**Capstone Project Submission**

| **Akash Choudhury –** akashchoudhury545@gmail.com |
| --- |
| * Data Wrangling * Loading and Preprocessing * Structuring Data * Enriching Data * Data Validation * Data Mining * Exploratory Data Analysis * Count Vectorizer * Stemming the data * Lemmatizing the data * Feature Extraction (Vectorization) * Model Development * Latent Dirichlet Allocation (LDA) * Summarization * Observations * Conclusions |
| **Link for GitHub and Drive as follow:** |
| GitHub Link:- https://github.com/akashbzs/Topic\_Modelling\_On\_BBC\_News  Drive Link:- https://drive.google.com/drive/folders/1O6lswfIRxXmnMnSDi7wLvQW7Q-uIMBtb?usp=sharing |
| **Summary for Topic Modelling on News Articles**:  [BBC](https://www.bbc.com/) stands for British Broadcasting Corporation.  It is an operational business division of the British Broadcasting Corporation (BBC) responsible for the gathering and broadcasting of news and current affairs in the UK and around the world. The department is the world's largest broadcast news organisation and generates about 120 hours of radio and television output each day, as well as online news coverage.  The **main objective** of the project is to identify major themes/topics across a collection of BBC news articles.  The dataset contains a set of news articles for each major segment consisting of business, entertainment, politics, sports and technology**.** There are over 2000 news article available in these categories.  Scikit-learn’s **Countvectorizer** is used to convert a collection of text documents to a vector of term/token counts. It also enables the pre-processing of text data prior to generating the vector representation. This functionality makes it a highly flexible feature representation module for text.  **NLTK** (Natural Language Toolkit) in python has a list of stop words stored in 16 different languages. Hence, we'll use it to extract our data without all such words. Going through the process of Stemming/Lemmatization of the data it was just an experimental, lemmatizer seems to be working fine, the words are segregated into a list. Now that all the necessary preprocessing is done, we'll implement the ML model for topic modelling on the BBC news articles.  **Conclusion:**   1. While reading the text files, we noticed that the file encoding was different in a few off-cases. We found that considering such factors, and engineering based on such knowledge, is very important while handling such data, in order to do so efficiently. 2. Upon experimenting with stemming and lemmatization on our dataset, we found that although it saves space and perhaps time, in our case, it's better to focus on quality, and avoid nuances. In our own 'cost-benefit' analysis, the difference weren't all that significant. Perhaps at a massive scale, the former approach would be ideal. 3. We noticed that it's more optimal to tokenize with no factual differences. So we lowercased the contents to unify tokens that may have just case-differences. 4. Best LDA model's params {'n\_components': 5}   Best log likelihood Score for the LDA model -643494.9704171557  LDA model Perplexity on train data 1696.6352006244963   1. Upon looking at the top n topics generated, we were able to correlate it with relevance to what was expected at a significant degree, whilst also shedding light on some unseen aspects.. Hence, we see that the model effectively beared fruit. |