Prepared by:
Khaled Hamdi
Youssef Khaled
Abdullah mousaad
Ziead tarek

Topic Modelling for NLTK

1.Introduction

Topic modelling is a powerful technique used to uncover latent themes or topics within a collection of documents. It has various applications in fields such as information retrieval, text mining, and social media analysis. Utilizing topic modeling we can scan large volumes of unstructured text to detect keywords, topics, and themes.

In this project, the objective is to develop a topic modelling system using NLTK (Natural Language Toolkit) and the LDA (Latent Dirichlet Allocation) algorithm.

The project aims to explore the dataset, preprocess it, train an LDA model using NLTK, and evaluate the performance of the model in terms of topic coherence and interpretability.

2. NLTK Overview

The Natural Language Toolkit (NLTK) is a widely used Python library for Natural Language Processing (NLP) tasks. It provides a comprehensive set of tools and resources for working with human language data. NLTK is designed to assist in various stages of NLP workflows, including text preprocessing, linguistic analysis, machine learning, and information retrieval.

Key features and functionalities of NLTK include:

1. Tokenization: NLTK offers robust tokenization methods to split text into individual words or sentences. It supports different tokenization strategies, such as word tokenization, sentence tokenization, and regular expression-based tokenization.
2. Stemming and Lemmatization: NLTK provides algorithms for stemming and lemmatization, which help reduce words to their base or root forms. Stemming simplifies words by removing suffixes, while lemmatization considers the context and transforms words to their dictionary form.
3. Part-of-speech Tagging: NLTK includes part-of-speech (POS) tagging capabilities, enabling the identification of the grammatical category (e.g., noun, verb, adjective) of each word in a sentence. NLTK offers pre-trained taggers as well as the flexibility to train custom taggers using annotated corpora.
4. Named Entity Recognition (NER): NLTK includes built-in NER models for identifying and classifying named entities in text, such as person names, organizations, locations, and dates. These models are trained on large annotated datasets and can be further fine-tuned for domain-specific NER tasks.

Prepared by:
Khaled Hamdi
Youssef Khaled
Abdullah mousaad
Ziead tarek

5. Text Corpora and Resources: NLTK provides access to various text corpora and lexical resources, such as the Brown Corpus, WordNet, and the Gutenberg Corpus. These resources can be used for training and evaluating models, extracting linguistic features, or conducting research in NLP.

6. Sentiment Analysis: NLTK offers tools and resources for sentiment analysis tasks, enabling the classification of text into positive, negative, or neutral sentiment. It includes pre-trained sentiment analysis models and datasets for training custom models.

7. Machine Learning Integration: NLTK seamlessly integrates with popular machine learning libraries, such as scikit-learn, allowing for the incorporation of NLTK's preprocessing and feature extraction capabilities into machine learning pipelines.

The comprehensive documentation and active community support make NLTK a valuable resource for NLP practitioners and researchers. Its modular design and user-friendly interface enable efficient development and experimentation with various NLP techniques.

In this project, NLTK's functionalities, such as tokenization, stemming, and stop word removal, will be leveraged for text preprocessing in the topic modelling pipeline using LDA. The versatility and robustness of NLTK make it a suitable choice for NLP tasks, including topic modelling.

3.Dataset Description and implementation

We're going to be working on dataset "Articles1.csv" which a collection of over 55000 articles relating to US policies, News, and Covid-19 precautions.

The dataset consists of 10 columns but the column we're focusing on is the "content" column where the articles are

We're going to refer to the dataset as "content". The content is relatively clean but we need to clean it further

We form the bigram and trigrams using the Phrases model. This is passed to Phraser() for efficiency in speed of execution.

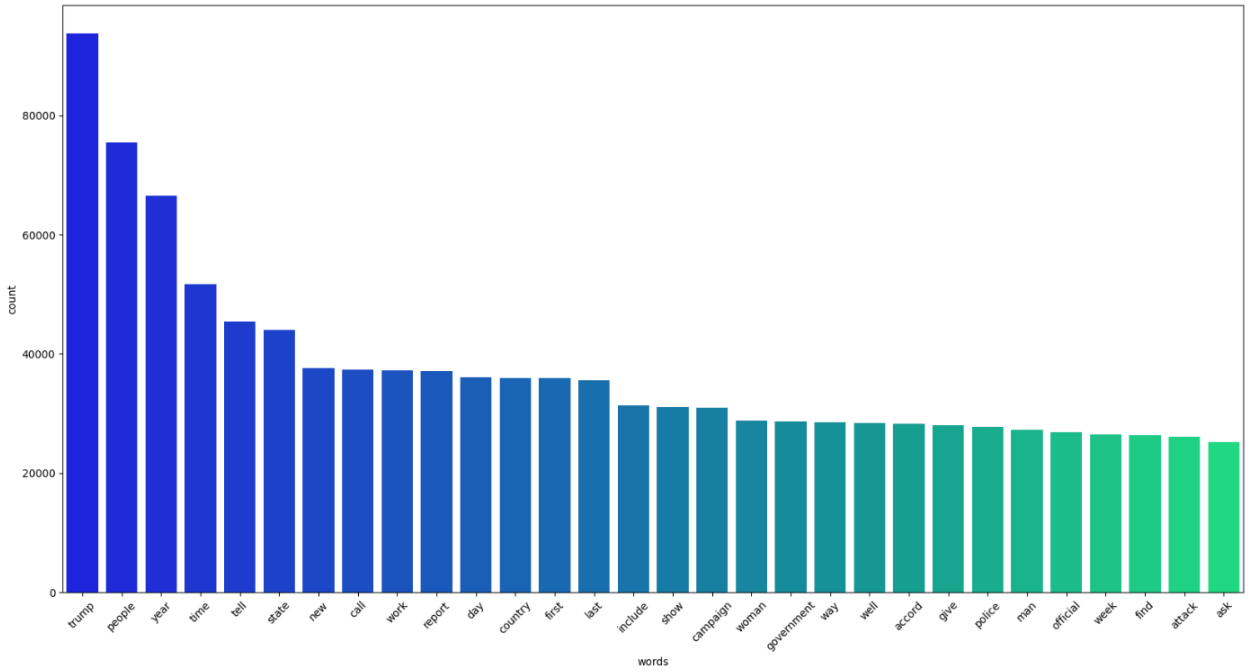
```
# Load the spaCy model outside the process_words function
nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])
bigram = gensim.models.Phrases(data_words, min_count=5, threshold=100)  # higher threshold fewer phrases.
trigram = gensim.models.Phrases(bigram[data_words], threshold=100)
bigram_mod = gensim.models.phrases.Phraser(bigram)
trigram_mod = gensim.models.phrases.Phraser(trigram)
```

Prepared by:
Khaled Hamdi
Youssef Khaled
Abdullah mousaad
Ziead tarek

Next, we remove stopwords and implement lemmatization

```
def process_words(texts, stop_words=stop_words, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV'], nlp=nlp):
    """Remove Stopwords, Form Bigrams, Trigrams, and Lemmatization"""
    texts = [[word for word in doc if word not in stop_words] for doc in texts]
    texts = [bigram_mod[doc] for doc in texts]
    texts = [trigram_mod[doc] for doc in texts]
    texts_out = [[token.lemma_ for token in nlp(" ".join(sent)) if token.pos_ in allowed_postags] for sent in texts]
    texts_out = [[word for word in doc if word not in stop_words] for doc in texts_out]
    return texts_out

# Process the data_words using the updated process_words function
data_ready = process_words(data_words)  # processed Text Data!
```

Finally, this is a simple representation of the word count after the preprocessing



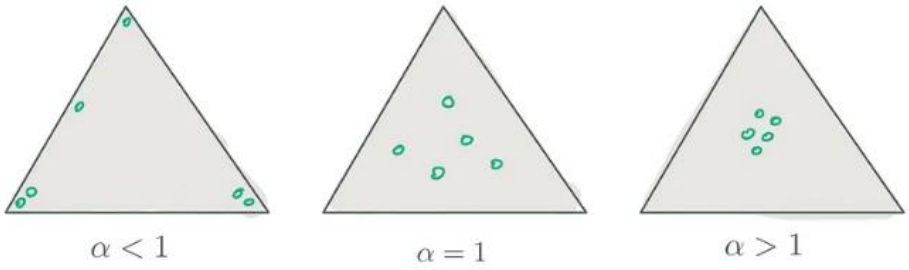
4.Topic modelling with LDA

Latent Dirichlet Allocation is a popular statistical unsupervised machine learning model for topic modeling. It assumes each topic is made up of words and each document (in our case each review) consists of a collection of these words. Therefore, LDA tries to find words that best describe each topic and matches reviews that are represented by these words.

LDA uses Dirichlet distribution, a generalization of Beta distribution that models probability distribution for two or more outcomes (K). For example, K = 2 is a special case of Dirichlet distribution for beta distribution.

Prepared by:
Khaled Hamdi
Youssef Khaled
Abdullah mousaad
Ziead tarek

Dirichlet distribution denoted with Dir(α) where α < 1 (symmetric) indicates sparsity, and it is exactly how we want to present topics and words for topic modeling. As you can see below, with α < 1 we have circles on sides/corners separated from each other (in other words sparse), and with α > 1 we have circles in the center very close to each other and difficult to distinguish. You can imagine these circles as topics.



LDA uses two Dirichlet distributions where

K is the number of topics

M denotes the number of documents

N denotes the number of words in a given document

Dir(alpha) is the Dirichlet distribution per-document topic distribution

Dir(beta) is the Dirichlet distribution per-topic word distribution

It then uses multinomial distributions for each word position

to choose a topic for the j-th word in document i; z_{i,j}

to choose a word for the specific word; w_{i,j}

Prepared by:
Khaled Hamdi
Youssef Khaled
Abdullah mousaad
Ziead tarek

If we bring all the pieces together, we get the formula below, which describes the probability of a document with two Dirichlet distributions followed by multinomial distributions.

$$P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{i=1}^{K} P(\varphi_i; \beta) \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j)\, P(W_{j,t} \mid \varphi_{Z_{j,t}})$$

Probability of a document

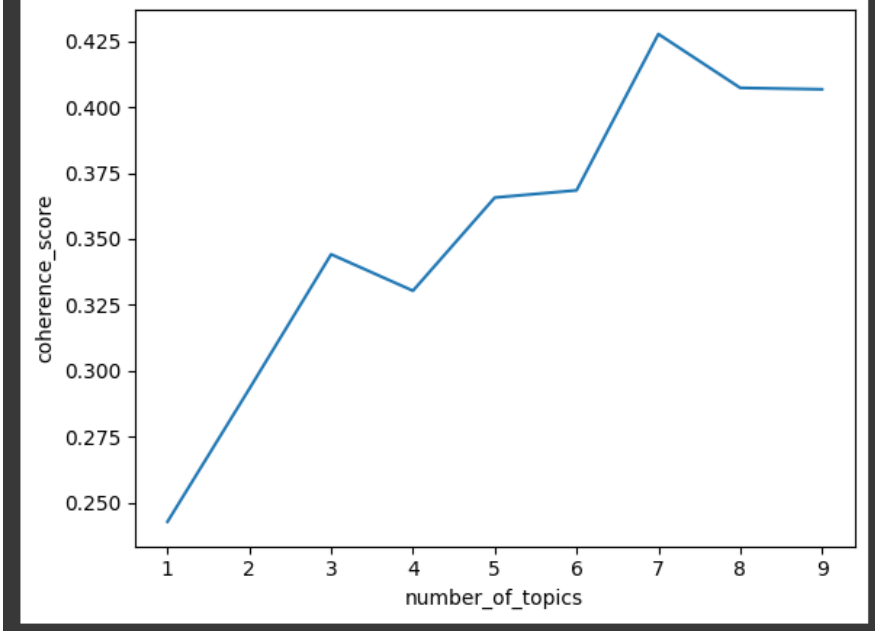
5. Determining the Number of Topics

Deciding on the number of topics for the topic modeling can be difficult. Since we have initial knowledge of the context, determining the number of topics for modeling wouldn't be too outraging. However, if this number is too much then the model might fail to detect a topic that is broader and if this number is too less then topics might have large overlapping words. Because of these reasons, we will use the topic coherence score.

```
from gensim.models import CoherenceModel

# Compute coherence score
number_of_topics = []
coherence_score = []
for i in range(1,10):
    print(i)
    lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                                id2word=id2word,
                                                iterations=50,
                                                num_topics=i)
    coherence_model_lda = CoherenceModel(model=lda_model,
                                         texts=data_ready,
                                         dictionary=id2word,
                                         coherence='c_v')
    coherence_lda = coherence_model_lda.get_coherence()
    number_of_topics.append(i)
    coherence_score.append(coherence_lda)

# Create a dataframe of coherence score by number of topics
topic_coherence = pd.DataFrame({'number_of_topics':number_of_topics,
                                'coherence_score':coherence_score})

# Print a line plot
sns.lineplot(data=topic_coherence, x='number_of_topics', y='coherence_score')
topic_coherence
```

Prepared by:
Khaled Hamdi
Youssef Khaled
Abdullah mousaad
Ziead tarek

	number_of_topics	coherence_score
0	1	0.242778
1	2	0.293162
2	3	0.344174
3	4	0.330358
4	5	0.365693
5	6	0.368435
6	7	0.427702
7	8	0.407293
8	9	0.406752



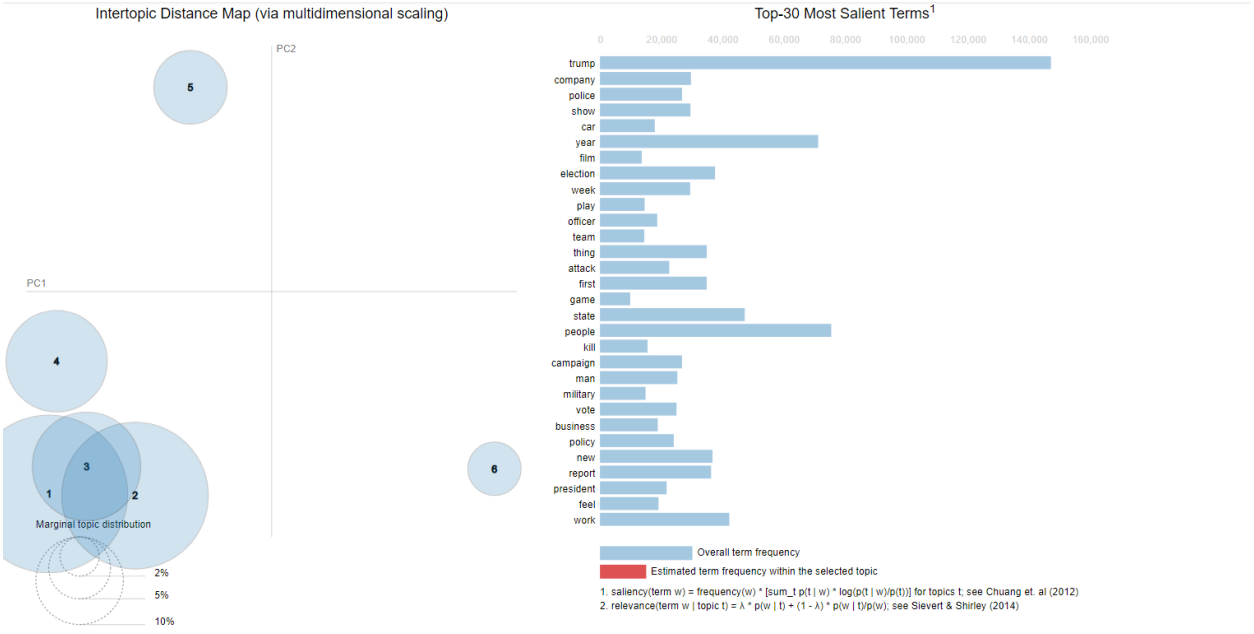
We recognize that the highest coherence score is at 7 topics, so we implement LDA for 6 topics,

Prepared by:
Khaled Hamdi
Youssef Khaled
Abdullah mousaad
Ziead tarek

```
# Build LDA model
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=6,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=10,
                                            passes=10,
                                            alpha='symmetric',
                                            iterations=100,
                                            per_word_topics=True)

pprint(lda_model.print_topics())
```

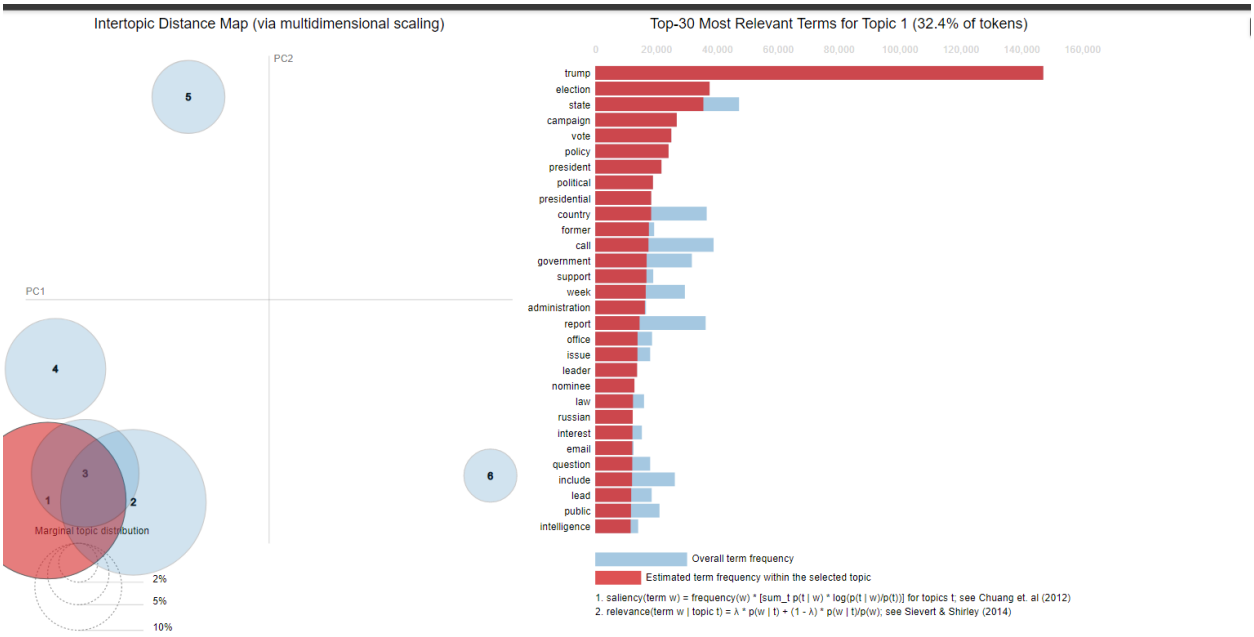
6.Conclusion

This is the most used words



And this is topic 1 for example.

Prepared by:
Khaled Hamdi
Youssef Khaled
Abdullah mousaad
Ziead tarek



In conclusion, this project successfully implemented topic modelling using NLTK and the Latent Dirichlet Allocation (LDA) algorithm. By leveraging the capabilities of NLTK for text preprocessing and the probabilistic modeling approach of LDA, meaningful topics were discovered within the given dataset.

The key findings and contributions of the project include:

1. Effective Topic Modelling: The NLTK and LDA-based approach proved to be effective in uncovering latent topics within the dataset. The LDA algorithm efficiently modeled the underlying topic structure by assigning probabilities to words and documents, allowing for the identification and interpretation of coherent topics.

2. Preprocessing with NLTK: NLTK's comprehensive set of tools for text preprocessing, such as tokenization, stemming, and stop word removal, played a crucial role in preparing the dataset for topic modelling. These preprocessing techniques helped to reduce noise and improve the quality of the topic modelling results.

3. Interpretability of Discovered Topics: The discovered topics demonstrated reasonable coherence and interpretability. By analyzing the top words associated with each topic and examining representative documents, meaningful themes were identified, providing valuable insights into the content and underlying patterns within the dataset.

The implications of this project are significant for various domains and applications:

Prepared by:
Khaled Hamdi
Youssef Khaled
Abdullah mousaad
Ziead tarek

1. Information Retrieval: The developed topic modelling system can be utilized for effective information retrieval, allowing users to search and access relevant documents based on specific topics of interest. This can enhance the efficiency of document retrieval systems in various fields, such as academia, business, and healthcare.

2. Content Recommendation: The discovered topics can be leveraged for personalized content recommendation systems. By matching user preferences and interests with the identified topics, relevant articles, news, or products can be recommended to users, enhancing their overall user experience.

3. Trend Analysis and Monitoring: The topic modelling system can be applied to monitor trends and changes in a given domain or industry. By analyzing the distribution of topics over time, organizations can gain insights into emerging trends, public opinion, or customer preferences, enabling them to make informed decisions and adapt their strategies accordingly.

Resources

1. "Text Preprocessing with NLTK", Ruthu S Sanketh, Published inTowards Data Science, Dec 3, 2020
2. "Topic modeling visualization – How to present the results of LDA models?", Selva Prabhakaran, December 4, 2018