

B.Tech Project Report

CSPE-30

on

**Modeling ‘NEWS’ Context using Word Embedding for Personalized
Recommendations**

BY

ABHISHEK SHARMA (21812019)

SOURABH SINGH (21812021)

PARIXIT CHAUDHARY (21812028)

Group No. :30

**Under the Supervision of
Dr. Vikram Singh , Asst. Prof
Comp. Engg. Dept. NIT Kurukshetra**





**DEPARTMENT OF COMPUTER ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY
KURUKSHETRA – 136119, HARYANA (INDIA)
May, 2022**

CERTIFICATE

We hereby certify that the work which is being presented in this B.Tech Project (CSPE-30) report entitled **“Modeling ‘NEWS’ Context using Word Embedding for Personalized Recommendations”**, in partial fulfillment of the requirements for the award of the **Bachelor of Technology in Computer Engineering** is an authentic record of our own work carried out during a period from January, 2022 to May, 2022 under the supervision of Dr. Vikram Singh Asst.Prof., Computer Engineering Department.

The matter presented in this project report has not been submitted for the award of any other degree elsewhere.

Parixit

Signature of Candidate

**ABHISHEK (21812019)
SOURABH (21812021)
PARIXIT (21812028)**

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Date:

Signature of Supervisor

Dr. Vikram Singh

Asst. Prof.
Computer Engineering Deptt.



TABLE OF CONTENTS

Section No.	TITLE	Page no.
	ABSTRACT	4
I	INTRODUCTION	
II	MOTIVATION	
III	LITERATURE SURVEY	
IV	PROPOSED APPROACH	
V	DATA FLOW DIAGRAM	
	V.1 Level 0 DFD	
	V.2 Level 1 DFD	
	V.3 Level 2 DFD	
VI	IMPLEMENTATION/ RESULTS/ OBSERVATIONS	
VII	CONCLUSION AND FUTURE PLAN	
VIII	REFERENCES (in IEEE format)	
APPENDIX:		
A	COMPLETE CONTRIBUTORY SOURCE CODE	

ABSTRACT

The popularity of various news forums and blogs poses curation challenges to the readers on the continuously generated news articles or items on a daily basis. It has been observed



that in several instances of a similar article are published in different forums and each instance captures some or other information context. The role of 'News Context', primarily defines the background of the NEWS article and its co-located links to a list of sources. The modeling of context serves several purposes. Topic modeling is one of the most powerful techniques in text mining for data mining, latent data discovery, and finding relationships among data and text documents. There are various methods for topic modeling; Latent Dirichlet Allocation (LDA) is one of the most popular in this field. In this project work we have utilized the word embedding techniques to the several the context of the news article and further LDA generated of the recommendations personalized to the user. In this, the designed system will also provide the single platform that fetches news from available sources and Describes and recommends news after some good preprocessing. LDA is a great way of Doing topic modeling and classification of news articles and we'll be using that.



BACKGROUND

Conventional news recommendation systems use a small set of keywords to identify the top recommended articles to users based on keywords frequency. We built upon this framework by enabling users to find recommended articles by providing an entire news article. The recommendation process uses article topics to evaluate which articles to recommend. Our system highlights topic insights through two different models: an unguided LDA for identifying topic recommended articles and a guided LDA with seed words from news to show interpretable topics. Our system also shows sentiment information and word cloud that summarize the query article for the user and personalized news recommendation for the user.

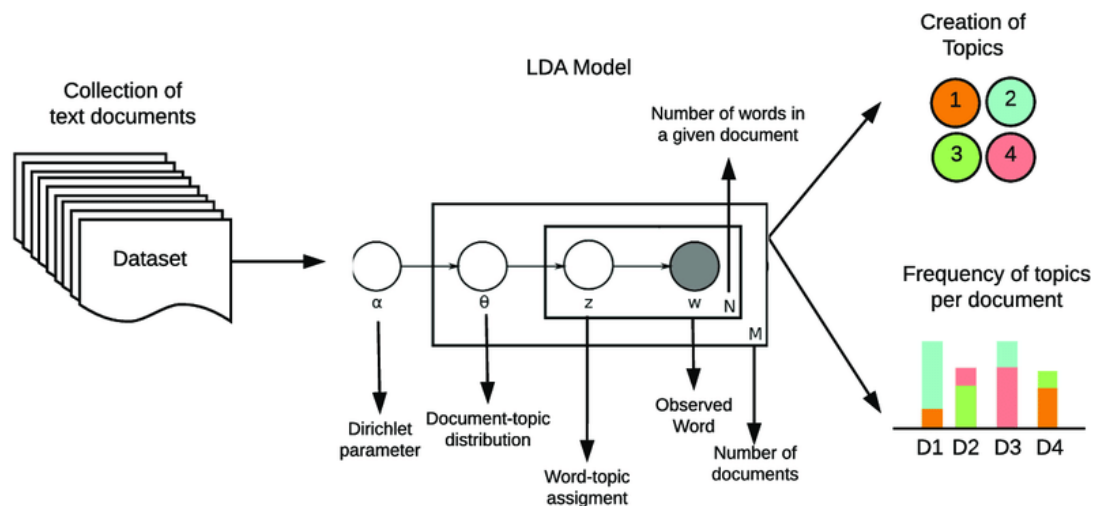


INTRODUCTION

What is LDA?

Latent Dirichlet Allocation (**LDA**) is a popular form of statistical topic modeling.. First, let us break down the word and understand what LDA means. Latent means hidden, something that is yet to be found. Dirichlet indicates that the model assumes that the topics in the documents and the words in those topics follow a Dirichlet distribution. Allocation means to give something, which in this case are topics.

In LDA, documents are represented as a mixture of topics and a topic is a bunch of words. Those topics reside within a hidden layer, also known as a latent layer. Some of the well known topic modeling techniques are Latent Semantic Analysis (LSA), Probabilistic Latent Semantic Analysis (PLSA), Latent Dirichlet Allocation (LDA), Correlated Topic Model (CTM). LDA looks at a document to determine a set of topics that are likely to have generated that collection of words. So, if a document uses certain words that are contained in a topic, you could say the document is about that topic.



LDA

LDA consists of two parts, the words within a document (a known factor) and the probability of words belonging to a topic, which is what needs to be calculated. The algorithm tries to determine, for a given document, how many words belong to a specific

topic. Plus it attempts to determine how many documents belong to a specific topic because of a certain word.

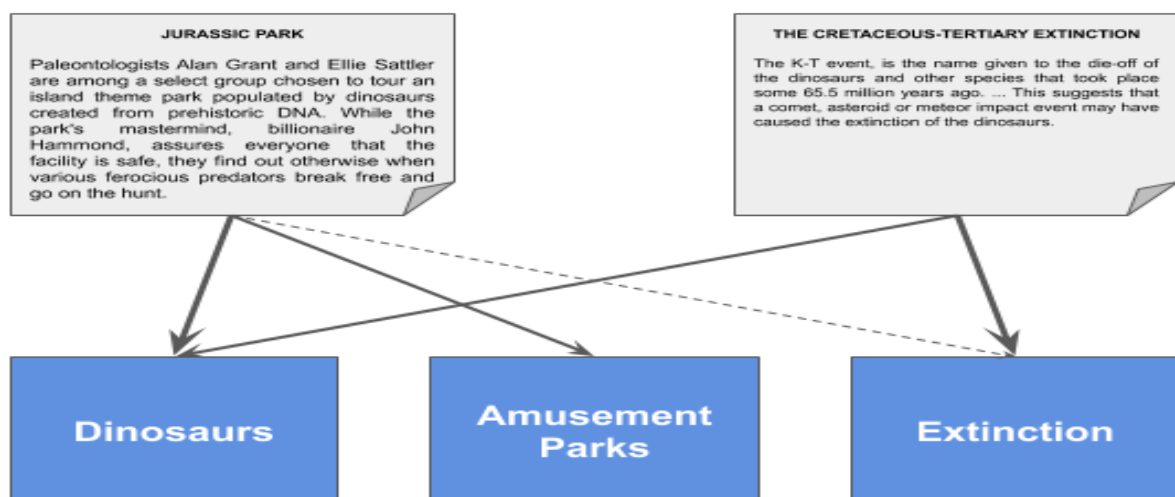
	Word1	word2	word3	word4
Topic1	0.01	0.23	0.19	0.03	
Topic2	0.21	0.07	0.48	0.02	
Topic3	0.53	0.01	0.17	0.04	

Hyperparameters in LDA

There are three hyperparameters in LDA

1. $\alpha \rightarrow$ document density factor
2. $\beta \rightarrow$ topic word density factor
3. $K \rightarrow$ number of topics selected

The α hyperparameter controls the number of topics *expected* in the document. The β hyperparameter controls the distribution of words per topic in the document, and K defines how many topics we need to extract.



- **Document:** Each text file (article in our case study)
- **Corpus:** Collection of all the documents
- **Dictionary:** Collection of mapping of each unique word to a unique index



- M : is the total documents in the corpus
- N : is the number of words in the document
- w : is the Word in a document
- z : is the latent topic assigned to a word
- θ (Θ): is the topic distribution
- LDA model's parameters: Alpha (α) and Beta (β)

In the example with documents, topics, and words, we'll have two PMFs:

- the probability of topic k occurring in document d - (θ_d)
- the probability of word w occurring in topic k - (ϕ_k)



MOTIVATION

Growth in unstructured data and popularity of online news forums and blogs

The growing rate of unstructured textual data has made an open challenge for the knowledge discovery which aims at extracting desired information from large collections of data.

Media, journals and newspapers around the world every day have to cluster all the data they have into specific topics to show the articles or news in a structured manner under specific **topics**. All the social network companies like **Facebook**, **Twitter**, etc do some sort of topic modeling on the posts and advertisements before using the recommendation engines to recommend stuff to users. Even **Google** runs topic modeling in their search to identify the documents relevant to the user search. Imagine having a digital library where the books are randomly placed irrespective of their topics. How difficult it will be to search for them or search for the books that belong to specific topics we are interested in.

Inefficiency in generation of news classification over large arrays of data

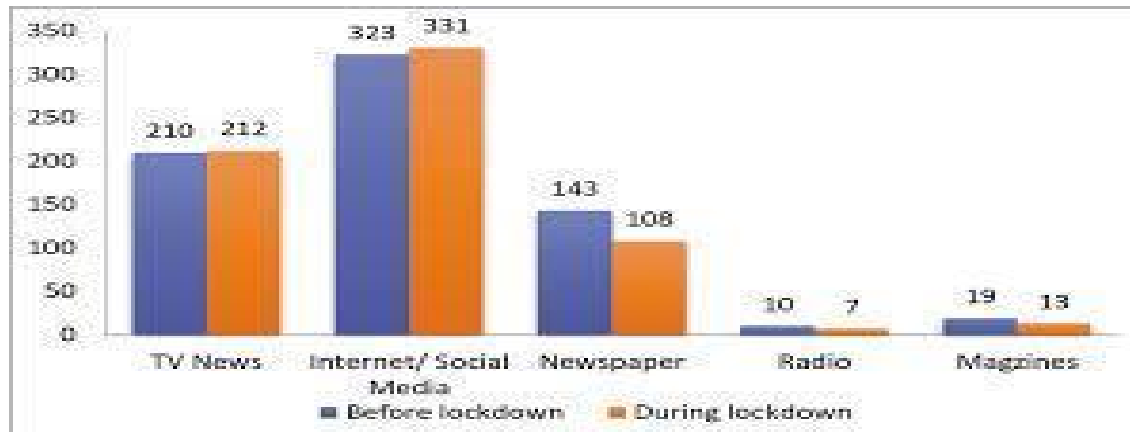
Since, topic modeling techniques can be very useful and effective in natural language processing to semantic mining and latent discovery in documents and datasets.

Hence, our motivation is to generate personalized news recommendation system subjects with the coverage of various aspects such as models, tools, dataset and applications. The main goal of this project is to provide a personalized news recommendation system using topic modeling based on LDA. Social network sites are becoming essential to how people experience news. The social media feed is made up of a mixture of private and public postings, and news is intertwined with all sorts of activities. What people are exposed to partly depends on the behavior of their fellow networkers.

Covid a big boost for blogs and online news

This shift towards digital news outlets and SNSs is especially evident among young people in large parts of the Western world (Associated Press-NORC Center for Public Affairs Research [2015](#); Kohut et al. [2012](#); Reuters Institute Digital News Report [2016](#)). For this reason, this latter group's news consumption habits have been regarded as a

challenge for news industries, journalism practice and democracy in total (Elvestad, Blekesaune, and Aalberg [2014](#)).



Content Recommendation

With online news it is possible to read a large amount of different news sources on a large array of topics, but since there are more articles available, finding interesting ones becomes more difficult, as reading even just every headline becomes cumbersome. Consequently automatic filtering with recommender systems, becomes attractive. We consider the specific problem of how to build a news recommender system to find interesting news within a specific language group, Finnish. We use content-based recommender systems, which is the less studied of the two main paradigms of recommender systems (Adomavicius and Tuzhilin, 2005). The main benefit of this content-based recommendation is that it allows others than large user communities and their parent companies to build recommender systems for any material they want to, also any niche content, be it by language, topic or anything else. By contrast, collaborative filtering requires a large user community, since it bases its recommendation on how similar users have rated the items. Another problem for news recommendation and collaborative filtering is that, even if there is a sufficient number of similar users, the items need to have been rated by at least some of these users, and in the case of news recommendation, it is the new items which are most interesting, and these are the ones



least likely to have been rated. Because of this, news recommendation systems usually include content-based recommendation, where items are recommended to users based on a profile of what kind of content the user has liked in the past (Lang, 1995; Billsus and Pazzani, 2000).

In late 2015, the New York Times (NYT) changed the way it recommends content to its readers, switching from a filtering approach to one that uses topic modeling. The NYT seeks to personalize content for its readers, placing the most relevant content on each reader's screen. It used to do this by a simple keyword matching approach for each reader, later changing to a collaborative matching approach for groups of readers with similar interests. The switch to topic modeling improves on both these approaches.

The NYT uses topic modeling in two ways—firstly to identify topics in articles and secondly to identify topic preferences amongst readers. The two are then compared to find the best match for a reader.

Goals

- a) Development of an integrated NEWS recommendation system, with extensive exploration features for user-support.
- b) Implementing a NEWS platform based on LDA and adapting word embedding schemes for the generation of 'Background' or 'context' of a news and for personalized recommendations.
- c) Learning everything from idea to deployment of machine Learning and Web development.



LITERATURE SURVEY

Topic models have many applications in natural processing languages. Many articles have been published based on topic modeling approaches in various subjects such as Social Network, software engineering, Linguistic science, etc. There are some works that have focused on surveys in Topic modeling.

The authors presented a survey on topic modeling in software engineering field to specify how topic models have thus far been applied to one or more software repositories. They focused on articles written between Dec 1999 to Dec 2014 and surveyed 167 articles that use topic modeling in the software engineering area. They identified and demonstrated the research trends in mining unstructured repositories by topic models. They found that most studies focused on only a limited number of software engineering tasks and also most studies use only basic topic models. The authors focused on survey in Topic Models with soft clustering abilities in text corpora and investigated basic concepts and existing models classification in various categories with parameter estimation (such as Gibbs Sampling) and performance evaluation measures. In addition, the authors presented some applications of topic models for modeling text corpora and discussed several open issues and future directions.

In introduced and surveyed the field of opinion mining and sentiment analysis, which helps us to observe elements from the intimidating unstructured text. The authors discussed the most extensively studied subject of subjectivity classification and sentiment which specifies whether a document is opinionated. Also, they described aspect-based sentiment analysis which exploits the full power of the abstract model and discussed aspect extraction based on topic modeling approaches. They discussed challenges of text mining techniques in information systems research and undigested the practical application of topic modeling in combination with explanatory regression analysis, using online customer reviews as an exemplary data source , the authors presented a survey of how topic models have thus far been applied in Software Engineering tasks from four SE journals and eleven conference proceedings. They considered 38 selected publications from 2003 to 2015 and found that topic models are widely used in various SE tasks in an increasing tendency, such as social software engineering, developer recommendation etc. Content-based news



recommendation has been addressed in the literature by several authors. Lang (1995) describes a method combining nearest neighbors and linear regression, using a combination of TF-IDF features and features selected with Minimum Description Length (MDL). Billsus and Pazzani (2000) used a combination of Nearest Neighbor Classification and Naïve Bayes classification for explicit feedback in an interesting/not interesting classification. Naïve Bayes has also been applied to content-based book recommendation (Mooney and Roy, 2000).

Topic models have not been used directly in content-based recommendation with explicit feedback, but with implicit feedback, a method similar to probabilistic Latent Semantic Indexing (Hofmann, 1999) has been applied by (Cleger-Tamayo et al., 2012). The topic model used is adapted to user behavior. We don't consider topic model adaptation, but rather use **Latent Dirichlet Allocation (Blei et al., 2003)** as an unsupervised feature extraction procedure.. Rashid et al. (2008) derive information theoretic strategies for how the first recommendations for a new user should be made. Recommender systems have been of great interest among the scientific community in the last few years. Several successful approaches have been proposed. These approaches can be classified by the domains they have been used in or applied to. Approaches are further classified by the type of data used for recommendation (implicit or explicit) and lastly by the type of algorithm used for recommendation (collaborative, content based filtering or hybrid approach). Amongst the many papers which both inspired and motivated us to work on this topic, one was [Li et al., 2010]. Here, news recommendation problem was modelled as a contextual bandit problem. Authors devised a new algorithm named as LinUCB to solve the problem. LinUCB is a general contextual bandit algorithm and can be applied to domains other than news recommendation. Using LinUCB, number of clicks increased by 12.5% as compared to a standard context-free bandit algorithm.

While talking about news recommendations, one cannot miss [Das et al., 2007]. Here, both model and memory based algorithms have been used. Memory based algorithms use weighted average of past ratings from other users where weight is proportional to the 'similarity' between users. Pearson correlation coefficient and cosine similarity are the typical

measures used for 'similarity'. Model based ratings model the user preferences based on past information. From model based, they used clustering techniques PLSI and MinHash, and from memory based, they used item co-visitation. All the three algorithms assign a



score to a story. Finally, two combinations of three techniques (PLSI, MinHash and co-visitation) were used for evaluation. In one combination, co-visitation was given a higher weight than the other techniques (2.0 instead of 1.0). This method was named CVBiased and in another combination PLSI and MinHash were given higher weight (2.0 instead of 1.0), which is known as CSBiased. The baseline used was a recommendation based on recent popularity called Popular. It was observed, on average both CVBiased and CSBiased performed 38% better than the baseline Popular on live traffic. Another influential paper is [Liu et al., 2010] , the work done in this paper was built on [Daset al., 2007] . Here, information filtering was used. Information filtering removes unwanted information from an information stream. First, they did an analysis of a users' interest over a 14 month period for each category of article (categories are pre-defined). They used click distribution of a user to study the interest. Then they used Bayes rule to predict a users' interest for a particular period of time. Then, predictions made for particular period were combined to predict a users interest in a long period. The predictions made till now were users' genuine interest. But a user also gets influenced by current news trends, so Bayes rule is used to make predictions using click distribution of a short recent period (example last hour). Finally, users' genuine interest is combined with current news trends to get IF (article).



PROPOSED WORK :

Proposed Approach for Modeling 'NEWS' Context using Word Embedding for Personalized Recommendations includes the following three phases:

PHASE 1 :

- News Fetching
- LDA model Training
- saving news in database

PHASE 2:

- Flask Server Setup with same database
- Authentication Service as API for User

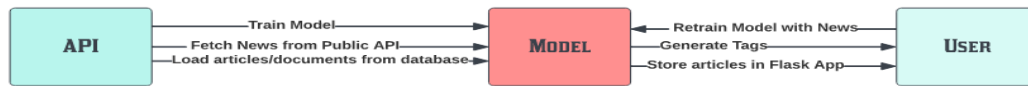
PHASE 3 :

- User Interface using React
- Visualize news Articles

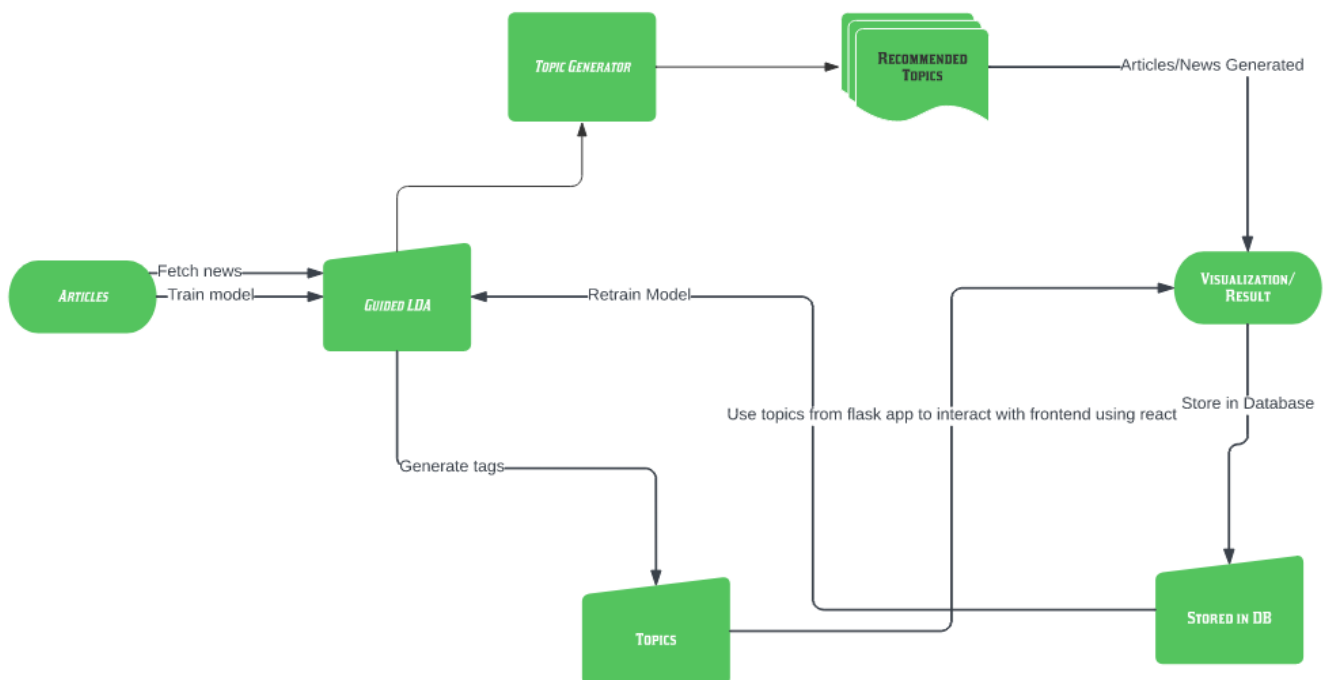


DATA FLOW DIAGRAMS

LEVEL 0:



LEVEL 1:





IMPLEMENTATION

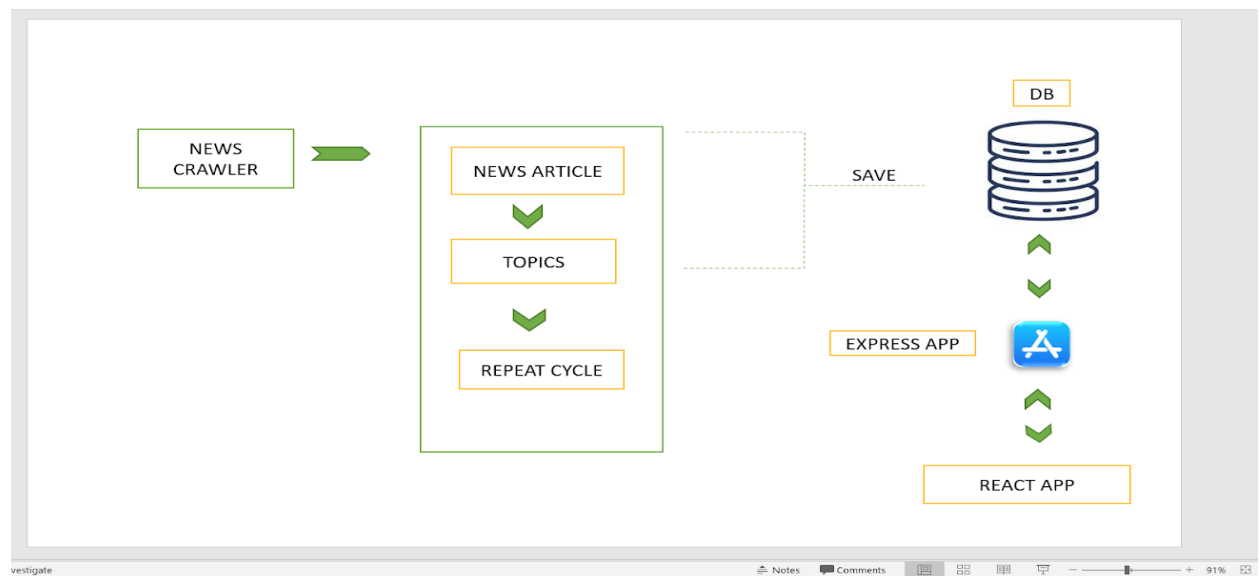
Phase 1:

The focus in this phase is to identify the prospective use- case of the proposed system and further novel aspects/features.

Ingredients to achieve topic modelling

- a. Guided-Lda, scikit-Learn, major python libraries to perform various NLP tasks
- b. LDA, one of the most popular topic modelling algorithms
 1. Topic Generation
 - a. Fetching news from internet
 - b. Saving news article in mongoDB
 - c. Topics Generation
 - d. Topics Stored in DB

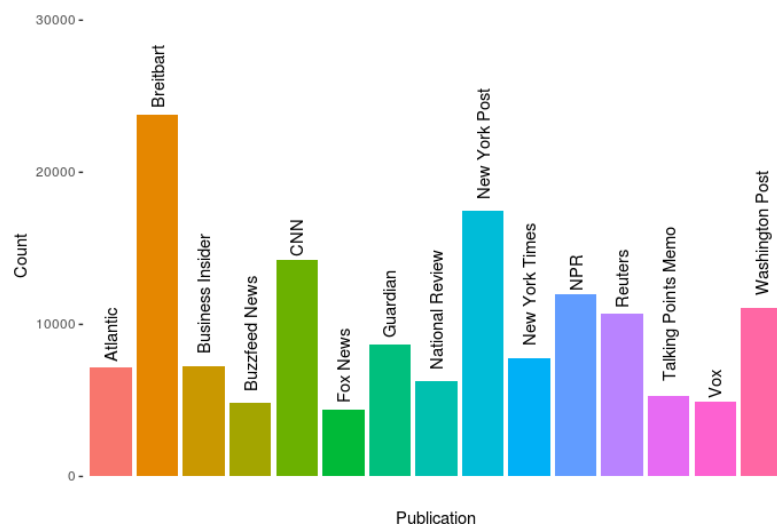
Architecture of the Proposed Approach



Topic Generation:

Fetching News:

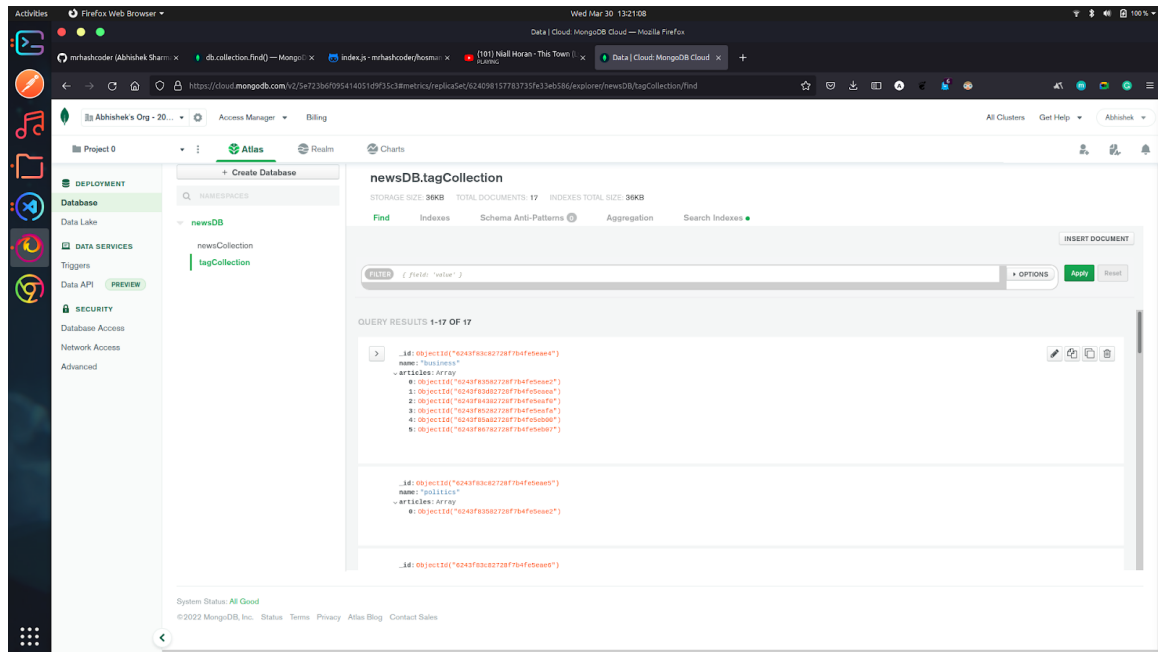
This process involves the data set we'll use for further pre-processing. A list of over one million news headlines published over a period of 15 years and can be downloaded from [Kaggle](https://www.kaggle.com/datasets/robertblackburn/news-headlines). News is fetched using API's





Saving articles in Mongo DB

Labeling the dataset as required i.e. giving id's to topics under topic generation from raw document files and saving it in database





Topics Stored in DB

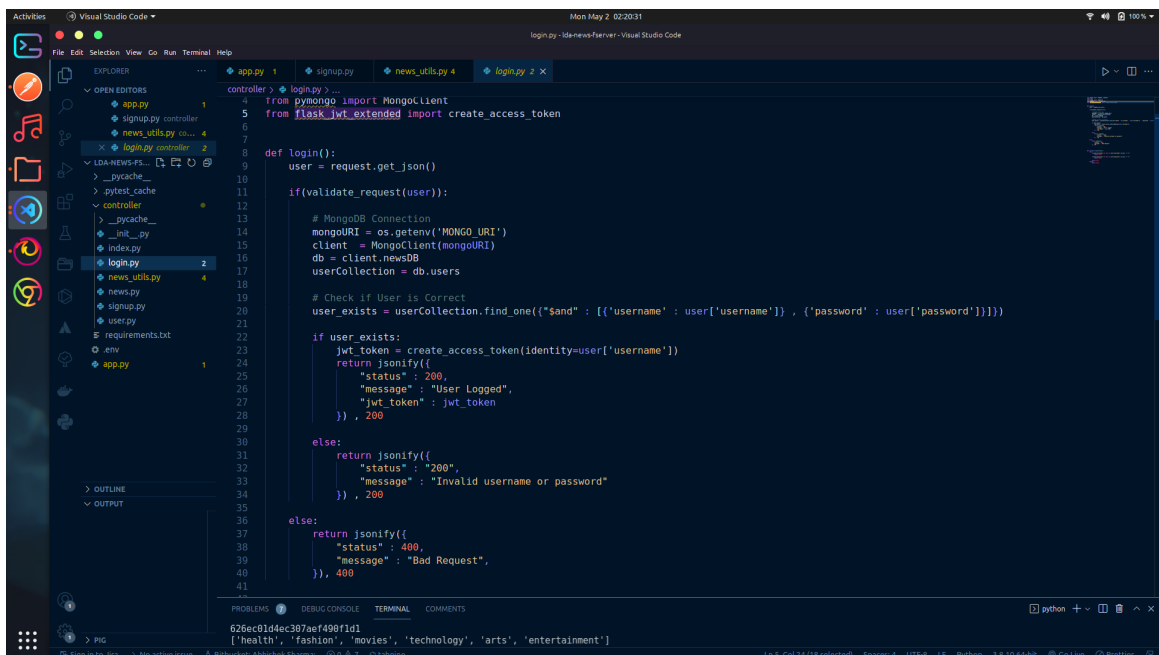
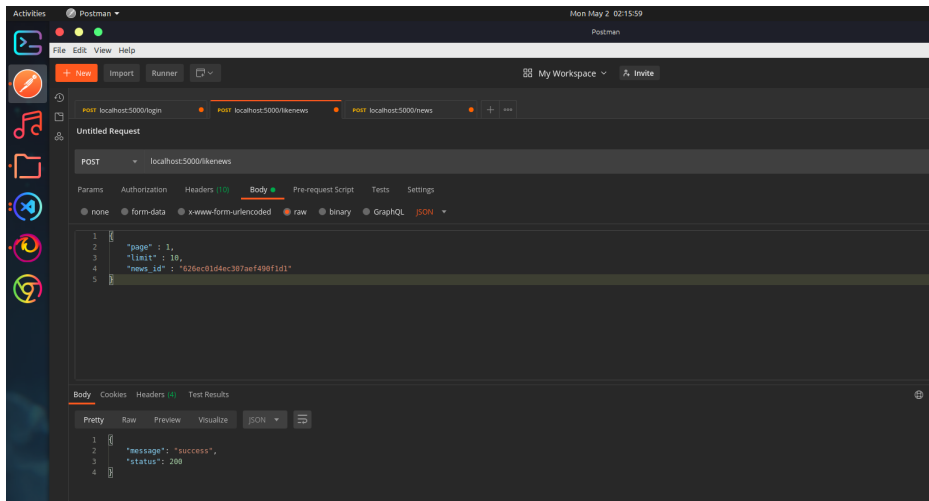
The screenshot shows the MongoDB Atlas web interface. The left sidebar contains navigation options: Project 0, Atlas, Realm, Charts, Deployment, Database, Data Lake, Data Services, Triggers, Data API, Security, Database Access, Network Access, and Advanced. The main panel displays the 'newsDB.newsCollection' database. The top of the main panel shows 'STORAGE SIZE: 52KB', 'TOTAL DOCUMENTS: 10', and 'INDEXES TOTAL SIZE: 36KB'. Below this, there are tabs for 'Find', 'Indexes', 'Schema', 'Anti-Patterns', 'Aggregation', and 'Search Indexes'. The 'Find' tab is active, showing a search bar with the filter '{ field: 'value' }'. Below the search bar, there are two documents displayed. The first document has a title 'PMD invites tender for the improvement including retailing and blacktea...' and a link 'https://www.sentinelnews.com/tender/pmd-invites-tender-for-the-improvement...'. The second document has a title 'Entrepreneurs have the potential of generating bigger dreams, says Nar...' and a link 'https://www.ebnlive.com/news/2022/mar/30/entrepreneurs-have-the-poten...'. The bottom of the interface shows the system status as 'All Good' and the copyright notice '©2022 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales'.



Phase 2:

Flask Server Setup:

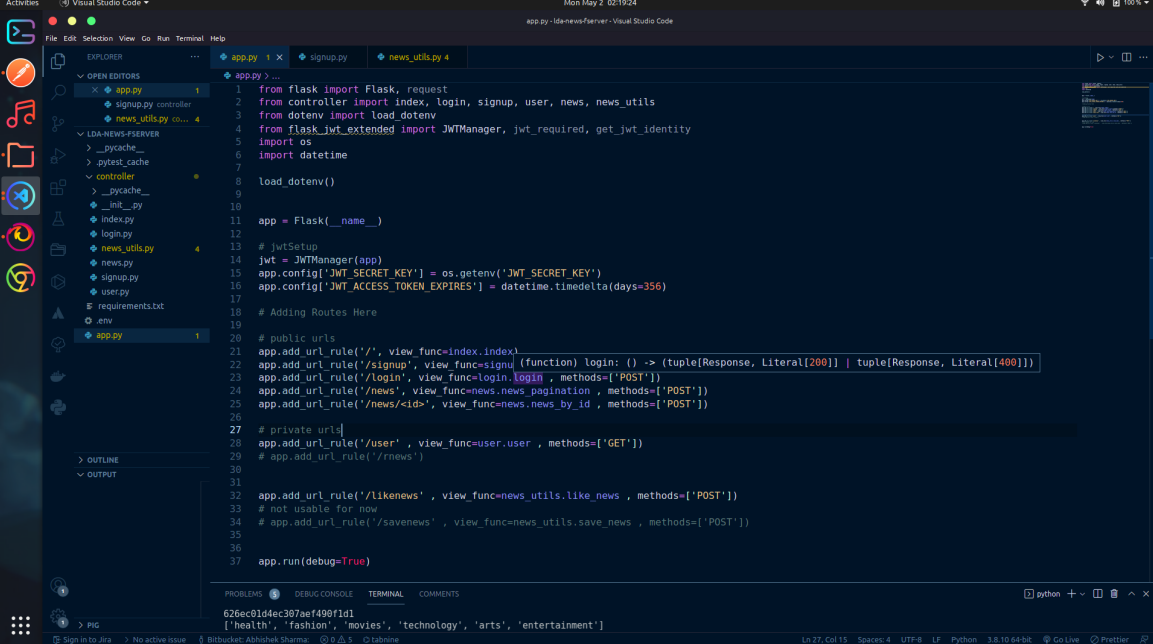
Flask Server Setup with the same database and Authentication Service as API for User news. Also used Postman to test API here.



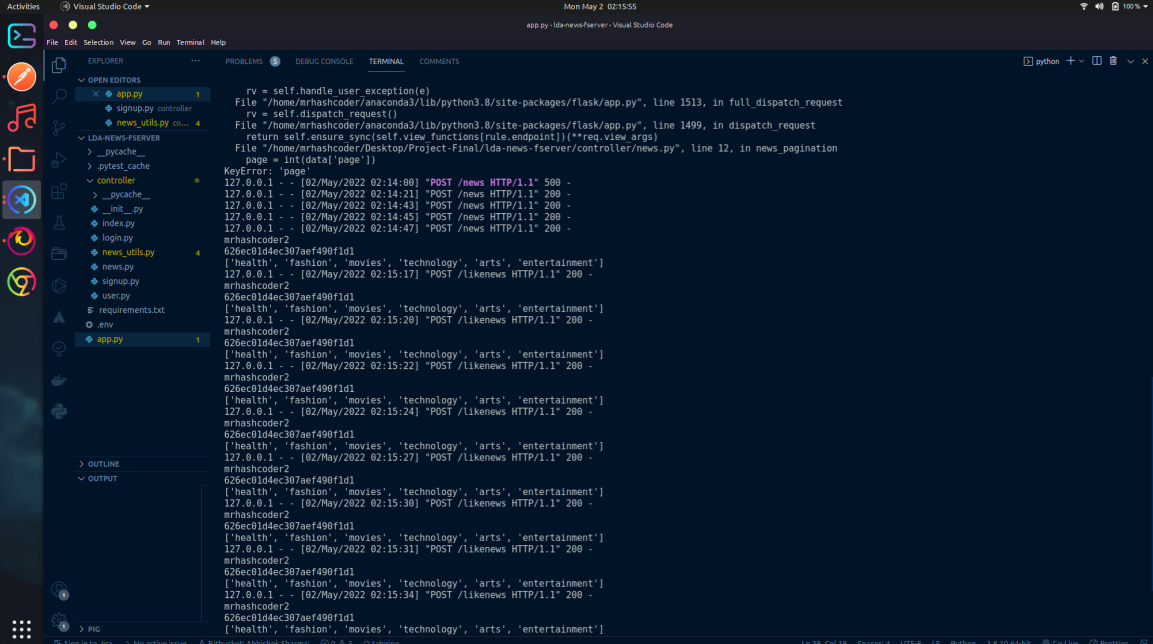


Flask Server App

This phase involves adding Flask Server App to fetch already generated data and remodel the database to recommend topics to the user. Provide user authentication api as service. we used jsonwebtoken along with flask for login here. pymongo library handle data communication with database.



```
1 from flask import Flask, request
2 from controller import index, login, signup, user, news, news_utils
3 from dotenv import load_dotenv
4 from flask_jwt_extended import JWTManager, jwt_required, get_jwt_identity
5 import os
6 import datetime
7
8 load_dotenv()
9
10
11 app = Flask(__name__)
12
13 # jwtSetup
14 jwt = JWTManager(app)
15 app.config['JWT_SECRET_KEY'] = os.getenv('JWT_SECRET_KEY')
16 app.config['JWT_ACCESS_TOKEN_EXPIRES'] = datetime.timedelta(days=356)
17
18 # Adding Routes Here
19
20 # public urls
21 app.add_url_rule('/', view_func=index.index)
22 app.add_url_rule('/signup', view_func=signup, methods=['POST'])
23 app.add_url_rule('/login', view_func=login, methods=['POST'])
24 app.add_url_rule('/news', view_func=news.pagination, methods=['POST'])
25 app.add_url_rule('/news/<id>', view_func=news.news_by_id, methods=['POST'])
26
27 # private urls
28 app.add_url_rule('/user', view_func=user.user, methods=['GET'])
29 app.add_url_rule('/rnews')
30
31
32 app.add_url_rule('/likenews', view_func=news_utils.like_news, methods=['POST'])
33 # not usable for now
34 # app.add_url_rule('/save news', view_func=news_utils.save_news, methods=['POST'])
35
36
37 app.run(debug=True)
```



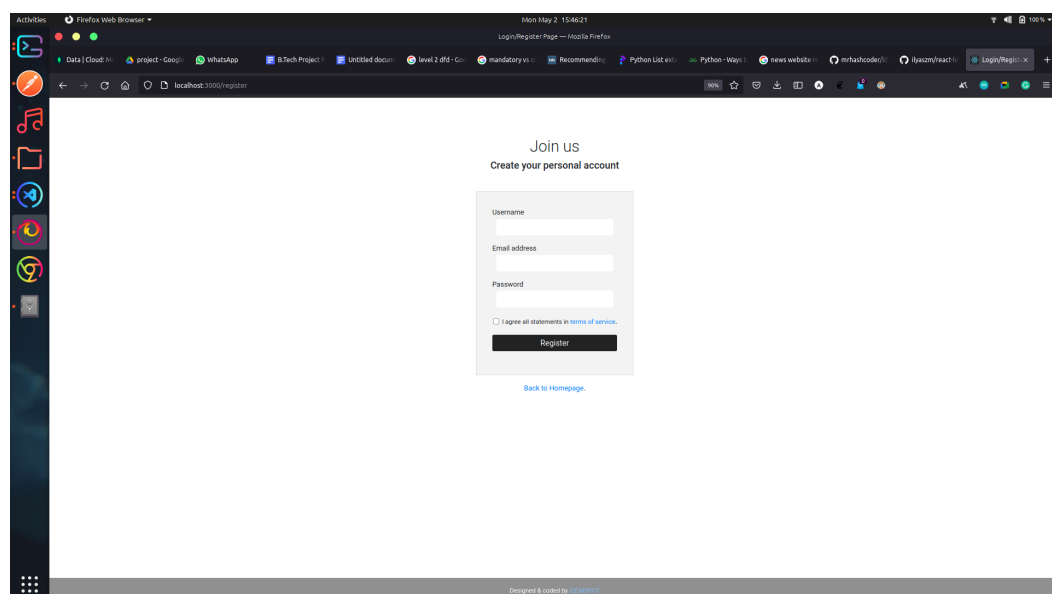
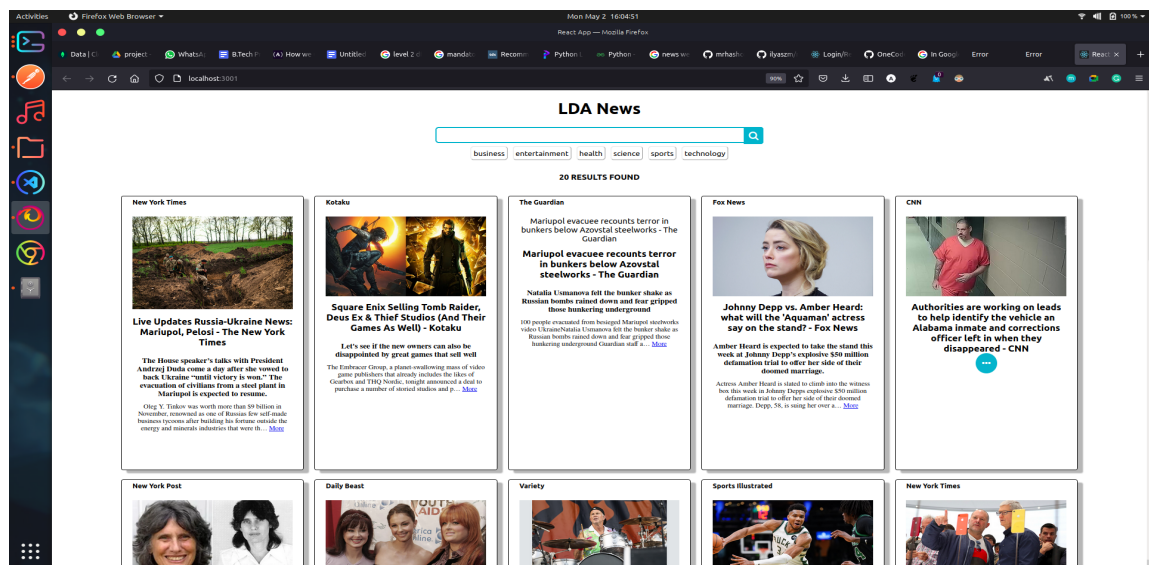
```
1 rv = self.handle_user_exception(e)
2 File "/home/mrhashcode/anaconda3/lib/python3.8/site-packages/flask/app.py", line 1513, in full_dispatch_request
3 rv = self.dispatch_request()
4 File "/home/mrhashcode/anaconda3/lib/python3.8/site-packages/flask/app.py", line 1499, in dispatch_request
5 return self.ensure_sync(self.view_functions[rule.endpoint])(**req.view_args)
6 File "/home/mrhashcode/Desktop/Project-Final/lda-news-fserver/controller/news.py", line 12, in news.pagination
7 page = int(data['page'])
8 KeyError: 'page'
9
10 127.0.0.1 - - [02/May/2022 02:14:00] "POST /news HTTP/1.1" 500 -
11 127.0.0.1 - - [02/May/2022 02:14:21] "POST /news HTTP/1.1" 200 -
12 127.0.0.1 - - [02/May/2022 02:14:43] "POST /news HTTP/1.1" 200 -
13 127.0.0.1 - - [02/May/2022 02:14:45] "POST /news HTTP/1.1" 200 -
14 127.0.0.1 - - [02/May/2022 02:14:47] "POST /news HTTP/1.1" 200 -
15 mrhashcode2
16 626ec01d4ec307aef490f1d1
17 ['health', 'fashion', 'movies', 'technology', 'arts', 'entertainment']
18 127.0.0.1 - - [02/May/2022 02:15:17] "POST /likenews HTTP/1.1" 200 -
19 mrhashcode2
20 626ec01d4ec307aef490f1d1
21 ['health', 'fashion', 'movies', 'technology', 'arts', 'entertainment']
22 127.0.0.1 - - [02/May/2022 02:15:20] "POST /likenews HTTP/1.1" 200 -
23 mrhashcode2
24 626ec01d4ec307aef490f1d1
25 ['health', 'fashion', 'movies', 'technology', 'arts', 'entertainment']
26 127.0.0.1 - - [02/May/2022 02:15:22] "POST /likenews HTTP/1.1" 200 -
27 mrhashcode2
28 626ec01d4ec307aef490f1d1
29 ['health', 'fashion', 'movies', 'technology', 'arts', 'entertainment']
30 127.0.0.1 - - [02/May/2022 02:15:24] "POST /likenews HTTP/1.1" 200 -
31 mrhashcode2
32 626ec01d4ec307aef490f1d1
33 ['health', 'fashion', 'movies', 'technology', 'arts', 'entertainment']
34 127.0.0.1 - - [02/May/2022 02:15:27] "POST /likenews HTTP/1.1" 200 -
35 mrhashcode2
36 626ec01d4ec307aef490f1d1
37 ['health', 'fashion', 'movies', 'technology', 'arts', 'entertainment']
38 127.0.0.1 - - [02/May/2022 02:15:30] "POST /likenews HTTP/1.1" 200 -
39 mrhashcode2
40 626ec01d4ec307aef490f1d1
41 ['health', 'fashion', 'movies', 'technology', 'arts', 'entertainment']
42 127.0.0.1 - - [02/May/2022 02:15:31] "POST /likenews HTTP/1.1" 200 -
43 mrhashcode2
44 626ec01d4ec307aef490f1d1
45 ['health', 'fashion', 'movies', 'technology', 'arts', 'entertainment']
46 127.0.0.1 - - [02/May/2022 02:15:34] "POST /likenews HTTP/1.1" 200 -
47 mrhashcode2
48 626ec01d4ec307aef490f1d1
49 ['health', 'fashion', 'movies', 'technology', 'arts', 'entertainment']
```



Phase 3:

User Dashboard using React

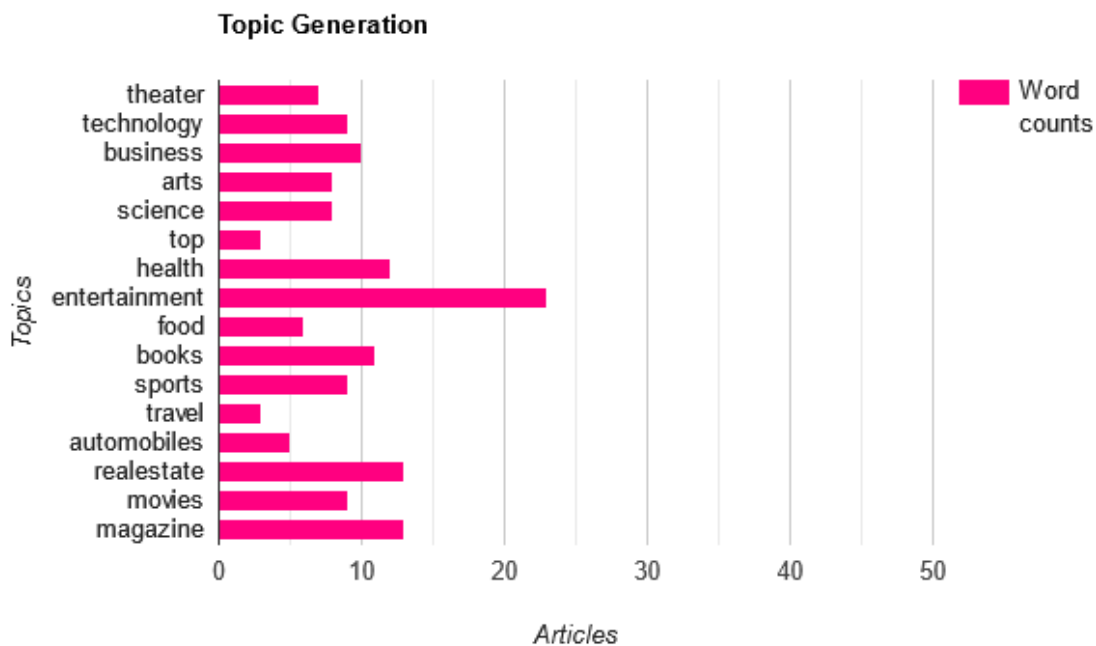
This phase involves showing user interface using react involving Visualization of News and Personalized User Authentication,





RESULTS

We used LDA for classification and it turns out to be a great method for classification. Here we are showing the result for topic generation for a dataset with 50 news. Using LDA topic generation we generated a total 23 topics for 50 news. Here is the visualization for that



We used a news dataset of the latest news from the internet for testing of classification. There were a total of 100 **news items in the json** file and each news entry had a maximum of 3 tags listed with it. We used each news entry to generate tags for news and generated 5 tags using the LDA model. Results were very intriguing. Although the result might be a little biased here as we used a dataset from the same website to train our model also. As shown in the table our lda model has **64% Accuracy** in topic generation.

Type	Count
One Match with News Tag	64
Two Match With News Tag	31
Three Match with News Tag	9



CONCLUSION AND FUTURE PLAN

In this Project, we have shown how the class of an article can be used for news recommendation. We have seen step by step how our system developed. First, we defined and formalized the problem. Then, we pre-processed the data thereby handling ambiguities and missing values. After that, we extracted two features. And lastly, we used the extracted features and features available in the data set to predict how much a user likes a news article.

The most important conclusion that we can draw from our results is that the class of the article can be used to improve a recommendation algorithm. News articles are strongly related to its class or category and this has an impact on users preferences. Using a supervised algorithm we were able to observe the impact of this feature (i.e. articles' class). Out of the many learning models i.e. linear regression, logistic regression and Naive Bayes, we used guided lda .

Another interesting approach would be to solve the problem of news recommendation as un- supervised learning problem. For example, models that are able to detect underlying variables that are enclosed in the news articles (i.e. LDA).

However, our approach is one of the many approaches and in the future we would like to propose different and more complex approaches with ultimate goal of designing a personalized news recommendation systems which are able to provide sentiment analysis and fine tune the model, word cloud for news and the accuracy of prediction of topics..

- Improve recommender system
- Add additional non-topic related features in the recommendation system
- Further optimize hyperparameters to the LDA models
- Sentiment analysis



REFERENCES

- [1] Khloponin, P., & Kosseim, L. (2021, June). [Online]. Using Document Embeddings for Background Linking of News Articles. In the International Conference on Applications of Natural Language to Information Systems (pp. 317-329). Springer, Cham. Available: https://doi.org/10.1007/978-3-030-80599-9_28. [Accessed 01 05 2022].
- [2] Ma, Y., Zong, L., Yang, Y., & Su, J. (2019, November). [Online]. News2vec: News network embedding with subnode information. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 4843-4852). Available: [10.18653/v1/D19-1490](https://arxiv.org/abs/10.18653/v1/D19-1490). [Accessed 01 05 2022].
- [3] Singh, V. (2017, October). [Online]. How we changed unsupervised LDA to SemiSupervised GuidedLDA. Machine learning. Available: <https://www.freecodecamp.org/news/how-we-changed-unsupervised-lda-to-semi-supervised-guidedlda-e36a95f3a164>. [Accessed 01 05 2022].

APPENDIX



COMPLETE CONTRIBUTORY SOURCE CODE

For Complete Code Visit:

Model: <https://github.com/mrhashcoder/lda-news-backend/>

Server: <https://github.com/mrhashcoder/lda-news-fserver/>

Frontend: <https://github.com/mrhashcoder/lda-news-frontend/>

Flask Server Code

```
from flask import Flask, request
from controller import index, login, signup, user, news, news_utils, rnews, tag_utils
from dotenv import load_dotenv
from flask_jwt_extended import JWTManager, jwt_required, get_jwt_identity
import os
import datetime

load_dotenv()

app = Flask(__name__)

# jwtSetup
jwt = JWTManager(app)
app.config['JWT_SECRET_KEY'] = os.getenv('JWT_SECRET_KEY')
app.config['JWT_ACCESS_TOKEN_EXPIRES'] = datetime.timedelta(days=356)

# Adding Routes Here

# public urls
app.add_url_rule('/', view_func=index.index)
app.add_url_rule('/signup', view_func=signup.signup, methods=['POST'])
app.add_url_rule('/login', view_func=login.login, methods=['POST'])
app.add_url_rule('/news', view_func=news.news_pagination, methods=['POST'])
app.add_url_rule('/news/<id>', view_func=news.news_by_id, methods=['POST'])
app.add_url_rule('/taglist', view_func=tag_utils.tag_list, methods=['POST'])
app.add_url_rule('/newsbytag', view_func=tag_utils.news_by_tag, methods=['POST'])
app.add_url_rule('/taglistcount', view_func=tag_utils.tag_list_with_count, methods=['POST'])
```



```
# private urls
app.add_url_rule('/user', view_func=user.user , methods=['GET'])
app.add_url_rule('/rnews', view_func=rnews.recommand_news, methods=['POST'])

app.add_url_rule('/likenews', view_func=news_utils.like_news , methods=['POST'])
# not usable for now
# app.add_url_rule('/savenews', view_func=news_utils.save_news , methods=['POST'])

app.run(debug=True)
```

Auth Flask Code :

```
from flask import request, jsonify
import os
from numpy import identity
from pymongo import MongoClient
from flask_jwt_extended import create_access_token

def login():
    user = request.get_json()

    if(validate_request(user)):

        # MongoDB Connection
        mongoURI = os.getenv('MONGO_URI')
        client = MongoClient(mongoURI)
        db = client.newsDB
        userCollection = db.users

        # Check if User is Correct
        user_exists = userCollection.find_one( {"$and" : [{ 'username' : user['username'] } , { 'password' :
user['password'] } ] })

        if user_exists:
            jwt_token = create_access_token(identity=user['username'])
            return jsonify( {
                "status" : 200,
                "message" : "User Logged",
                "jwt_token" : jwt_token
```



```
}), 200
```

```
else:
```

```
return jsonify({
```

```
"status" : "200",
```

```
"message" : "Invalid username or password"
```

```
}), 200
```

```
else:
```

```
return jsonify({
```

```
"status" : 400,
```

```
"message" : "Bad Request",
```

```
}), 400
```

```
def validate_request(user):
```

```
    try:
```

```
        if(user["username"] == None or user["username"].strip() == ""):
```

```
            return False
```

```
        if(user["password"] == None or user["password"].strip() == ""):
```

```
            return False
```

```
    return True
```

```
    except:
```

```
        return False
```