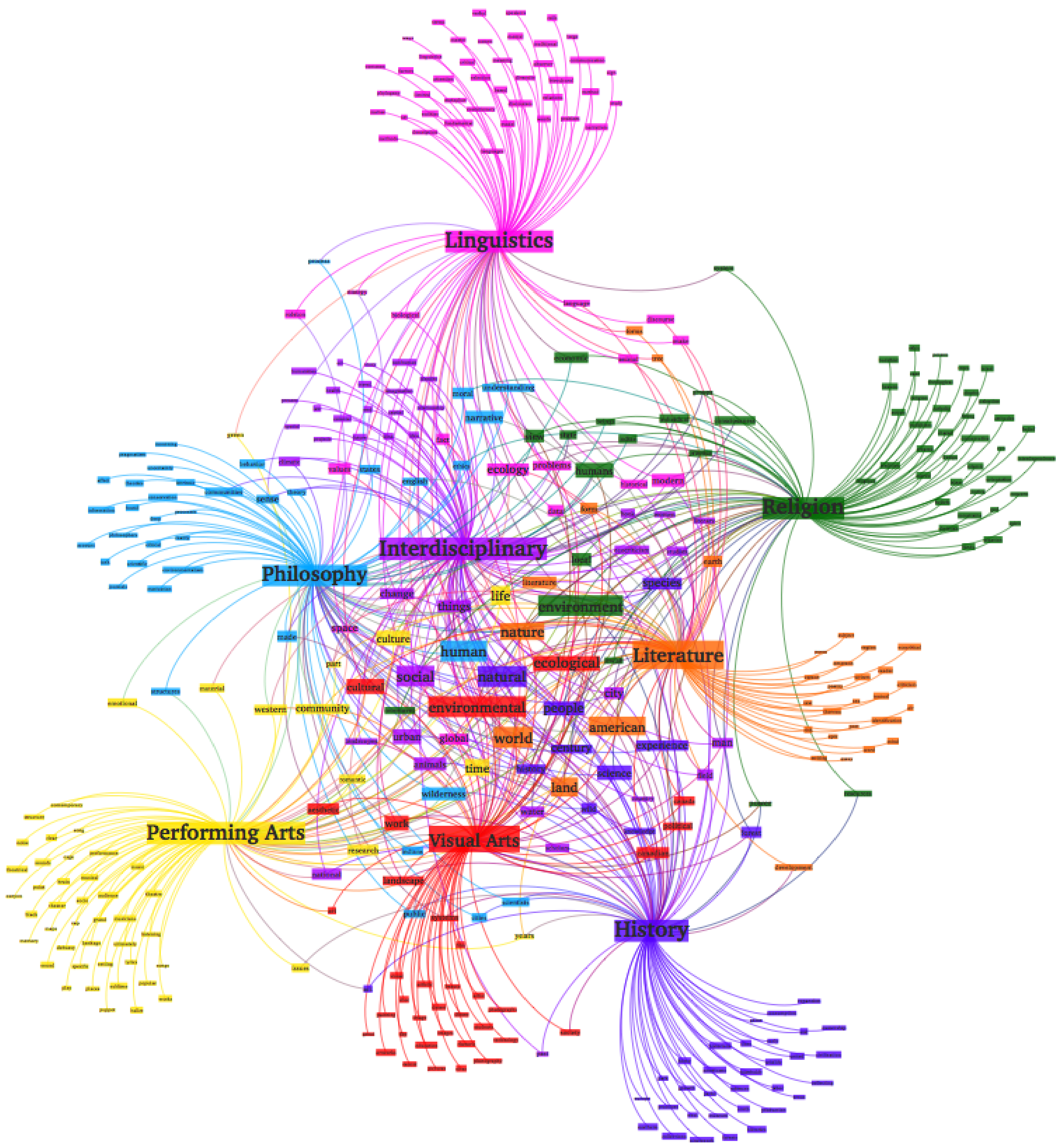
TOPIC MODELLING
PROJECT-X: NERDS_OF_A_FEATHER



INTRODUCTION

Topic analysis or Topic Modelling is an Unsupervised machine learning technique which organizes and understands large collections of text data, infers patterns, creates clusters of similar expressions by assigning “tags” or categories according to each individual text’s topic or theme.

With the aid of Natural Language Processing (NLP), Topic Modelling sorts human language to find patterns, and find language semantics within textual data, and derives important conclusions and insights to facilitate an appropriate data-driven decision.

OBJECTIVE

Creating a Topic Model using the LDA Algorithm on the scraped data (from Jan1-Jan14 2021) from “TheHindu” e-newspaper.

BODY

The entire process of Topic Modelling can be divided into two main components:

A. **Web Scraping** - Scraping the relevant data from various sources (TheHindu newspaper in this project). Libraries such as ‘requests’, ‘BeautifulSoup’ are used.

B. **Natural Language Processing** - Removing unwanted data elements, phrases, symbols, performing Lemmatization, Stemming, Tokenization, to obtain data which is ready to be fed to the Topic Model.

C. **Training and Testing the Topic Model** - Training the Model on a certain percentage of the entire dataset (90% in this project) also called the Train set, and test the performance of the model on the Testing set (10% in this project).

In order to understand the entire pipeline, let us go over all the steps sequentially:

1. WEB SCRAPING-

Web Scraping is the process of programmatically collecting data from various websites and sources. Python is the programming language preferred for this task because of its plethora of options and libraries for Web Scraping.

- [ ] **MAIN LIBRARIES USED:**

‘Requests’ - Allows us to make HTTP requests in python.

'BeautifulSoup' - Used to extract data in a hierarchical and readable format from HTML and XML files.

☐ <u>ARCHITECTURE FLOW:</u>

1. **IDENTIFYING DATA REQUIREMENT -** Scraping data from the website is easy, however, it is important to scrape the **right data**. For this purpose, first understand the components of the website which are of relevance to the Topic Model, and must be scraped.
2. **STUDYING WEBSITE STRUCTURE -** To be able to extract the right data, we must understand the layout of the website.
3. **WEB SCRAPING ETHICS -** While scraping any website, it is extremely important to take into account the crawling politeness policy; a practice meant to save server overload. For this, it is a good practice to go through the ***'robot.txt'*** file which most websites keep to inform scrapers of the sections of the website which are allowed to be scraped, and which ones are not. ***Time delays*** should also be used while scraping.
4. **ROTATING IP ADDRESS:** Getting blocked by a website is a common problem which web scrapers face, when trying to crawl the website multiple times, as this increases the server load. To tackle this, various methods can be used, one of which is **'Proxy Rotation'.** Proxies are scraped from free-proxy sites (one can also use a private proxy) and a rotating pool of proxies is created which is used to send a request to the website with a different IP address every time. This way, the website believes that multiple real users are accessing the website, instead of a single bot or person.
5. **EXTRACTING RELEVANT DATA:** After creating a scraper which works along the lines of the crawling politeness policy, we must extract the data from the website which is required. Irrelevant objects such as ads, photographs should not be scraped.
6. **DATA VISUALIZATION:** After having scraped all the webpages, we must ensure that none of the extracted elements are empty or are falling out of the confidence range(5% - 95% of the average document length). And with a lot of data files, the best way to address these outliers is by data visualization. We must identify the upper and lower bounds and remove any document which does not lie within the range, and can be problematic for the Topic Model.
7. **DATA FILE CREATION:** All the data scraped from all the URL can be structured as JSON files, and be saved on Cloud.

2. Natural Language Processing -

Natural Language Processing (NLP) forms the second component of the project. NLP is the ability of a computer program to understand human language as it is spoken and written. It helps in creating a reliable dataset for topic modelling which improves the quality of the training data for analytics and enables accurate decision-making.

For implementing the NLP pipeline, web scraped data (JSON files) is imported and a corpus (collection of documents) is created. A data frame is created using the ‘pandas’ library to organize the data. Following this, we go through the data cleaning procedure to identify and remove any unnecessary data present in the dataset which involves the following steps:

- ☐ **ARCHITECTURE FLOW:**

1. Data Cleansing:

* **Removing Whitespaces**: Remove extra whitespaces (Removes line characters too) using ‘regex (re)’ library.

* **Removing Punctuations and case**: Remove punctuations and convert to lower case using ‘string’ library.

* **Removing Numbers**: Remove numbers present in the data using ‘isdigit()’ function.

2. Removing Stopwords: Stopwords are the most commonly used words in any natural language. For the purpose of analyzing text data and building NLP models, these stopwords are removed using ‘NLTK library's’ stopwords function.

3. Lemmatization: Lemmatization refers to grouping together words with the same root or lemma but with different inflections or derivatives of meaning so they can be analyzed as one item. Here, ‘Spacy’ library is used for lemmatization.

4. Removing Person Names: The dataset consists of many names of famous persons which are irrelevant for topic modeling, hence we remove them. Using the ‘Spacy’ library, load the trained NLP pipeline using en_core_web_sm.load() and pass the corpus through this pipeline. The document returned from the pipeline tags each word with certain characteristics such as parts of speech, bigrams, etc. In this, the ‘label_’ categorises the words with their type like PERSON, ORGANISATION, etc. Making use of this, check if a token is a PERSON and does not include it in our corpus and if it’s not then include it in the corpus. In this way, all tokens with PERSON tags are removed.

5. **Bigram Formation:** Most frequently occurring words are paired using the 'Gensim' library and the model is trained by providing corpus and hyperparameters.

6. **Removing words having less than three letters**: The words having less than three letters are insignificant for topic modeling, hence they are removed using a nested loop through tokens.

The processed text is obtained which can be used for topic modelling.

3. TOPIC MODELLING -

Topic Modeling is a part of NLP, which is used to understand a large corpus of texts through topics extraction.

LDA (Latent Dirichlet Allocation) is an unsupervised machine-learning model that takes documents as input and finds topics as output. The model also says in what percentage each document talks about each topic. A topic is represented as a weighted list of words.

In this project, we use the scraped data from the given webpages and the corpus of text, upon which the LDA model will be trained and tested.

- ☐ **MAIN LIBRARIES USED:**

  Gensim - Used for unsupervised topic modelling and Natural Language processing, and is designed to handle large text collections.

- ☐ **ALGORITHM USED:** Latent Dirichlet Allocation (LDA)

'*Latent*' in LDA refers to finding the ***hidden*** topics and '*Dirichlet*' is the ***probability distribution*** used to understand how a keyword and topic are related. LDA assigns each document ("TheHindu" news articles in this project) to a unique topic and each topic to a set of keywords using the topic probability in each document and word probability in each topic.

The steps taken for Topic Modeling are as follows:

- ☐ **ARCHITECTURE FLOW:**

1. **Reading the .csv files:** All the text files stored in .csv files are read into the code, and all the text is appended into a single list.

2. **Data Cleansing**: Once again, these data are subjected to the aforementioned process of Data Cleansing. This is done in order to cherry pick and remove any remaining unwanted elements.

3. **Frequency Distribution Graph:** Using the 'nltk.probability' library, a Word Frequency Distribution Graph is drawn, which depicts which words occur most frequently in the entire text. Those words that have a frequency, but seem like they would not add much value to the model are identified intuitively, and then are removed using the 'spacy.lang.en' library.

4. **Finding the Optimal Number of Topics:** The Optimal Number of Topics are obtained for the LDA Model using a line plot.

5. **Test Train Split**: The entire corpus is split into training and testing data, in the ratio 90:10 respectively.

6. **Hyperparameter Tuning**: Using Gridsearch, we obtain the most optimal parameters for the LDA model.

7. **Building the Model**: Then, using the 'gensim.models.LdaMulticore' library, a Latent Dirichlet Model is built, using the obtained optimal number of topics and the training data.

8. **Visualizing the Topics:** The topics are printed, and we use Word Clouds to visualize each topic, and to just get an overview of the word frequency distribution.

9. **Coherence and Perplexity**: The coherence and perplexity for this model is calculated using the 'gensim.models' library.

10. **Topic Assignment:** Assigning a name for each established topic using the IAB Taxonomy, which is a standard followed to provide the web page content a consistent Topic naming convention.

11. **Model Testing:** The model is then tested on the Test Data, which is 10% of the total data. The coherence is calculated once again to test the efficacy of the model.

12. Topic Allocation and topic Dominance in each document is visualized using various visualization techniques such as the bar plot.