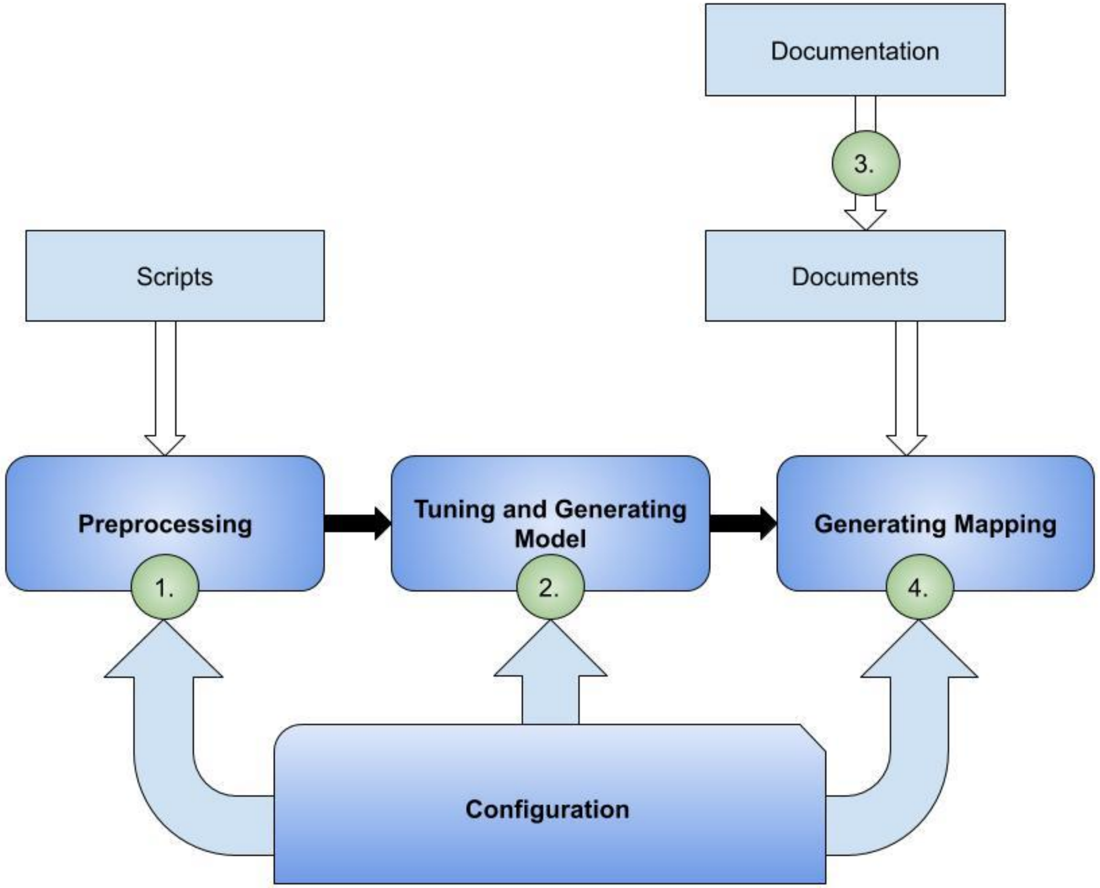
Conclusion

- Both JSD and cosine similarity perform well to capture the similar distributions.
- The functionality of the scripts were identified the following way:
  1. Different documents were extracted from documentation pertaining to different functionalities.
  2. LDA model was trained over the processed corpus of scripts.
  3. The model was used to generate topic distributions for the documents.
  4. The scripts and the documents were then compared based on similarity in their distributions.
  5. By setting a threshold, the scripts were mapped to documents having qualifying similarity scores.

Doc2Script

A product for mapping documents to scripts.

Architecture



Configuration

By abstracting the technical stuff away, the users are provided with few configuration options. These options are provided with default values as well. With these options, the users can aid the final mapping with their intuition.

Configuration

1. Reduction of scripts from JSON to txt: The scripts are converted from the JSON file to text.

   By default, the descriptions, field names, and applicationid which are present in 'sstObjectList' and "ssaObject" are selected. Users can specify if anything else should be added.

2. Filtering Dictionary extremes: The model uses a dictionary for training/mapping. The user can filter out the words from the dictionary that occur in total less than m times and in more than n percent of scripts.
3. Tendency to catch phrases: The model will catch more/fewer phrases based on the value given for this parameter.
4. Adding Stopwords: The user can add to the list of stopwords if they feel like certain words are not significant for mapping.

4. **Dirichlet Parameters:** The default values are set to auto (“alpha”=auto,”eta” =auto). The model will automatically learn asymmetrical “alpha” and “eta”(recommended). The user can also set symmetrical “alpha”, “eta”. With lower values, the model assumes that any given script/document is made up of fewer dominant topics(themes).

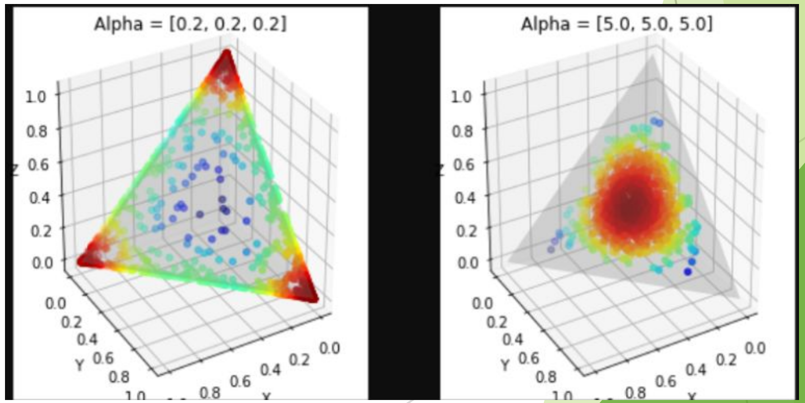


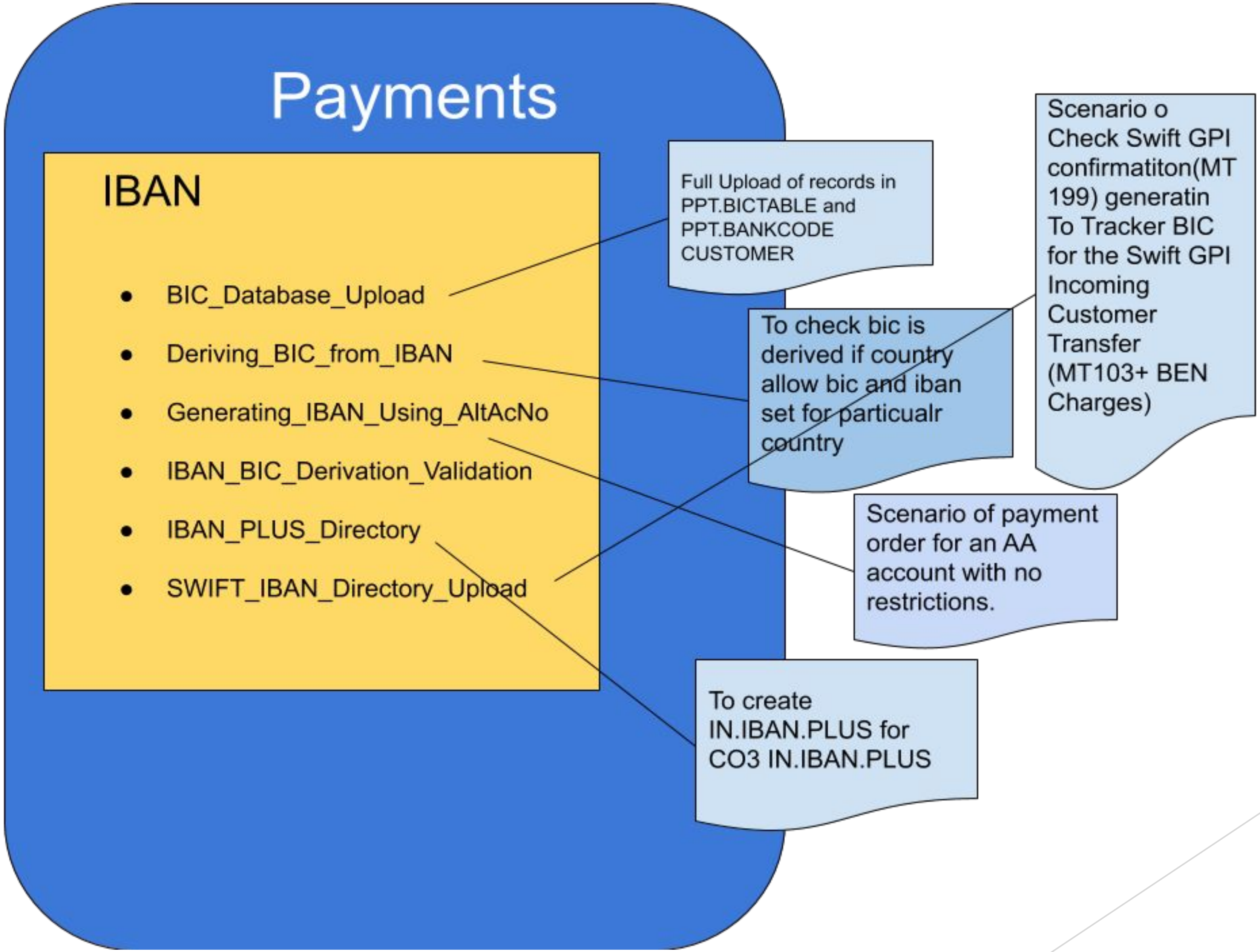
Figure: The corners represent the different topics for a 3-topic model.

5. Similarity measure used for mapping: We have two measures of similarity in distributions of scripts and documents over the topics. i) Cosine similarity ii) JSD.

- The user will be able to set threshold value (in the interval [0,1]). The scripts will get mapped to the documents for a qualifying similarity score.

Outputs

1. **Mappings:** For each document, the user will get a list of mapped scripts(which are labelled with scenario-id).
2. For each documents, the user will also get a list of words/phrases which contributed in the mapping.(topwords)



Payments

IBAN

- BIC_Database_Upload
- Deriving_BIC_from_IBAN
- Generating_IBAN_Using_AltAcNo
- IBAN_BIC_Derivation_Validation
- IBAN_PLUS_Directory
- SWIFT_IBAN_Directory_Upload

Full Upload of records in PPT.BICTABLE and PPT.BANKCODE CUSTOMER

To check bic is derived if country allow bic and iban set for particualr country

Scenario o Check Swift GPI confirmatiton(MT 199) generatin To Tracker BIC for the Swift GPI Incoming Customer Transfer (MT103+ BEN Charges)

Scenario of payment order for an AA account with no restrictions.

To create IN.IBAN.PLUS for CO3 IN.IBAN.PLUS

Thank You