Probabilistic Graphical Models

Project 2 - Latent Dirichlet Allocation

Mohammadjavad Matinkia - 96131043

1 Preliminaries

In this project we are going to implement the **Latent Dirichlet Allocation** algorithm from scratch and examine its parameters and performance on two different dataset. Latent Dirichlet Allocation is a probabilistic model used in *Natural Language Processing* problems. In simple words, this model considers random topics for the wrods of each document in a given corpus. These topics are unknown a priori but the number of the topics must be given as the hyperparameter of the model. The distribution of the topics for the words of the corpus is defined as a multinomial distribution with parameter θ and the distribution of the topics in the documents is also defined as a multinomial distribution with parameter ϕ.
Latent Dirichlet Allocation (LDA) considers **dirichlet** prior distribution with parameters α and β for θ and ϕ, respectively. The goal of LDA is to learn the parameters θ and ϕ from the available documents. For learning these parameters, there are two important methods: **Gibbs Sampling**, and **Variational Inference**. In this experiment we only implement the method of Gibbs sampling. For the first three parts of the project we use the first database which is the synthetic graphical dataset and for the fourth part we use the given attached dataset. It's worth to note that for the first three parts of the project we use 30 iterations to train the LDA model and examine its performance.

2 Implementation Details

In this project we have implemented two different Latent Dirichlet Allocation model. One is for the first graphical dataset and is inplemented in **new_lda.py** file and another is for the second dataset which was attached as the project dataset and is implemented in **utilities.py** file. There are minor differences between these two implementation, the first one is customized to work with the graphical dataset and the second one is customized to work on a corpus of documents of texts. For simplicity we explain the structure of the LDA in **utilities.py**. The LDA model receives a corpus variable which is an instance of the **Corpus** class, a value for α and a value for β. The number of titles is a property of the corpus variable. LDA model extracts the predefined number of titles from the corpus and assigns random titles to each word of each document in corpus. This is done by **assign_initial_topics_to_words** method. Then LDA

creates a matrix which maintains the number of words assigned to each topic. This is done by the **compute_word_topic_count** method and this matrix, builds the ϕ matrix. LDA also creates a matrix which holds the number of each topics in each document. This is done by **compute_document_topic_count** method and this matrix, builds the θ matrix. After creating initial matrices, the **train** method of the LDA is called. This is where the **Gibbs Sampling** occurs and the posterior distribution of z and the θ and ϕ matrices get updated. After a certain amount of iterations the model gets trained and one can get the θ and ϕ matrices using **get_theta** and **get_phi** methods.
The **Corpus** class which is defined before the **LDA** class is for preprocessing the corpus. This class gets the corpus and the predefined number of titles. The goal of this class is to extract documents, make a vocabulary out of documents, convert the text document to numerical arrays, in which the number assigned to each word is its corresponding index in the vocabulary, and finally deliver the structured documents to the LDA model.

3 Part 1

In this part, α and β have single values. We provide different values for α and β and measure the performance of LDA model in terms of likelihood (or perplexity) and the time consumed for learning. In Table(1) you can see these results.

α	β	**Perplexity / Likelihood**	**Execution Time(s)**
0.1	0.001	-135580.472	66
0.1	0.01	-129620.909	64
0.1	**1**	**-128810.115**	**69**
0.1	10	-141705.823	69
1	0.001	-138971.719	68
1	0.01	-139364.315	71
1	1	-154204.443	72
1	10	-152065.978	70
10	0.001	-166860.875	69
10	0.01	-167633.558	68
10	1	-173524.738	69
10	10	-190235.346	68

Table 1: The effect of α and β on the LDA model

From Table(1) we can conclude that the best performance is for the values $\alpha = 0.1$ and $\beta = 1$. Futhermore we can see that changing the values of α and β does not have a significant effect on the training time of the model.

4 Part 2

In this part we intend to examine the effect of the prescribed number of topics (the value of T) on the LDA model. To do so, we chose the number of topics from $\{5, 7, 9, 11, 13, 15\}$, and like before, we train the LDA model with 30 iterations The results of LDA are given in Table(2), and Figure(1). Figure(1) show the perplexity of the model for each number of topics.

Number of Topics	**Perplexity**	**Execution Time**
5	57411.685	67
7	88667.155	66
9	107322.932	69
11	130670.090	72
13	149156.922	71
15	167610.489	72

Table 2: The effect of number of topics on perplexity of the LDA model

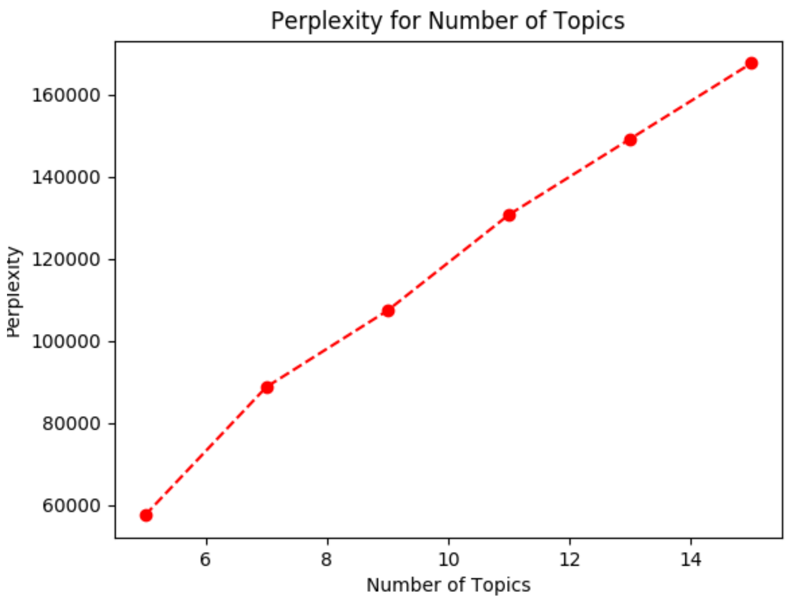

Figure 1: The effect of number of topics on perplexity of the LDA model

As it is clear from Table(2) and Figure(1), as the number of topics increases the perplexity of the model increases.

5 Part 3

In this part we are going to examine different kinds of sampling from the posterior distribution of the parameter z_i. In the first case we try the simple sampling in which at each iteration we choose only one sample from z. In the second case we try multiple sampling in which at each iteration we choose 5 samples from z.In the third case we try multiple sampling in which at each iteration we choose 10 samples from z. In the fourth case we generate 5 samples and we discard the first 4 sample and use the 10th sample. And at last in the final case we generate 10 samples and we discard the first 9 samples and use the 20th sample. The results of these different sampling types are presented in Table(3).

Samapling Description	**Likelihood Score**	**Execution Time**
Simple Sampling	-140237.605	70
Multisampling: 5 Samples	-135067.212	110
Multisampling: 10 Samples	-131385.763	113
Sequence Sampling: Choose the 5th Sample	-132345.987	111
Sequence Sampling: Choose the 10th Sample	-135012.453	111

Table 3: The effect of different sampling techniques on the performance of the LDA model

In all cases we track the model to find out when it reaches to the mixing state. Then, in the case of multisampling, five (or ten) times we sample from the data and generate the ϕ matrix, then produce a new ϕ matrix which the average of these 5 (or 10) ϕ matrices.
As we can see from Table(3), multisampling may increase the likelihood score of the model but, this increase is negligible in samll datasets, such as the one we used in this project, and also it is achieved with cost of increase in the execution time.

6 Part 4

In this part we use the second database which is a corpus containing 2248 documents. Initially we extract the documents and normalize each document so that there are no punctuations and no upper case letters in the docuement. Then we extract a vocabulary based on the all of the documents of the corpus. Then we assign to the words of the documents, a number which is equal to the index of the word in the vocabulary. Now that we have a numbered set of documents we can run LDA on them. Here we assume that the number of topics is 10 and the initial values for α and β are 1 and 0.1 respectively. We also assume that the number of iterations is 30.
After we trained the LDA model we obtain the θ and ϕ matrices, which we save them for further analysis. Recall that the θ matrix is a $D \times K$ matrix, in which D is the number of documents and K is the number of topics. $\theta_{i,j}$ is the distribution of the topic j in the document i. Now if we want to cluster the documents based on the θ, we consider the following: each row of θ is a vector corresponding to a document and portrays the distribution of the topics in the document. Using **K-Means** clustering algorithm we cluster

these vectors. So the hyperparameter we need to determine here is the number of clusters. These clusters represent the document which are closest in sense of common contents and topics. We use four different number of clusters as 5, 10, 15, and 20 clusters. In order to have a visualized representation of clusters we use the **Principal-Components Analysis** algorithm to reduce the dimensions of the vectors of θ to show the clusters.Figure(2), Figure(3), Figure(4) and Figure(5) shows these clusterings for different values of K (parameter of K-Mean clustering algorithm).

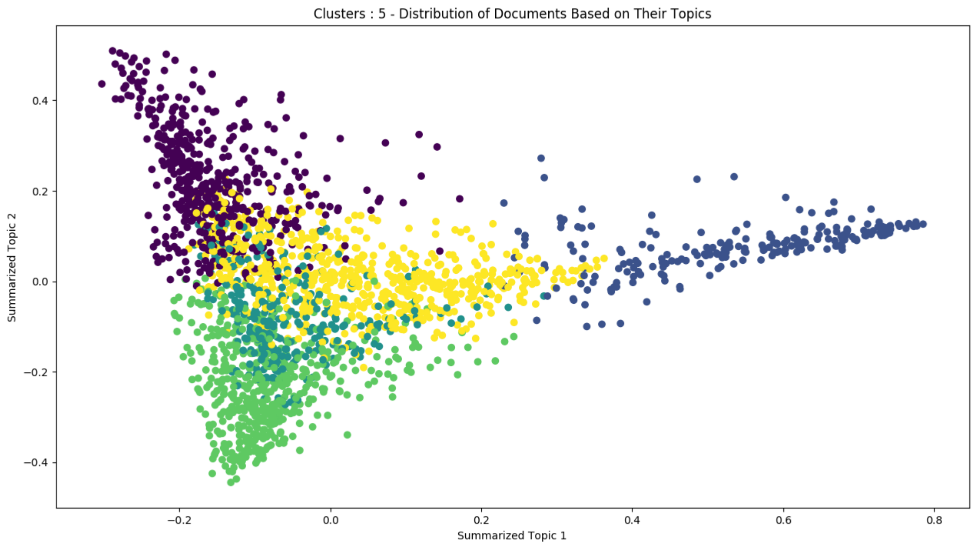

Figure 2: The demonstration of the clustering for $K = 5$. Note that the dimensionality is reduced from 10 to 2

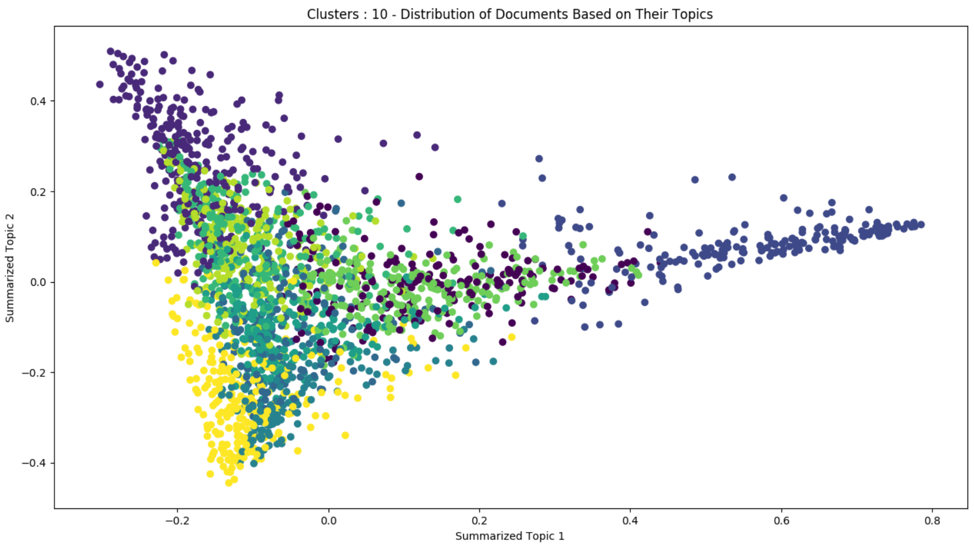

Figure 3: The demonstration of the clustering for $K = 10$. Note that the dimensionality is reduced from 10 to 2

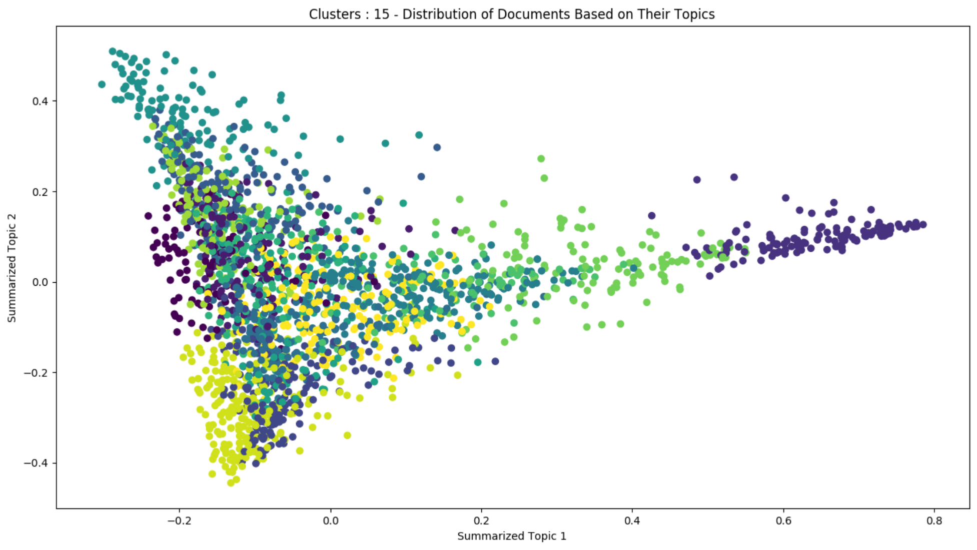

Figure 4: The demonstration of the clustering for $K = 15$. Note that the dimensionality is reduced from 10 to 2

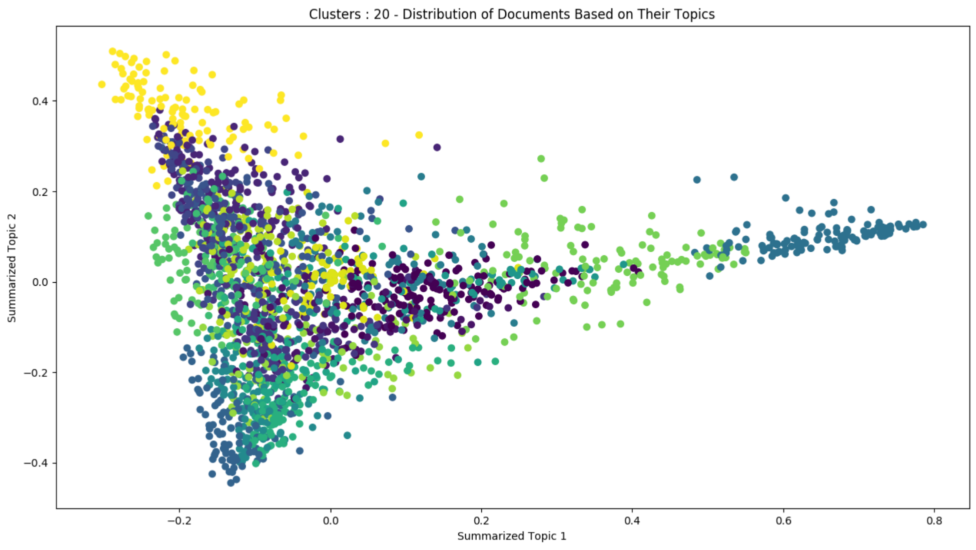

Figure 5: The demonstration of the clustering for $K = 20$. Note that the dimensionality is reduced from 10 to 2