CAPSTONE PROJECT ON:

TOPIC MODELING ON NEWS ARTICLES

Presented by:

Subhadip Ghosh

Journey Roadmap:

- Problem Statement
- Discussing our documents
- Dataset Inspection
- Text Preprocessing
- Vectorization of tokens
- Topic Modeling using LDA (Latent Dirichlet Allocation)
- Discussing tokens in each topic
- Discussing our LDA visualization chart
- Conclusion
- Plans to improve

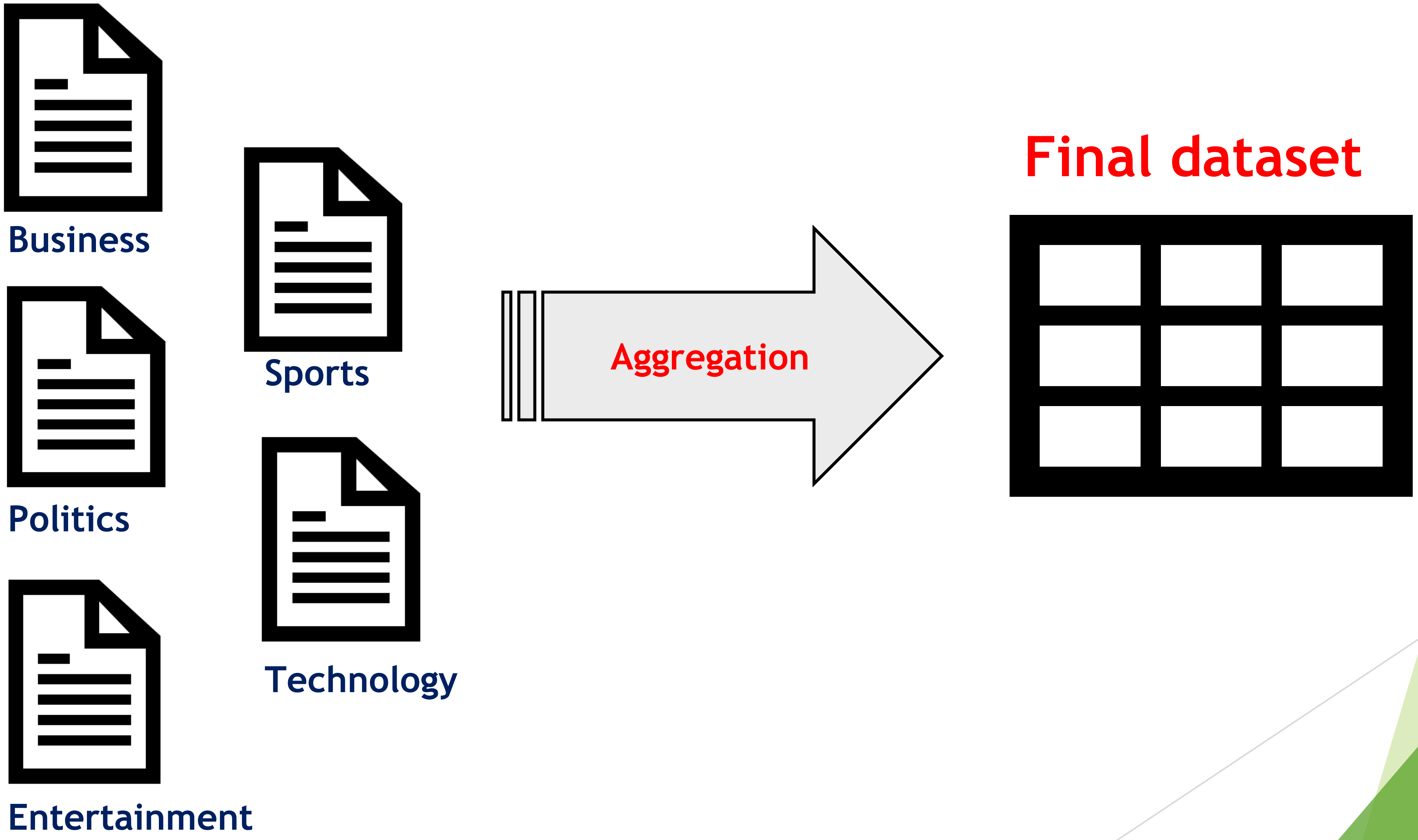
Problem Statement:

We are provided with a collection of articles corresponding to BBC News and the major topics/themes they have covered from 2004-2005

Objective of our model:

The objective of our model is to aggregate all these articles in one data frame and then identify these major topics/themes by clustering the appropriate terms that correctly correspond to the given themes.

Discussing our documents:



Data Inspection:

- We have got 2225 documents in total
- We only have one column which is ‘Articles’ corresponding to the news reports of BBC News
- After aggregating all the documents, we resampled the dataset to mix up the articles
- By visually inspecting the first few documents of our dataset, we found a lot of unwanted characters like ‘/n, &, %, #’
- We need to remove all these special characters and all the stop words to work on only those terms which will be crucial to our topic-modelling algorithm.

Text Pre-processing

- Tokenization
- Removing Stop-Words
- Lemmatization
- Checking document complexity

Tokenization:

- Tokenization is the process of turning the entire corpus into several ‘tokens’ or tiny pieces of terms which best describes the document when aggregated
- We used the split() method to tokenize our corpus and removed any unwanted characters and punctuations in our dataset
- We filtered out the crucial tokens by rejecting all the stopwords (is, and, the, are, they, etc…) by using the nltk library
- We used Lemmatization to convert these tokens into their base form in accordance to the English dictionary and these modified terms are called ‘lemma’.

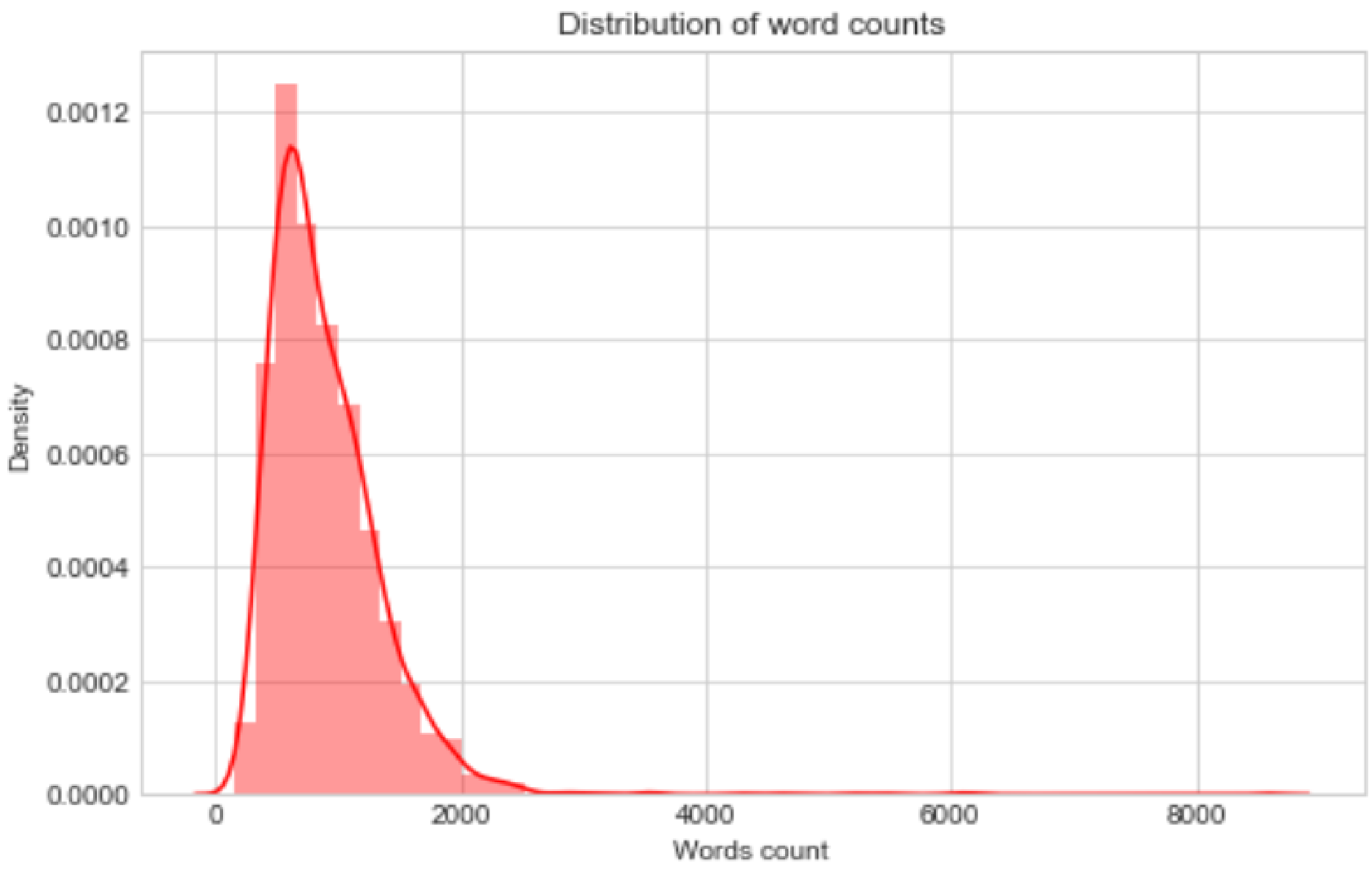
  Example:

  ‘neighbouring, neighbourhood, neighbours' turns into ‘neighbour’
- Used Text blob to extract only the noun phrases from our corpus

Checking document complexity:

Complexity can be calculated for each document by:

- Calculating the number of characters in each document
- We will be looking into the distribution of character counts in our entire corpus



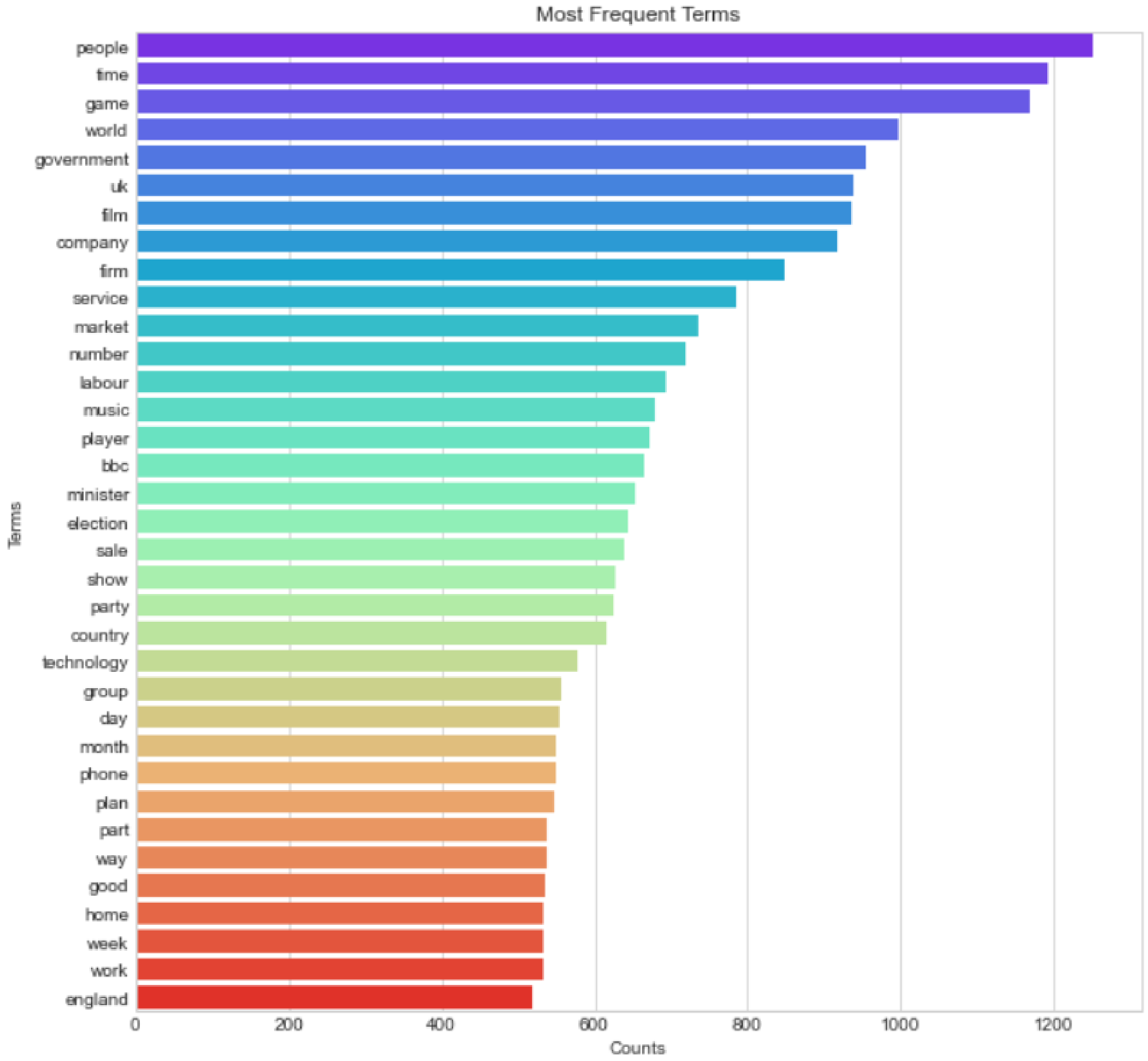
Vectorization:

- Term vectorization
- Checking the frequency of each token
- Updating Stop-words and vectorizing using TF-IDF

Term-Vectorization:

- Unfortunately, our machine doesn't understand raw text data. These terms need to be converted into vectors for our machine to understand
- Term vectorization is the process of converting each and every token from the vocabulary into a corresponding vector of real numbers
- We will be using Count-Vectorizer to check the terms that occur most frequently in our corpus
- We will use the term frequency to update the stop-words and then apply TF-IDF for vectorization of the tokens

Most frequent terms:




```
[('mr', 2874),
 ('year', 2198),
 ('new', 1793),
 ('people', 1252),
 ('time', 1192),
 ('game', 1170),
 ('world', 996),
 ('government', 954),
 ('uk', 938),
 ('film', 936),
 ('company', 918),
 ('bn', 871),
 ('firm', 848),
 ('service', 785),
 ('market', 736),
 ('number', 718),
 ('labour', 693),
 ('music', 678),
 ('player', 672),
```

Updating Stopwords:

- In the previous section we checked the most frequently occurring words in our entire corpus
- The top three words “new, year, mr” won’t be of much help to us in topic modelling as these three words doesn't correspond to any such major theme/topic that have been covered by BBC
- We have excluded all the words that have a frequency of more than 1500 and less than 4
- We added some more tokens in the stopwords list such as ‘abc, eu, th, aaa…etc’ as these tokens will also be of no help to us
- We will now use Tfidf-Vectorizer with these updated stopwords

Using TF-IDF Vectorizer:

Parameters used in TF-IDF Vectorizer:

- **ngram_range:** tells the model the range of word lengths it needs to consider while vectorization. We have used (1,2), meaning maximum two words can be collectively selected by the model
- **min_df:** Minimum document frequency. We have used 0.003, meaning any term present in less than 0.3 percent of the documents will not be considered
- **max_df:** Maximum document frequency. Used 0.60, meaning any term present in greater than 60 percent of the documents will not be considered
- After vectorizing using TF-IDF, we have a sparse matrix having shape (2215, 5647), meaning we have 2215 documents and 5647 tokens in the form of vectors

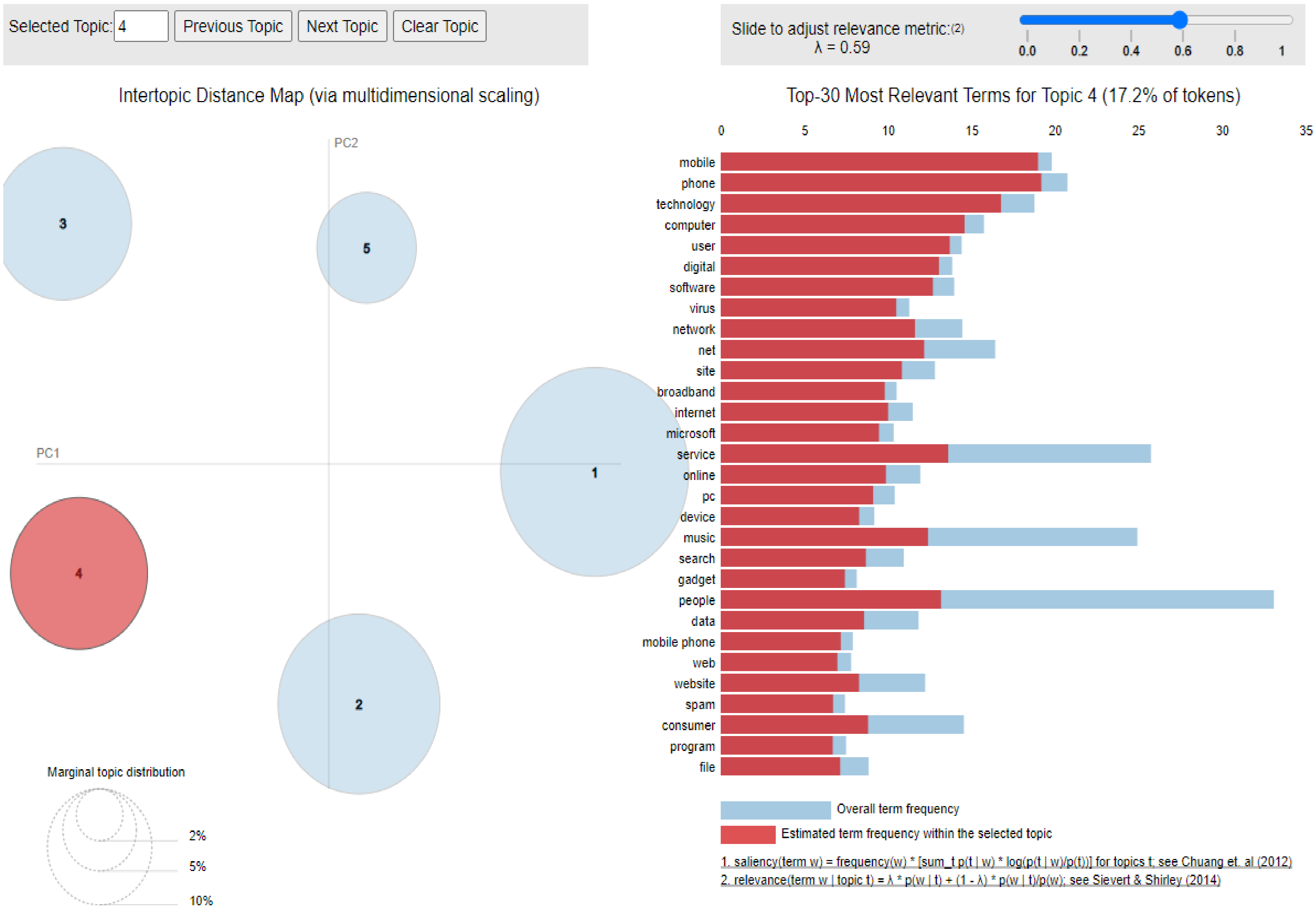
Fitting our Model

- Libraries we will be using
- Fitting our model using LDA
- Checking out the topics
- Discussing LDA visualisation chart

Libraries we will be using and why:

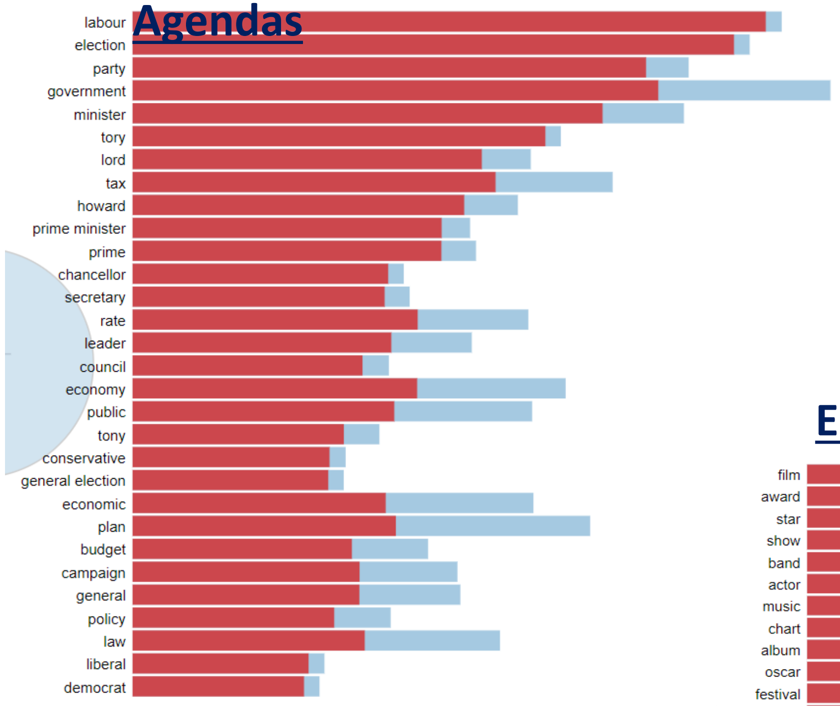
- Importing pyLDAvis from sklearn
- Importing T-SNE (t-distributed stochastic neighbor embedding)
- Importing Latent Dirichlet Allocation

Topic Modeling using LDA

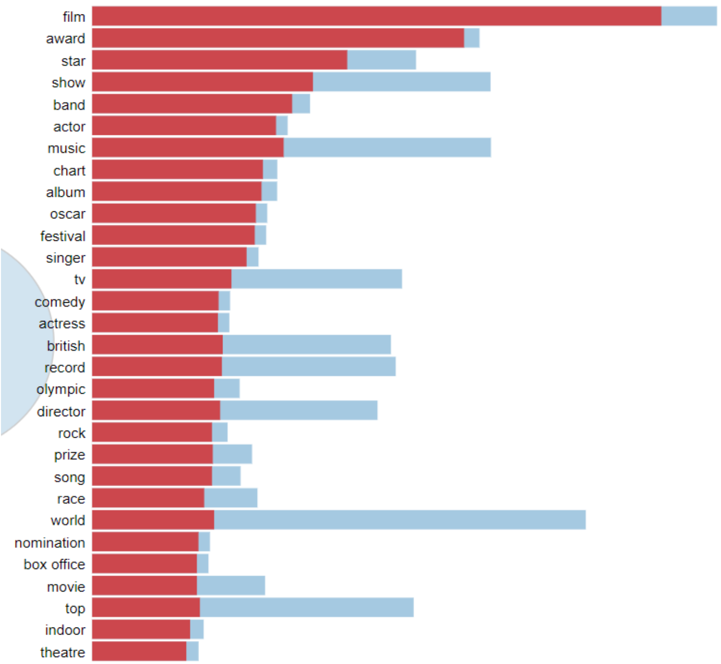


Terms in each topics:

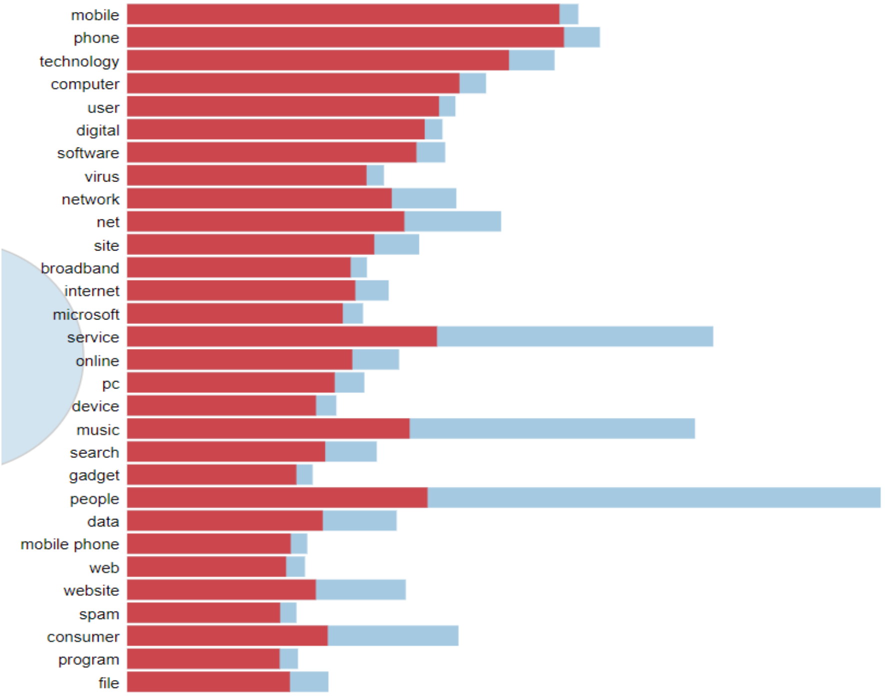
Political Agendas



Entertainment

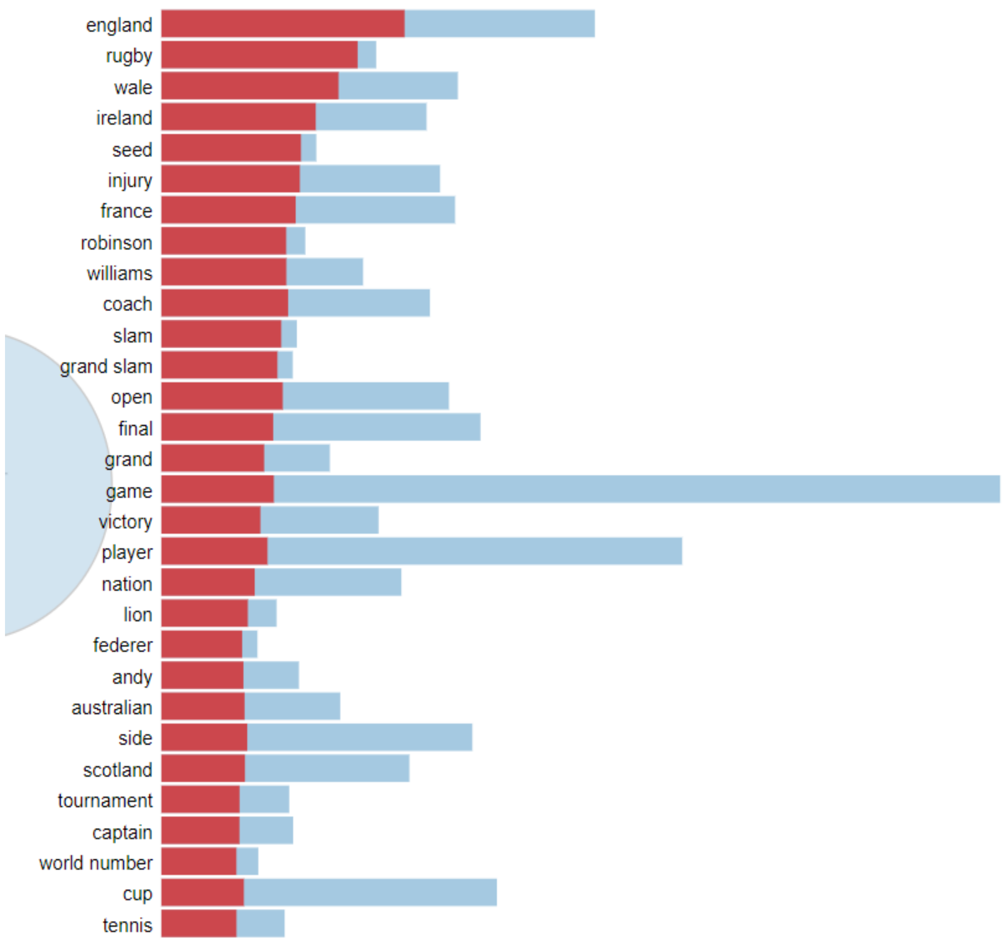


Technology

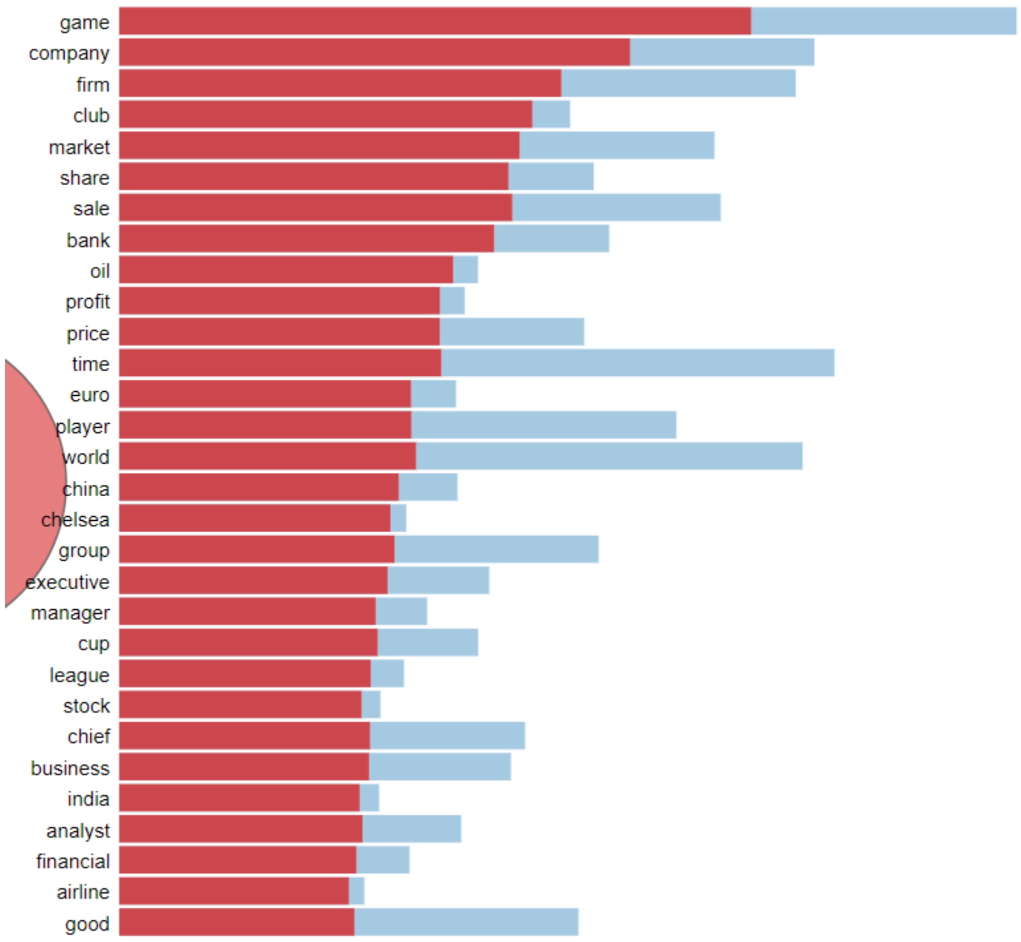


Terms in each topic(contd.)

Sports



Economy



Discussing the LDA visualisation chart:

- The white bars define the overall term frequency of a particular token
- The red bars define the estimated term frequency of a particular token
- In each topic, the top 30 most relevant tokens have been used for visualization
- The relevance metric (lambda) when equal to 1, the terms are arranged according to their probability in a particular topic. Lambda when equal to zero, the tokens get arranged in the decreasing order of their respective lifts.

Lift: ratio of a term's probability to it's marginal probability in the entire corpus

- We have used T-SNE as a dimensionality reduction technique to visualise the clusters form in a 2-dimensional feature space

Conclusions:

- Out of the five major themes/topics that had been provided to us, our LDA (Latent Dirichlet Allocation) has correctly clustered most of the terms corresponding to a topic

- Business and Political articles constitute about 56% of the overall tokens and the rest three topics correspond to the remaining 44%

- Our LDA model works best at relevance level 0.6, correctly classifying almost 85% of the tokens in their respective clusters

- Adjusted the hyper-parameters doc_topic_prior and topic_word_prior, but for 5 topics, both corresponding to None gave the best results

Plans to improve:

- Most of the tokens from articles “Sports” have been removed either in the pre-processing stages or while updating the stop-words

- We could have use different parameters for TF-IDF for capturing more number of tokens by reducing the min_df even more. But due to higher computational powers required for a machine to generate a sparse matrix with more than 20,000 tokens, we were unable to test that out.