LDA MODEL FOR ANALYZING COURSE DESCRIPTIONS & VISUALIZING CREATED GRAPH USING GEPHI

.

A Project

Presented to the

Faculty of

University of North Carolina,

Greensboro

.

In Partial Fulfillment

of the Requirements for the Degree

Master of (Arts/Science)

in

Computer Science

.

by

(Saipavan Tadikonda)

(May 2023)

ABSTRACT

Topic Modeling falls under unsupervised technique which aims to analyze the large volumes of text data by clustering documents into groups. It tries to group the documents into clusters based on similar characteristics. One of the approaches which is mainly used for topic modeling is LDA (Latent Dirichlet Allocation). The goal of this project is to develop a LDA (Latent Dirichlet Allocation) topic model for course descriptions across the selected four universities North Carolina State University (NCSU), University of North Carolina at Chapel Hill (UNC), University of North Carolina at Charlotte (UNCC) and Georgia State University (GSU), find the courses to obtained LDA topics similarity, create a networkx graph with node attributes and edge attributes, visualize the created graph using visualization tool Gephi and finding out the cluster which contains topics and the courses in the selected universities related to data science using different filters and applying modularity which are available in the Gephi.

Course information (Coursenumber, CourseTitle, CourseDescription, CourseDepartment and University) used for developing LDA model are web scrapped using Beautiful Soup library as it is used for pulling out data from HTML and XML files. The input to the LDA Topic model is the key words generated from cleaned, lemmatized course descriptions of the courses using corpora, corpus generated from the key words and the optimal number of topics which is obtained from perplexity and coherence plot. Node attributes of the graph are the CourseNumbers and LDA topics and edge attributes are the weight between the course numbers and topics.

Libraries used: Beautiful Soup, genism, networkx, pyLDAvis, corpora, nltk and matplotlib.

Programming Language used: Python   Visualization tool: Gephi.

ACKNOWLEDGEMENTS

It is not possible to achieve our goal in a day or overnight. It is a process of activities which follows one after other. This is a small gesture that I can express for the help of my Professors Dr. Lixin Fu and Dr. Somya Mohanty for helping, guiding, and exhibiting keen interest in my project work throughout the process.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

In the recent years with the increasing amount of unstructured data it is difficult to obtain the required information out of unstructured data but it is not impossible to obtain. With the help of technology, we can use some powerful methods to mine the data and fetch the required information. One such method in this aspect is Topic Modeling. From the name itself we can understand that it is a process to identify the topics present in the text objects and to derive the hidden patterns exhibited by a text corpus. Topic Modeling is an unsupervised approach which is used for finding and observing the bunch of words called topics in large clusters of texts. This approach is very useful for the purpose for document clustering, organizing large blocks of textual data, information retrieval from the unstructured text and feature selection. This project is one of the applications of the topic modeling. Here the unstructured data is nothing, but the course information details across the four selected universities North Carolina State University, University of North Carolina at Chapel Hill, University of North Carolina at Charlotte, and Georgia State University. Now we will see some of the libraries and its functions which are required for applying topic modeling on the above-mentioned unstructured data.

1.1 Libraries and Functions used:

Beautiful Soup [1]:

Beautiful Soup is a python library which is used for pulling data out from the HTML and XML files. This library provides a few methods for navigating, searching, and modifying a parse tree. To install this library, we need to run pip install beautifulsoup4 command. This library supports the inbuilt html parser which is included with python's standard library and also supports other third-party python parsers. This library saves programmer’s time.

Gensim [2]:

Gensim is a free open-source python library for representing documents as semantic vectors. It is designed to process raw, unstructured text using unsupervised machine learning algorithms. The algorithm which we use for this project is Latent Dirichlet Allocation (LDA, LdaModel). To install or upgrade genism library we need to run pip install –upgrade gensim.

Let us see some of the functions which I used under this gensim library.

simple_preprocess:

Under the gensim library there are various utility functions. One of the functions which we use in this project is simple_preprocess. The functionality of this function is it converts a document into a list of tokens. So, in our project each course description information of the course acts as a document and each document is converted into list of tokens.

models.Phrases:

We use this function to detect phrases on the tokens which we obtained based on the collected collocation counts. The adjacent words in the tokens that appear together more frequently than expected are joined together with the _ character.

Corpora.Dictionary:

This Dictionary function encapsulates the mapping between the normalized words i.e., the final lemmatized words of the course descriptions and their integer ids.

doc2bow:

This is the main function for this project as this function doc2bow converts the collection of the words to its bag-of-words representation. (i.e., a list of (word_id, word_frequency)2-tuples) which is nothing but the corpus. We use this corpus as one of the inputs to the LDA model.

models.LdaModel:

This function estimates Latent Dirichlet Allocation model parameters based on a training corpus.

models.coherencemodel:

This constructor initializes the four stage pipeline by accepting a coherence measure and by using the get_coherence() method we can get the topic coherence of the trained LDA model.

The other python library which I used in this project is NLTK.

NLTK [3]:

NLTK or Natural Language Toolkit is a python package which we can use for natural language processing.

One function under this library which I used in this project is stopwords.

Stopwords:

As our course description information contains many function words such as ‘the’, ‘in’, ‘of’, ‘at’ etc., that are grammatically important, but they do not carry much semantic meaning in comparison to main words such as nouns and verbs. So, by using this stopwords function we can remove those function words in our course description tokens information.

Another library which I used in this project is NetworkX.

NetworkX [4]:

NetworkX is a python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. In this project we use this library to create a network graph in the format of graphml so that we can visualize the created graph using a tool called Gephi. We can get to know more about this library and the visualization tool Gephi further in the report.

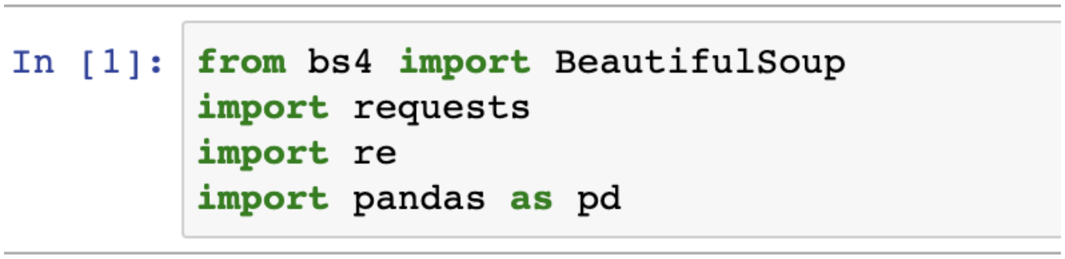
To reach the final goal, the project is divided into four stages. Below are the stages.

1) Web Scraping (Raw Data): In this stage we need to extract the required data i.e., CourseNumber, CourseTitle, CourseDescription and CourseDepartment from the selected four universities using the required library and store them into a csv file.
2) Text preprocessing: In this stage we clean the obtained raw data that includes removing unwanted spaces, unwanted text, converting the course description sentences into words, removing stop words from the words, lemmatization, finding the bi & trigrams and generating the corpus using the dictionary.
3) Topic Modeling: In this stage with the generated corpus, lemmatized texts, and the number of topics we build an LDA model and calculate coherence and perplexity measures. To build a good model, we need to find out the optimal number of topics from the coherence and perplexity plots. Then we will find the obtained topics to course similarity using the LDA model which we generated.
4) Network Build: This is the final stage of the project in which we build a network graph and store the graph in graphml format and visualize this graph using the Gephi tool and apply the modularity concept in the Gephi and find the clusters.

CHAPTER 2 WEB SCRAPING (RAW DATA)

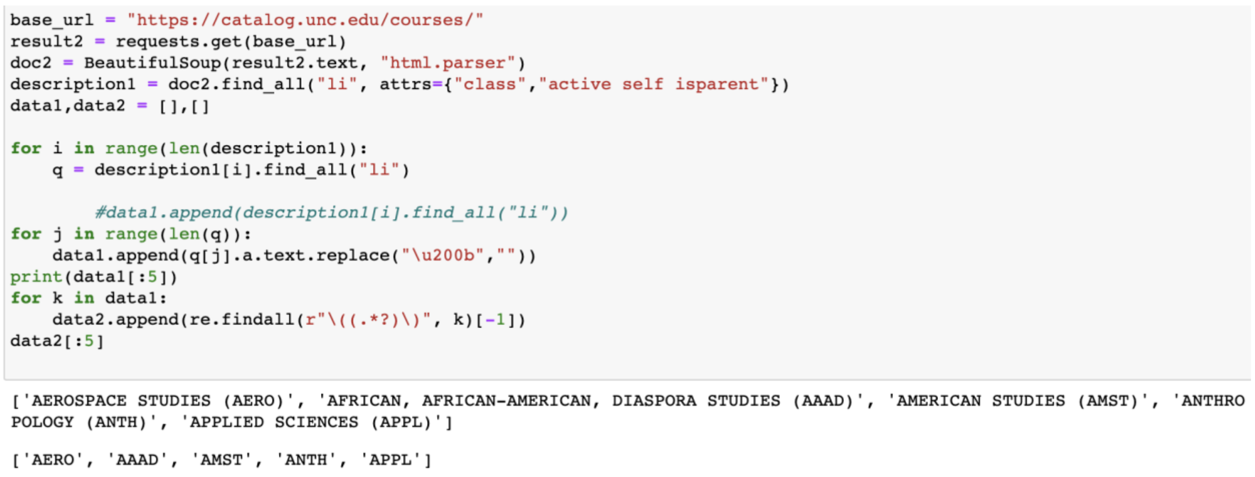
2.1 Scraping using Beautiful Soup

Each university has its own website, and, in those websites, we can observe what all courses that university is offering for both undergraduates and graduate students in course description pages. Our aim is to scrap the required details i.e., CourseNumber, CourseTitle, CourseDescription and CourseDepartment from each course description page. We can achieve this by using the library Beautiful Soup. First, we need to fetch the web page of course description using the requests library and we need to apply the html parser of the beautiful soup library on the fetched web page as it parses the html code from the text. From the obtained html code, we need to separate and store the required details which helps in navigating through all pages of the course descriptions. Then by inspecting each page of the course description we need to identify the html tags under which our coursenumber, coursetitle, coursedescription and coursedepartment information is placed and by using those html tags we need to scrap our required details and store them into a csv file. Below is one of the examples of scrapping the required details from one of the selected universities i.e., University of Chapel Hill (https://catalog.unc.edu/courses/)



```
In [1]: from bs4 import BeautifulSoup
        import requests
        import re
        import pandas as pd
```

Fig 1. Libraries used for scrapping and storing data into csv file.

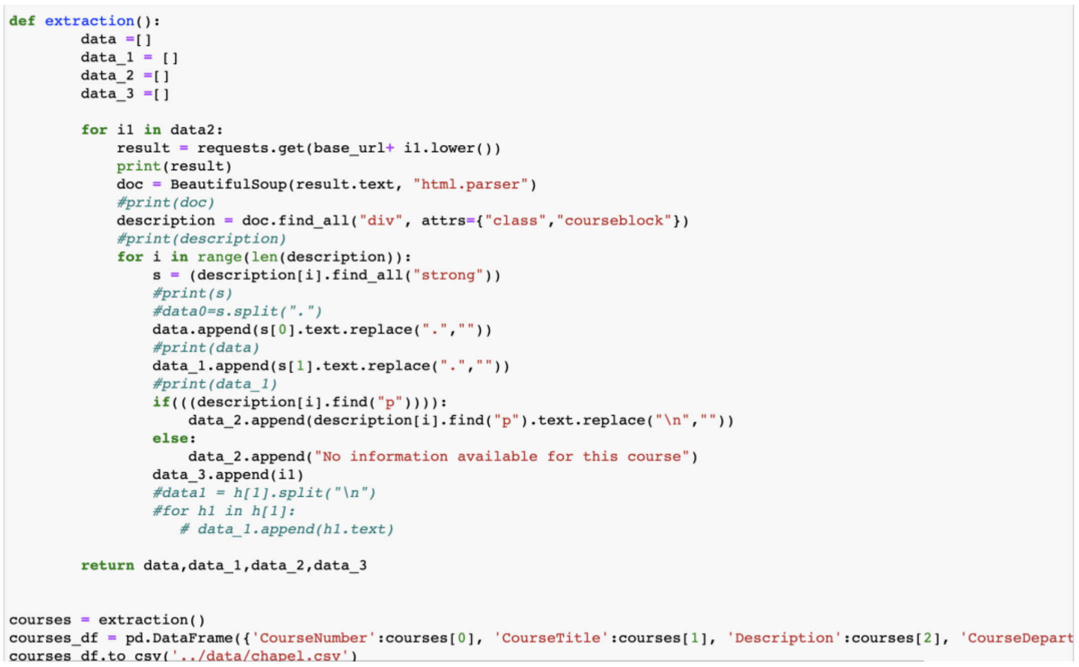


```
base_url = "https://catalog.unc.edu/courses/"
result2 = requests.get(base_url)
doc2 = BeautifulSoup(result2.text, "html.parser")
description1 = doc2.find_all("li", attrs={"class","active self isparent"})
data1,data2 = [],[]

for i in range(len(description1)):
    q = description1[i].find_all("li")

        #data1.append(description1[i].find_all("li"))
for j in range(len(q)):
    data1.append(q[j].a.text.replace("\u200b",""))
print(data1[:5])
for k in data1:
    data2.append(re.findall(r"\((.*?)\)", k)[-1])
data2[:5]
```

```
['AEROSPACE STUDIES (AERO)', 'AFRICAN, AFRICAN-AMERICAN, DIASPORA STUDIES (AAAD)', 'AMERICAN STUDIES (AMST)', 'ANTHRO
POLOGY (ANTH)', 'APPLIED SCIENCES (APPL)']

['AERO', 'AAAD', 'AMST', 'ANTH', 'APPL']
```

Fig 2: Code which gives course departments and the extension for navigating through each course department descriptions page.

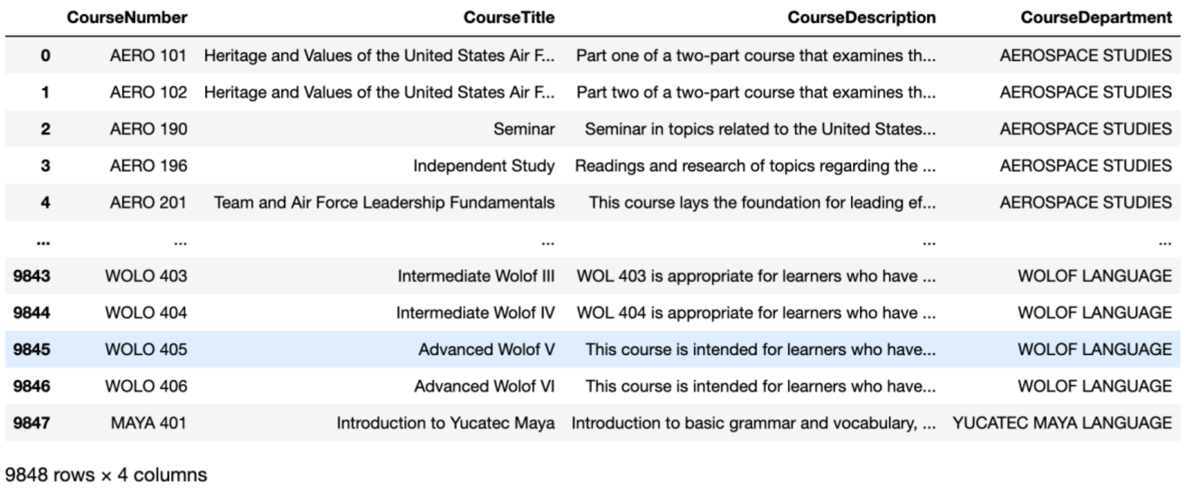
From the above Fig2 we can observe that first we are getting the web page by providing the URL using the requests library and then we are parsing the html code using the html parser and then we are getting one of the required details i.e., course department (i.e., data1 in the Fig2) and also the extensions (data2 in the Fig2) which helps us to navigate each course department descriptions page.



```
def extraction():
        data =[]
        data_1 = []
        data_2 =[]
        data_3 =[]

        for i1 in data2:
            result = requests.get(base_url+ i1.lower())
            print(result)
            doc = BeautifulSoup(result.text, "html.parser")
            #print(doc)
            description = doc.find_all("div", attrs={"class","courseblock"})
            #print(description)
            for i in range(len(description)):
                s = (description[i].find_all("strong"))
                #print(s)
                #data0=s.split(".")
                data.append(s[0].text.replace(".",""))
                #print(data)
                data_1.append(s[1].text.replace(".",""))
                #print(data_1)
                if((description[i].find("p"))):
                    data_2.append(description[i].find("p").text.replace("\n",""))
                else:
                    data_2.append("No information available for this course")
                data_3.append(i1)
                #data1 = h[1].split("\n")
                #for h1 in h[1]:
                   # data_1.append(h1.text)

        return data,data_1,data_2,data_3


courses = extraction()
courses_df = pd.DataFrame({'CourseNumber':courses[0], 'CourseTitle':courses[1], 'Description':courses[2], 'CourseDepart
courses_df.to_csv('../data/chapel.csv')
```

Fig3. Code that navigates to each course department page and extract the required details using the extensions (data2 from Fig2) obtained and storing the obtained details into a csv file.

The code in the above Fig3 shows how to use the extensions obtained to navigate through all the pages and extract the coursenumber, coursetitle, coursedescription if exists and coursedepartment information. The final data frame before storing it into a csv file which contains the above-mentioned details is as follows:



	CourseNumber	CourseTitle	CourseDescription	CourseDepartment
0	AERO 101	Heritage and Values of the United States Air F...	Part one of a two-part course that examines th...	AEROSPACE STUDIES
1	AERO 102	Heritage and Values of the United States Air F...	Part two of a two-part course that examines th...	AEROSPACE STUDIES
2	AERO 190	Seminar	Seminar in topics related to the United States...	AEROSPACE STUDIES
3	AERO 196	Independent Study	Readings and research of topics regarding the ...	AEROSPACE STUDIES
4	AERO 201	Team and Air Force Leadership Fundamentals	This course lays the foundation for leading ef...	AEROSPACE STUDIES
...	...	...	...	...
9843	WOLO 403	Intermediate Wolof III	WOL 403 is appropriate for learners who have ...	WOLOF LANGUAGE
9844	WOLO 404	Intermediate Wolof IV	WOL 404 is appropriate for learners who have ...	WOLOF LANGUAGE
9845	WOLO 405	Advanced Wolof V	This course is intended for learners who have...	WOLOF LANGUAGE
9846	WOLO 406	Advanced Wolof VI	This course is intended for learners who have...	WOLOF LANGUAGE
9847	MAYA 401	Introduction to Yucatec Maya	Introduction to basic grammar and vocabulary, ...	YUCATEC MAYA LANGUAGE

9848 rows × 4 columns

Fig 4. Course details obtained from the University of North Carolina Chapel Hill

This is the scrapping procedure followed for the University of North Carolina Chapel Hill. We need to do the scrapping for the remaining three universities i.e., North Carolina State University, University of North Carolina at Charlotte, and Georgia State University in this manner. Procedure differs because each university website is different but the order of steps which followed remains the same. After scrapping add the respective University column to each data frame and finally combine them into a one csv file as we will be building one LDA model further for all these four selected universities.

CHAPTER 3 TEXT PREPROCESSING

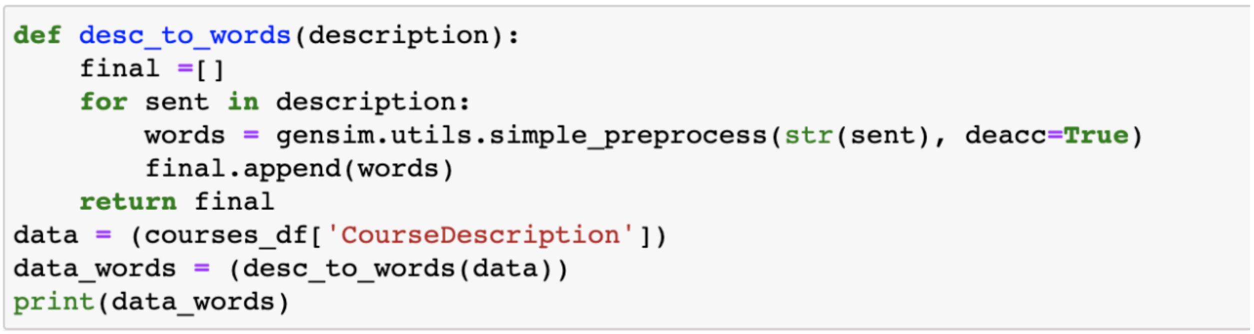
In the previous stage we stored our final combined course details of all selected four universities into a csv file. The goal of this stage is to preprocess the text i.e., cleaning the data, converting course descriptions sentences into words, stop words removal, find bigrams, trigrams, finding the lemmatized version of the words and generate the corpus which is required for the building the LDA model.

3.1 Data Cleanup:

In this step read the stored csv file into a data frame using the panda's library. Then remove the extra spaces in the beginning and in the ending if there exists any by using the strip () function. Also confirm if there are any nulls present in the data frame. If we observe every department in each university have some common courses like Directed Study, Independent Study, Thesis, Project, etc. which doesn't contribute much to build the LDA model. So, remove the repeated courses from the data frame so that our final LDA model will be efficient.

3.2 Sentences into words:

At the end of the previous we obtained the cleaned data frame. But the data under course descriptions is in sentence format. To build an LDA model on the course descriptions column we need to convert each course description into words as the model can only understand and get useful information from the words. Also, the LDA model is an unsupervised machine learning model. Using the simple_preprocess () function which is provided by the gensim library, convert each course description sentence into words.

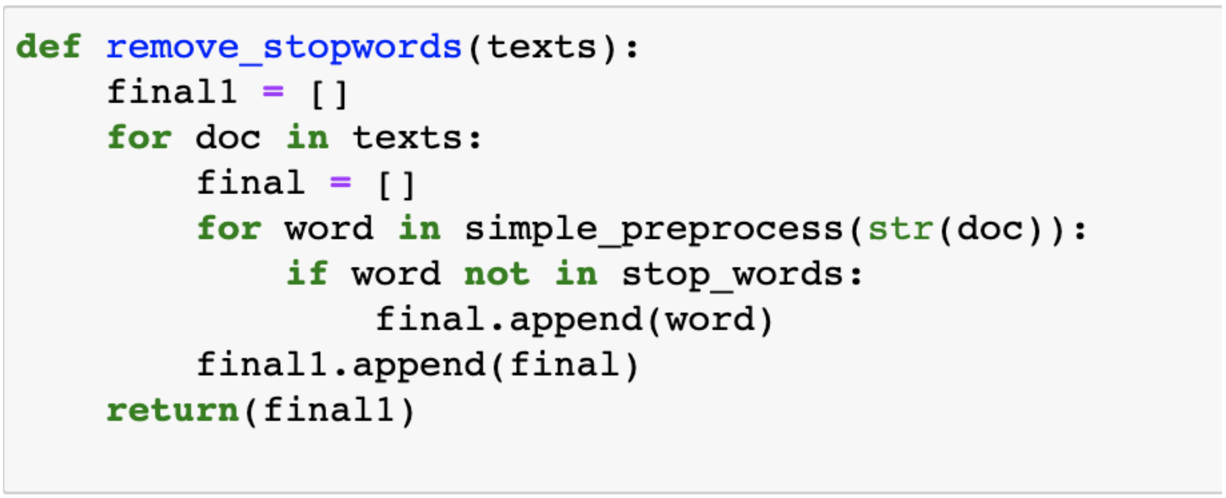


```python
def desc_to_words(description):
    final =[]
    for sent in description:
        words = gensim.utils.simple_preprocess(str(sent), deacc=True)
        final.append(words)
    return final
data = (courses_df['CourseDescription'])
data_words = (desc_to_words(data))
print(data_words)
```

Fig. 5 Code used to convert course descriptions into words.

3.3 Removal of Stopwords:

From the obtained words of each course description, we need to remove the words which don't add much semantic value. So, by using the stopwords from the NLTK library we can remove all the words which don't add much meaning.



```python
def remove_stopwords(texts):
    final1 = []
    for doc in texts:
        final = []
        for word in simple_preprocess(str(doc)):
            if word not in stop_words:
                final.append(word)
        final1.append(final)
    return(final1)
```

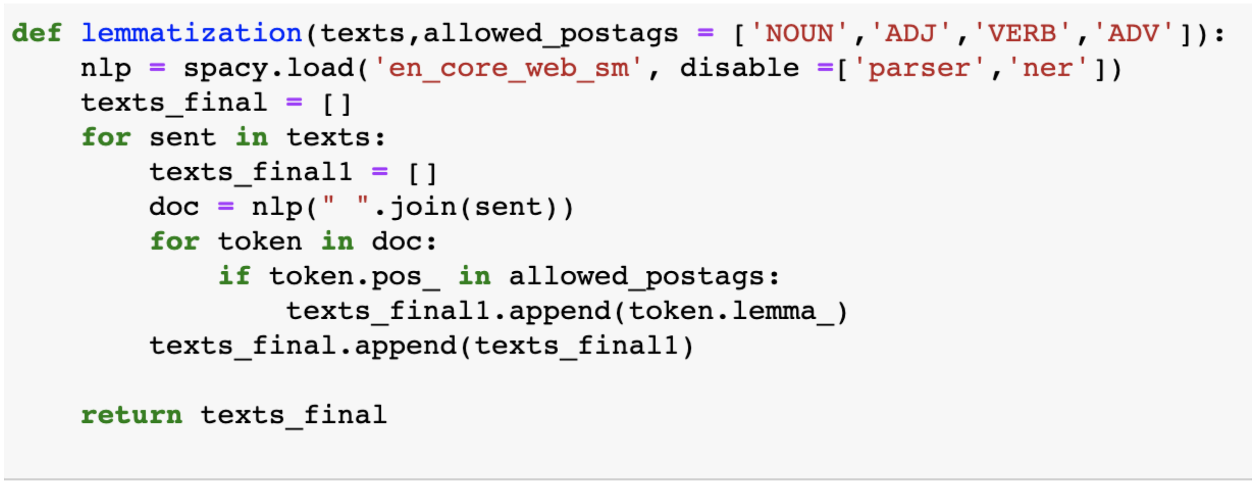
Fig. 6 Code used to remove the stopwords from the words obtained.

3.4 Bi and Trigrams:

After removing the stopwords from the words obtained we need to find if there are any bigrams i.e., any two words occurring together frequently and trigrams i.e., any three words occurring together frequently. We can do this by using the Phrases () function provided by the gensim library. So here if there are any bigrams then those words will be joined by ‘_’ and if there are any trigrams even those three words are joined by ‘_’.

3.5 Lemmatization:

Lemmatization is nothing but converting the word into its root word. One example is the root word of machines would be machine. We need to lemmatize the words which we obtained after finding bigrams and trigrams as it would help the topic model to categorize things more easily. Below is the code which is used to lemmatize the words.



```
def lemmatization(texts,allowed_postags = ['NOUN','ADJ','VERB','ADV']):
    nlp = spacy.load('en_core_web_sm', disable =['parser','ner'])
    texts_final = []
    for sent in texts:
        texts_final1 = []
        doc = nlp(" ".join(sent))
        for token in doc:
            if token.pos_ in allowed_postags:
                texts_final1.append(token.lemma_)
        texts_final.append(texts_final1)

    return texts_final
```

Fig. 7 Code which is used to perform lemmatization on the data.

From the above Fig 7 we can observe that this function accepts the texts and returns only the lemmatized version of words for each course description which are nouns, adjectives, verbs, and adverbs.

3.6 Creating Dictionary and Corpus:

Dictionary (Id2word) and corpus are the two essential inputs for the LDA model. Id2word identifies the keywords of the words that we obtained after lemmatization and gives ids to each word. This can be done by using the Dictionary () function from the gensim.corpora library. Coming to the corpus it contains the id and frequency of that word in the form of tuple for each course description. We can achieve this by using the doc2bow

() function which is used to convert the collection of words into a bag of word representation.

By the end of this stage, we obtained the lemmatized version of the texts i.e., course descriptions, dictionary and corpus which are the essential inputs for training an LDA model.

CHAPTER 4 TOPIC MODELING

As mentioned before Topic modeling is a technique to extract the hidden topics from large volumes of text. Latent Dirichlet Allocation (LDA) is a popular algorithm for topic modeling. The goal of this stage is to train an LDA model which extracts good quality topics.

4.1 Inputs for LDA Model:

Apart from the inputs which we obtained in the previous stage i.e., Dictionary and corpus we also need some other inputs such as number of topics, Chunksize, Update_every and passes. Chunksize is nothing but the number of documents to be used in each training chunk. Update_every is nothing but how often the model parameters should be updated. Passes are nothing but the total number of the training passes. Basically, the LDA model considers each document as a collection of topics in a certain proportion and each topic as a collection of keywords in a certain proportion. If we provide the LDA model with a number of topics, it rearranges the topics distribution within the documents. But the efficiency of the LDA model depends on the number of topics given. We can find the optimal number of topics for the LDA model by plotting perplexity vs coherence plot.

4.2 Perplexity and Coherence:

Perplexity and Coherence score is used as a convenient measure to judge how good the trained model is performing. Perplexity tries to measure how the trained model is surprised when it is given with the new data. Generally, the lower the perplexity the better is the model. Coherence score is used to distinguish between the good and the bad topics. Generally, a score more than 0.4 is considered good. So, if we plot the graph of perplexity and coherence values for different numbers of topics, we can get the ideal number of topics where the perplexity and coherence values meet. Below is the plot

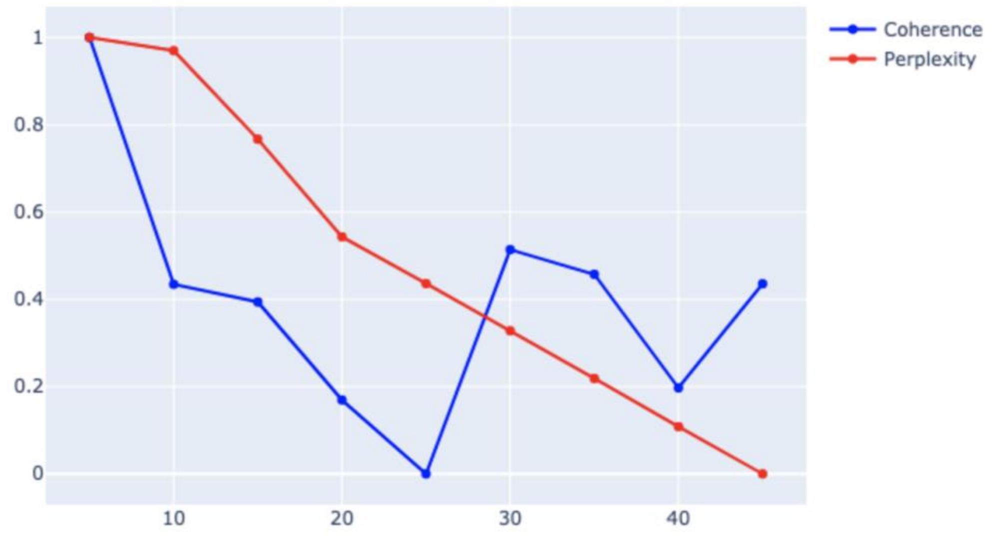
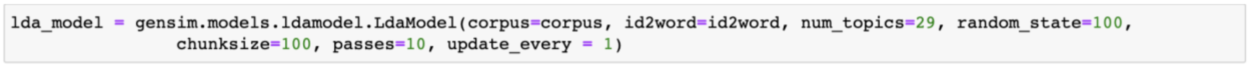


Fig. 8 Optimal number of topics obtained.

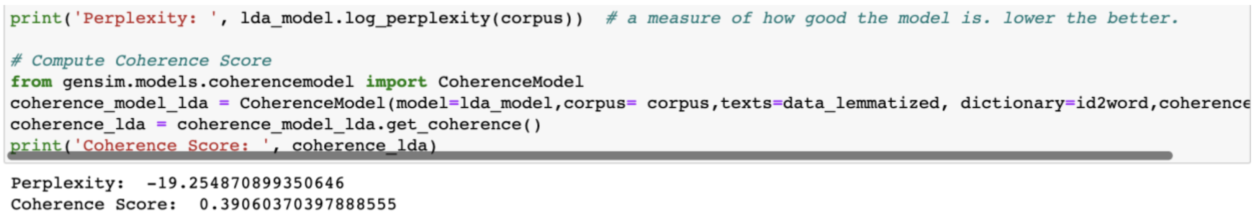
From the above Fig.8, we can observe that both perplexity and coherence values meet at 29. Therefore, the optimal number of topics for our LDA model would be 29. Now we have all the required inputs to train an LDA model. If we provide these inputs to the genism's library LDA model using the following code, we will get the LDA topics.



```
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus, id2word=id2word, num_topics=29, random_state=100,
                chunksize=100, passes=10, update_every = 1)
```

Fig. 9 Code used to build the LDA model.

The perplexity and coherence score obtained for the above trained LDA model.

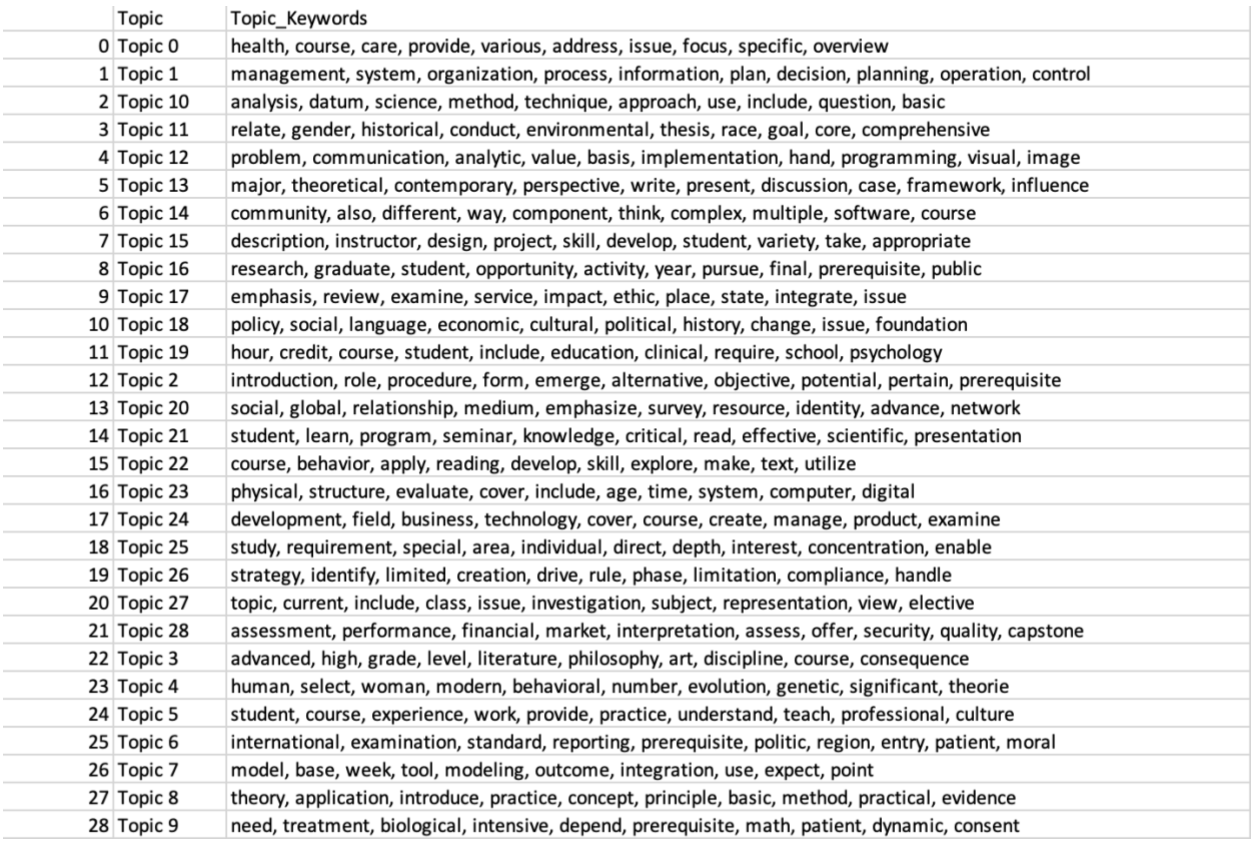


```
print('Perplexity: ', lda_model.log_perplexity(corpus))  # a measure of how good the model is. lower the better.

# Compute Coherence Score
from gensim.models.coherencemodel import CoherenceModel
coherence_model_lda = CoherenceModel(model=lda_model,corpus= corpus,texts=data_lemmatized, dictionary=id2word,coherence
coherence_lda = coherence_model_lda.get_coherence()
print('Coherence Score: ', coherence_lda)
```

```
Perplexity:  -19.254870899350646
Coherence Score:  0.39060370397888555
```

Fig.10 Perplexity and the Coherence Scores obtained for the LDA model.

The LDA topics obtained for the trained LDA model is as follows:



	Topic	Topic_Keywords
0	Topic 0	health, course, care, provide, various, address, issue, focus, specific, overview
1	Topic 1	management, system, organization, process, information, plan, decision, planning, operation, control
2	Topic 10	analysis, datum, science, method, technique, approach, use, include, question, basic
3	Topic 11	relate, gender, historical, conduct, environmental, thesis, race, goal, core, comprehensive
4	Topic 12	problem, communication, analytic, value, basis, implementation, hand, programming, visual, image
5	Topic 13	major, theoretical, contemporary, perspective, write, present, discussion, case, framework, influence
6	Topic 14	community, also, different, way, component, think, complex, multiple, software, course
7	Topic 15	description, instructor, design, project, skill, develop, student, variety, take, appropriate
8	Topic 16	research, graduate, student, opportunity, activity, year, pursue, final, prerequisite, public
9	Topic 17	emphasis, review, examine, service, impact, ethic, place, state, integrate, issue
10	Topic 18	policy, social, language, economic, cultural, political, history, change, issue, foundation
11	Topic 19	hour, credit, course, student, include, education, clinical, require, school, psychology
12	Topic 2	introduction, role, procedure, form, emerge, alternative, objective, potential, pertain, prerequisite
13	Topic 20	social, global, relationship, medium, emphasize, survey, resource, identity, advance, network
14	Topic 21	student, learn, program, seminar, knowledge, critical, read, effective, scientific, presentation
15	Topic 22	course, behavior, apply, reading, develop, skill, explore, make, text, utilize
16	Topic 23	physical, structure, evaluate, cover, include, age, time, system, computer, digital
17	Topic 24	development, field, business, technology, cover, course, create, manage, product, examine
18	Topic 25	study, requirement, special, area, individual, direct, depth, interest, concentration, enable
19	Topic 26	strategy, identify, limited, creation, drive, rule, phase, limitation, compliance, handle
20	Topic 27	topic, current, include, class, issue, investigation, subject, representation, view, elective
21	Topic 28	assessment, performance, financial, market, interpretation, assess, offer, security, quality, capstone
22	Topic 3	advanced, high, grade, level, literature, philosophy, art, discipline, course, consequence
23	Topic 4	human, select, woman, modern, behavioral, number, evolution, genetic, significant, theorie
24	Topic 5	student, course, experience, work, provide, practice, understand, teach, professional, culture
25	Topic 6	international, examination, standard, reporting, prerequisite, politic, region, entry, patient, moral
26	Topic 7	model, base, week, tool, modeling, outcome, integration, use, expect, point
27	Topic 8	theory, application, introduce, practice, concept, principle, basic, method, practical, evidence
28	Topic 9	need, treatment, biological, intensive, depend, prerequisite, math, patient, dynamic, consent

Fig.11 LDA Topics and their keywords.

There is also a package named pyLDAvis which provides the interactive chart to visualize the obtained LDA topics. Below is the figure obtained by using the pyLDAvis on the LDA topics obtained.

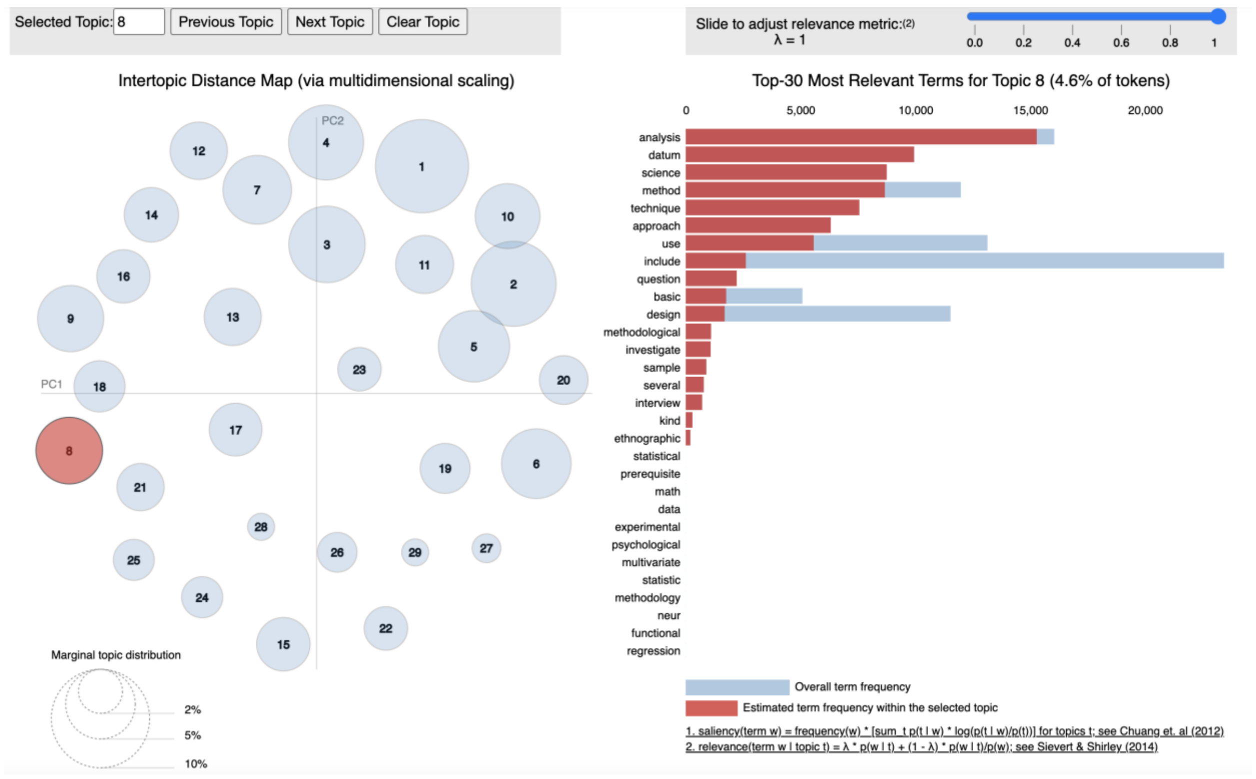
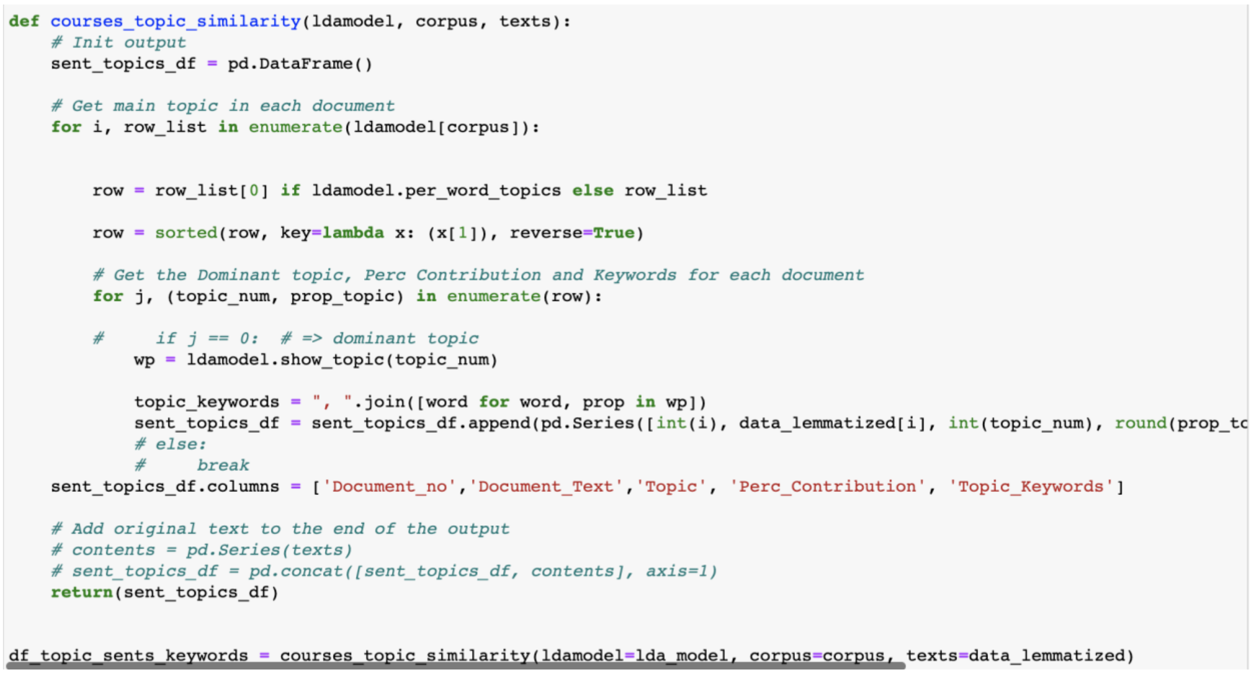

Selected Topic: 8
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 1
0.0
0.2
0.4
0.6
0.8
1
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
Marginal topic distribution
2%
5%
10%
Top-30 Most Relevant Terms for Topic 8 (4.6% of tokens)
0
5,000
10,000
15,000
20,000
analysis
datum
science
method
technique
approach
use
include
question
basic
design
methodological
investigate
sample
several
interview
kind
ethnographic
statistical
prerequisite
math
data
experimental
psychological
multivariate
statistic
methodology
neur
functional
regression
Overall term frequency
Estimated term frequency within the selected topic
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)

Fig. 12 pyLDAvis on the LDA topics obtained.

From the above Fig.12, you can observe the bubbles on the left-hand side of the figure where each bubble represents a topic. The bigger the bubble, the more prevalent the topic. If we move our cursor on each bubble the right-side part of the figure gets updated. It gives the keywords under that topic.

4.3 Topic to Course Relationship:

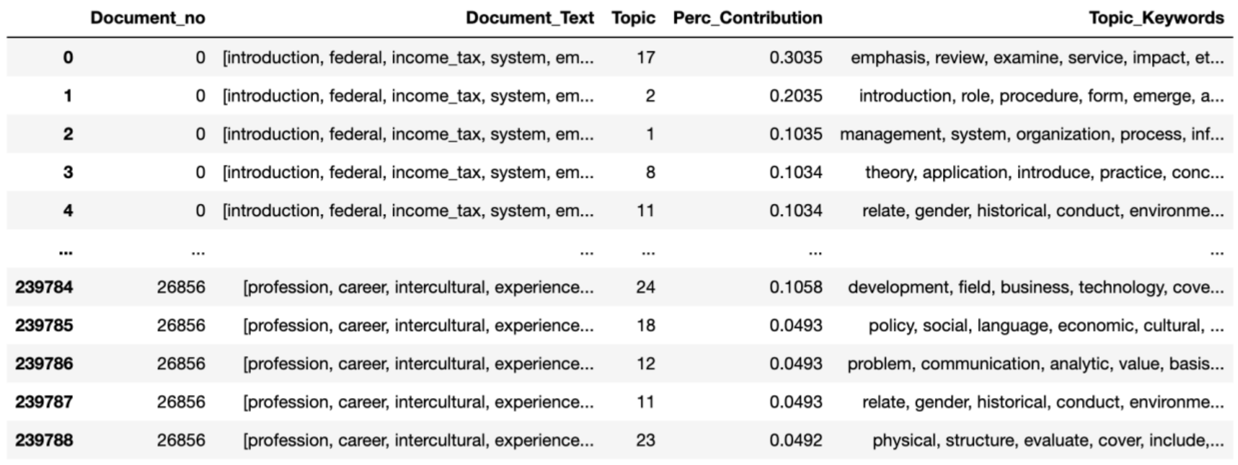
We obtained the LDA topics using the gensim's LDA model. In this step we will find relation between the courses and the topics obtained. i.e., what topics among the obtained topics relates to each course and with what percentage contribution that course is related to the each obtained topics. To obtain this we need the lda model, corpus, and the lemmatized texts. Below is the image which contains the code that is used to find the topics to courses relationship. A function is written which accepts the lda model, corpus and the lemmatized texts and returns a data frame which contains the Document_no , Document_Text, Topic to which the document is related, percentage contribution and the topic keywords.



```
def courses_topic_similarity(ldamodel, corpus, texts):
    # Init output
    sent_topics_df = pd.DataFrame()

    # Get main topic in each document
    for i, row_list in enumerate(ldamodel[corpus]):


        row = row_list[0] if ldamodel.per_word_topics else row_list

        row = sorted(row, key=lambda x: (x[1]), reverse=True)

        # Get the Dominant topic, Perc Contribution and Keywords for each document
        for j, (topic_num, prop_topic) in enumerate(row):

        #     if j == 0:  # => dominant topic
            wp = ldamodel.show_topic(topic_num)

            topic_keywords = ", ".join([word for word, prop in wp])
            sent_topics_df = sent_topics_df.append(pd.Series([int(i), data_lemmatized[i], int(topic_num), round(prop_tc
            # else:
            #     break
    sent_topics_df.columns = ['Document_no','Document_Text','Topic', 'Perc_Contribution', 'Topic_Keywords']

    # Add original text to the end of the output
    # contents = pd.Series(texts)
    # sent_topics_df = pd.concat([sent_topics_df, contents], axis=1)
    return(sent_topics_df)

df_topic_sents_keywords = courses_topic_similarity(ldamodel=lda_model, corpus=corpus, texts=data_lemmatized)
```

Fig. 13 Code for finding the topic to course relationship.

The data frame that was obtained by calling this function to find the topic to course relationship is as follows:



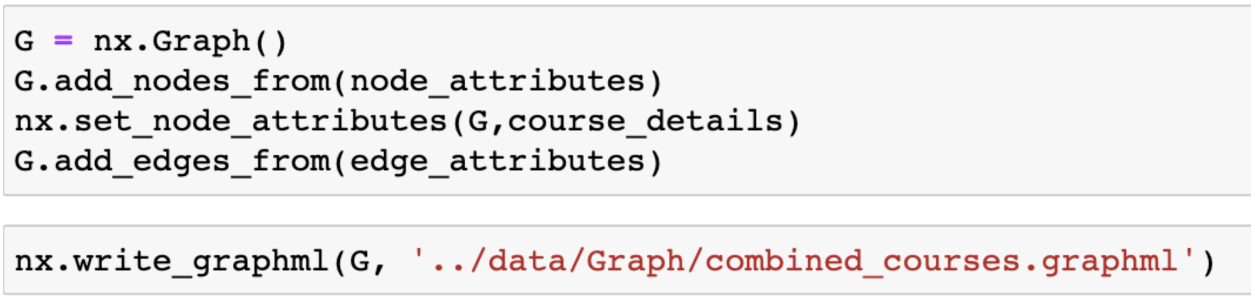
	Document_no	Document_Text	Topic	Perc_Contribution	Topic_Keywords
0	0	[introduction, federal, income_tax, system, em...	17	0.3035	emphasis, review, examine, service, impact, et...
1	0	[introduction, federal, income_tax, system, em...	2	0.2035	introduction, role, procedure, form, emerge, a...
2	0	[introduction, federal, income_tax, system, em...	1	0.1035	management, system, organization, process, inf...
3	0	[introduction, federal, income_tax, system, em...	8	0.1034	theory, application, introduce, practice, conc...
4	0	[introduction, federal, income_tax, system, em...	11	0.1034	relate, gender, historical, conduct, environme...
...	...	...	...	...	...
239784	26856	[profession, career, intercultural, experience...	24	0.1058	development, field, business, technology, cove...
239785	26856	[profession, career, intercultural, experience...	18	0.0493	policy, social, language, economic, cultural, ...
239786	26856	[profession, career, intercultural, experience...	12	0.0493	problem, communication, analytic, value, basis...
239787	26856	[profession, career, intercultural, experience...	11	0.0493	relate, gender, historical, conduct, environme...
239788	26856	[profession, career, intercultural, experience...	23	0.0492	physical, structure, evaluate, cover, include,...

Fig. 14 Data Frame obtained using the code in the Fig 13.

From the Fig 14 we can observe that the data frame contains information regarding how each document is related to what LDA topics & with what percentage contribution.

CHAPTER 5 VISUALIZATION USING GEPHI

To visualize the obtained topic to course relationship we need to create a network graph with the help of the networkx library. In this stage we will create a network graph and visualize the graph using Gephi. To create a graph, we need edge attributes and node attributes. In our case the edge attributes would be the course number, and the topic to which it is related and with what percentage contribution i.e., the weight between them. These will act as edge attributes. Coming to the nodes it contains the course numbers along with the LDA topics i.e., we need to append the obtained LDA topics to the course numbers. Also, we can set node attributes which contain the document text, course title, course description, course department and university. After obtaining both nodes and edges create a graph and add the node, node attributes and edges. Store the created graph in the graphml format.



```
G = nx.Graph()
G.add_nodes_from(node_attributes)
nx.set_node_attributes(G,course_details)
G.add_edges_from(edge_attributes)
```

```
nx.write_graphml(G, '../data/Graph/combined_courses.graphml')
```

Fig. 15 Code to add the nodes, node attributes and edges to networkx graph.

We obtained the graphml file. Now we need to visualize the created graph using the visualization tool Gephi.

5.1 Introduction to Gephi:

Gephi is an open-source software for visualizing and analyzing large network graphs. We can use this to explore, analyze, spatialize, filter, manipulate and export all types of the graphs. Gephi application is downloaded using the link [8]. After installing this application, we need to import the graphml file which we created in this application. When we open the graphml file we will be shown the import report that contains the information regarding the number of nodes, edges in the created graph. Then we could see the created graph. The image of the graph is as follows:

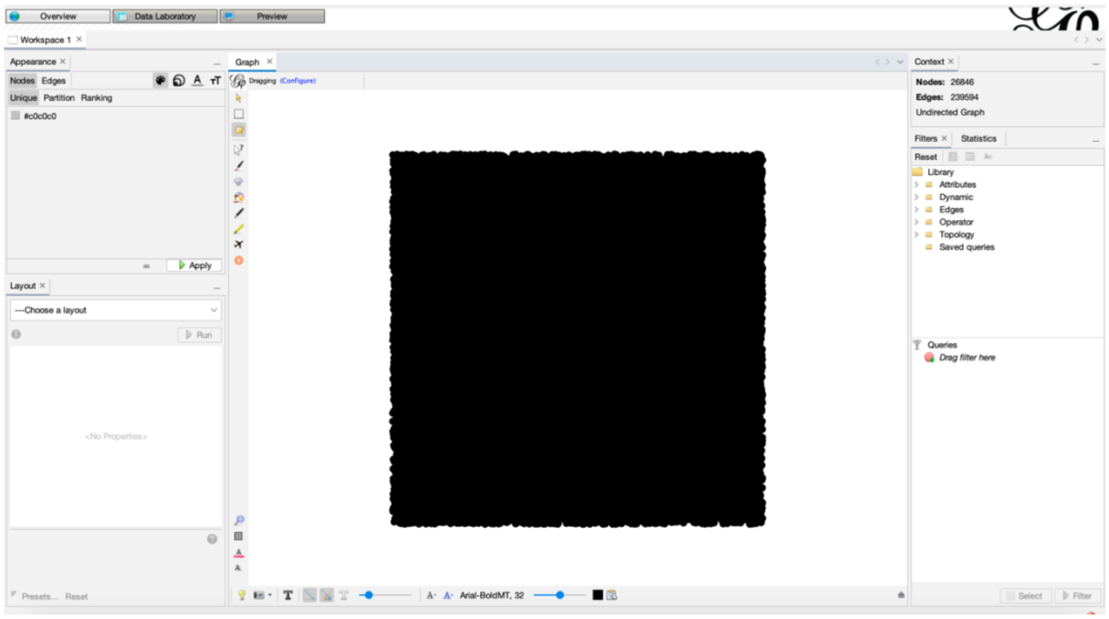

Nodes: 26846
Edges: 239594
Undirected Graph

Fig. 16 Initial Graph

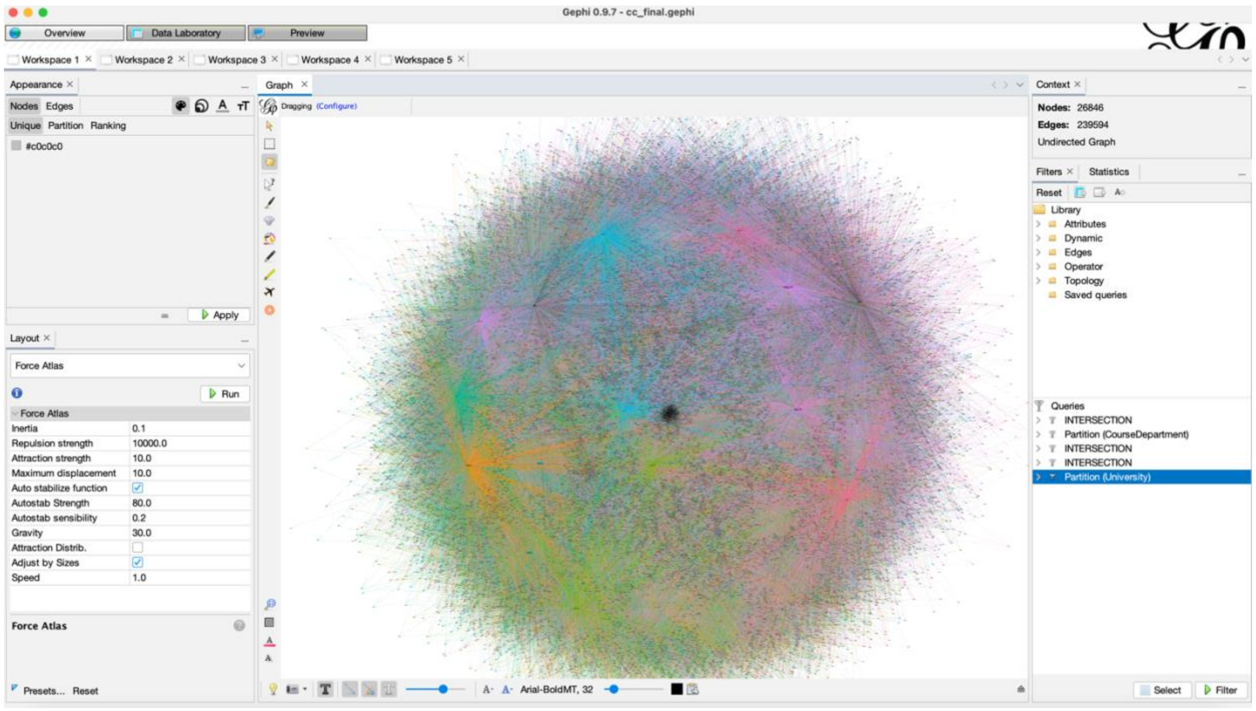
From Fig 16 we can see and understand that node positions are so close, and we cannot understand anything from the graph. We need to perform required operations. One such operation is adjusting the layout. Layout algorithms set the graph shape. It is the most essential step. In the layout module choose Force Atlas in the dropdown and in the layout, properties set the repulsion strength at 10000 to expand the graph. Run this layout algorithm until the graph expands after the expansion stop the layout algorithm. Here all the nodes move far away from other nodes. Next locate the ranking module which will be in the top left. This ranking module will help us in configuring the node’s color and size. For that to happen we need to choose the rank parameter as degree and click on apply so that we can see that change in the color based on the degree.

Next locate the Statistics module on the right panel. Under this module click the run button beside average path length. It computes the path length for all possible pairs of nodes and gives information about how nodes are close to each other. When we run the average path length, we will get three new values; those are betweenness centrality, closeness centrality and eccentricity. After obtaining these three values go back to the ranking module and select betweenness centrality in the list as this metric indicates the influential nodes for the highest value. Click on the diamond icon in the toolbar for size and set the minimum size and maximum size of the nodes. Once when we click on apply then we can see the network graph with colored nodes by the degree and size of nodes by betweenness centrality metric. Now go back to the layout module and check the adjust by sizes box in the layout properties and run the layout algorithm for a bit so that we cannot see any overlapping nodes.

Now find the ‘T’ button in the bottom of the Gephi interface and click on it as it displays the node labels and set the label size proportional to the node size so that the label of the smaller node appears small, and the label of the larger node appears big. Also set the label size with the scale slider which is available next to the label size. By now we will be able to see a graph which contains all the nodes with their respective sizes and edges between the nodes.

5.2 Community-detection:

As mentioned earlier the goal of this project is to identify the topics and courses related to the data science field which is nothing but finding the topics and courses falling under the data science community. In the statistics panel of the Gephi interface we can find the modularity if we click on run next to it, we will get a popup check on the randomize and click ok to launch the detection. In the background Gephi will implement the Louvain method to detect the communities in the graph. Now locate the Partition module which will be left to the Ranking module. In the dropdown we can see the Modularity class which is obtained by running the Modularity from the statistics panel before. A random color would be set for each community identifier and when we click on apply the graph will be updated with the respective colors based on the community they belong to. Below is the figure of the graph:



Nodes: 26846
Edges: 239594
Undirected Graph

Force Atlas

Inertia	0.1
Repulsion strength	10000.0
Attraction strength	10.0
Maximum displacement	10.0
Auto stabilize function	✓
Autostab Strength	80.0
Autostab sensibility	0.2
Gravity	30.0
Attraction Distrib.	
Adjust by Sizes	✓
Speed	1.0

Fig. 17 Graph after community detection

From the above Fig 17 we can observe the different communities represented by different colors. From those obtained communities we need to find the community which is related to data science. Below is the figure of a community which is related to data science.

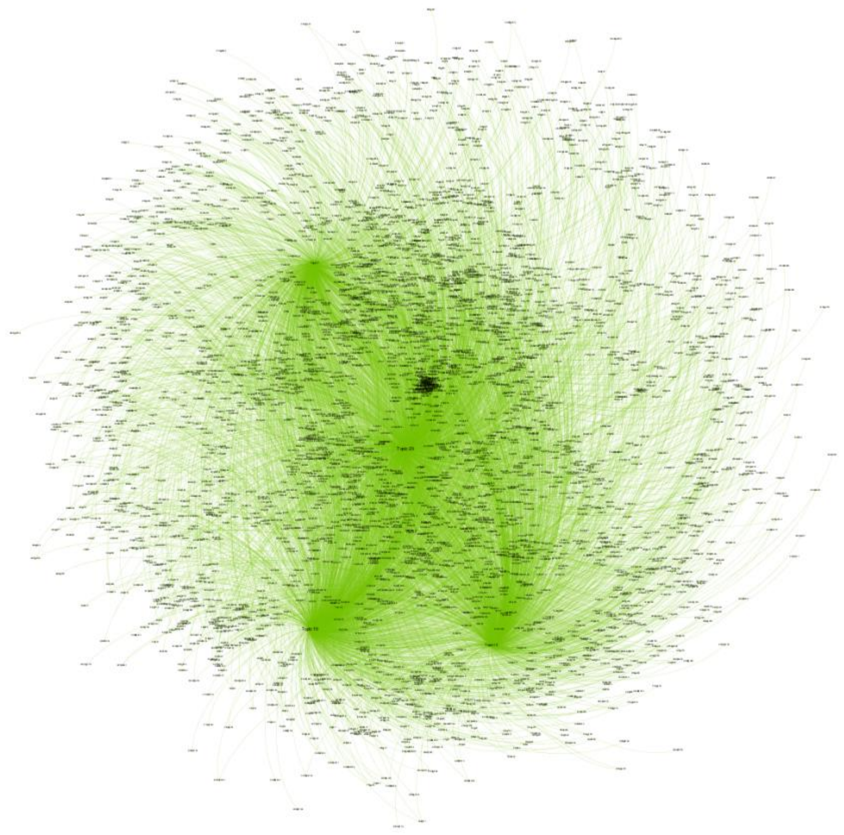
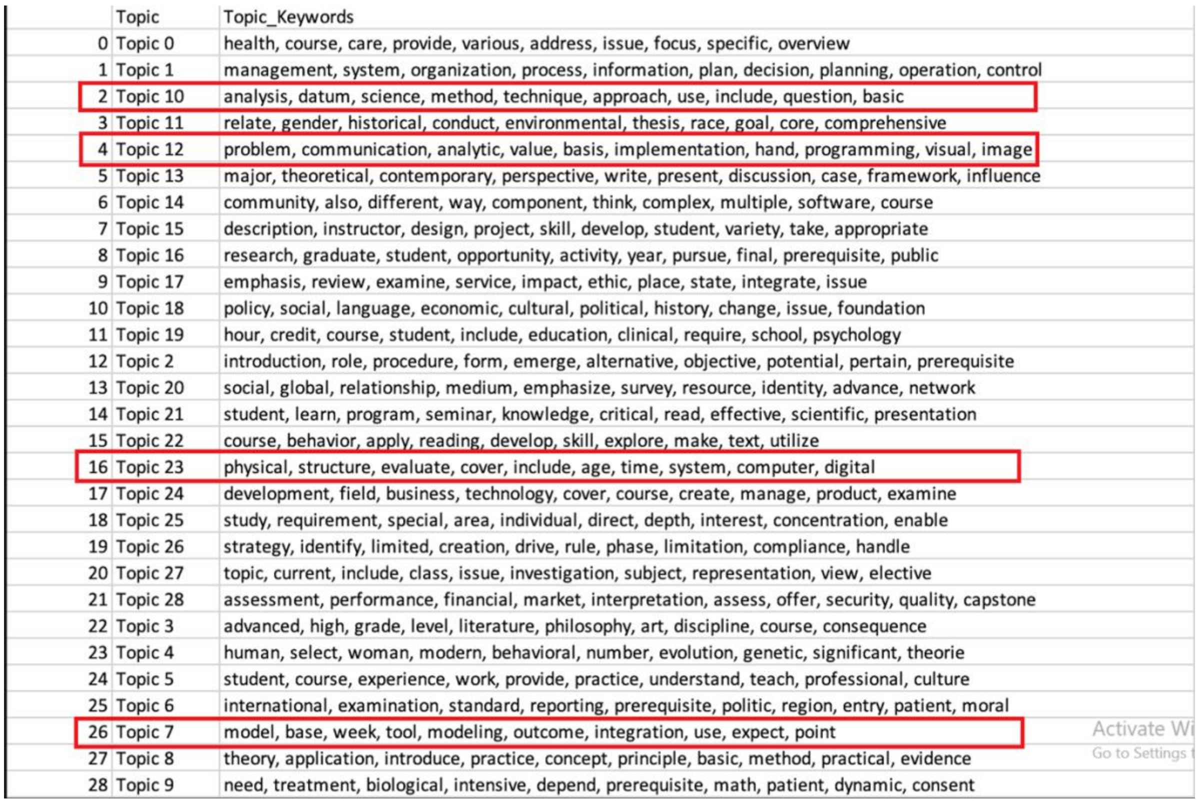


Fig. 18 Modularity class with number 10

The topics which are falling under this modularity class 10 are Topic 10, Topic 12, Topic 23, and Topic 7. If we observe the key words of each topic under this modularity class 10, they all are somehow like the terms which occur in the field of data science. Below is the image in which the topics and the key words of that topics related to the data science are highlighted in the red:



	Topic	Topic_Keywords
0	Topic 0	health, course, care, provide, various, address, issue, focus, specific, overview
1	Topic 1	management, system, organization, process, information, plan, decision, planning, operation, control
2	Topic 10	analysis, datum, science, method, technique, approach, use, include, question, basic
3	Topic 11	relate, gender, historical, conduct, environmental, thesis, race, goal, core, comprehensive
4	Topic 12	problem, communication, analytic, value, basis, implementation, hand, programming, visual, image
5	Topic 13	major, theoretical, contemporary, perspective, write, present, discussion, case, framework, influence
6	Topic 14	community, also, different, way, component, think, complex, multiple, software, course
7	Topic 15	description, instructor, design, project, skill, develop, student, variety, take, appropriate
8	Topic 16	research, graduate, student, opportunity, activity, year, pursue, final, prerequisite, public
9	Topic 17	emphasis, review, examine, service, impact, ethic, place, state, integrate, issue
10	Topic 18	policy, social, language, economic, cultural, political, history, change, issue, foundation
11	Topic 19	hour, credit, course, student, include, education, clinical, require, school, psychology
12	Topic 2	introduction, role, procedure, form, emerge, alternative, objective, potential, pertain, prerequisite
13	Topic 20	social, global, relationship, medium, emphasize, survey, resource, identity, advance, network
14	Topic 21	student, learn, program, seminar, knowledge, critical, read, effective, scientific, presentation
15	Topic 22	course, behavior, apply, reading, develop, skill, explore, make, text, utilize
16	Topic 23	physical, structure, evaluate, cover, include, age, time, system, computer, digital
17	Topic 24	development, field, business, technology, cover, course, create, manage, product, examine
18	Topic 25	study, requirement, special, area, individual, direct, depth, interest, concentration, enable
19	Topic 26	strategy, identify, limited, creation, drive, rule, phase, limitation, compliance, handle
20	Topic 27	topic, current, include, class, issue, investigation, subject, representation, view, elective
21	Topic 28	assessment, performance, financial, market, interpretation, assess, offer, security, quality, capstone
22	Topic 3	advanced, high, grade, level, literature, philosophy, art, discipline, course, consequence
23	Topic 4	human, select, woman, modern, behavioral, number, evolution, genetic, significant, theorie
24	Topic 5	student, course, experience, work, provide, practice, understand, teach, professional, culture
25	Topic 6	international, examination, standard, reporting, prerequisite, politic, region, entry, patient, moral
26	Topic 7	model, base, week, tool, modeling, outcome, integration, use, expect, point
27	Topic 8	theory, application, introduce, practice, concept, principle, basic, method, practical, evidence
28	Topic 9	need, treatment, biological, intensive, depend, prerequisite, math, patient, dynamic, consent

Fig. 19 Topics and Topic Keywords related to the data science.

Now let us see which courses of the Computer Science department from the selected four universities are falling under this Modularity class. (by using Partition filters)

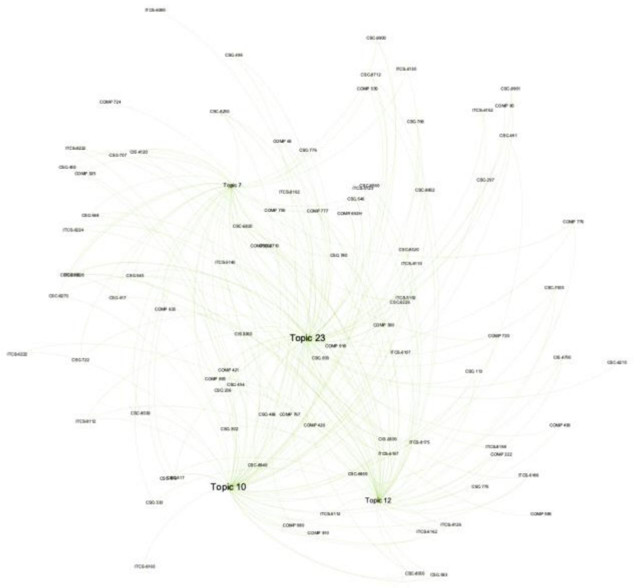


Fig. 20 Computer Science courses under Modularity class 10

Now let us see the computer science courses from University of North Carolina Chapel Hill (UNC) falling under this Modularity class.

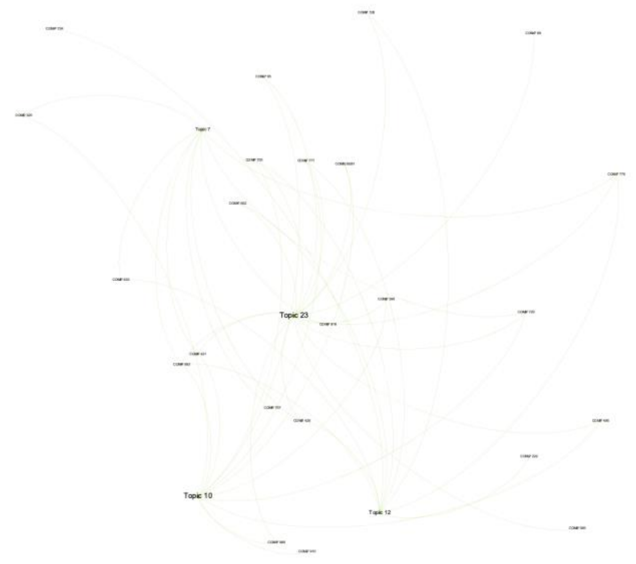


Fig. 21 Computer Science courses from UNC falling under this Modularity Class.

Now let us see the computer science courses from North Carolina State University (NCSU) falling under this Modularity class.

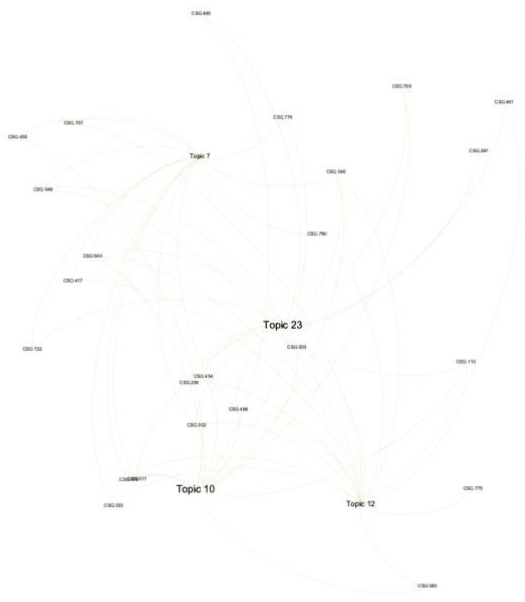


Fig. 22 Computer Science courses from NCSU falling under this Modularity Class.

Now let us see the computer science courses from Georgia State University (GSU) falling under this Modularity class.

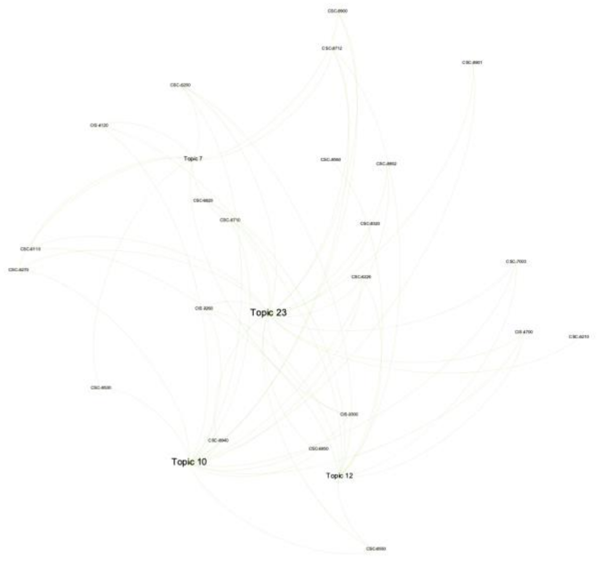


Fig. 23 Computer Science courses from GSU falling under this Modularity Class.

Now let us see the computer science courses from University of North Carolina at Charlotte (UNCC) falling under this Modularity class.

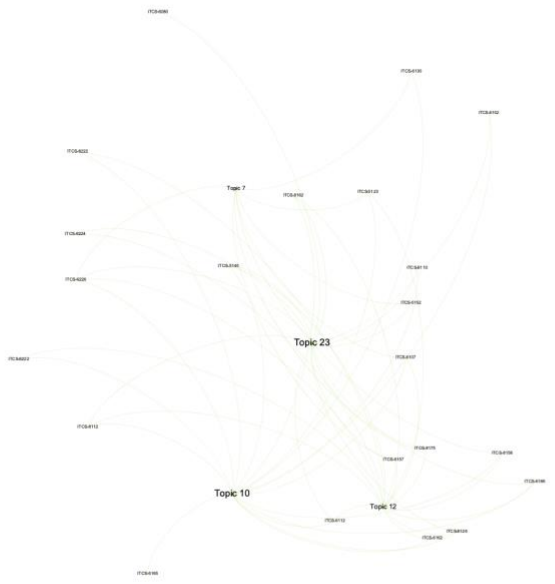


Fig. 24 Computer Science courses from UNCC falling under this Modularity Class

CHAPTER 6 CONCLUSION

This project helps to understand how to web scrape a website, importance of the topic modeling, building an LDA model, and how to detect communities on the scraped data. This project is one of the applications of the topic modeling. In this project LDA model is generated on the data of the selected four universities. In a way it is relating the courses from one university to another university. Also, this model is helpful in finding courses within a university which relate with other departments in the same university.

This type of approach and analysis helps us to understand and find out which departments in the university are related to each other and this type of model particularly helps the students to compare the courses from one university to another university and even with one department to another department of a university. This helps to get better insights on course offerings in a particular university and to which field a particular university or group of universities are related. Also, this project helped in understanding how the Gephi application works and how to filter or partition the data in Gephi using the inbuilt metrics.

In this project we build the LDA model which is an unsupervised machine learning algorithm. We can also try using other efficient algorithms like LDA's Mallet implementation etc., to get better efficient topics so that we can relate courses across different universities more efficiently. The performance of the model depends on how well the data which was scrapped from the website was preprocessed.

REFERENCES

1. https://beautiful-soup-4.readthedocs.io/en/latest/#
2. https://radimrehurek.com/gensim/intro.html
3. https://www.nltk.org/
4. https://networkx.org/documentation/networkx-1.10/reference/introduction.html
5. https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/
6. https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/#9createbigramandtrigrammodels
7. https://datascienceplus.com/evaluation-of-topic-modeling-topic-coherence/
8. https://gephi.org/users/download/
9. https://gephi.org/users/quick-start/
10. https://www.crummy.com/software/BeautifulSoup/bs4/doc/#quick-start
11. https://www.analyticsvidhya.com/blog/2016/08/beginners-guide-to-topic-modeling-in-python/
12. https://catalogs.gsu.edu/
13. https://catalogs.gsu.edu/content.php?catoid=13&navoid=1427
14. https://catalog.unc.edu/courses/
15. http://catalog.ncsu.edu/course-descriptions/
16. https://catalog.uncc.edu/content.php?catoid=19&navoid=1167