Final Group Report – Topic Classifier

Team Members: Isha Vohra, Srishti Garg, Pranav Setlur, Nathan Heo, Shaam Balasubramanian

Introduction

Research in its purest form is finding more problems from the mere chase of your solution to the original problem, and this feedback loop is the reason this sub-team was formed. The topic classifier sub team was formed due to the needs of the sub-teams in the VIP-Getting to Know U. Other sub-teams discovered problems that needed solutions to effectively tackle their respective goal. A communal problem discovered among the sub-teams was the absence of a typographic hierarchy. The team to tackle this issue was formed, Topic Classifier, and our goal was set to develop a model that can automatically classify, and model topics through semantics.

User Stories

- Dashboard Data and Design: Earlier this year, during the launch of the Getting to Know U VIP (Vertically Integrated Projects), this sub team discovered a need for topic classification since one of the issues with the Tableau dashboards that were designed was that topic keywords with the same definition were not being grouped together. Therefore, a need for a semantics-based topic modelling method was uncovered. Now, with this sub team being split up into the Proposals, Affiliations, and Provost teams, more sub teams stand to benefit from our work than even in previous semesters.
- Collab Web: The data this sub team is using includes thousands of article documents with no structure. Knowing what field/topic these articles are a part of would supply an effortless way to structure them. While there are keywords in every article, that is too granular as there would be thousands of categories making the whole thing useless. But something as broad as 'physics' is too broad. That balance is key for this team, which is where Topic Classifier can step in.

Our Process

At the start of the semester, we focused on trying to build from Spring 2022's work with Latent Dirichlet Allocation, otherwise known as LDA. However, we quickly realized that we were coming to a crossroads and pivoted to using k-means clustering in tandem with other similar clustering methods to see which one would be the best fit for our purpose. Some other clustering methods included Gaussian Mixture Modeling and Hierarchical Clustering. Gaussian Mixture Modeling is a probabilistic model that gathers all data points that are generated from a mixture of finite number of distributions. Even though it was a fast algorithm to learn, the algorithm is known for causing divergence and using up all the components. Hierarchical Clustering is geared towards creating clusters that have similar objects in the group. The elements in each cluster are very similar to

each other but the clusters themselves are different from other ones. After finding clusters that are somewhat related, clusters were grouped together. Even though this is an effective technique, it involves making many arbitrary decisions and does not work well with mixed data types and large datasets.

K-means clustering, on the other hand, is an unsupervised learning technique. Unsupervised learning techniques use machine learning algorithms to group datasets that have not been labeled. These techniques allow for the discovery of hidden patterns without human knowledge. This type of clustering takes a similar approach to hierarchical clustering where collections of data points are grouped together because of similar similarities. We looked through similar past projects and realized that the best course of action would be to use k-means clustering on a dataset that already is loaded with the metadata of over 1.7 million papers and has a Kaggle usability score of 8.75. We used TF-IDF vectorization, which is a machine learning technique that is an alternative to Bag of Words, to assign weights to words within the papers' abstracts based on semantics. For our case, we chose this method over Bag of Words since Bag of Words does not consider the "importance" of certain words over others and instead assigns a more arbitrary count value to the words based on occurrences. When visualizing the clusters, we chose to go with the t-SNE method of dimensionality reduction over other methods like LDA, UMAP, and PCA since this method yielded the most distinct clusters based on our quick seaborn visualization implementations.

Another avenue the team decided to explore is the NLTK library. NLTK is unique due to its abundance of pre-trained models and corpora which allowed us to analyze textual information with ease. Although it has been established that topic classification is a more involved process in comparison to topic modeling, the improved accuracy was a pull toward implementing this method in our research. During our extensive research, an online textbook that walks through the use of NLTK was discovered and studied. The book had been recently updated to support Python 3 and NLTK 3. It covered a range of topics on NLTK, including Accessing Text Corpora, Categorizing and Tagging Words, and Extracting Information from Text. Furthermore, a site that describes the process of text classification was also utilized to further our progress. Both are listed below.

After our preliminary research, we decided to implement topic classification on an actual dataset. Rather than using a Georgia Tech dataset, which would have to be cleaned, we used the Arxiv Dataset from Kaggle. This dataset contained research paper titles with about 14 different attributes (comments, licenses, versions, etc.) that could be used to organize the information. We then installed the NLTK module on a Google Colab notebook to organize the testing. The NLTK package was then downloaded onto the notebook following some initial errors. From the ArXiv dataset, we decided to focus on each research paper's abstracts and titles to assist with the topic classifications of the papers. The narrowing of applicable attributes allowed us to set out a manageable process toward classifying the textual information with an intermediate goal of relation detection and identification. With the .json file imported, the functions contained within the

NLTK “popular” subset were used for the sentence segmentation and tokenization of the dataset. In order to test the code’s applicability, a Python slice of ten was used due to the large number of titles contained within the dataset (2158809). Given these ten titles and abstracts, the sentences contained within the dataset were tokenized.

After our initial tokenization attempt, we ran a frequency distribution on the titles and abstracts to get the most common words used. However, we discovered that the most common words were either stop words (like ‘the’, ‘are’, etc.) or punctuation marks. This does not provide us with any useful information on the data. Therefore, we decided to first clean the titles and abstracts by removing the stop words and punctuation marks. We also chose to lemmatize the words for consistency. For instance, before cleaning, one of the abstracts contained 983 words. After cleaning, we were able to reduce this to 713 words, a significant improvement. Then, our tokenization provided more usable results. In addition to tokenizing, we also tagged each token with a part-of-speech (POS). Now, the titles and abstracts had been converted to a list of tuples, where the first part of the tuple was a word and the second was its POS.

Next, we attempted chunking the data, which is a basic technique for entity detection. We used the ne_chunker, which helps with Named Entity Recognition. The goal of a named entity recognition system is to find all textual mentions of a named entity. The result of our chunking algorithm was a NLTK tree for each title and abstract. The tree contained all of the words in the abstract and its associated POS, but it also labeled some words and phrases with a named entity, like ‘ORGANIZATION’, ‘PERSON’, or ‘LOCATION.’

Finally, we ran some visualizations on the chunked data. We used the first 10 research papers from the ArXiv dataset to create a representation of how we could potentially use NLTK to assist in topic classification. After retrieving the abstract and titles for the papers, we lemmatized the texts to compare the abstract and title against the rest of the research papers. We completed this process by creating a frequency distribution of the common words between two research papers and continued the process until each paper was compared to each other. Through this process we were able to derive a percentage that displayed similarities amongst the different papers. With this data, we decided to create a heatmap to identify the relationships of the research papers.

———

Next Steps

The research, once again, unfolded predictably, and left us with more questions than answers. The research we conducted through k-means clustering left us with more technical questions to answer. To create an effective model, having fundamental background knowledge is especially important, which is why this was focused on this semester amongst the new members of the sub team. The sub team as a whole made a lot of progress and was able to display effective results from real datasets. The next step for this sub team in terms of k-means clustering is to work with new datasets that closely resemble the datasets that the other sub-teams might work with. Additionally, for

next semester, the k-means clustering method can be combined with the NLTK method to classify datasets that the other sub-teams provide. This will involve constant contact with the professor and other sub-team leaders for working on valid datasets. We also want to work to find new visualizations sources such as the ones we used this semester (Seaborn, Plotly) or gain a better understanding of them for our use case.

Appendices

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction import text
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from nltk.tokenize import RegexpTokenizer
from nltk.stem.snowball import SnowballStemmer
%matplotlib inline
```

Figure 1 - the packages that need to be imported -> numpy, pandas, matplotlib.pylot

```
vectorizer3 = TfidfVectorizer(stop_words = stop_words, tokenizer = tokenize, max_features = 1000)
X3 = vectorizer3.fit_transform(desc)
words = vectorizer3.get_feature_names()
```

```
from sklearn.cluster import KMeans
wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
    kmeans.fit(X3)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11),wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.savefig('elbow.png')
plt.show()
```

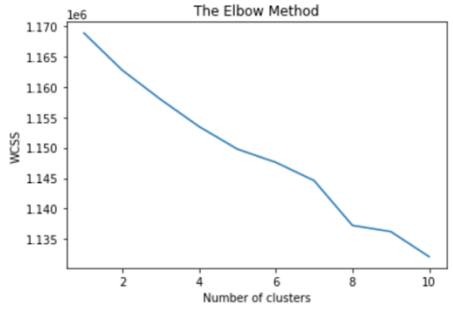

Figure 2: The graph shows how many clusters could be created with the data set.

```
print(words[250:300])
```

```
['decis', 'declar', 'defenc', 'defend', 'delay', 'deliv', 'demand', 'deni', 'despit', 'destroy', 'detent', 'develop',
'die', 'director', 'disabl', 'disast', 'discuss', 'diseas', 'dismiss', 'disput', 'doctor', 'dog', 'dollar', 'domest',
'donald', 'donat', 'doubl', 'doubt', 'draw', 'dri', 'drink', 'drive', 'driver', 'drop', 'drought', 'drown', 'drug',
'drum', 'dump', 'dure', 'eagl', 'earli', 'eas', 'east', 'econom', 'economi', 'edg', 'educ', 'effort', 'elder']
```

```
# 3 Clusters
kmeans = KMeans(n_clusters = 3, n_init = 20, n_jobs = 1) # n_init(number of iterations for clsutering) n_jobs(number of
kmeans.fit(X3)
# We look at 3 the clusters generated by k-means.
common_words = kmeans.cluster_centers_.argsort()[:,-1:-26:-1]
for num, centroid in enumerate(common_words):
    print(str(num) + ' : ' + ', '.join(words[word] for word in centroid))
```

```
/Users/srishtigarg/opt/anaconda3/lib/python3.9/site-packages/sklearn/cluster/_kmeans.py:792: FutureWarning: 'n_jobs'
was deprecated in version 0.23 and will be removed in 1.0 (renaming of 0.25).
  warnings.warn("'n_jobs' was deprecated in version 0.23 and will be"
```

```
0 : polic, new, man, plan, charg, australia, council, court, kill, govt, australian, death, nsw, report, face, warn,
crash, year, sydney, water, urg, attack, fund, murder, open
1 : say, polic, need, minist, govt, expert, trump, plan, mp, australia, union, council, report, govern, labor, pm, wo
nt, group, new, public, opposit, chang, health, wa, time
2 : win, award, cup, titl, open, gold, stage, world, final, tour, elect, australia, lead, seri, claim, aussi, austral
ian, second, grand, england, big, race, record, battl, m
```

Figure 3: Three clusters have been created with words that are somewhat related.

```
kmeans = KMeans(n_clusters = 8, n_init = 20, n_jobs = 1)
kmea
# Fi                                    enerated by k-means.
common_words = kmeans.cluster_centers_.argsort()[:,-1:-26:-1]
for num, centroid in enumerate(common_words):
    print(str(num) + ' : ' + ', '.join(words[word] for word in centroid))
```

click to scroll output; double click to hide

```
/Users/srishtigarg/opt/anaconda3/lib/python3.9/site-packages/sklearn/cluster/_kmeans.py:792: FutureWarning: 'n_jobs'
was deprecated in version 0.23 and will be removed in 1.0 (renaming of 0.25).
  warnings.warn("'n_jobs' was deprecated in version 0.23 and will be"
```

```
0 : plan, council, court, govt, charg, kill, nsw, water, face, death, crash, attack, new, urg, woman, car, murder, ac
cus, driver, consid, hear, fatal, teen, road, report
1 : new, australia, report, warn, interview, sydney, open, day, chang, wa, elect, urg, hit, world, hous, home, test,
claim, set, talk, qld, final, china, return, nation
2 : say, polic, need, minist, expert, govt, trump, plan, mp, australia, union, report, council, labor, govern, pm, wo
nt, public, new, group, opposit, chang, time, wa, lawyer
3 : man, charg, murder, jail, polic, court, die, stab, arrest, face, miss, assault, accus, kill, death, guilti, attac
k, car, crash, shoot, child, sydney, alleg, sex, search
4 : win, australian, year, open, new, old, award, jail, cup, celebr, titl, world, gold, final, tour, south, australi
a, elect, stage, lead, end, share, market, dollar, day
5 : polic, investig, probe, offic, search, hunt, arrest, death, car, shoot, drug, charg, miss, seek, driver, crash, a
ttack, murder, assault, suspect, fatal, raid, protest, warn, woman
6 : fund, boost, govt, feder, council, school, cut, health, road, govern, budget, urg, hospit, centr, seek, help, reg
ion, group, servic, research, project, rais, announc, m, pledg
7 : health, record, defend, mental, servic, new, minist, break, case, world, set, govt, nsw, qld, covid, plan, hospi
t, indigen, coronavirus, report, worker, sa, m, warn, high
```

Figure 4: Eight clusters have been created. The clusters in the previous clusters are broken up into more specific clusters.

```
First abstract before cleaning
'  A fully differential calculation in perturbative quantum chromodynamics is\npresented for the production of massive photon pairs at hadron colliders. All
\nnext-to-leading order perturbative contributions from quark-antiquark,\ngluon-(anti)quark, and gluon-gluon subprocesses are included, as well as\nall-orde
rs resummation of initial-state gluon radiation valid at\nnext-to-next-to-leading logarithmic accuracy. The region of phase space is\nspecified in which the
calculation is most reliable. Good agreement is\ndemonstrated with data from the Fermilab Tevatron, and predictions are made for\nmore detailed tests with C
DF and DO data. Predictions are shown for\ndistributions of diphoton pairs produced at the energy of the Large Hadron\nCollider (LHC). Distributions of the
diphoton pairs from the decay of a Higgs\nboson are contrasted with those produced from QCD processes at the LHC, showing\nthat enhanced sensitivity to the
signal can be obtained with judicious\nselection of events.\n'
983

First abstract after cleaning
'fully differential calculation perturbative quantum chromodynamics presented production massive photon pair hadron collider nexttoleading order perturbativ
e contribution quarkantiquark gluonantiquark gluongluon subprocesses included well allorders resummation initialstate gluon radiation valid nexttonexttolead
ing logarithmic accuracy region phase space specified calculation reliable good agreement demonstrated data fermilab tevatron prediction made detailed test
cdf data prediction shown distribution diphoton pair produced energy large hadron collider lhc distribution diphoton pair decay higgs boson contrasted produ
ced qcd process lhc showing enhanced sensitivity signal obtained judicious selection event'
713
```

Figure 5: Effects of removing the stop words and punctuation and lemmatizing the first abstract for the Arxiv Dataset.

```
display(tokenized_sents[0])
#print(tokenized_abstracts)
#print(len(tokenized_sents))
#print([len(a) for a in tokenized_sents])
```

```
[('Calculation', 'NN'),
 ('of', 'IN'),
 ('prompt', 'JJ'),
 ('diphoton', 'NN'),
 ('production', 'NN'),
 ('cross', 'NN'),
 ('sections', 'NNS'),
 ('at', 'IN'),
 ('Tevatron', 'NNP'),
 ('and', 'CC'),
 ('LHC', 'NNP'),
 ('energies', 'NNS')]
```

Figure 6: Tokenization of the first title – tuple containing the word and its POS

```
chunked_abstracts = []
chunked_titles = []

for title, abstract in zip(tokenized_sents, tokenized_abstracts):
  #chunked_abstract = chunk_parser.parse(abstract)
  chunked_abstract = nltk.ne_chunk(abstract)
  chunked_title = nltk.ne_chunk(title)
  chunked_abstracts.append(chunked_abstract)
  chunked_titles.append(chunked_title)


#print(chunked_abstracts[0])
print(chunked_titles[0])
```

```
(S
  (GPE Calculation/NN)
  of/IN
  prompt/JJ
  diphoton/NN
  production/NN
  cross/NN
  sections/NNS
  at/IN
  (ORGANIZATION Tevatron/NNP)
  and/CC
  (ORGANIZATION LHC/NNP)
  energies/NNS)
```

Figure 7: Result of the ne_chunker on the first title. The title has been converted to a tree with some relational entities identified.

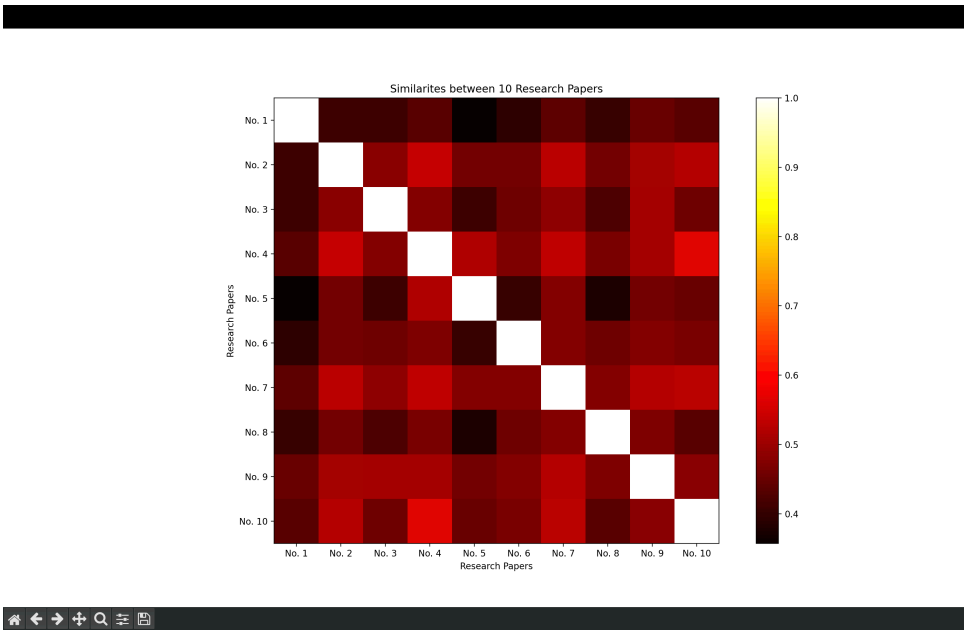

Figure 8: NLTK Effort - Heat map showing similarities found from algorithm amongst first 10 research papers in dataset.

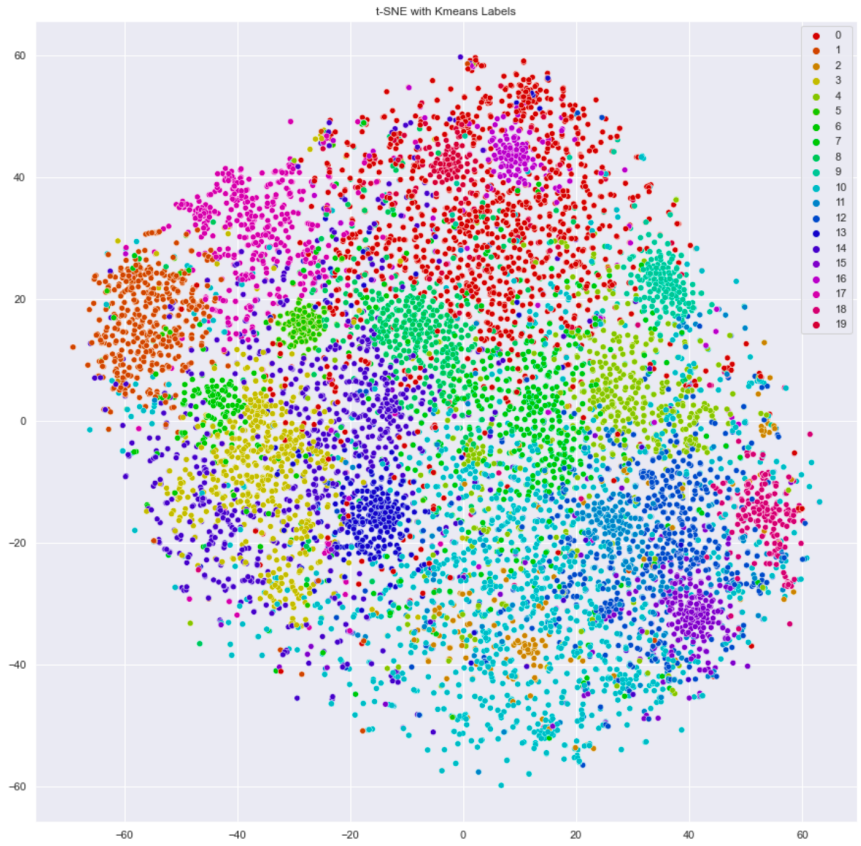

Figure 9: Kmeans Effort - Distinct clusters based on TF-IDF vectorization and t-SNE for dimensionality reduction

Important Links

Link to NLTK textbook: https://www.nltk.org/book/

Link to text classification tutorial: https://medium.com/analytics-vidhya/nlp-tutorial-for-text-classification-in-python-8f19cd17b49e

Link to code: https://github.gatech.edu/Getting-to-know-U/ClassifiedTopics