Topic Modelling of articles with K-Means , LDA and visualization using SoM

- **Introduction**

  In today's digital age, the exponential growth of online news articles presents both an opportunity and a challenge. While the abundance of information empowers individuals to stay informed, the sheer volume can overwhelm readers and hinder efficient access to relevant content. As a result, there's an increasing need for automated methods to categorize, summarize, and organize news articles effectively. In this project we are trying to solve problem of categorizing news articles by showcasing certain techniques using unsupervised machine learning, because there can be many articles which have never been categorized, thus finding new categories for them.

- **Problem Statement**

  Through this I wanted to solve categorization or segregation of topics in news articles based on unsupervised machine learning approaches (K-Means, LDA) and visualize those new articles post segregation (SoM) based on how similar they are to one another.

  We have taken 2 approaches of Unsupervised Machine Learning into consideration

  1. K-Means Clustering
  2. Latent Dirichlet Allocation (LDA)

  For Visualization , we have taken Self Organizing Maps (SoM)

  By leveraging unsupervised machine learning algorithms, we seek to uncover underlying themes and patterns within large collections of news articles.

  Moreover, the significance of topic modelling is, it can facilitate content recommendation, trend analysis, and information retrieval, thereby enhancing the user experience and engagement with news content.

- **Data used and its description.**

  Currently we are using dataset from Kaggle .

  The URL Link of the dataset is https://www.kaggle.com/datasets/benjaminawd/new-york-times-articles-comments-2020?resource=download&select=nyt-articles-2020.csv
  This is New York Times News Articles of the year 2020.

  The snippet of the data collected and used in this assignment is :

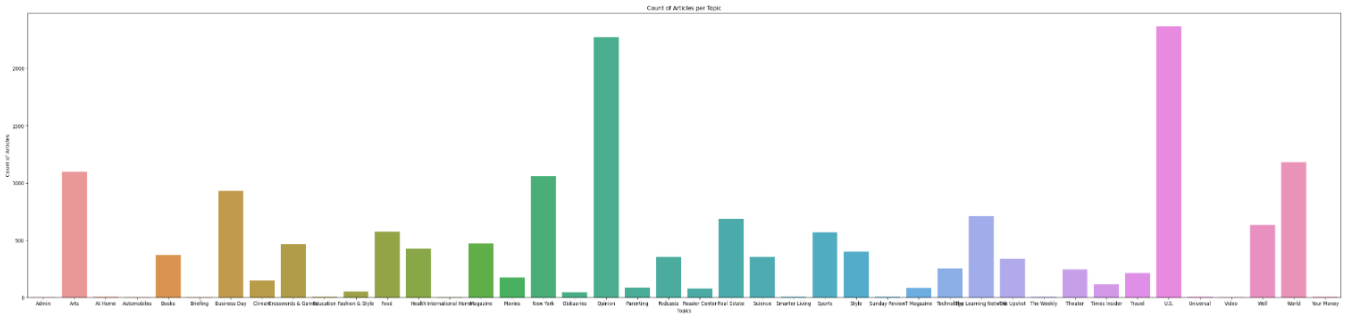
	newsdesk	section	subsection	material	headline	abstract	keywords	word_count	pub_date	n_comments	uniqueID
0	Editorial	Opinion	NaN	Editorial	Protect Veterans From Fraud	Congress could do much more to protect America...	['Veterans', 'For-Profit Schools', 'Financial ...	680	2020-01-01 00:18:54+00:00	186	nyt://article/69a7090b-9f36-569e-b5ab-b0ba5bb3...
1	Games	Crosswords & Games	NaN	News	'It's Green and Slimy'	Christina Iverson and Jeff Chen ring in the Ne...	['Crossword Puzzles']	931	2020-01-01 03:00:10+00:00	257	nyt://article/9edddb54-0aa3-5835-a833-d311a76f...
2	Science	Science	NaN	News	Meteor Showers in 2020 That Will Light Up Nigh...	All year long, Earth passes through streams of...	['Meteors and Meteorites', 'Space and Astronom...	1057	2020-01-01 05:00:08+00:00	6	nyt://article/04bc90f0-b20b-511c-b5bb-3ce13194...
3	Science	Science	NaN	Interactive Feature	Sync your calendar with the solar system	Never miss an eclipse, a meteor shower, a rock...	['Space and Astronomy', 'Moon', 'Eclipses', 'S...	0	2020-01-01 05:00:12+00:00	2	nyt://interactive/5b58d876-9351-50af-9b41-a312...
4	Science	Science	NaN	News	Rocket Launches, Trips to Mars and More 2020 S...	A year full of highs and lows in space just en...	['Space and Astronomy', 'Private Spaceflight',...	1156	2020-01-01 05:02:38+00:00	25	nyt://article/bd8647b3-8ec6-50aa-95cf-2b81ed12...

Note : Already the data has the column **section** which defines the category of news articles

But while running the models K-Means or LDA , we are not utilising that column **section** we are only focussing on **abstract** and **headline** columns. The abstract column defines the abstraction of a particular news article, headline defines the news article headlines.

But while trying to interpret the clusters and how many articles were initially grouped under certain sections & clusters pre and post Clustering , the column **section** was a reference check.

The pic is of Plotting a bar graph of **sections** vs **Count of articles** pre **K-Means Clustering** on original dataset



For future improvement we could have used , research papers , journals from various topics to create our own dataset.

The data has 11 columns and 16787 rows

• Main findings & outcomes

- **Brief on working and Findings :** In this assignment first, we tried to segregate or categorize topics with the help of K-Means clustering and then we also used an advanced clustering mechanism Latent Dirichlet allocation which is predominantly used for topic modelling.
- But before that, we did some preprocessing of the text data. Like, converting into lowercase the abstract, extractive summarization of abstract, and removal of stop words, punctuations and lemmatization, tokenization.
- Then we calculated TF-IDF values on the processed text.
- We also calculated the similarity matrix, based on the TF-IDF values.
- We also tried to calculate the optimal number of K , by elbow method. We tried using silhouette score method, but somehow the processing took way more than few hours.
- We took optimal number of clusters as 43, here. Elbow method was not though showing a definite elbow , but as our observations, 43 was fetching better results. (*We though tried with other numbers in the given range , which ever was forming an elbow dip in diagram)*

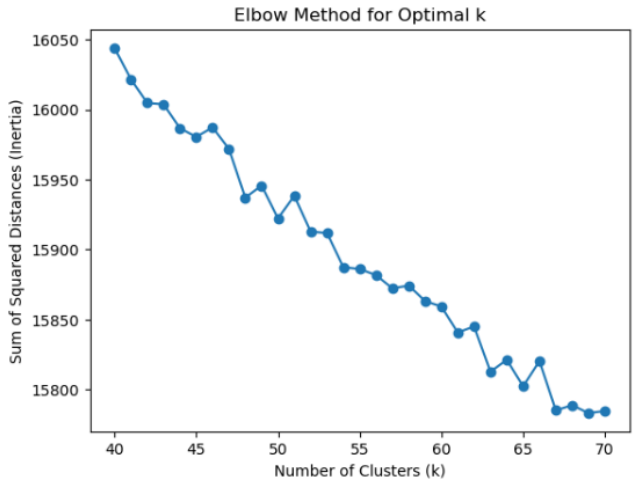
Estimating optimized cluster count in K-Means

```
k_values = range(40, 71)
inertia_values = []

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(tfidf_matrix)
    inertia_values.append(kmeans.inertia_)

# Plot the elbow curve
plt.plot(k_values, inertia_values, marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Sum of Squared Distances (Inertia)')

# Show the plot
plt.show()
```



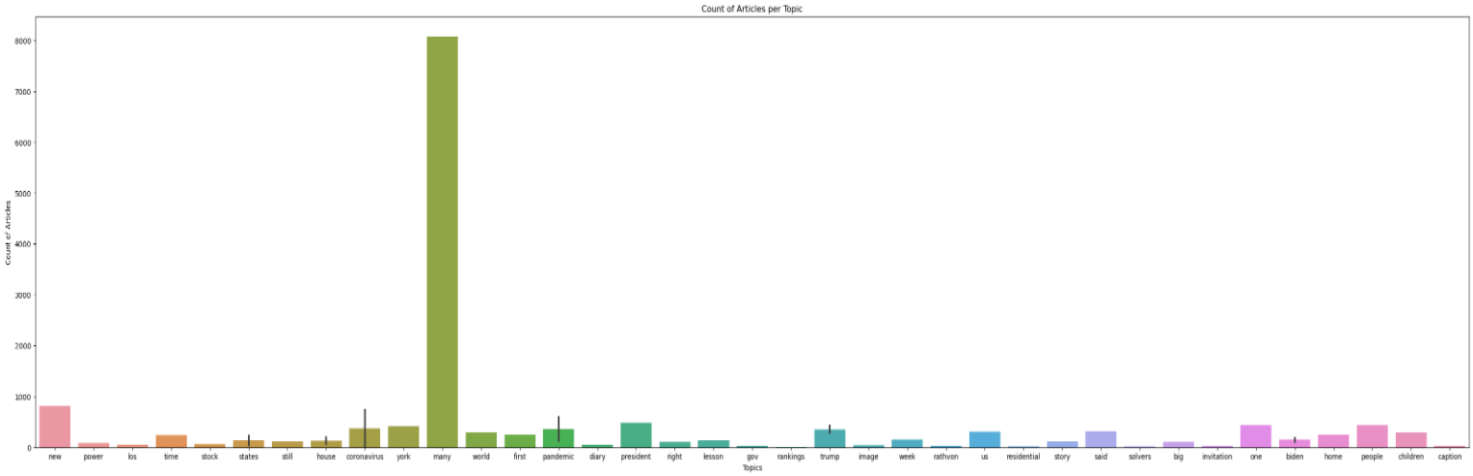
- We next did K-means Clustering on the similarity matrix we got. The similarity data frame looked like as below. Used Co-sine Similarity.

Out[16]:

Document_ID	(0, Protect Veterans From Fraud)	(1, 'It's Green and Slimy')	(2, Meteor Showers in 2020 That Will Light Up Night Skies)	(3, Sync your calendar with the solar system)	(4, Rocket Launches, Trips to Mars and More 2020 Space and Astronomy Events)	(5, Pro-Iranian Protesters End Siege of U.S. Embassy in Baghdad)	(6, Judge John Hodgman on Uncle Money Bags)	(7, She Felt Fine, but Her M.R.I. Showed Several Strokes. What Was Wrong?)	(8, These Armenian Flatbreads Stuffed With Greens Are the Perfect Snack)	(9, 'Don't Believe a Word,' a Look at Language and Power (and Why Dolphins Have Accents))	...	(16777, Homes for Sale in New York and Connecticut)	(16778, Homes for Sale in Brooklyn, Manhattan and Staten Island)	(16779, Becky Hammon Becomes First Woman to Serve as Head Coach in N.B.A. Game)
Document_ID														
(0, Protect Veterans From Fraud)	1.0	0.000000	0.000000	0.000000	0.000000	0.054836	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
(1, 'It's Green and Slimy')	0.0	1.000000	0.059155	0.000000	0.065580	0.000000	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
(2, Meteor Showers in 2020 That Will Light Up Night Skies)	0.0	0.059155	1.000000	0.000000	0.043868	0.000000	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
(3, Sync your calendar with the solar system)	0.0	0.000000	0.000000	1.000000	0.052099	0.000000	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
(4, Rocket Launches, Trips to Mars and More 2020 Space and Astronomy Events)	0.0	0.065580	0.043868	0.052099	1.000000	0.000000	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0

- The words with the highest TF-IDF in the respective clusters were dominant , so those can be interpreted as topics
- The number of Articles we got from the respective topics of K-Means Clustering can be depicted by a bar plot shown below.

	Cluster_ID	Topic	Articles_per_Cluster
0	0	new	809
1	1	power	85
2	2	los	54
3	3	time	236
4	4	stock	58



With K-Means , we can interpret that K-Means does not do much good in clustering with TF-IDF and similarity of articles, if we see the clusters, we do not see that it matches with sections which were predefined in the dataset.

We did not find any appropriate metric for the accuracy of K-Means clustering so went for visual interpretation as mentioned in code

- Now, we will try LDA or Laten Dirichlet Allocation.
- *LDA considers two important parameters*
- 1.**Document-topic distribution** : Each row represents a document, and each column represents a topic. Initialize these probabilities randomly.
  2.**Topic-word distribution** : Each row represents a topic, and each column represents a word in the vocabulary. Initialize these probabilities randomly.
- Iterate over each document in the corpus
- For each word in the document:
  Calculate the probability of each topic being the source of the word, given the current state of the model. This is done by multiplying the probability of the word given the topic with the probability of the topic given the document.
- Sample a new topic assignment for the word based on these probabilities.
  Update the document-topic distribution and topic-word distribution based on the new topic assignments.

Repeat the iterative process until the model converges, meaning that the topics and their distributions stabilize and don't change significantly between iterations.

- Post creating a corpus from our pre-processed texts , we trained our LDA and got 43 clusters or topic.

```
Topic 0: 0.073*"help" + 0.041*"nation" + 0.038*"keep" + 0.035*"top" + 0.035*"ways" + 0.030*"businesses" + 0.023*"vote" + 0.023
*"voting" + 0.019*"elections" + 0.017*"spring"
Topic 1: 0.058*"family" + 0.046*"story" + 0.040*"shows" + 0.040*"news" + 0.037*"lost" + 0.028*"support" + 0.025*"access" + 0.01
9*"set" + 0.019*"critics" + 0.018*"tell"
Topic 2: 0.076*"say" + 0.052*"vaccine" + 0.043*"among" + 0.036*"death" + 0.031*"experts" + 0.023*"service" + 0.022*"led" + 0.02
1*"thousands" + 0.021*"growing" + 0.018*"whose"
Topic 3: 0.096*"home" + 0.067*"house" + 0.031*"working" + 0.029*"politics" + 0.027*"star" + 0.027*"music" + 0.026*"woman" + 0.0
23*"helped" + 0.023*"give" + 0.022*"congress"
Topic 4: 0.057*"children" + 0.056*"social" + 0.047*"season" + 0.033*"outbreak" + 0.032*"live" + 0.031*"market" + 0.029*"media"
+ 0.023*"career" + 0.020*"others" + 0.020*"coronavirus"
Topic 5: 0.100*"election" + 0.047*"might" + 0.040*"end" + 0.024*"summer" + 0.023*"using" + 0.022*"research" + 0.021*"british" +
0.021*"columnist" + 0.021*"bring" + 0.019*"magazine"
Topic 6: 0.071*"states" + 0.063*"health" + 0.043*"public" + 0.039*"united" + 0.039*"officials" + 0.036*"workers" + 0.032*"care"
+ 0.028*"coronavirus" + 0.026*"system" + 0.021*"federal"
Topic 7: 0.136*"us" + 0.055*"next" + 0.043*"millions" + 0.039*"puzzle" + 0.028*"spent" + 0.026*"start" + 0.025*"doctors" + 0.02
1*"moment" + 0.020*"michigan" + 0.019*"changes"
Topic 8: 0.039*"different" + 0.036*"well" + 0.031*"restrictions" + 0.030*"several" + 0.026*"long" + 0.025*"cities" + 0.024*"isl
and" + 0.024*"travel" + 0.023*"created" + 0.021*"let"
Topic 9: 0.064*"business" + 0.023*"matter" + 0.022*"normal" + 0.021*"potential" + 0.021*"red" + 0.018*"heart" + 0.016*"course"
+ 0.016*"pain" + 0.016*"activists" + 0.016*"watch"
Topic 10: 0.072*"life" + 0.059*"women" + 0.034*"part" + 0.030*"without" + 0.030*"food" + 0.029*"families" + 0.024*"art" + 0.023
*"men" + 0.023*"best" + 0.022*"scientists"
Topic 11: 0.039*"see" + 0.036*"another" + 0.031*"return" + 0.025*"second" + 0.025*"largest" + 0.024*"number" + 0.023*"image" +
0.021*"george" + 0.019*"running" + 0.019*"south"
Topic 12: 0.058*"companies" + 0.035*"open" + 0.034*"parents" + 0.033*"office" + 0.028*"lot" + 0.023*"favorite" + 0.021*"tech" +
```

- Now , we want to see for the articles , what are its dominant topics, which is as follows.

Optionally, print the dominant topic for each document

```
for i, article in enumerate(tokenized_texts):
    print(f"Article {i+1} - Dominant Topic: {lda_model.get_document_topics(corpus[i])[0][0]}")
```

```
Article 1 - Dominant Topic: 3
Article 2 - Dominant Topic: 18
Article 3 - Dominant Topic: 8
Article 4 - Dominant Topic: 4
Article 5 - Dominant Topic: 10
Article 6 - Dominant Topic: 1
Article 7 - Dominant Topic: 18
Article 8 - Dominant Topic: 3
Article 9 - Dominant Topic: 8
Article 10 - Dominant Topic: 1
Article 11 - Dominant Topic: 2
Article 12 - Dominant Topic: 2
Article 13 - Dominant Topic: 4
Article 14 - Dominant Topic: 18
Article 15 - Dominant Topic: 3
Article 16 - Dominant Topic: 2
Article 17 - Dominant Topic: 4
Article 18 - Dominant Topic: 1
Article 19 - Dominant Topic: 1
```

- Also, we tried to analyse how many documents or articles are under each dominant topics.

```
Dominant Topics:
Topic 41: 381 documents
Topic 8: 230 documents
Topic 32: 285 documents
Topic 28: 519 documents
Topic 20: 299 documents
Topic 22: 348 documents
Topic 15: 387 documents
Topic 19: 325 documents
Topic 26: 217 documents
Topic 38: 350 documents
Topic 3: 446 documents
Topic 24: 319 documents
Topic 1: 340 documents
Topic 0: 414 documents
Topic 30: 454 documents
Topic 29: 236 documents
Topic 31: 1083 documents
Topic 23: 1187 documents
Topic 39: 412 documents
Topic 37: 513 documents
Topic 13: 493 documents
Topic 25: 340 documents
Topic 40: 461 documents
Topic 21: 261 documents
Topic 9: 174 documents
Topic 36: 281 documents
Topic 5: 309 documents
Topic 34: 270 documents
```

- We have then tried to visualize the clusters formed by LDA (Latent Dirichlet Allocation) using a **Self-Organizing Map (SOM),** this was carried out as follows by the following steps

    1. **Assign Topics**: Assign each news article to its dominant topic based on the LDA model.
    2. **Feature Vector Representation**: Represent each article as a feature vector, where each dimension corresponds to the probability of the article belonging to a specific topic.
    3. **Train SOM**: Train a Self-Organizing Map (SOM) using the feature vectors.
    4. **Visualize Clusters**: Visualize the clusters formed by the SOM on a 2D

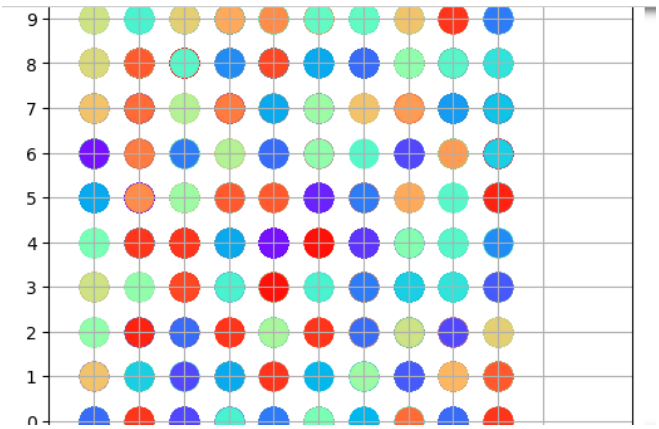
Post assigning the topics , we did the feature vector representation as shown below

	Topic_0	Topic_1	Topic_2	Topic_3	Topic_4	Topic_5	Topic_6	Topic_7	Topic_8	Topic_9	...	Topic_33	Topic_34	Topic_35	Topic_36	Topic_37
0	0.000000	0.000000	0.000000	0.102330	0.000000	0.0	0.000000	0.0	0.000000	0.000000	...	0.0	0.000000	0.000000	0.114812	0.000000
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.000000	0.000000	...	0.0	0.000000	0.000000	0.000000	0.000000
2	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.238408	0.000000	...	0.0	0.000000	0.000000	0.000000	0.000000
3	0.000000	0.000000	0.000000	0.000000	0.113734	0.0	0.000000	0.0	0.000000	0.000000	...	0.0	0.000000	0.000000	0.000000	0.000000
4	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.000000	0.000000	...	0.0	0.000000	0.000000	0.000000	0.075485
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
16782	0.170487	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.000000	0.000000	...	0.0	0.000000	0.170612	0.000000	0.000000
16783	0.000000	0.000000	0.000000	0.066168	0.000000	0.0	0.000000	0.0	0.000000	0.000000	...	0.0	0.000000	0.000000	0.000000	0.065179
16784	0.000000	0.078723	0.078704	0.000000	0.000000	0.0	0.078691	0.0	0.155667	0.000000	...	0.0	0.000000	0.000000	0.078727	0.000000
16785	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.113622	0.0	0.000000	0.113693	...	0.0	0.000000	0.000000	0.000000	0.000000
16786	0.000000	0.155677	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.078703	0.000000	...	0.0	0.078732	0.000000	0.000000	0.078677

16787 rows × 43 columns

Here each column represents a topic , so as we can see there are 43 topics against each row (rows represent each article or document), and the scores represent the probability of one article to be in a specific topic

- We tried to train and visualize SOM by using these feature vectors , as they will act as weights of how similar one neighbour is to the other.



Unlike supervised learning algorithms, Latent Dirichlet Allocation (LDA) is an unsupervised algorithm, meaning it doesn't have a target variable to predict, and therefore, there is no concept of accuracy in the traditional sense. Instead, the evaluation of an LDA model typically involves assessing the quality of the topics it discovers and how well it fits the data.
We can check the coherence score for the same, which is a measure used to evaluate the quality of the topics generated by the model

In our observation the coherence score , we received is NaN.

• Areas of Improvement

The dominant topics extracted from LDA maybe needs relevance , for example the topic modelling algorithm can fail to extract any topics from the corpus, coherence calculation cannot proceed, resulting in NaN scores

Quality of topics in the datasets , I think played a factor.

From the elbow method , the optimal number of clusters we got may not be great, that can be improved.

The quality of abstract in topic could be better.

Currently we have taken a concise dataset, but we need to look like in future we can perform this model on real data like journals and research papers by web-scrapping then we can check for the coherence score.

We Could have used some dimensionality reduction for faster processing

Also, as an area of improvement, we should have considered epochs of learning rate and sigma in SOM

• References

Topic Modeling in Python: Latent Dirichlet Allocation (LDA) | by Shashank Kapadia | Towards Data Science

Latent Dirichlet Allocation - GeeksforGeeks

Topic Modelling With LDA -A Hands-on Introduction - Analytics Vidhya

Gensim LDA Coherence Score Nan – Python (tutorialink.com)

Self Organizing Map(SOM) with Practical Implementation | by Amir Ali | The Art of Data Scicne | Medium

Self Organizing Maps - Kohonen Maps - GeeksforGeeks

Self-Organizing Maps: Theory and Implementation in Python with NumPy (stackabuse.com)