Deliverable 2

March 1, 2023

1) **Problem Statement:** A latent Dirichlet allocation (LDA) unsupervised model is used to perform topic modelling on a set of about 10,000 research papers from the NeurIPS conferences. The model attempts to predict what each document is about. Additionally, a TF-IDF preprocessing step paired with k-means algorithm is applied to the same dataset. Finally, the aim is to compare which of the 2 models is better for performing topic modeling on this particular dataset and why.

2) **Data Preprocessing:**

   - Dataset: https://www.kaggle.com/code/richardnch/topic-modelling-on-neurips-papers/input. The dataset consists of almost 10,000 research papers submitted to the annual Conference on Neural Information Processing Systems (NeurIPS) between 1987 and 2019.

   - For preprocessing, I only dealt with the actual text of each document and ignored metadata (authors, id, titles, abstracts). I lowercased all the text and removed punctuation, numbers, and special characters to clean the text. Then, I performed tokenization where I turned each document's text into a list of individual words. I filtered out words whose length is shorter than 3 characters to reduce noise as these words are likely to be irrelevant to the main topics and ideas of the research paper. Next, I removed stop words. In addition to the usual stop words from NLTK library such as 'the', 'and', 'for', 'how'…, I added my own custom stop words that I noticed were frequent in this specific dataset to further reduce noise and keep only relevant features that contribute to the main topics. Finally, I lemmatized the text to turn each word in third person to first person and verbs in past and future tenses to present.

   N.B. I have not done this yet, but I am looking into making my own custom bigrams and trigrams as the NeurIPS papers are domain-specific and a lot of the terminology used comes in pairs or groups of 3 words to mean one thing (for example, machine learning, neural network, object-oriented programming are examples of bigrams and trigrams). The goal of introducing bigrams and trigrams is to potentially improve the results of the model by considering multiple words that often appear together as just one word.

   The bag-of-words representation of a document was made using gensim.corpora.Dictionary() and the doc2bow() method. We represent each document in the collection as a list of tuples where the first element of the tuple is an integer id corresponding to a specific word and the second element is the word's frequency inside the document.

The above is enough for LDA. For k-means, however, we must calculate, for each word in the collection, its TF-IDF score (See Theory Notes). I used sklearn to compute the TF-IDF score for each word.

3) **Machine Learning Model:** A Latent Dirichlet Allocation (LDA) is used. It is a type of unsupervised classification model. It considers each document as talking about various topics, and each topic also has various words belonging to it. The aim of LDA is to discover the topics a document belongs to, based on the words in it. The algorithm iteratively estimates the distribution of topics in documents and the distribution of words in topics until a steady state is reached. (See Theory Notes file for how the models work.) TF-IDF results in a high weight for words that appear frequently in a particular document but are rare in the overall corpus, and a low weight for words that are common across many documents in the corpus. Then, the k-means algorithm is used to calculate the closeness of two weighted TF-IDF vectors.

   A) I used the built-in LDA model provided by the gensim library in Python. For TF-IDF and k-means, I used sklearn.

   B) I split the data into 20% testing and 80% training to give the LDA model enough data to learn from. I experimented with the number of topics/clusters (which is a hyperparameter) and found that 10 topics is good enough. Choosing a larger number of topics results in overfitting and choosing fewer than 10 leads to underfitting. The output is a list of 10 topics with each topic having a list of top keywords and their associated probabilities (See Theory Notes for how these probabilities are calculated).

   C) **Validation Methods:** We can detect overfitting in the intertopic distance model if the topic clusters are very small, (indicating that a small number of the documents talk about this particular topic), and we can also detect underfitting on the map as well if we have very large circles/clusters indicating that a lot of documents belong to that specific topic. Another sign of overfitting we can see with the map is when clusters are close to each other indicating that the topics are conceptually similar and would be better off grouped together (i.e. we should decrease the number of topics).

   D) **Challenges Faced:** A persistent challenge is making sure the google colab notebook does not crash especially during preprocessing as the RAM is being used up very quickly.

4) **Evaluation metric:** Perplexity score is often used to evaluate LDA models. For some reason, my perplexity score is coming out negative, which should not be possible. I am still trying to find the bug(s) causing this because everything seems to work fine up until that point. The intertopic distance model is also considered an evaluation metric. To evaluate TF-IDF and k-means, the silhouette coefficient provided by sklearn metrics will be used.

5) **Next Steps:** I will look into creating my own custom bigrams and trigrams to improve the preprocessing and hopefully ameliorate the results. I will be doing some debugging to discover why my perplexity score is being returned as negative. (might be due to underflow and small probabilities being multiplied together… https://stats.stackexchange.com/questions/322809/inferring-the-number-of-topics-for-gensims-lda-perplexity-cm-aic-and-bic). I am also still trying to understand intuitively the meaning of a perplexity score. Also, in preprocessing, I will try to experiment with the idea of filtering out words that appear in more than 80% percent of documents and those that appear in very few documents (filtering extremes). Moreover, I am still looking into how to visualize k-means clusters and evaluate its accuracy.