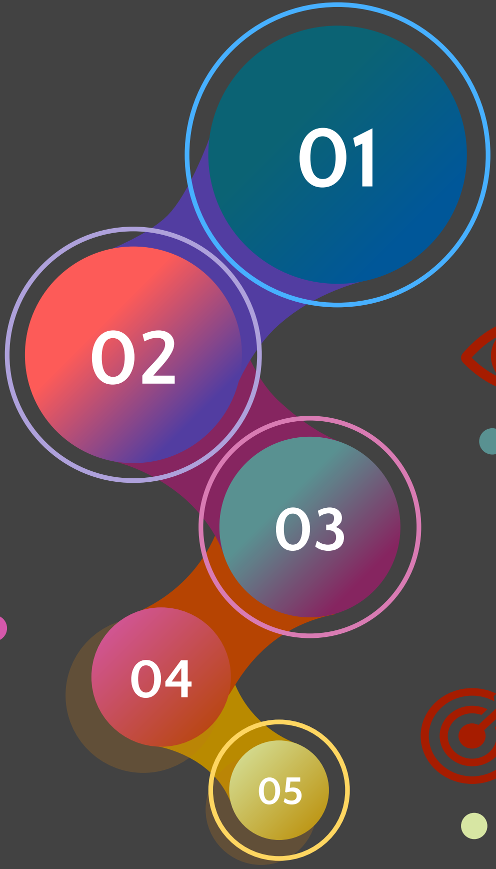
Capstone Project

Unsupervised ML (Topic Modeling) On BBC News Articles

ROADMAP TO PRESENTATION



Introduction

A brief note about the project.

Data Cleaning

Loading, treating and preparing the data to be model-friendly.

EDA and Feature Extraction

Analysing and transforming data into features for unsupervised learning.

Model Development and Visualization.

Implementation, tuning the model, and visualizing the results.

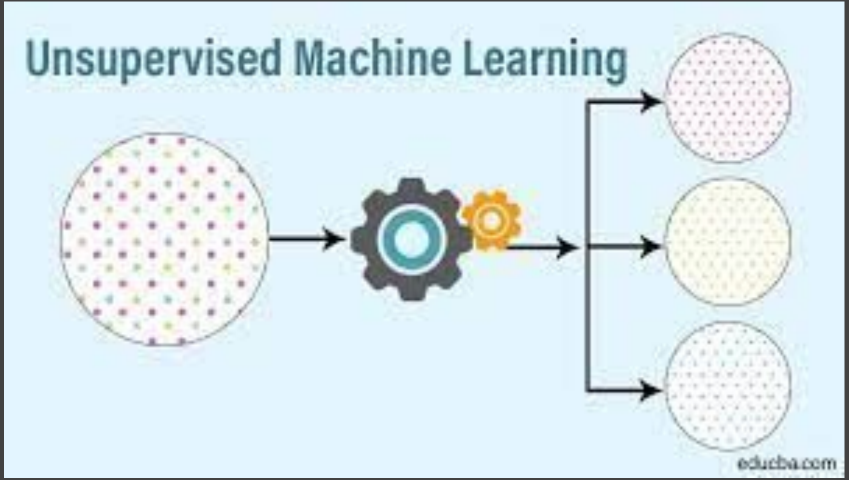
Conclusion and References

Unsupervised ML

Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention.

A few popular unsupervised ML algorithms:

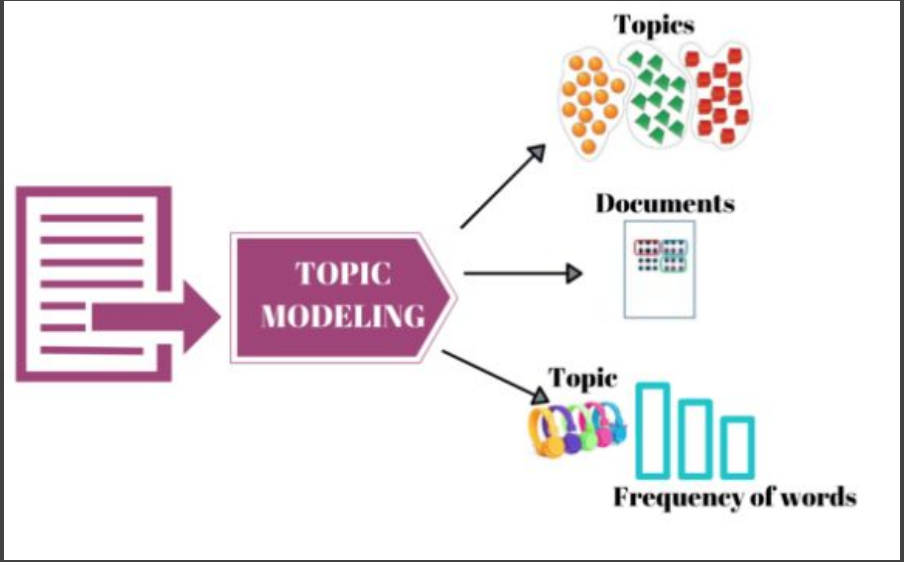
- **K-means clustering**
- **KNN (k-nearest neighbors)**
- **Hierarchical clustering**
- **Anomaly detection**
- **Neural Networks**



Topic Modeling

Topic Modeling is an unsupervised ML approach that is useful in segmenting documents into relevant topics based on frequencies of words.

In this project, we will be using an algorithm called **LDA (latent dirichlet allocation)** to perform topic modeling on BBC news articles



About the Project/Dataset

BBC which is short for the **British Broadcasting Corporation**, is primarily known for their efforts in the gathering and broadcasting of news and current affairs in the **UK** and around the world.

BBC News Online is BBC's news website. It is one of the most popular news websites in the UK, reaching **over a quarter of the UK's internet users**, and worldwide, with around **14 million** global readers every month.

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.

Objective

To implement Topic Modelling using LDA, and identify major themes/topics across a collection of BBC news articles, and to group the articles accordingly.

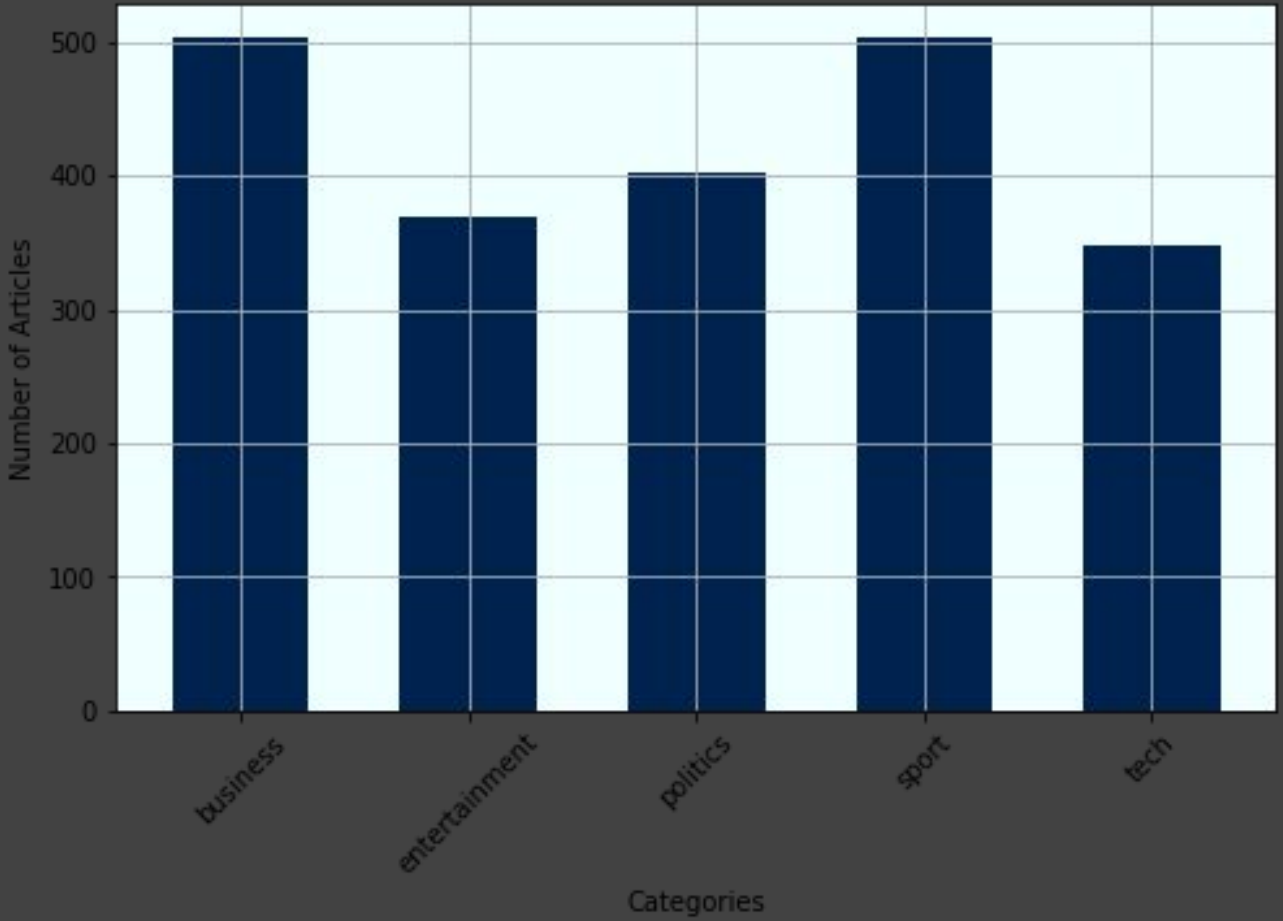
Data Cleaning

Data cleaning

The dataset in this case,is an **aggregation** of numerous **text files** that are **sub-categorized** into 5 different domains. Hence, for loading the data we iterated over all the text files, and copied the contents to a dataframe along with their category.

There was a particular file in this dataset which was formatted differently from the rest of the files, hence while reading the data, the text-formatting related exception was thrown. However, since this was apparently just a one-time occurrence, we disregarded it.

Distribution of the Data



Feature Extraction

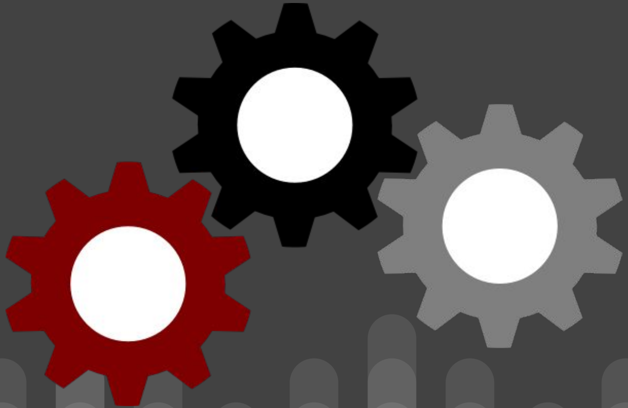
In order to use textual data for predictive modeling, the text must be **parsed** to remove certain stopwords.

These words then need to be encoded as integers, or floating-point values, such that they can be used as inputs in machine learning algorithms. This process is called **Feature Extraction (or Vectorization).**

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.

Model Development



Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) is one of most popular **topic modeling** technique to extract topics from a given corpus. The term latent conveys something that exists but is not yet developed. In other words, latent means hidden or concealed. The Dirichlet model d**escribes the pattern of the words that are repeating together, occurring frequently, and these words are similar to each other.**

And this **stochastic** process uses **Bayesian inferences** for explaining “the prior knowledge about the distribution of random variables”. In the case of topic modeling, the process helps in estimating what are the chances of the words, which are spread over the document, will occur again?
This enables the model to build data points, estimate probabilities, that’s why LDA is a breed of **generative probabilistic model.**

Latent Dirichlet Allocation (LDA)

LDA generates probabilities for the words using which the **topics are formed** and eventually the **topics are classified into documents**.

LDA makes two key assumptions:

1. **Documents are a mixture of topics**
2. **Topics are a mixture of tokens (or words)**

The end goal of LDA is to find the most optimal representation of the **Document-Topic** matrix and the **Topic-Word** matrix to find the most optimized **Document-Topic distribution** and **Topic-Word distribution.**

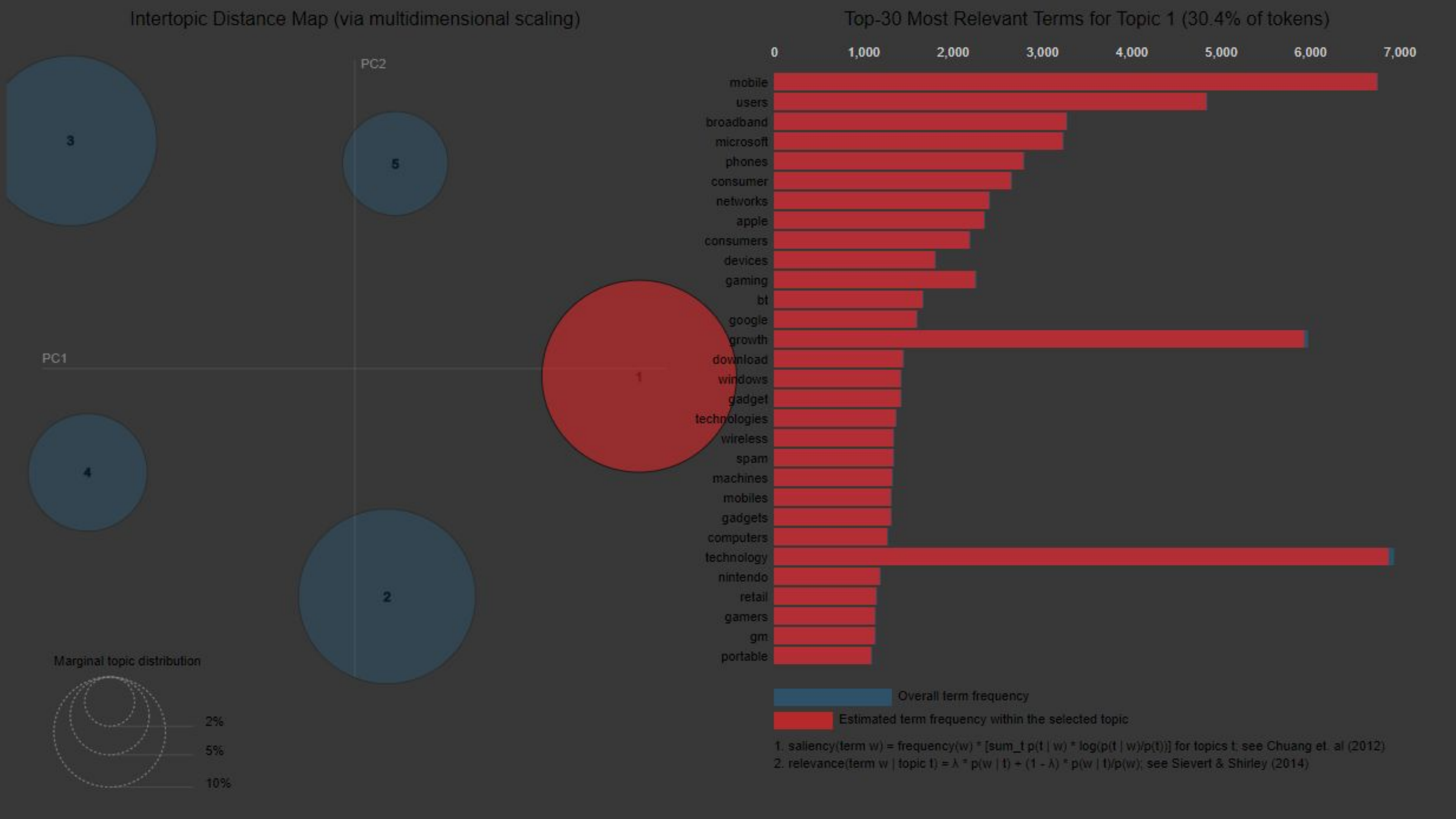
As LDA assumes that documents are a mixture of topics and topics are a mixture of words so LDA backtracks from the document level to identify which topics would have generated these documents and which words would have generated those topics

Visualization

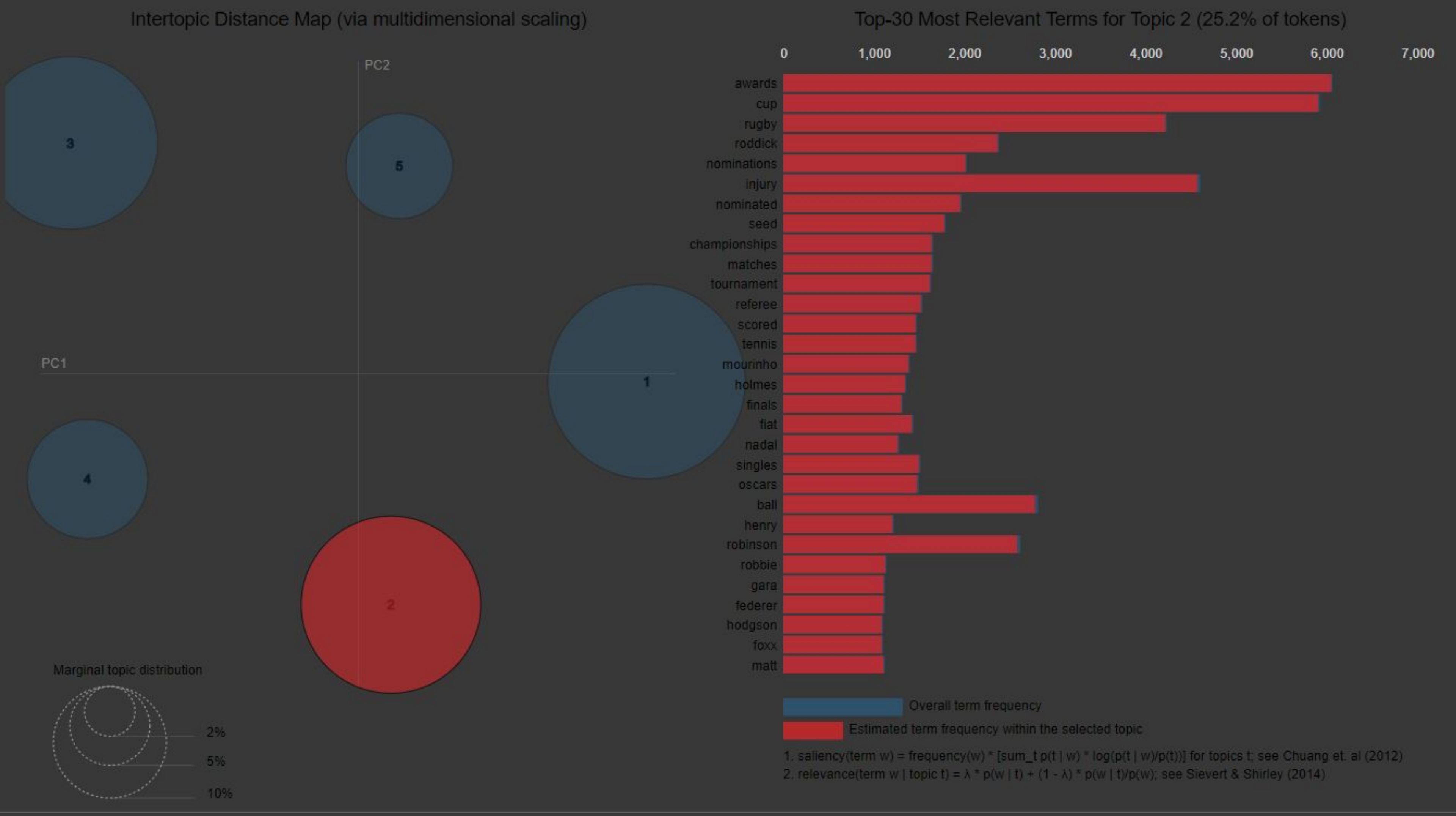
Visualizations using pyLDAvis

pyLDAvis is a python package that allows us to visualize our LDA model, and helps us to get a better understanding of the model. This will show us the topics found, categories and how they're distributed based on preferences.

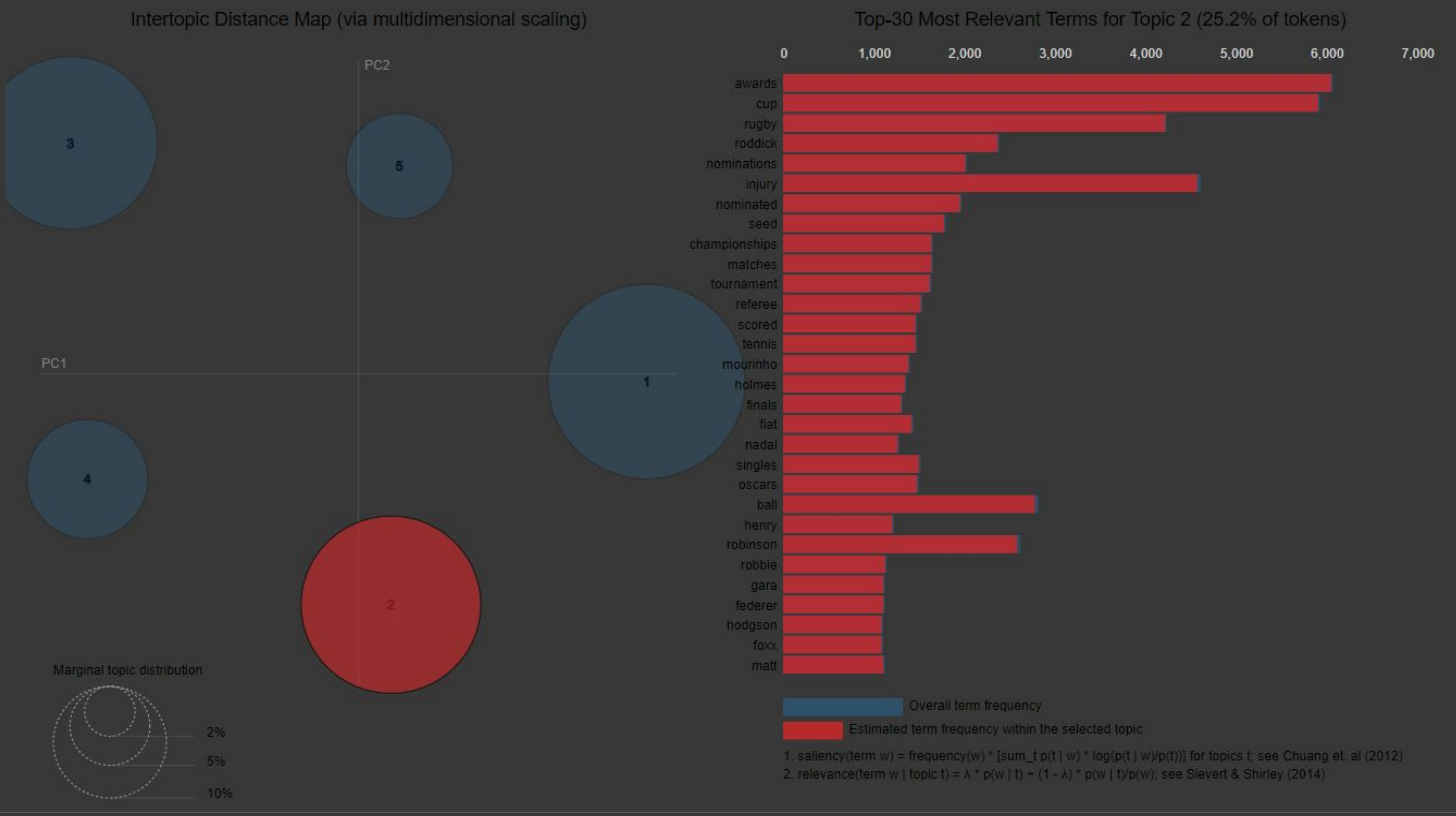
We shall be looking at pyLDAvis Visualizations for the 5 Topics identified by the LDA model, in the following slides



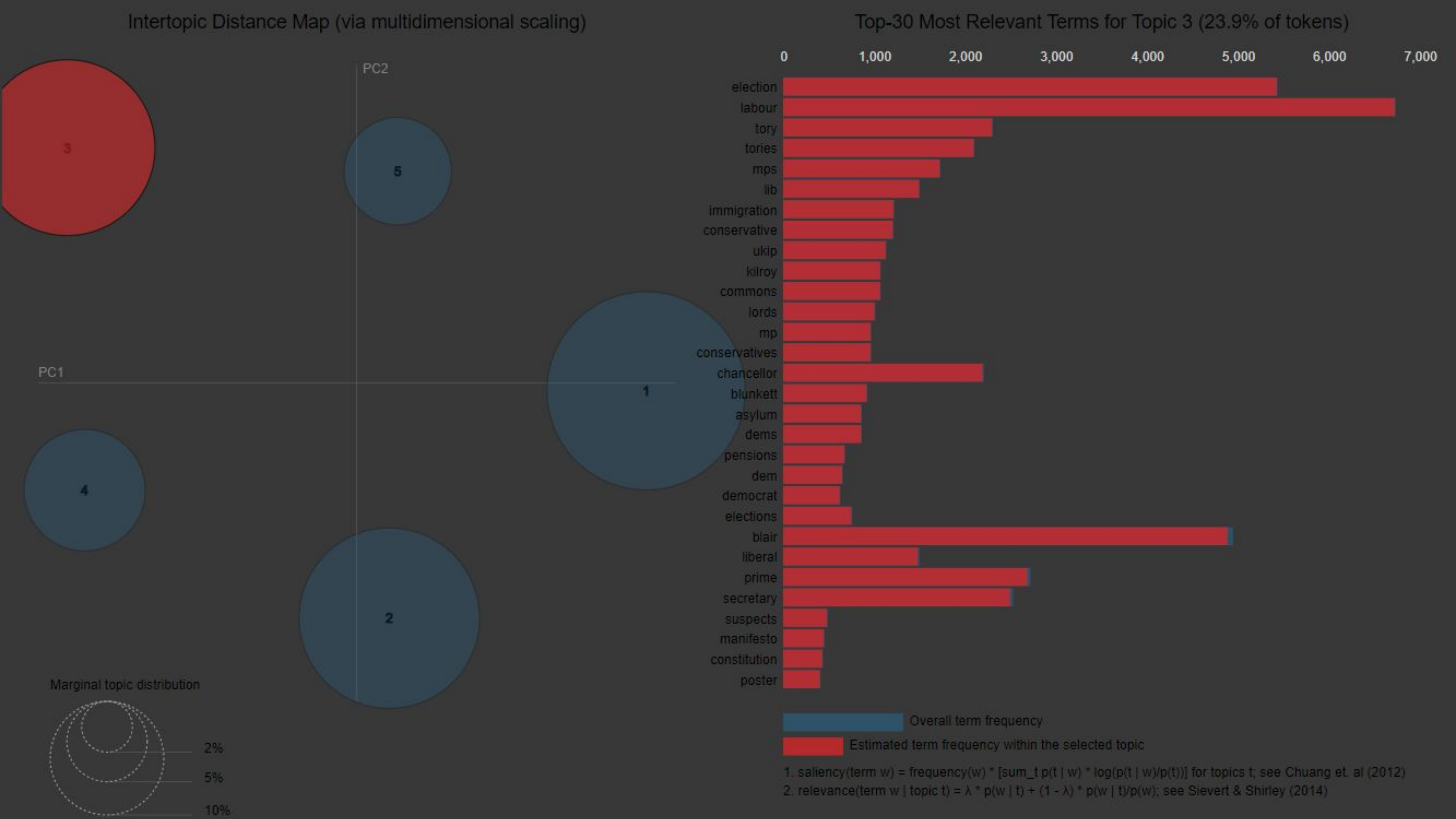
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)



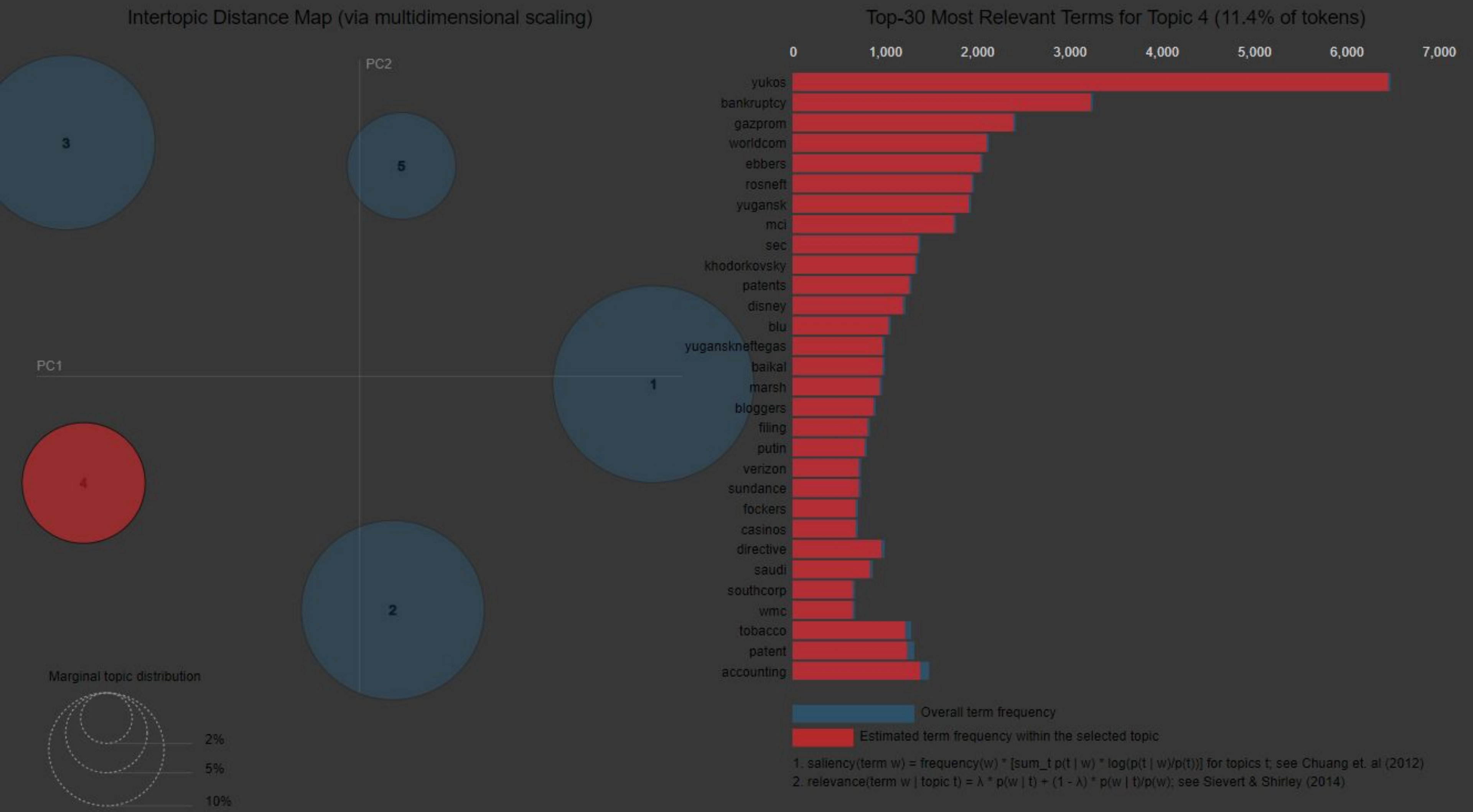
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)



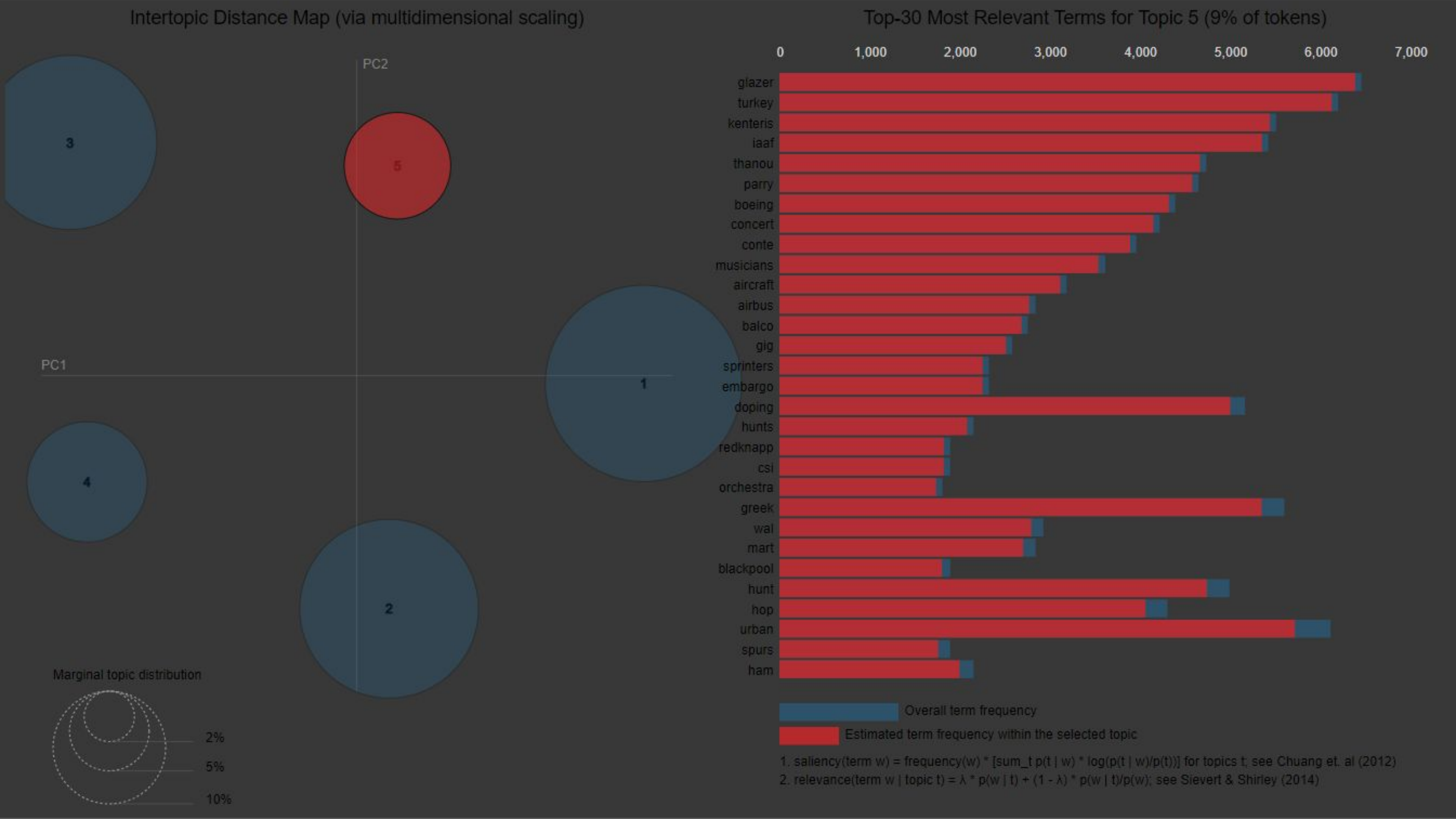
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)



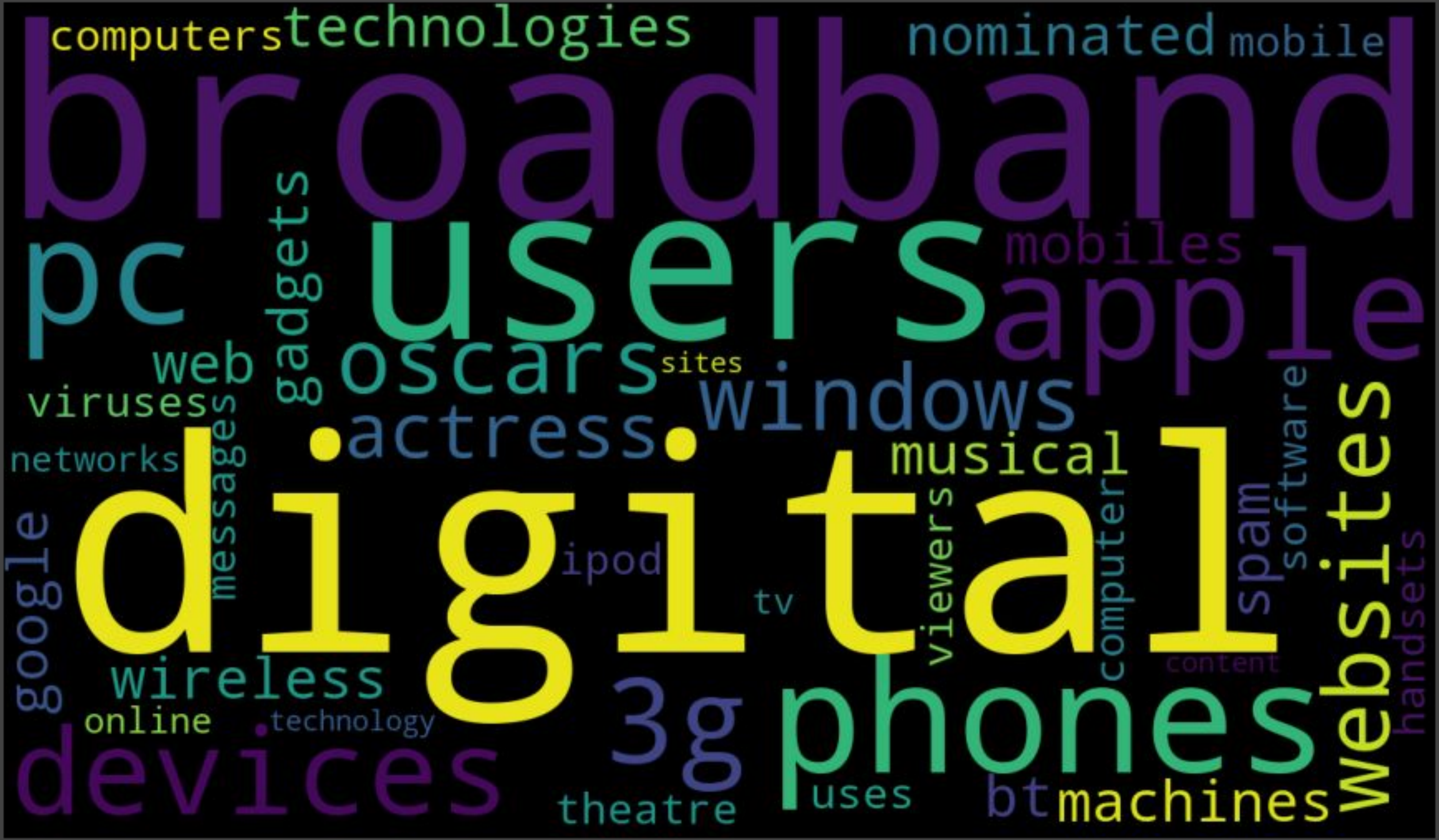
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

WordCloud Visualizations

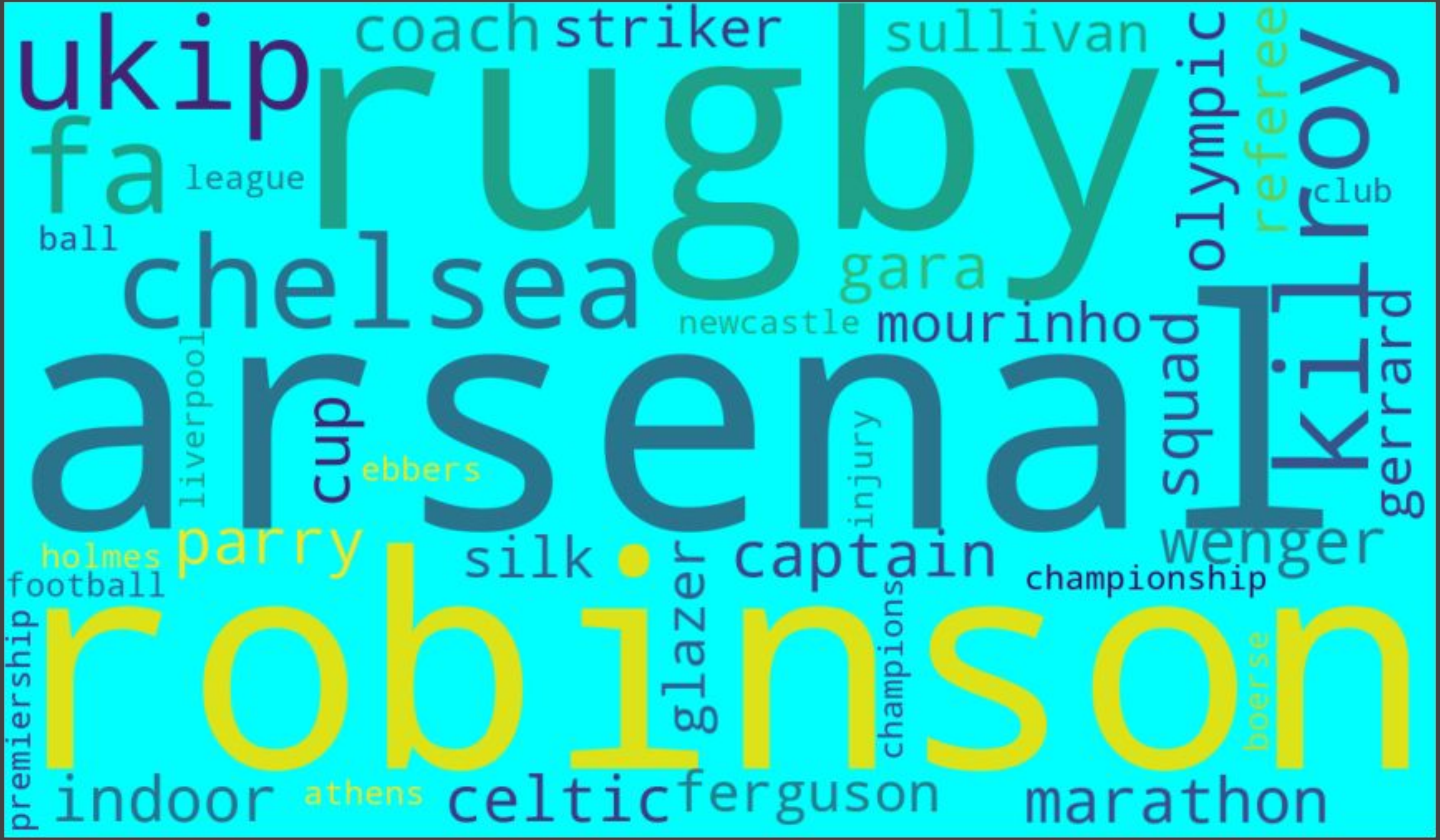
WordCloud is a python package that helps us visualize words in a wordcloud format. We can also provide the frequencies of the words, and they will be emphasized accordingly.

We shall look at the wordcloud generated for each topic, in the subsequent slides

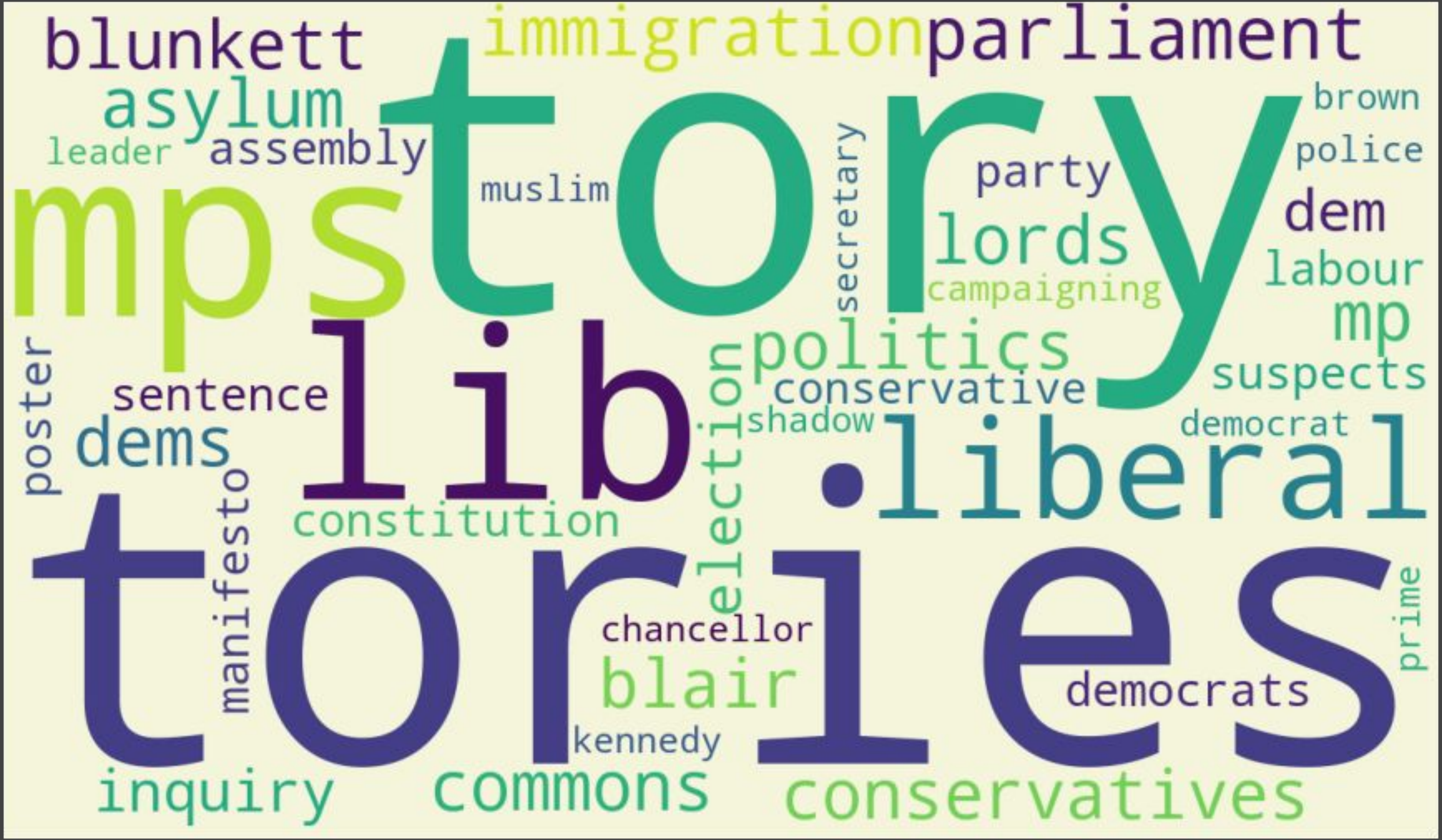
Topic 1 : Tech


broadband
digital
users
pc
apple
phones
devices
3g
websites
windows
oscars
actress
technologies
computers
nominated
mobile
mobiles
gadgets
web
sites
viruses
networks
messages
musical
google
ipod
tv
viewers
computer
spam
software
handsets
content
wireless
online
technology
theatre
uses
bt
machines

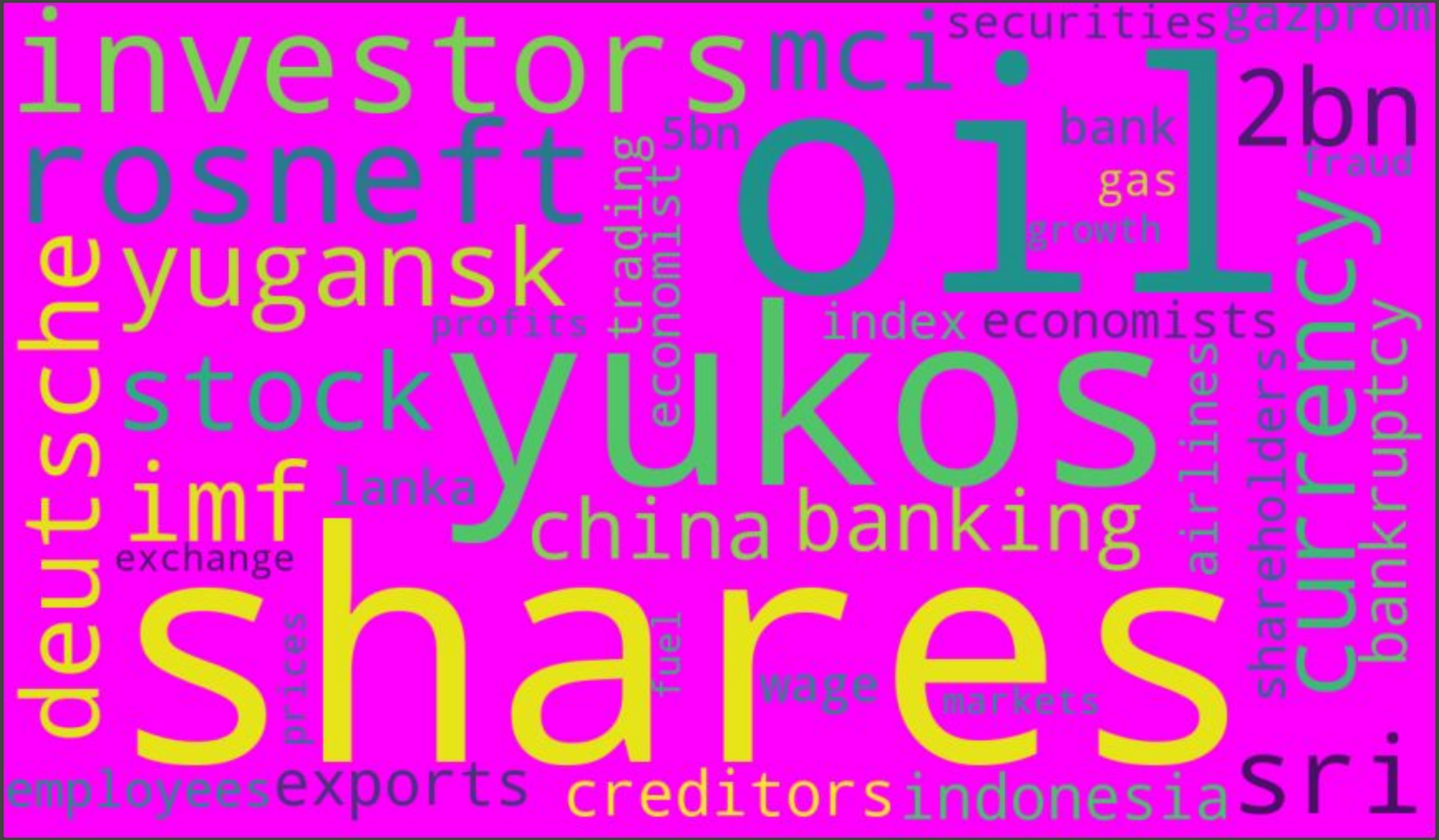
Topic 2 : Sports



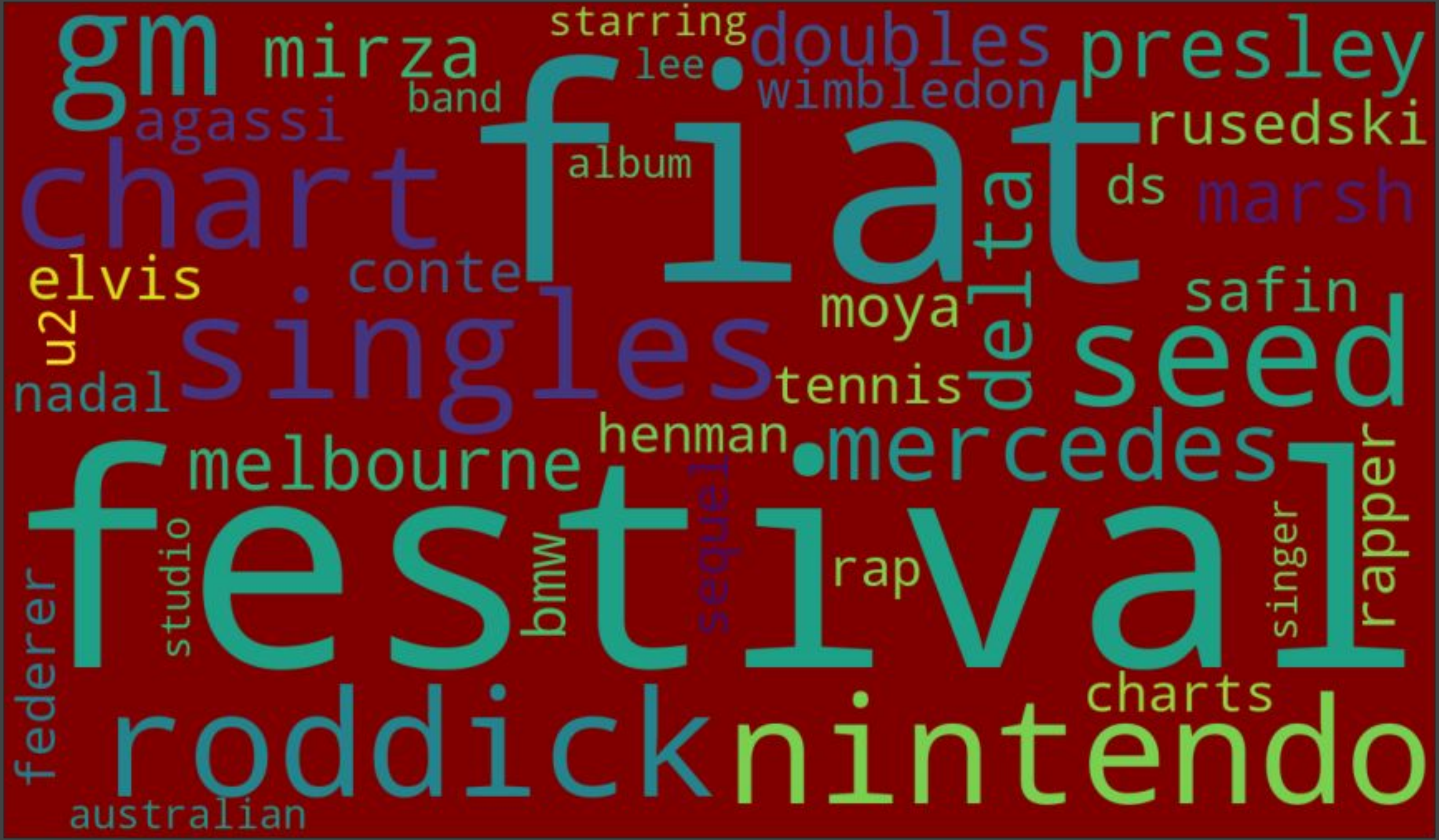
Topic 3 : Politics



Topic 4 : Business



Topic 5 : Entertainment



Conclusions

- 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.

- 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.

- 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.
-
- These are the optimal lda metrics that we got after implementing GridSearchCV:
    - **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**

- 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 observe that the model effectively beared fruit.

References

- **Python NLTK API Documentation**
  https://www.nltk.org/api/nltk.html

- **Python SkLearn LDA Documentation**
  https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html

- **pyLDAvis API Documentation**
  https://pyldavis.readthedocs.io/en/latest/modules/API.html

- **Python Wordcloud Usage Reference**
  https://python-course.eu/applications-python/python-wordcloud-tutorial.php

Thank You