TOPIC MODELLING ON TELUGU-ENGLISH CODE-MIXED DATA USING LDA

A Project Progress Report

Submitted in partial fulfillment for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

P.SANTOSH	(N170425)
A.SHANIL KUMAR	(N170778)
B.NAGAJYOTHI	(N170264)
K.YOGINI	(N171004)
P.KHYATHI	(N170245)

Under the Esteem Guidance of

Mr. Rayala Upendar Rao

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Rajiv Gandhi University of Knowledge Technologies – Nuzvid

Nuzvid, Krishna, Andhra Pradesh – 521202.

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Rajiv Gandhi University of Knowledge Technologies – Nuzvid

Nuzvid, Krishna, Andhra Pradesh – 521202.

CERTIFICATE OF COMPLETION

This is to certify that the work entitled, **Topic Modelling on Telugu-English Code-Mixed Data using LDA** is the Bonafide work of **P.SANTOSH (ID No: N170425), A.SHANIL (ID No:N170778), B.NAGAJYOTHI (ID No: N170264), K.YOGINI (ID No: N171004), P.KHYATHI (ID No:N170245),** carried out under my guidance and supervision for 3rd year project of **Bachelor of Technology** in the department of Computer Science and Engineering under RGUKT IIIT Nuzvid. This work is done during the academic session May 2022 – September 2022, under our guidance.

--------------------------------------	-----------------------------------
Mr. Rayala Upendar Rao	**Mr. Chiranjeevi Sadu**
Assistant professor,	Assistant Professor,
Department of CSE,	Head of the Department,
RGUKT Nuzvid	Department of CSE,
	RGUKT Nuzvid.

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Rajiv Gandhi University of Knowledge Technologies – Nuzvid

Nuzvid, Krishna, Andhra Pradesh – 521202.

<u>CERTIFICATE OF EXAMINATION</u>

This is to certify that the work entitled, **"Topic Modelling on Telugu-English Code Mixed Data using LDA"** is the bonafide work of **P.SANTOSH (ID No*:* N170425), A.SHANIL KUMAR *(*ID No:N170778), B.NAGAJYOTHI *(*ID No*:* N170264), K.YOGINI (ID No*:* N171004), P.KHYATHI *(*ID No*:*N170245),** and here by accord our approval of it as a study carried out and presented in a manner required for its acceptance in final year of **Bachelor of Technology** for which it has been submitted. This approval does not necessarily endorse or accept every statement made, opinion expressed or conclusion drawn, as a recorded in this thesis. It only signifies the acceptance of this thesis for the purpose for which it has been submitted.

-------------------------------------	-------------------------------------
Mr. Rayala Upendar Rao	**Mrs. M. Baby Anusha**
Assistant Professor,	Assistant Professor©,
Department of CSE,	Department of CSE,
RGUKT-NUZVID	RGUKT-NUZVID

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Rajiv Gandhi University of Knowledge Technologies – Nuzvid

Nuzvid, Krishna, Andhra Pradesh – 521202.

DECLARATION

We **"P.SANTOSH (ID No*:* N170425), A.SHANIL *(*ID No:N170778), B.NAGAJYOTHI *(*ID No*:* N170264), K.YOGINI (ID No*:* N171004), P.KHYATHI *(*ID No*:*N170245),** hereby declare that the project report entitled **"Topic Modelling on Telugu-English Code Mixed Data using LDA"** done by us under the guidance of Mr. Upendar Rao, Assistant Professor is submitted for the partial fulfillment for the award of degree of Bachelor of Technology in Computer Science and Engineering during the academic session May 2022-September 2022 at RGUKT - Nuzvid.

We also declare that this project is a result of our own effort and has not been copied or imitated from any source. Citations from any websites are mentioned in the references. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Date: 03-09-2022

Place: Nuzvid

P. SANTOSH	N170425
A. SHANIL	N170778
B. NAGAJYOTHI	N170264
P. KHYATHI	N170245
K. YOGINI	N171004

ACKNOWLEDGEMENT

We would like to express our profound gratitude and deep regards to our guide **Mr. Rayala Upendar Rao** for his exemplary guidance, monitoring and constant encouragement to us throughout the B. Tech course. We shall always cherish the time spent with him during the course of this work due to the invaluable knowledge gained in the field of reliability engineering.

We are extremely grateful for the confidence bestowed in us and entrusting our project entitled **“Topic Modelling on Telugu-English Code-Mixed Data using LDA”**

We express gratitude to Mr. Chiranjeevi sadhu (HOD of CSE) and other faculty members for being source of inspiration and constant encouragement which helped us in completing the project successfully.

Our sincere thanks to all the batch mates of 2017 CSE, who have made our stay at RGUKT-NUZVID, a memorable one.

Finally, yet importantly, we would like to express our heartfelt thanks to our beloved God and parents for their blessings, our friends for their help and wishes for the successful completion of this project.

ABSTRACT

Detection of theme word or key word describing a collection of words is an important text processing method in natural language processing known as topic detection (TD). An accurate topic detection method depends upon goodness of topic modelling technique. There are several topic modelling techniques implemented successfully, some prominent names are LSA, LDA, Hierarchical Dirichlet Process, Non-Negative Matrix Factorization. Among these Latent Dirichlet Allocation (LDA) is considered the most prevalent topic modeling method. Most topic modelling/detection techniques applied well for English corpus but very little work is available when it comes to Indian languages and code-mixed data so here, we used Latent Dirichlet Allocation(LDA) for detecting the hidden topics for the Telugu English code mixed data and measured the accuracy of the topic modeling model using coherence score.

Table of Contents

List of Figures

CHAPTER 1
INTRODUCTION

1.1 NATURAL LANGUAGE PROCESSING(NLP)

NLP stands for Natural Language Processing, which is a part of Computer Science, Human language**,** and Artificial Intelligence. It is the technology that is used by machines to understand, analyse, manipulate, and interpret human's languages. It helps developers to organize knowledge for performing tasks such as translation, automatic summarization, Named Entity Recognition (NER), speech recognition, relationship extraction, and topic segmentation**.**

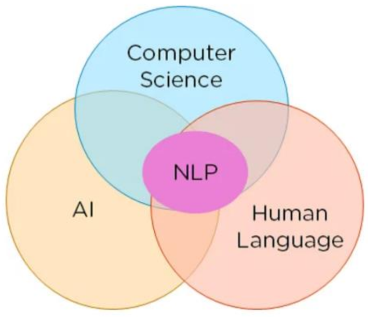


Fig1. Constituents of NLP

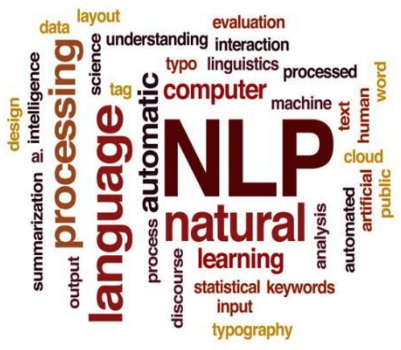


Fig2.NLP word cloud

Human language is filled with ambiguities that make it incredibly difficult to write software that accurately determines the intended meaning of text or voice data. Homonyms, homophones, sarcasm, idioms, metaphors, grammar and usage exceptions, variations in sentence structure—these just a few of the irregularities of human language that take humans years to learn, but that programmers must teach natural language-driven applications to recognize and understand accurately from the start, if those applications are going to be useful.

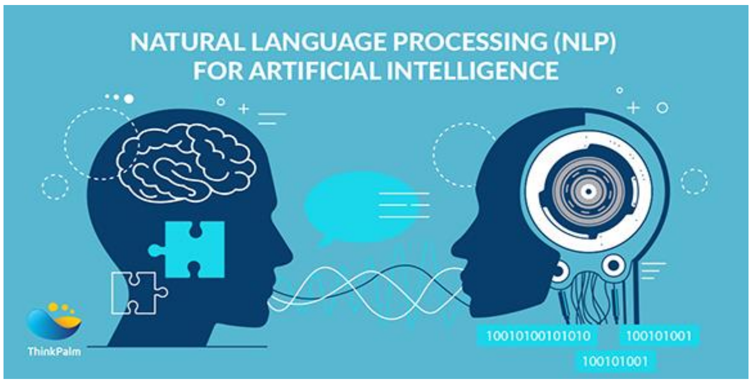

NATURAL LANGUAGE PROCESSING (NLP) FOR ARTIFICIAL INTELLIGENCE
ThinkPalm
10010100101010
100101001
100101001

Fig3. Interaction between machine and human

1.2 TOPIC MODELLING

- Topic modelling is an unsupervised technique that intends to analyse large volumes of text data by clustering the documents into groups. In the case of topic modelling, the text data do not have any labels attached to it. Rather, topic modelling tries to group the documents into clusters based on similar characteristics.

- A **topic model** is a model, which can automatically detect topics based on the words appearing in a document.

- It is important to note that topic modelling is different to topic classification. Topic classification is a supervised learning while topic modelling is an unsupervised learning algorithm.

- Some of the well-known topic modelling techniques are

  1. Latent Semantic Analysis (LSA)
  2. Probabilistic Latent Semantic Analysis (PLSA)
  3. Latent Dirichlet Allocation (LDA)
  4. Correlated Topic Model (CTM)
  5. Non negative factorization.

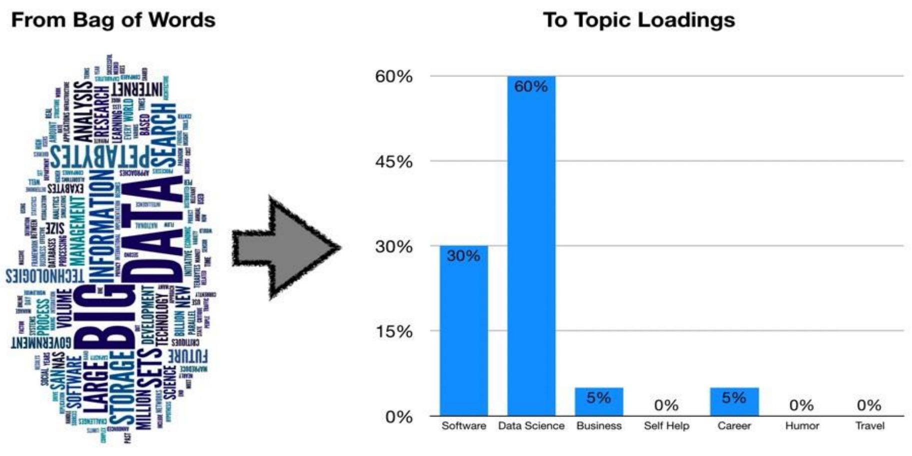


Fig4.Topic modelling

1.3 LATENT DIRICHLET ALLOCATION

A tool and technique for Topic Modelling, Latent Dirichlet Allocation (LDA) classifies or categorizes the text into a document and the words per topic, these are modelled based on the Dirichlet distributions and processes.

The LDA makes two key assumptions:

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

And, these topics using the probability distribution generate the words. In statistical language, the documents are known as the probability density (or distribution) of topics and the topics are the probability density (or distribution) of words.

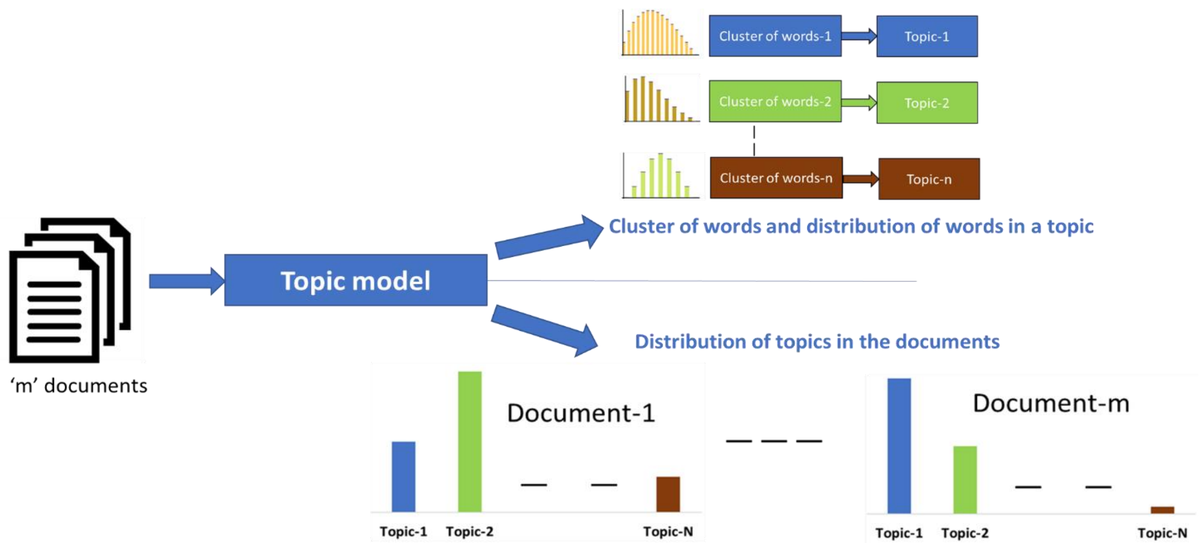


Fig5.LDA Model

The Algorithm to find the latter

- Go through each document and randomly assign each word in the document to one of ***k*** topics (*k* is chosen beforehand).
- For each document ***d***, go through each word ***w*** and compute :

- **p(topic *t* | document *d*)**: the **proportion of words in document *d* that are assigned to topic *t***. Tries to capture how many words belong to the topic *t* for a given document *d*. Excluding the current word.
  If a lot of words from *d* belongs to *t*, it is more probable that word *w* belongs to *t*.
  (#words in *d* with *t* +*alpha*/ #words in *d* with any topic+ *k***alpha*)

- **p(word *w*| topic *t*)**: the proportion of assignments to topic *t* over all documents that come from this word *w*. Tries to capture how many documents are in topic *t* because of word *w*.
  LDA represents documents as a mixture of topics. Similarly, a topic is a mixture of words. If a word has high probability of being in a topic, all the documents having *w* will be more strongly associated with *t* as well. Similarly, if *w* is not very probable to be in *t*, the documents which contain the *w* will be having very low probability of being in *t*, because rest of the words in *d* will belong to some other topic and hence *d* will have a higher probability for those topics. So even if *w* gets added to *t*, it won't be bringing many such documents to *t*.

- Update the probability for the word *w* belonging to topic *t*, as
  p(word w with topic t) = p(topic t | document d) * p(word w | topic t) $\longrightarrow$ eq1

PARAMETERS OF LDA

- alpha:- Document-topic density
- beta:- Topic-word density
- Chunk size:-number of documents to be used in each training chunk
- update every:-how often the model parameters should be updated
- passes:-total number of training passes

1.4 COHERENCE SCORE

We can use the coherence score in topic modelling to measure how interpret-able the topics are to humans. In this case, topics are represented as the top N words with the highest probability of belonging to that particular topic. Briefly, the coherence score measures how similar these words are to each other.

CV Coherence Score

One of the most popular coherence metrics is called CV. It creates content vectors of words using their co-occurrences and, after that, calculates the score using normalized point wise mutual information (NPMI) and the cosine similarity. This metric is popular because it's the default metric in the Gensim topic coherence pipeline module, but it has some issues. Even the author of this metric doesn't recommend using it.

Because of that, we don't recommend using the CV coherence metric.

UMass Coherence Score

Instead of using the CV score, we recommend using the UMass coherence score. It calculates how often two words, w_i and w_j appear together in the corpus and it's defined as

$$C_{UMass}(w_i, w_j) = \log \frac{D(w_i, w_j) + 1}{D(w_i)}, \quad \longrightarrow \text{eq2}$$

where $D(w_i, w_j)$ indicates how many times words w_i and w_j appear together in documents, and $D(w_i)$ is how many times word w_i appeared alone. The greater the number, the better is coherence score. Also, this measure isn't symmetric, which means that $C_{UMass}(w_i, w_j)$ is not equal to $C_{UMass}(w_j, w_i)$. We calculate the global coherence of the topic as the average pairwise coherence scores on the top N words which describe the topic.

UCI Coherence Score

This coherence score is based on sliding windows and the point wise mutual information of all word pairs using top N words by occurrence. Instead of calculating how often two words appear in the document, we calculate the word co-occurrence using a sliding window. It means that if our sliding window has a size of 10, for one particular word w_i, we observe only 10 words before and after the word w_i.

Therefore, if both words w_i and w_j appeared in the document but they're not together in one sliding window, we don't count as they appeared together. Similarly, as for the UMass score, we define the UCI coherence between words w_i and w_j as

$$C_{UCI}(w_i, w_j) = \log \frac{P(w_i, w_j) + 1}{P(w_i) \cdot P(w_j)}. \quad \longrightarrow \quad \text{eq3}$$

where $P(w)$ is probability of seeing word w in the sliding window and $P(w_i, w_j)$ is probability of appearing words w_i and w_j together in the sliding window. In the original paper, those probabilities were estimated from the entire corpus of over two million English Wikipedia articles using a 10-words sliding window. We calculate the global coherence of the topic in the same way as for the UMass coherence.

Word2vec Coherence Score

One smart idea is to utilize the word2vec model for the coherence score. This will introduce the semantic of the words in our score. Basically, we want to measure our coherence based on two criteria:

Intra-topic similarity – the similarity of words in the same topic.

Inter-topic similarity – the similarity of words across different topics.

The idea is pretty simple. We want to maximize intra-topic and minimize inter-topic similarity. Also, by similarity, we imply the cosine similarity between words represented by word2vec embedding.

Following that, we compute intra-topic similarity per topic as an average similarity between every possible pair of top N words in that topic. Consequently, we compute the inter-topic similarity between two topics as an average similarity between top N words from these topics.

Finally, the word2vec coherence score between two topics, t_i and t_j, is calculated as

$$C_{word2vec}(t_i, t_j) = \frac{\frac{intra\_topic\_similarity(t_i) + intra\_topic\_similarity(t_j)}{2}}{inter\_topic\_similarity(t_i, t_j)}. \longrightarrow \text{eq4}$$

Choosing the Best Coherence Score

There is no one way to determine whether the coherence score is good or bad. The score and its value depend on the data that it’s calculated from. For instance, in one case, the score of 0.5 might be good enough but in another case not acceptable. The only rule is that we want to maximize this score.

Usually, the coherence score will increase with the increase in the number of topics. This increase will become smaller as the number of topics gets higher. The trade-off between the number of topics and coherence score can be achieved using the so-called elbow technique. The method implies plotting coherence score as a function of the number of topics. We use the elbow of the curve to select the number of topics.

The idea behind this method is that we want to choose a point after which the diminishing increase of coherence score is no longer worth the additional increase of the number of topics.

CHAPTER 2

REQUIREMENTS AND ANALYSIS

2.1 Hardware components

- Processor: 64-bit, quad-core, 2.5 GHz minimum per core
- RAM: 4 GB or more.
- HDD: 20 GB of available space or more.
- Display: Dual XGA (1024 x 768) or higher resolution monitors.
- Keyboard: A standard keyboard

2.2 Software components

- **Python:** Python offers concise and readable code. While complex algorithms and versatile workflows stand behind machine learning and AI, Python's simplicity allows developers to write reliable systems.

- **Gensim:** Gensim is a Python library for topic modelling, document indexing and similarity retrieval with large corpora. Target audience is *the* natural language processing (NLP) and information retrieval (IR) community.

- **Scikit-Learn:** Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

- **Numpy:** Numpy is the fundamental package for array computing with python.

- **NLTK:** NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

- **RE:** A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern. RegEx can be used to check if a string contains the specified search pattern.

- **pyLDAvis:** pyLDAvis is an open-source python library that helps in analyzing and creating highly interactive visualization of the clusters created by LDA. In this article, we will see how to use LDA and pyLDAvis to create Topic Modelling Clusters visualizations.

- **Label Studio:** Label Studio is an open-source tool for labelling and exploring multiple types of data. Different types of labeling can be performed with many data formats. Label Studio can also be integrated with machine learning models to supply predictions for labels (pre-labels), or perform continuous active learning. To perform labeling the 15 data file has to be imported and after labeling it can be exported in desired file format.

- **Collaboratory Notebook**

- **Windows OS 64-bit.**

CHAPTER 3

PROPOSED MODEL AND FLOW OF THE PROJECT

3.1 Proposed model

Detection of theme word or key word describing a collection of words is an important text processing method in natural language processing known as topic detection (TD).This project focuses on detecting the hidden topics of the Telugu English code mixed data and measured the accuracy of the topic modelling model using coherence score in python.

3.2 Flow of the project

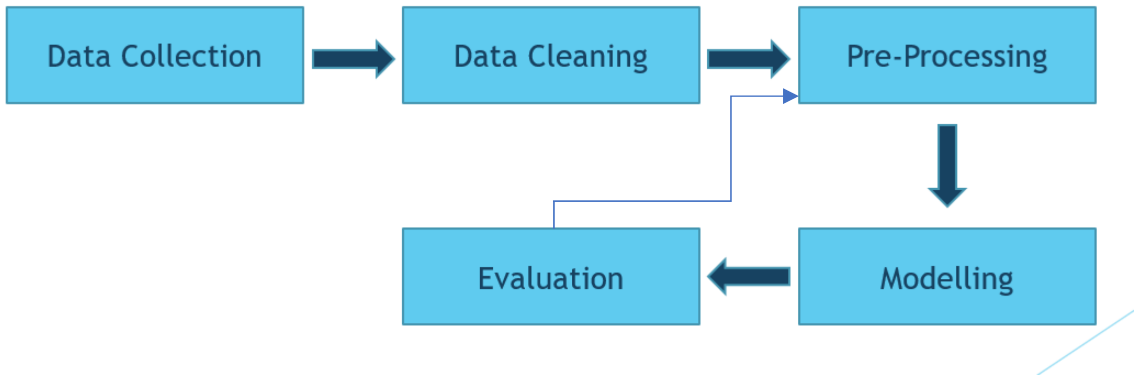


Fig6. Flow of the project

3.3 Advantages and Disadvantages

3.3.1 Advantages

- Used in graph-based models to obtain semantic relationship between words
- Used in text summarization to quickly find out what the document or book is explaining about
- provides improved customer service by identifying the keyword the customer is asking about and acting accordingly
- can identify the keywords of search and recommend products to the customers accordingly

3.3.2 Disadvantages / Limitations

- Researcher has to interpret categorizations
- Topic models can easily be abused if they are wrongly understood as an objective representation of the meaning of a text.
- The results of topic models should not be over-interpreted unless the researcher has strong theoretical apriori about the number of topics in a given corpus, or if the researcher has carefully validated the results of a topic model using both the quantitative and qualitative techniques

3.4 Applications

- Topic tracking
- Spam filter
- Recommender system
- Text Summarization
- Entity Recognition
- Detecting patterns
- Identify Emerging Trends
- Enhance customer service

CHAPTER 4

IMPLEMENTATION

4.1 Data Collection

- Data collection is the process of gathering the data from sources like YouTube, Twitter.

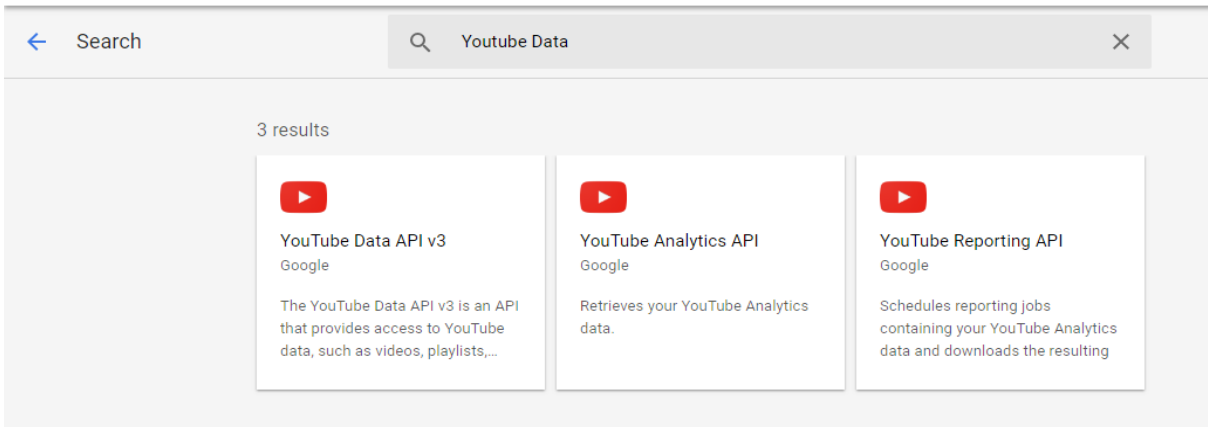
Extracting YouTube comments using YouTube API

Google provides a large set of APIs for the developer to choose from. Each and every service provided by Google has an associated API. Being one of them, YouTube Data API is very simple to use.

Enabling the API

1)Create New Project, Enable API and Create Credentials: In this step we will create a project and will enable the API.

- Go to **Google Developers** Console and Click on **Sign In** in the upper rightmost corner of the page. Sign In using the credentials of the valid Google Account. If you don't have a google account, setup an account first and then use the details to Sign In on the Google Developers Homepage.
- Now navigate to the Developer Dashboard and create a new Project.
- Click on **Enable API option**.
- In the search field, search for **YouTube Data API** and select the **YouTube Data API** option that comes in the drop-down list.



YouTube Data API v3
Google
The YouTube Data API v3 is an API that provides access to YouTube data, such as videos, playlists,...

YouTube Analytics API
Google
Retrieves your YouTube Analytics data.

YouTube Reporting API
Google
Schedules reporting jobs containing your YouTube Analytics data and downloads the resulting

Fig7.YouTube api key generation

- You will be redirected to a screen that says information about the YouTube Data API, along with two options : ENABLE and TRY API
- Click on ENABLE option to get started with the API.
- In the sidebar under APIs & Services, select Credentials.
- In the Credentials tab, select the Create credentials drop-down list, and choose API key.

  There are two types of Credentials: API Key and OAuth. OAuth provides you with Client Id and a Secret Key in the form of a .json file. OAuth is generally used where authorization is required like in the case of retrieving liked videos of a user. So for the rest cases where authorization is not required like searching for the videos using a keyword or for searching for the related videos etc we will be using API Key.

2.Installation

- Google API client for python can be installed using simple **pip** command:

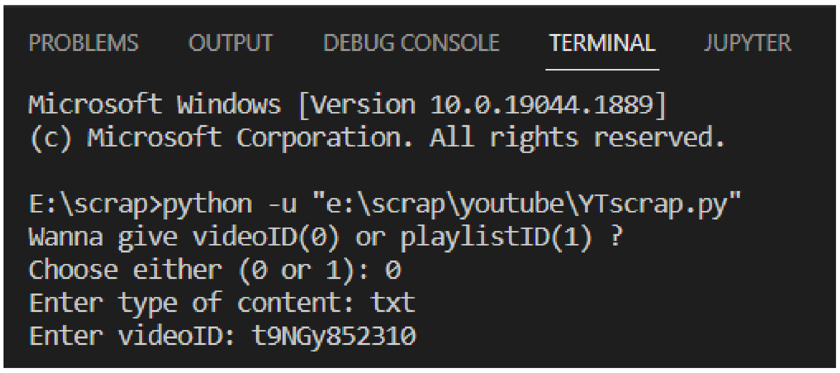
  pip install --upgrade google-api-python-client

- Python dotenv: Python-dotenv reads key-value pairs from a .env file and can set them as environment variables

  pip install python -dotenv

3.Then create .env file.

4.By using all these perform scrapping as it takes input as video/playlist id and creates the YouTube comments as the txt file named by video/playlist id.



```
Microsoft Windows [Version 10.0.19044.1889]
(c) Microsoft Corporation. All rights reserved.

E:\scrap>python -u "e:\scrap\youtube\YTscrap.py"
Wanna give videoID(0) or playlistID(1) ?
Choose either (0 or 1): 0
Enter type of content: txt
Enter videoID: t9NGy852310
```

Fig8.scrapping

4.2 DATA CLEANING

- Removing punctuation's
- Removing emojis
- Removing URLs

4.3 IMPORT ALL NECESSARY LIBRARIES

```
import pandas as pd
import gensim
import gensim.corpora as corpora
from gensim.utils import simple_preprocess
from gensim.utils import tokenize
from gensim.models import CoherenceModel
from gensim.parsing.preprocessing import
preprocess_string,strip_tags,strip_punctuation,strip_numeric,remove_stopwords,strip_short
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
import string
import re
import pyLDAvis
import pyLDAvis.gensim_models
```

4.4 DATA PREPROCESSING

- Removing punctuations like . , ! $( ) * % @
- Removing URLs
- Removing Stop words
- Lower casing
- Removing numbers
- Removing duplicates
- Removing other language data(manual)
- Tokenization

1)For Removing punctuations

```
import string
def remove_punctuation(txt):
   txt_nopunct="".join([word for word in txt if word not in string.punctuation])
   return txt_nopunct
data['msg_clean']=data['comments'].apply(lambda x: remove_punctuation(x))
print(data[‘msg_clean’])
```

Data after removing punctuations

```
                                                             msg_clean
0       aa judges ni change cheysey daka programme chuddam manestey better
1                                        Aa juhi Chawla pakshi Raj avutad emo
2                                     Aa kajal dharidram pothene bhaguntundhi
3                       Aa kamma koritala gaadu kavalni chetta cinema teesadu
4                      Aa Kapil gaadu aithe over action tho savadenguthunnadu
...                                                                      ...
213104                                         Zte nubia m2 is good or bad 
213105                 zubaan sambhal k amandeep  jeet achchi nhi lagti kyaa 
213106                                      Zuckerberg and Gates and Buffet
213107        Zuckerberg investment started from ad in insta next whats app
213108                                  Zz Bacon sockets are as David Beckham
```

2)Tokenization and lowercasing

- For Topic modelling we have to tokenize the sentences as words.

```
import re
def tokenize(txt):
   tokens=re.split('\W+',txt)
   return tokens
data['tokenized']=data['msg_clean'].apply(lambda x: tokenize(x.lower()))
print(data.head())
data['tokenized']=data['msg_clean'].apply(lambda x: tokenize(x.lower()))
print(data['tokenized'].head())
```

- **Data After Tokenization and lowercasing**

```
 [aa, judges, ni, change, cheysey, daka, programme, chuddam, manestey, better]
                                    [aa, juhi, chawla, pakshi, raj, avutad, emo]
                                   [aa, kajal, dharidram, pothene, bhaguntundhi]
                  [aa, kamma, koritala, gaadu, kavalni, chetta, cinema, teesadu]
                 [aa, kapil, gaadu, aithe, over, action, tho, savadenguthunnadu]
: tokenized, dtype: object
```

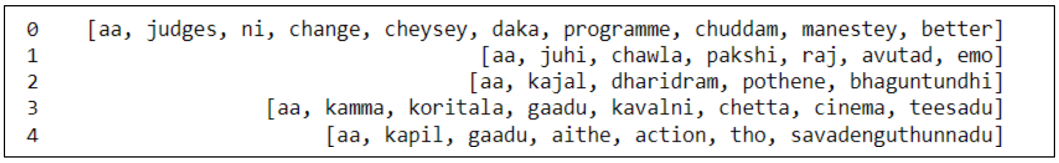
2) Remove stop words

We have to remove both English and Telugu stop words from the data. First we have to import them then compare and remove them from the data.

- <u>For English Stop words</u>

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
stopwords = set(stopwords.words("english"))
def dele_stopwords_english(txt):
   txt_rsw=[word for word in txt if word not in stopwords]
   return txt_rsw
data['english']=data['tokenized'].apply(lambda x: dele_stopwords_english(x))
data['english'].head()
```

- **After removing of English stop words**



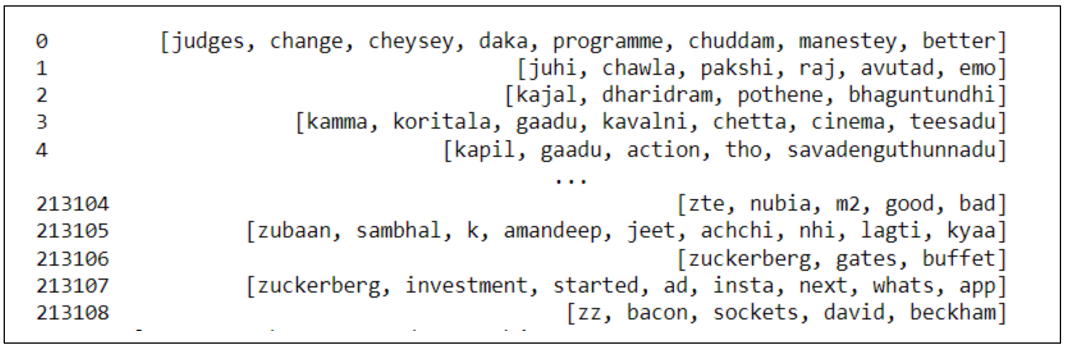
```
0    [aa, judges, ni, change, cheysey, daka, programme, chuddam, manestey, better]
1                                  [aa, juhi, chawla, pakshi, raj, avutad, emo]
2                                 [aa, kajal, dharidram, pothene, bhaguntundhi]
3                  [aa, kamma, koritala, gaadu, kavalni, chetta, cinema, teesadu]
4                       [aa, kapil, gaadu, aithe, action, tho, savadenguthunnadu]
```

- **For Telugu stop words**
  As there is no inbuilt library for Telugu. Here we take a list of Telugu stop words. Compare with the tokenized data, if match then remove them.

```
stop_words=['la','du','vu','tho','ki','ku','chetha','thoda','valana','ooree','orey','arey','are','ore','
ooyee','oosee','lopala','kante','koraku','patti','yokka','okka','aa','yenni','enni','e','ra','raa',"","",'
ka','ani','ni','lo','ee','me','a','ne','lu','em','em','nee','u','lodho','bhiya','chyandi','cheyandi','chud
aleka','meeku','meku','pedatha','indhi','aadhi','gadi','akkuva','ekkuva','vundhi','inka','mee','k','
a','emi','malli','matrame','mechuko','mecuko','vadda','vodda','vadha','vodha','venta','veruga','
vyatirekanga','vyathirekanga','sambandham','sambandam']

def dele_stopwords_telugu(txt):
    txt_rsw=[word for word in txt if word not in stop_words]
    return txt_rsw
data['telugu']=data['english'].apply(lambda x: dele_stopwords_telugu(x))
data['telugu']
```

- **Data after removing Telugu stopwords**



```
0         [judges, change, cheysey, daka, programme, chuddam, manestey, better]
1                                       [juhi, chawla, pakshi, raj, avutad, emo]
2                                      [kajal, dharidram, pothene, bhaguntundhi]
3                       [kamma, koritala, gaadu, kavalni, chetta, cinema, teesadu]
4                                   [kapil, gaadu, action, tho, savadenguthunnadu]
                                               ...
213104                                               [zte, nubia, m2, good, bad]
213105          [zubaan, sambhal, k, amandeep, jeet, achchi, nhi, lagti, kyaa]
213106                                               [zuckerberg, gates, buffet]
213107          [zuckerberg, investment, started, ad, insta, next, whats, app]
213108                                      [zz, bacon, sockets, david, beckham]
```

3)Removing special characters

```
def del_special_chars(txt):
    txt_rsw=[word for word in txt if word not in special_chars]
    return txt_rsw
```

4)Removing spaces and empty sentences

```
data['no_space']=data['telugu'].apply(lambda x:[word for word in x if word!=""])
data['no_space2']=data['no_space'].apply(lambda x:[word for word in x if word!=" "])
copy=data['tokenized'].apply(lambda x:tuple(word for word in x))
res = [ele for ele in copy if ele != ()]
from collections import Counter
counter=Counter(res)
most=counter.most_common(150)
data['no_space2']
```

- **Data after removing spaces and empty sentences**

```
0         [judges, change, cheysey, daka, programme, chuddam, manestey, better
1                                        [juhi, chawla, pakshi, raj, avutad, emo
2                                       [kajal, dharidram, pothene, bhaguntundhi
3                            [kamma, koritala, gaadu, kavalni, chetta, cinema, teesadu
4                                    [kapil, gaadu, action, tho, savadenguthunnadu
                                                  ...
213104                                                [zte, nubia, m2, good, bad
213105              [zubaan, sambhal, k, amandeep, jeet, achchi, nhi, lagti, kyaa
213106                                               [zuckerberg, gates, buffet
213107              [zuckerberg, investment, started, ad, insta, next, whats, app
213108                                          [zz, bacon, sockets, david, beckham
```

- **Final data**

```
0         [judges, change, cheysey, daka, programme, chuddam, manestey, better]
1                                        [juhi, chawla, pakshi, raj, avutad, emo]
2                                       [kajal, dharidram, pothene, bhaguntundhi]
3                            [kamma, koritala, gaadu, kavalni, chetta, cinema, teesadu]
4                                    [kapil, gaadu, action, tho, savadenguthunnadu]
                                                  ...
213104                                                [zte, nubia, m2, good, bad]
213105              [zubaan, sambhal, k, amandeep, jeet, achchi, nhi, lagti, kyaa]
213106                                               [zuckerberg, gates, buffet]
213107              [zuckerberg, investment, started, ad, insta, next, whats, app]
213108                                          [zz, bacon, sockets, david, beckham]
```

4.5 BUILDING THE MODEL

Data transformation: Corpus and Dictionary

```
id2word = corpora.Dictionary(res)
corpus = []
for text in res:
   new = id2word.doc2bow(text)
   corpus.append(new)
```

Here we assigned id to the word and replaced the word with id and calculated the frequency of words.

Training the LDA model

We have everything required to train the base LDA model. In addition to the corpus and dictionary, you need to provide the number of topics as well.

Parameters in LDA model

alpha:- Document-topic density

beta:- Topic-word density

chunksize:-number of documents to be used in each training chunk

update_every:-how often the model parameters should be updated

passes:-total number of training passes

Compute Model Coherence Score

Let’s calculate the baseline coherence score

Hyperparameter Tuning

First, let’s differentiate between model hyperparameters and model parameters :

Model hyperparameters can be thought of as settings for a machine learning algorithm that are tuned by the data scientist before training. Examples would be the number of trees in the random forest, or in our case, number of topics K

Model parameters can be thought of as what the model learns during training, such as the weights for each word in a given topic

Now that we have the baseline coherence score for the default LDA model, let’s perform a series of sensitivity tests to help determine the following model hyperparameters:

1. Number of Topics (K)
2. Dirichlet hyperparameter alpha: Document-Topic Density
3. Dirichlet hyperparameter beta: Word-Topic Density

We’ll perform these tests in sequence, one parameter at a time by keeping others constant and run them over the two different validation corpus sets. We’ll use ***C_v*** as our choice of metric for performance comparison

```
def compute_coherence_values(dictionary, corpus, texts, limit, start, step):
  coherence_values = []
  model_list = []
  for num_topics in range(start, limit, step):
    model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=num_topics,
                                            update_every=1,
                                            chunksize=50,
                                            passes=10,
                                            alpha="auto",
                                            random_state=12
                                          )

    model_list.append(model)
    coherencemodel = CoherenceModel(model=model,texts=texts, corpus=corpus, coherence='c_v')
    coherence_values.append(coherencemodel.get_coherence())
    print(num_topics,"Topics Completed\n")
  return model_list, coherence_values
```

Let’s call the function, and iterate it over the range of topics, alpha, and beta parameter values

Investigate Results

Let’s start by determining the optimal number of topics. The chart below outlines the coherence score, C_v, for the number of topics across two validation sets, and a fixed alpha = “auto”

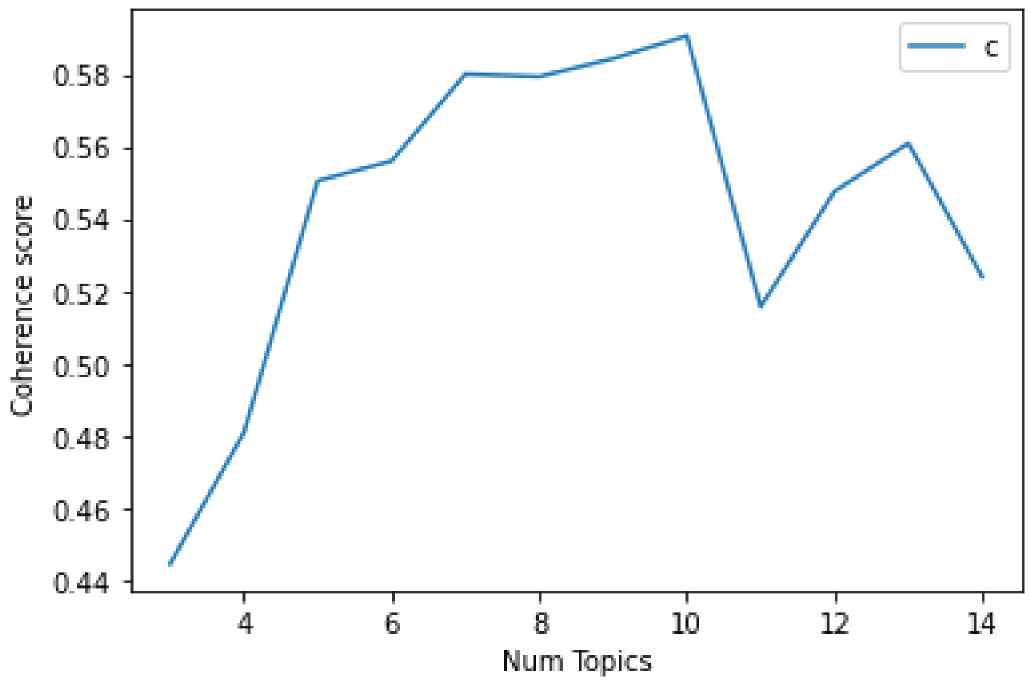


Fig9. Coherence graph

With the coherence score seems to keep increasing with the number of topics, it may make better sense to pick the model that gave the highest CV before flattening out or a major drop.

We should take the topics where we get the high coherence value.

```
for m, cv in zip(x, coherence_values):
    print("Num Topics =", m, " has Coherence Value of", round(cv, 4))
```

```
Num Topics = 3  has Coherence Value of 0.528
Num Topics = 4  has Coherence Value of 0.5584
Num Topics = 5  has Coherence Value of 0.5793
Num Topics = 6  has Coherence Value of 0.557
Num Topics = 7  has Coherence Value of 0.6155
Num Topics = 8  has Coherence Value of 0.5874
Num Topics = 9  has Coherence Value of 0.5845
Num Topics = 10  has Coherence Value of 0.5731
Num Topics = 11  has Coherence Value of 0.5592
Num Topics = 12  has Coherence Value of 0.5446
Num Topics = 13  has Coherence Value of 0.5505
Num Topics = 14  has Coherence Value of 0.5539
```

Input data to the model

```
0         [judges, change, cheysey, daka, programme, chuddam, manestey, better]
1                                    [juhi, chawla, pakshi, raj, avutad, emo]
2                                   [kajal, dharidram, pothene, bhaguntundhi]
3                      [kamma, koritala, gaadu, kavalni, chetta, cinema, teesadu]
4                                 [kapil, gaadu, action, tho, savadenguthunnadu]
                                        ...
213104                                           [zte, nubia, m2, good, bad]
213105           [zubaan, sambhal, k, amandeep, jeet, achchi, nhi, lagti, kyaa]
213106                                           [zuckerberg, gates, buffet]
213107           [zuckerberg, investment, started, ad, insta, next, whats, app]
213108                                   [zz, bacon, sockets, david, beckham]
```

Final model

```
model = gensim.models.ldamodel.LdaModel(corpus=corpus,

                                        id2word=id2word,

                                        num_topics=num_topics,

                                        update_every=1,

                                        chunksize=50,

                                        passes=10,

                                        alpha="auto",

                                        random  state=12   )
```

4.6 Evaluating the model

4.6.1 Testing

- **Testing with single sentence**

```
Input1:
            new_sentence ="esari world cup india gelustundi"
Output:
            Associated %:  52.22 %
            The topic name:  Sports
Input2:
            new_sentence ="ntr acting movies lo baguntundi"
Output:
            Associated %:  40.97 %
            The topic name:  Entertainment (movies,serials,webseries)
```

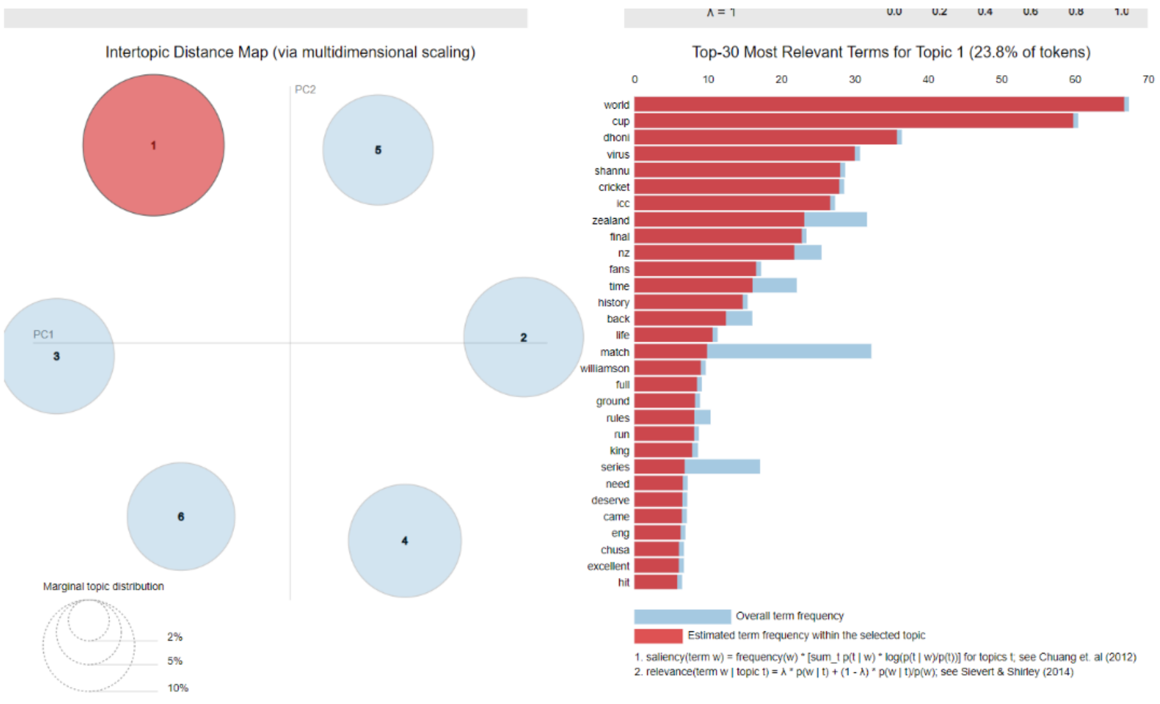
- **Testing with testing dataset**

Comment	**Topic Predicted**	**Actual Results**
esari world cup india gelustundi	Sports	Sports
ntr acting movies lo baguntundi	Entertainment	Entertainment
india team lo virat form loki vacchadu	Sports	Sports
kcr trs ki goppa nayakudu	Politics	Politics
Karthika deepam timings maristhey matches chuskovacchu	Entertainment	Sports
Dhoni next world cup ki vuntada	Sports	Sports

Accuracy: (Correctly Predicted / Total Predicted) *100 = **(376 / 500) *100 = 75.2%**

4.6.2 Visualization

```
import pyLDAvis
import pyLDAvis.gensim_models
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim_models.prepare(lda_model, corpus, id2word, mds="mmds", R=20)
pyLDAvis.save_html(vis,"lda3.html")
vis
```



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)

Fig10.visualization

CHAPTER 5
CONCLUSION

We started from scratch by collecting, importing, cleaning and processing the collected dataset to build the LDA model. Then we saw multiple ways to visualize and test the accuracy of the outputs of topic models including the word clusters and coherence score which intuitively tells you what topic is dominant in each document. The pyLDAVis provides more details into the clustering of the topics and the coherence score gives the accuracy.

REFERENCES

- https://www.researchgate.net/publication/361045137_Sentiment_Analysis_of_Code-Mixed_Social_Media_Text_SA-CMSMT_in_Indian-Languages
- https://www.researchgate.net/publication/358873355_Sentiment_Extraction_from_English-Telugu_Code_Mixed_Tweets_Using_Lexicon_Based_and_Machine_Learning_Approaches
- https://www.researchgate.net/publication/334447179_Text_Processing_of_Telugu-English_Code_Mixed_Languages
- http://qpleple.com/perplexity-to-evaluate-topic-models/
- https://www.amazon.com/Machine-Learning-Probabilistic-Perspective-Computation/dp/0262018020
- https://neptune.ai/blog/vectorization-techniques-in-nlp-guide
- https://medium.com/analytics-vidhya/topic-modeling-using-gensim-lda-in-python-48eaa2344920
- https://www.machinelearningplus.com/nlp/topic-modeling-visualization-how-to-present-results-lda-models/