Analyzing Reddit Posts

Using Reddit's Data to dive into the college experience

The Problem

- Build a binary classification model that can distinguish which of two subreddits, (r/rutgers or r/UPenn), a particular post belongs to
- This will lead to a better understanding of the similarities and differences between the two pages
- The classification can also be used by Reddit to make suggestions for the advertisements for the pages that are specific to it

The Data

- The dataset contains information (post title, created date & time, user, texts, urls and score) on posts acquired via Reddit's historical data
- Dataset contains 151914 rows × 8 columns of information of both the pages combined

The Data

```
1  combined_data
```

	author	title	score	created	link	text	url	subreddit
0	u/MsChanandalerBong	Tuition Rally at Brower	6	2010-04-28 20:11	https://www.reddit.com/r/rutgers/comments/bxhy...	NaN	http://www.dailytargum.com/multimedia/2.12020/...	Rutgers
1	u/MsChanandalerBong	Rutgers Mohamed Sanu for Heisman? More Likely ...	6	2010-04-28 20:22	https://www.reddit.com/r/rutgers/comments/bxi2...	NaN	http://bleacherreport.com/articles/357996-sanu...	Rutgers
2	u/rustoof	So how many of us are here?	18	2010-05-08 16:36	https://www.reddit.com/r/rutgers/comments/c1k7...	Hi guys/gals,\n\I'm gonna be a junior in the fa...	https://www.reddit.com/r/rutgers/comments/c1k7...	Rutgers
3	u/tomatohs	Gov. Chris Christie wants all N.J. medical mar...	11	2010-06-19 12:09	https://www.reddit.com/r/rutgers/comments/cgru...	NaN	http://www.nj.com/news/index.ssf/2010/06/chris...	Rutgers
4	u/thisismypasswordtoo	Setting up DC++ (Windows)	9	2010-08-30 20:55	https://www.reddit.com/r/rutgers/comments/d7hl...	(throwaway account)\n\n* you must be on the Ru...	https://www.reddit.com/r/rutgers/comments/d7hl...	Rutgers
...	...	...	...	...	...	...	...	...
151909	u/softalmondmilk	SUPPLEMENTAL ESSAY	1	2022-12-31 03:56	https://www.reddit.com/r/UPenn/comments/zzpsgn...	[removed]	https://www.reddit.com/r/UPenn/comments/zzpsgn...	UPenn
151910	u/softalmondmilk	ISP and BFS	3	2022-12-31 10:13	https://www.reddit.com/r/UPenn/comments/zzw6fa...	I was hoping to know if ISP is still offered f...	https://www.reddit.com/r/UPenn/comments/zzw6fa...	UPenn
151911	u/[deleted]	Is my schedule too much work?	1	2022-12-31 12:47	https://www.reddit.com/r/UPenn/comments/zzzhvj...	[deleted]	NaN	UPenn
151912	u/itsamayzing	Does anyone know what topics are covered in ST...	5	2022-12-31 16:24	https://www.reddit.com/r/UPenn/comments/100425...	Going straight into stat 1020 because I had a ...	https://www.reddit.com/r/UPenn/comments/100425...	UPenn
151913	u/[deleted]	Does Anyone Here use the universities HSA heal...	1	2022-12-31 18:50	https://www.reddit.com/r/UPenn/comments/1006xk...	[deleted]	NaN	UPenn

151914 rows × 8 columns

1

Data Cleaning & Exploration

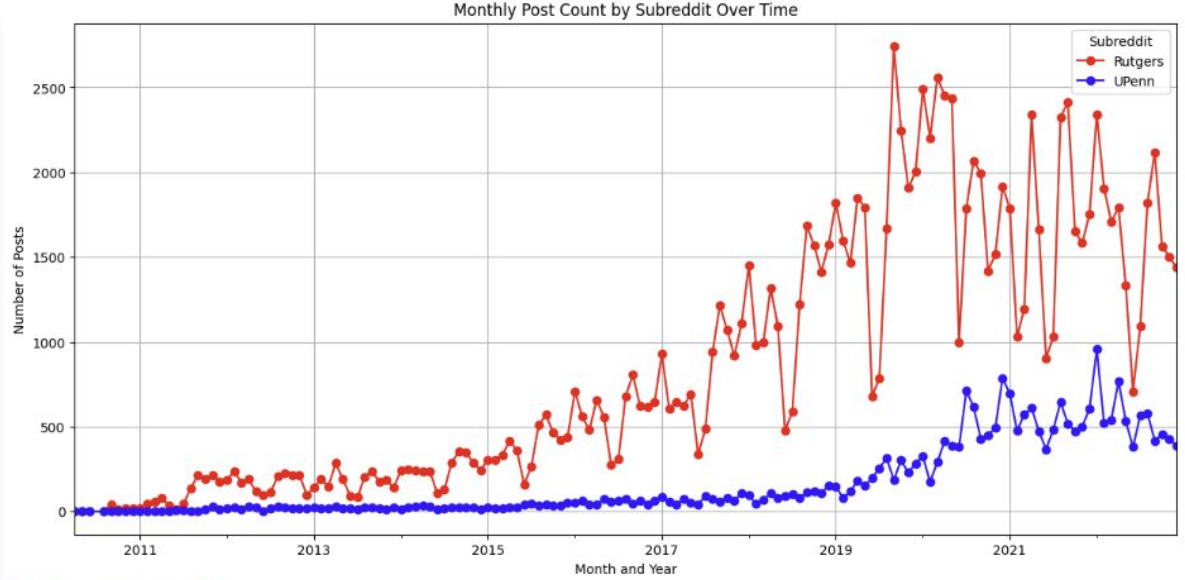
Checking for and cleaning null values, emojis, multiple languages and duplicates; and visualizing the data to better understand it.

Annual posts by subreddit

Monthly Post

Taking a look at our posts we can definitely see the number posts by the subreddit rutgers leads when compared to upenn it also attributes to the number of students going there.

Another thing to notice is that most of these posts in both the subreddits have a cyclical pattern so let's move and explore that a bit more



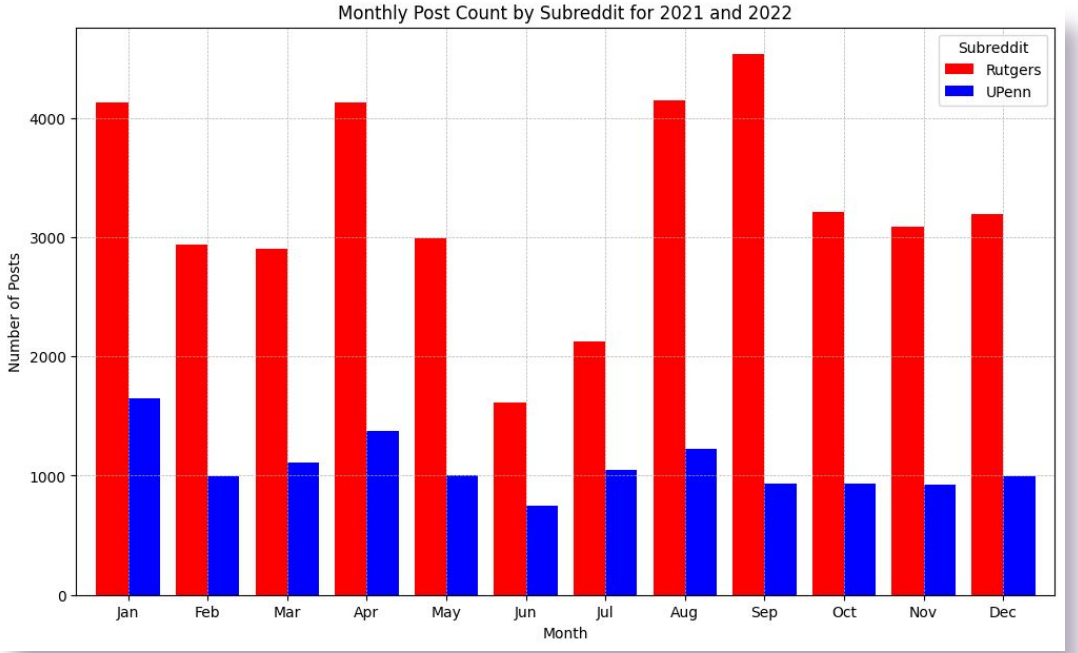

```
(Timestamp('2010-04-28 20:11:00'),
 Timestamp('2022-12-31 18:50:00'),
 subreddit
 Rutgers    127207
 UPenn       24707
 Name: count, dtype: int64)
```

Monthly posts by the subreddit (2021-2022)

Description

As we see there is a cyclical nature to the posts made by people during the academic year especially peaking when the semester starts and then again during exam season as well.

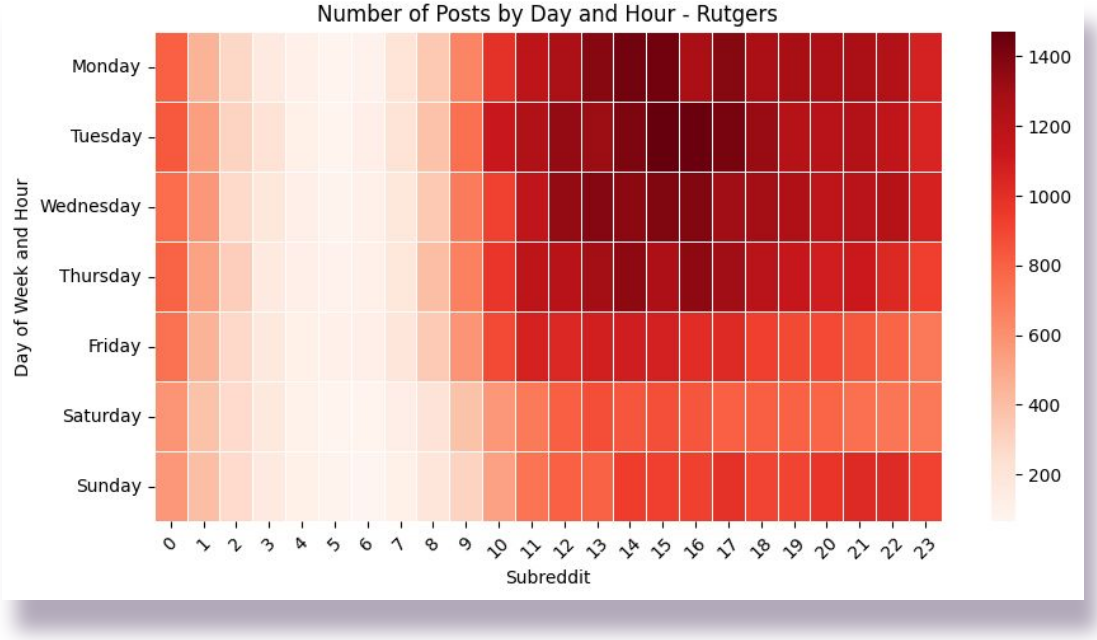
Let's also check how frequent the students post in the day.



Weekly posts by r/Rutgers average (2021-2022)

Description

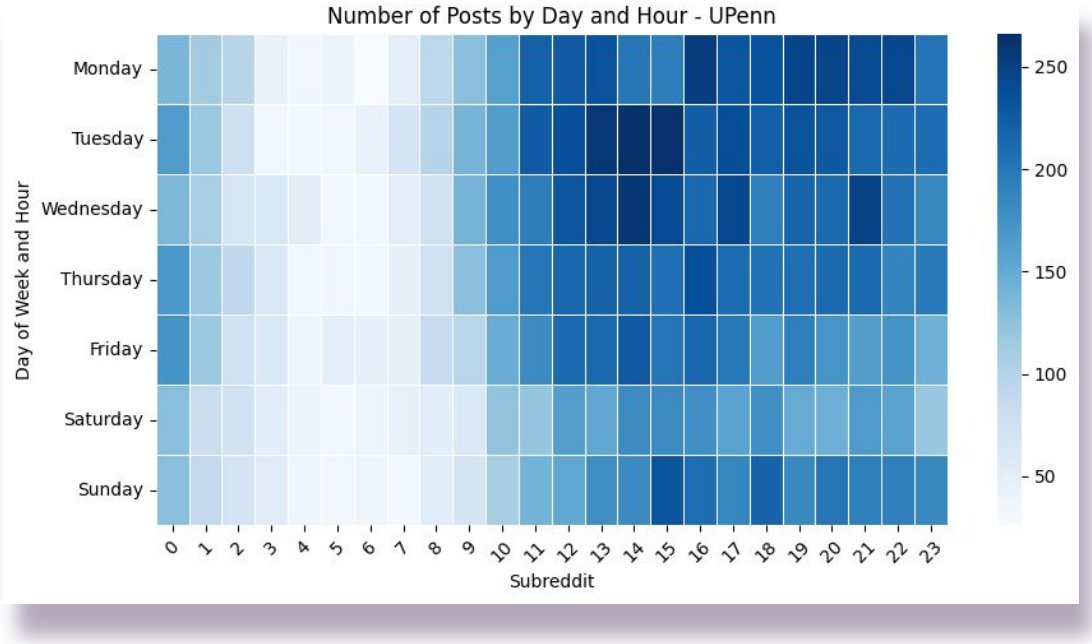
r/Rutgers shows higher posting frequency with significant peaks after 11:00 AM and slows down around 10:00 PM, while we also see that the posts peak during the weekdays and slow down during the weekends.



Weekly posts by r/UPenn average (2021-2022)

Description

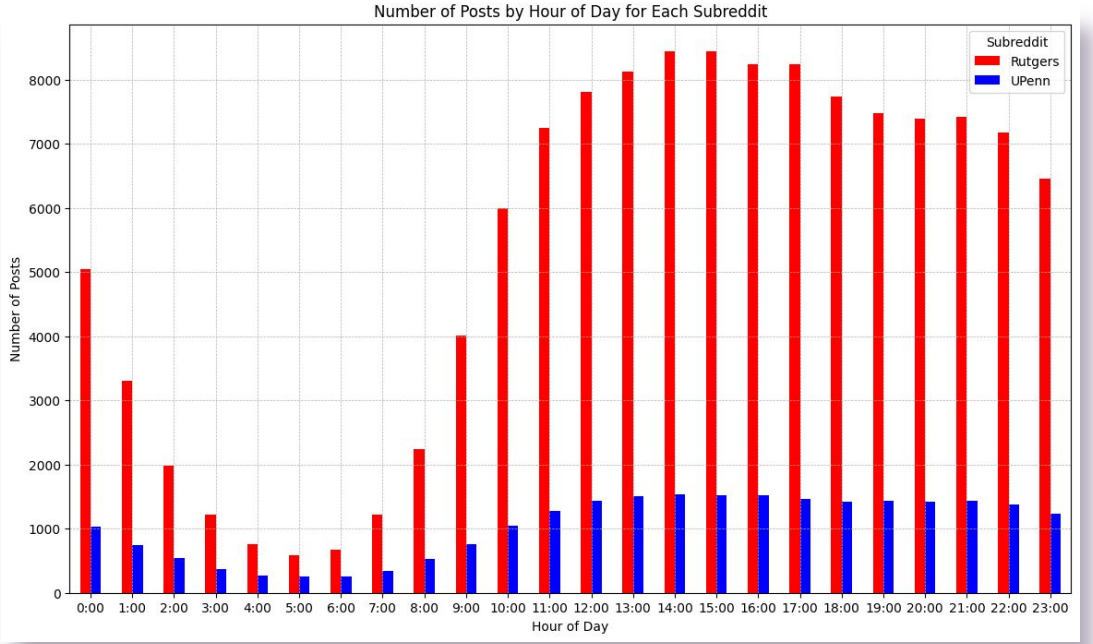
r/UPenn activity peaks in the late afternoons to evenings, with engagement from 3PM to midnight. Unlike Rutgers, UPenn maintains a steady flow of posts during weekends, though still with an emphasis on evening hours, indicating a user base that is actively posting throughout the week with a notable inclination for nighttime interaction.



Daily posts by the subreddit average (2021-2022)

Description

r/Rutgers shows higher posting frequency with significant peaks after 11:00 AM and slows down around 10:00 PM, while r/UPenn has a more consistent posting pattern throughout the day.

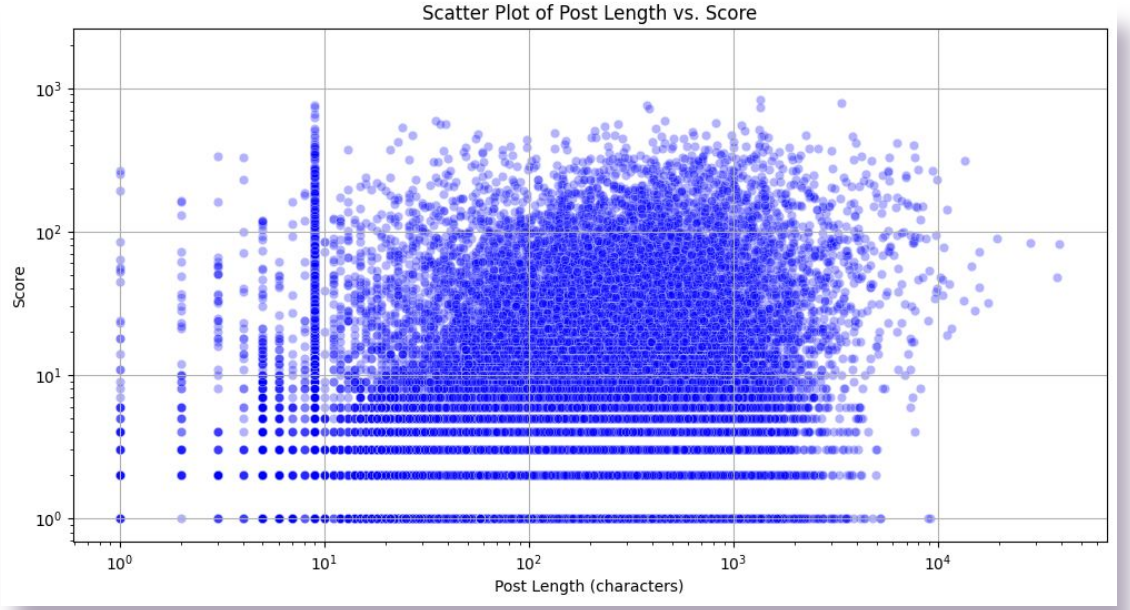


Post length vs Score

Description

The scatter plot depicts the relationship between post length and score on a logarithmic scale, with a very low correlation coefficient of 0.0158.

The plot shows a wide dispersion of points without a clear trend, indicating that post length does not strongly influence the score. The concentration of dots along the lower part of the y-axis suggests that most posts, regardless of length, receive a low to moderate score. Posts with a very high score are rare and do not appear to be dependent on length.

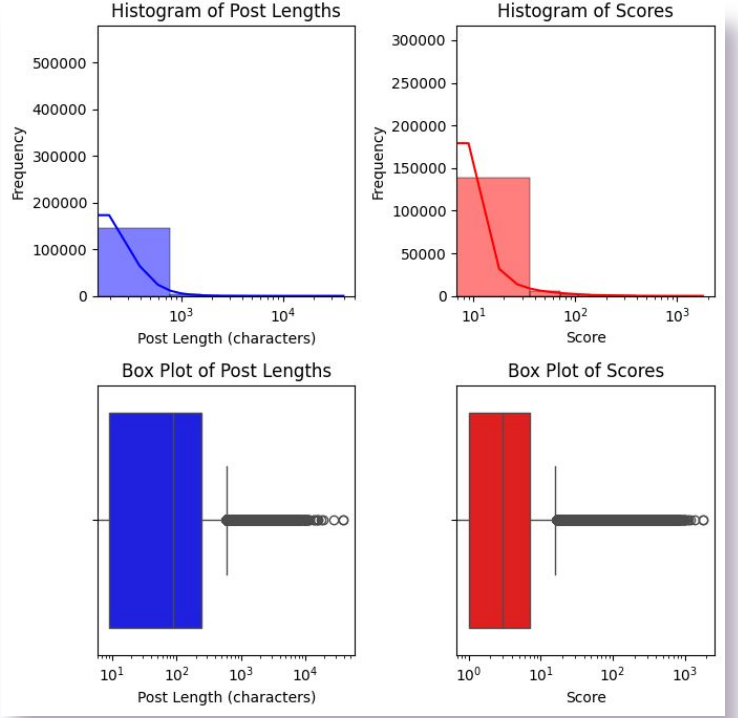


0.01576762917368809

Histogram & Box Plots for post length and score

Description

The data reveals a right-skewed distribution for both post lengths and scores, with the majority being short posts with lower scores and only a few outliers with high scores or long post lengths. The box plots further illustrate this concentration of lower values and the presence of extreme outliers.



If your wondering what's the post with the highest score?

Post with the largest score:

author	u/kingvincent25
title	Guys, I need your help. I need to create a subreddit for a class as a final project but I don’t have enough karma to create one. Can you guys please upvote this so I can finish my final!
score	1782
created	2020-12-14 10:24:00
link	https://www.reddit.com/r/rutgers/comments/kczmwd/guys_i_need_your_help_i_need_to_create_a/
text	NaN
url	https://www.reddit.com/r/rutgers/comments/kczmwd/guys_i_need_your_help_i_need_to_create_a/
subreddit	Rutgers
month_year	2020-12
hour_of_day	10
day_of_week	Monday
post_length	0.0

Data Cleaning
Text Cleaning

Description

A simple text cleaning function to clean our titles for each posts the only thing new here would be clearing the emojis out from our posts as well.

Text Cleaning

```
[27]
def Text_Cleaning(Text):
    # Remove emojis using the emoji package
    Text = emoji.replace_emoji(Text, replace='')

    # Remove URLs
    Text = re.sub(r'https?://\S+|www\.\S+', '', Text)

    # Remove newlines and tabs
    Text = re.sub(r'[\t\n\r]+', ' ', Text)

    # Remove punctuations
    punc_table = str.maketrans('', '', string.punctuation)
    Text = Text.translate(punc_table)

    # Remove non-ASCII characters
    Text = Text.encode("ascii", errors="ignore").decode()

    # Remove numbers and words with numbers
    Text = re.sub(r'\w*\d\w*', '', Text)

    # Remove single characters
    Text = re.sub(r'\s+[a-zA-Z]\s+', ' ', Text)
    return Text
```

Data Cleaning
Text Processing

Description

A double pass to get rid of the stop words and tokenizing our data along with it.

Text Processing

```
# Stopwords
Stopwords = set(nltk.corpus.stopwords.words("english"))

def Text_Processing(Text):
  Processed_Text = list()
  Lemmatizer = WordNetLemmatizer()

  # Tokens of Words
  Tokens = nltk.word_tokenize(Text)

  # Removing Stopwords and Lemmatizing Words
  for word in Tokens:
    if word not in Stopwords:
      Processed_Text.append(Lemmatizer.lemmatize(word))

  return(" ".join(Processed_Text))
```

Cleaning Null & Blank title

Description

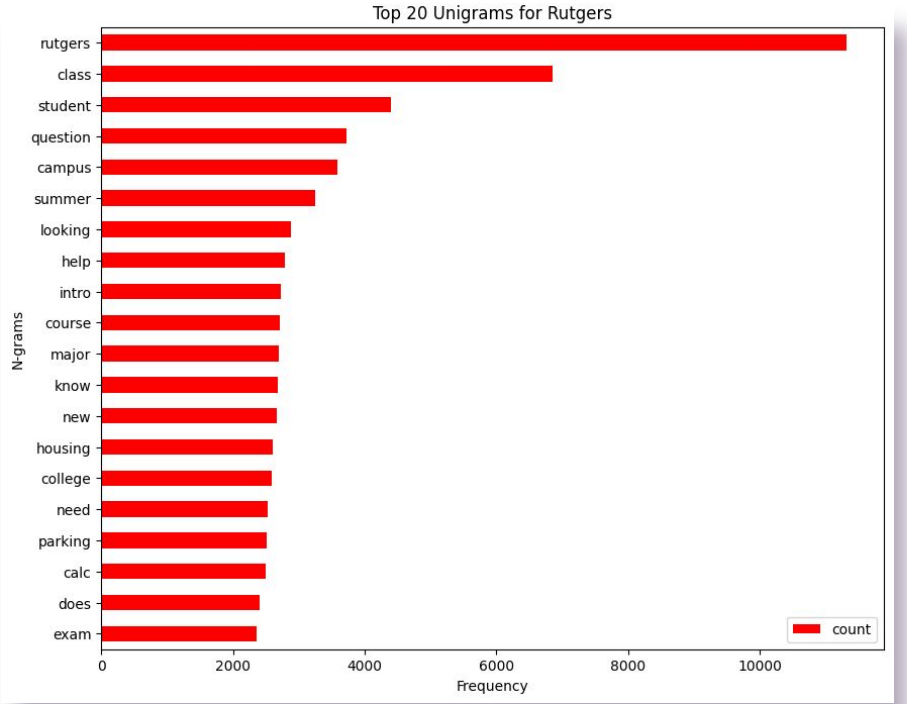
Finally lets get rid of the deleted titles and delete the blank posts as well

```
[30]  1  df = df[df['processed_title'] != 'deleted user']
      2  df = df[df['processed_title'].str.strip() != '']
```

```
1  df.value_counts('processed_title')
```

```
processed_title
Calc                                                                  102
Housing                                                                92
Math                                                                   82
CIS                                                                    53
Chem                                                                   53
                                                                     ...
Hey Penn international applicant Scotland considering applying Can help  1
Hey Penn engineer cool program unique opportunity                      1
Hey Penn GODS admission interview scheduled day need guidance          1
Hey Orgo tutor                                                         1
zoomin thrivin                                                         1
Name: count, Length: 135109, dtype: int64
```

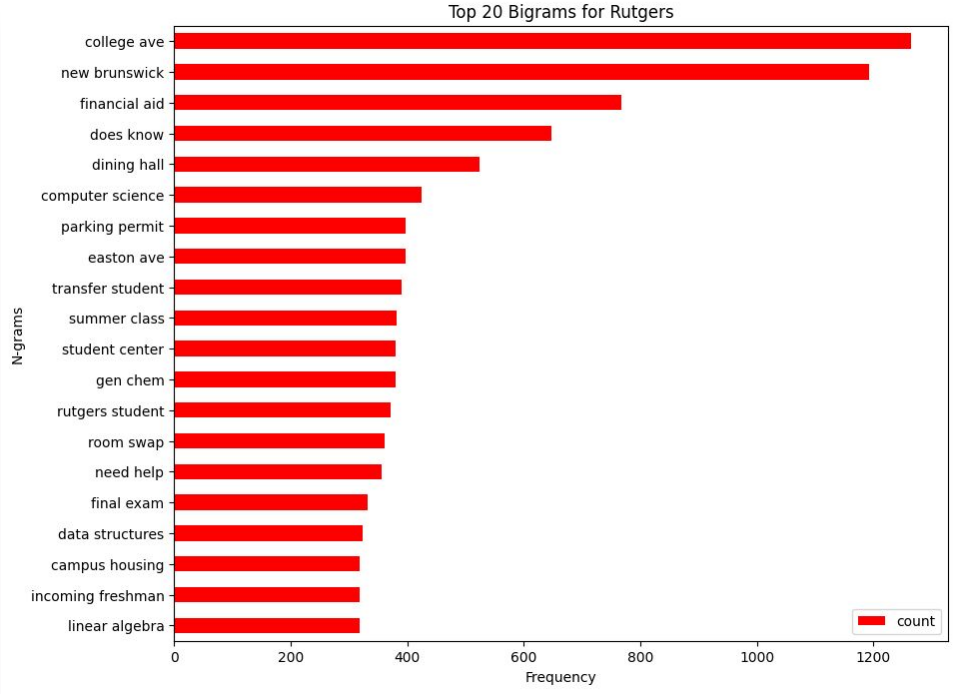
Unigram for Rutgers



r/Rutgers

The chart shows the most common words in Rutgers-related discussions, with "rutgers" leading, followed by academic terms like "class" and "student," and practical concerns such as "campus" and "housing." The frequency of these unigrams highlights the focus areas of the community's conversations.

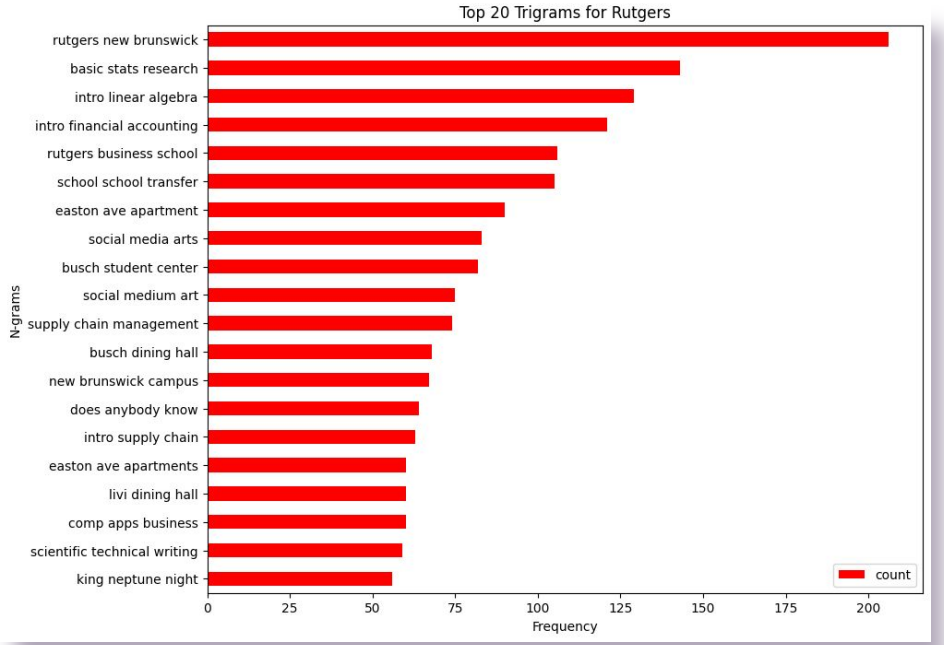
Bigram for Rutgers



r/Rutgers

The bar chart presents the top 20 bigrams related to Rutgers, with prominent phrases such as "college ave," "new brunswick," and "financial aid." The chart reflects key topics of conversation like locations, academic concerns, and student life logistics.

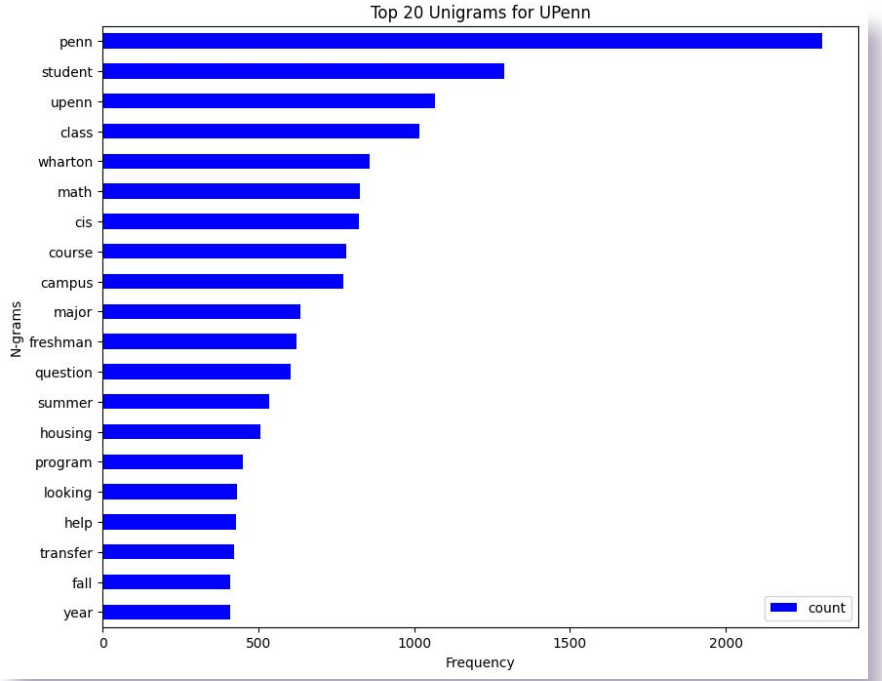
Trigrams for Rutgers



r/Rutgers

The bar chart shows the top 20 trigrams associated with Rutgers, focusing on specific courses like "intro linear algebra," facilities such as "busch dining hall," and events like "king neptune night." These trigrams indicate detailed academic, residential, and cultural aspects that are frequently discussed within the Rutgers community.

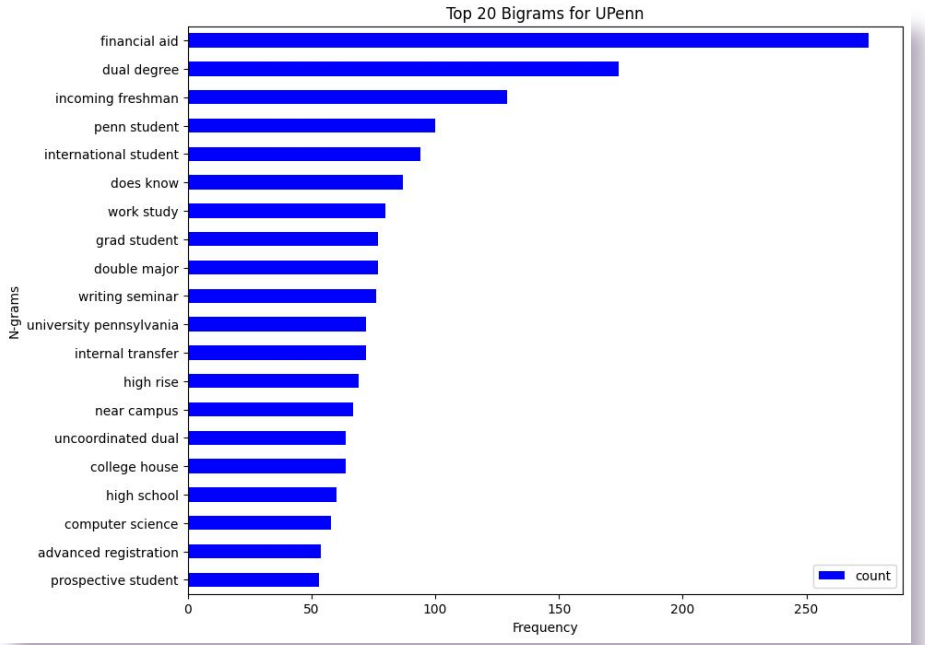
Unigrams for UPenn



r/UPenn

The chart highlights the top unigrams from UPenn discussions, featuring prominent mentions of "penn," "student," "upenn," and academic-related words like "class," "wharton," and "math." The prevalence of these words indicates a strong focus on the educational environment and student life at the University of Pennsylvania.

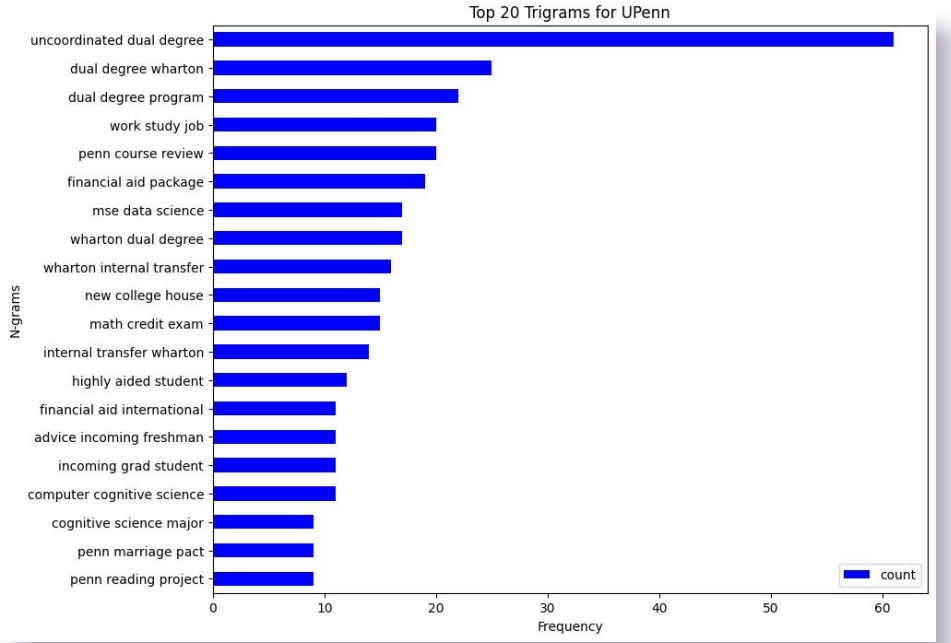
Bigrams for UPenn



r/UPenn

The chart depicts the top 20 bigrams from UPenn-related texts, focusing on "financial aid," "dual degree," and various student statuses like "incoming freshman" and "grad student." The terms reflect key aspects of the university experience, from administrative processes to student life and housing.

Trigrams for UPenn



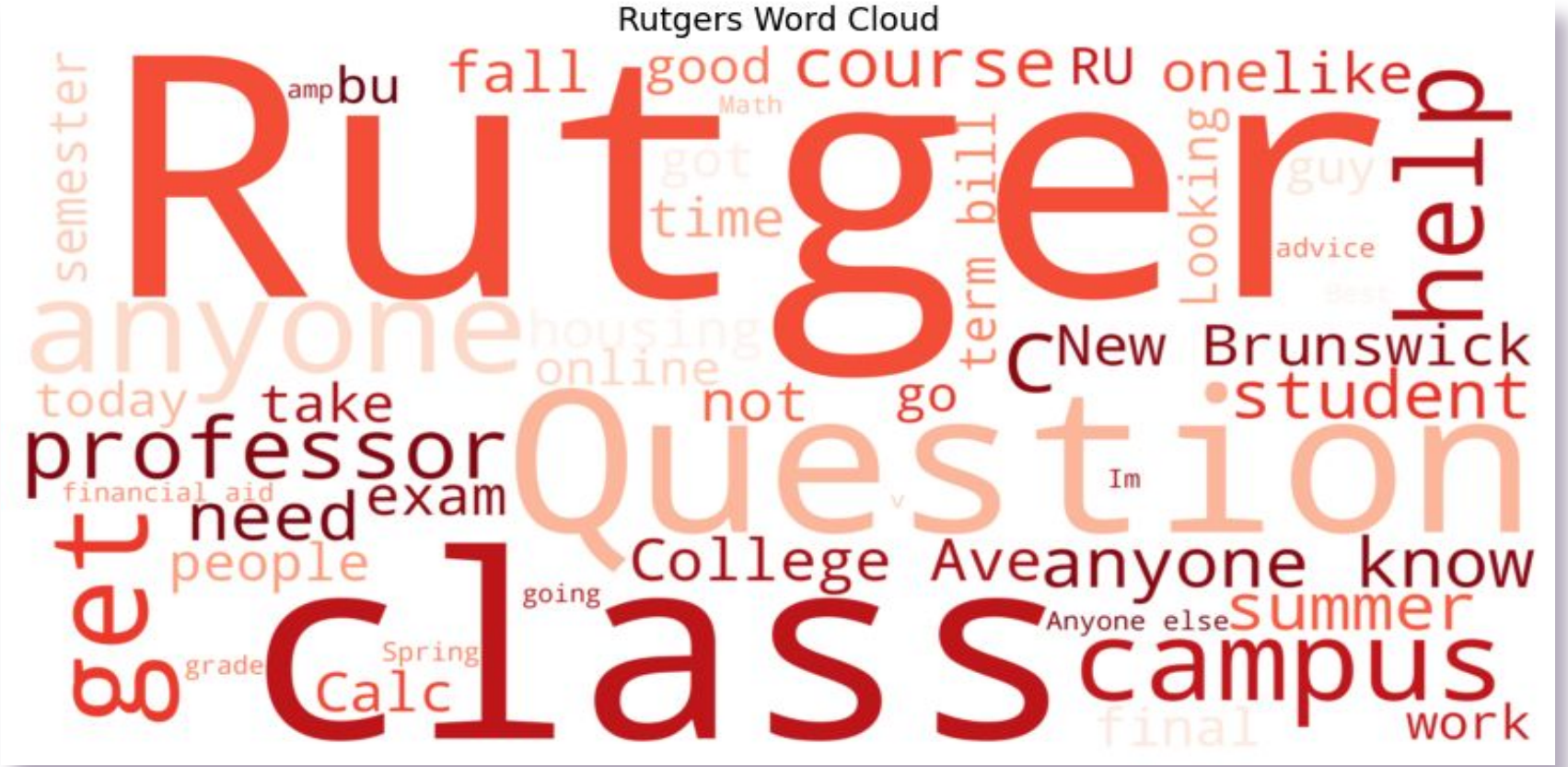
r/UPenn

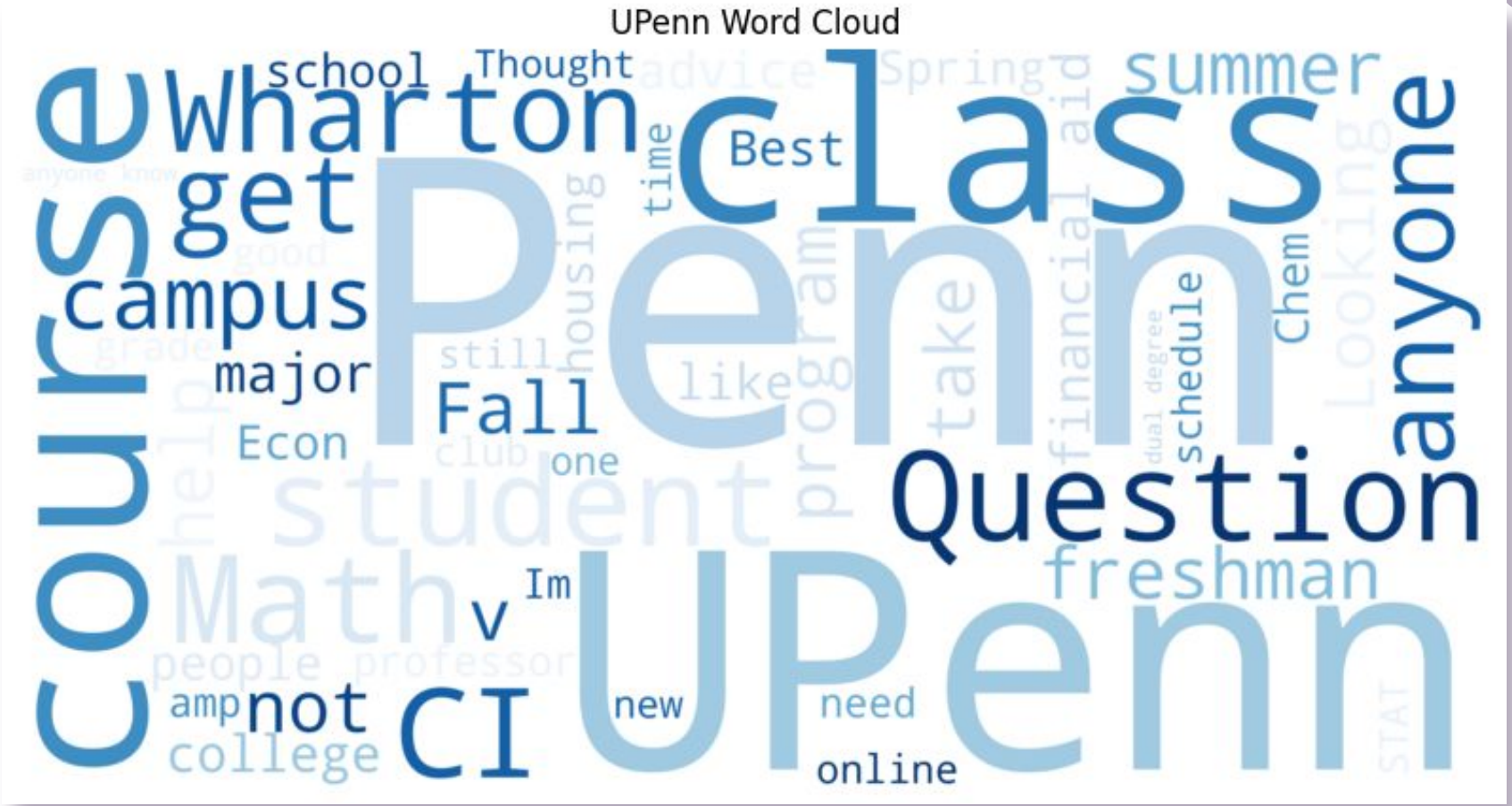
The bar chart illustrates the top 20 trigrams associated with UPenn, with a clear focus on academic programs like "dual degree Wharton" and "mse data science." Student-centric issues such as "financial aid package" and specific initiatives like "penn reading project" also feature prominently, reflecting the concerns and interests within the UPenn community discourse.

2

Word Cloud

VIsualizing the Word Cloud for r/Rutgers and r/UPenn


Rutgers Word Cloud
Rutger
Question
class
anyone
professor
campus
College Ave
anyone know
New Brunswick
student
help
course
fall
good
RU
onelike
bu
amp
semester
time
got
Math
term bill
Looking
guy
advice
housing
online
today
take
not
go
C
need
exam
financial aid
people
get
going
Anyone else
summer
Spring
grade
Calc
final
work
Im


UPenn Word Cloud
course
Wharton
school
Thought
class
Best
summer
get
Penn
anyone
campus
major
Fall
Econ
Question
schedule
Chem
student
Math
Im
v
UPenn
freshman
CI
not
amp
college
new
need
online

3

Word2Vec

Diving deep into the titles

Most similar for r/UPenn and r/Rutgers

```
1  model.wv.most_similar('rutgers', topn=10)  # get other similar words
```

```
[('penn', 0.786194920539856),
 ('pennsylvania', 0.7065936923027039),
 ('upenn', 0.7005419731140137),
 ('university', 0.6738671064376831),
 ('president', 0.6610144376754761),
 ('report', 0.6430330872535706),
 ('faculty', 0.6281049847602844),
 ('coronavirus', 0.6253470182418823),
 ('minecraft', 0.6171616911888123),
 ('rowan', 0.6158574819564819)]
```

```
1  model.wv.most_similar('upenn', topn=10)  # get other similar words
```

```
[('penn', 0.8254845142364502),
 ('mampt', 0.7774819135665894),
 ('competitive', 0.7641792297363281),
 ('future', 0.7527706623077393),
 ('current', 0.7508677244186401),
 ('alumnus', 0.7437769770622253),
 ('internally', 0.7241896986961365),
 ('program', 0.7209224104881287),
 ('wharton', 0.7206037640571594),
 ('disadvantage', 0.7199550271034241)]
```

Most Similar

The Word2Vec model's output showcases the words most closely associated with "rutgers" and "upenn," such as "pennsylvania" and "university" for "rutgers," and "wharton" and "program" for "upenn." These results highlight the contextual landscape surrounding each university name, reflecting common topics and themes within the discourse of their respective communities.

r/UPenn & r/Rutgers

```
1   model.wv.most_similar('covid', topn=10)  # get other similar words
```

```
[('vaccine', 0.8242006301879883),
 ('result', 0.804023802280426),
 ('coronavirus', 0.794784665107727),
 ('positive', 0.7930689454078674),
 ('saliva', 0.79248046875),
 ('testing', 0.7795373797416687),
 ('pcr', 0.7591186165809631),
 ('emergency', 0.7534778118133545),
 ('tested', 0.7283697724342346),
 ('notification', 0.725772500038147)]
```

Covid

The model identifies terms closely related to "covid," such as "vaccine," "result," and "coronavirus," suggesting a strong association with health, testing, and the pandemic response within the analyzed text corpus.

Similarity & Differences

Analyzing Similarities and Differences

```
[ ]  1  # Most Similar Words
     2  print("Words most similar to 'rutgers':\n", model.wv.most_similar('rutgers', topn=10))
```

```
Words most similar to 'rutgers':
 [('penn', 0.786194920539856), ('pennsylvania', 0.7065936923027039), ('upenn', 0.7005419731140137), ('university', 0.6738671064376831), ('presi
```

```
[ ]  1  print("Words most similar to 'upenn':\n", model.wv.most_similar('upenn', topn=10))
```

```
Words most similar to 'upenn':
 [('penn', 0.8254845142364502), ('mampt', 0.7774819135665894), ('competitive', 0.7641792297363281), ('future', 0.7527706623077393), ('current',
```

```
[ ]  1  # Similarity Scores
     2  print("Similarity score between 'rutgers' and 'upenn':", model.wv.similarity('rutgers', 'upenn'))
```

```
Similarity score between 'rutgers' and 'upenn': 0.7005419
```

```
[ ]  1  print("Similarity score between 'rutgers' and 'ivy':", model.wv.similarity('rutgers', 'ivy'))
```

```
Similarity score between 'rutgers' and 'ivy': 0.6040357
```

```
[ ]  1  # Odd-One-Out Analysis
     2  print("Odd one out for ['upenn', 'ivy', 'rutgers']:", model.wv.doesnt_match(['upenn', 'ivy', 'rutgers']))
```

```
Odd one out for ['upenn', 'ivy', 'rutgers']: ivy
```

```
[ ]  1  print("Odd one out for ['rutgers', 'loan', 'job']:", model.wv.doesnt_match(['rutgers', 'loan', 'job']))
```

```
Odd one out for ['rutgers', 'loan', 'job']: rutgers
```

Explanation

Our model shows around 70% between Rutgers and Upenn and considers Ivy to be the odd one out that's prob. Because both of them have more in common in terms of students and academics discussions and not just being ivy.

However UPenn does show competitive as more similar to it.

Campus vs Study Life

Campus Life vs. Study Balance:

```
1  model.wv.most_similar(positive=['dorm', 'study'], negative=['home'], topn=10)
```

```
[('participant', 0.5092398524284363),
 ('learning', 0.4986644685268402),
 ('swap', 0.4971344470977783),
 ('seminar', 0.4758041203022003),
 ('assistant', 0.46448397636413574),
 ('tutor', 0.45459285378456116),
 ('studying', 0.4524478614330292),
 ('tutoring', 0.4449842572212219),
 ('double', 0.4393843412399292),
 ('biology', 0.4374176859855652)]
```

Explanation

The analysis indicates that dormitory life is perceived as an integral part of the academic ecosystem on campus, with terms like 'participant', 'learning', 'seminar', 'tutor', and 'biology' implying a vibrant study culture that extends beyond classrooms into living spaces, where students actively engage in educational activities and access a network of academic support.

This suggests that dorms are not only residential areas but also social and learning hubs where collaborative study and academic discussions are a norm.

Undergraduate to Graduate

Undergraduate to Graduate Transition:

```
[ ]  1  model.wv.most_similar(positive=['graduate', 'undergraduate'], negative=['professor'], topn=10)

[('undergrad', 0.8129611015319824),
 ('nursing', 0.8065541982650757),
 ('applying', 0.7740465998649597),
 ('grad', 0.7657743692398071),
 ('pharmacy', 0.7549444437026978),
 ('current', 0.741146445274353),
 ('soe', 0.7401242852210999),
 ('master', 0.7288368344306946),
 ('sea', 0.720282793045044),
 ('apply', 0.7184532284736633)]
```

Explanation

The terms 'undergrad', 'nursing', 'applying', 'grad', 'pharmacy', 'current', 'lose', 'master', 'sea', and 'apply' capture the transitional phase from undergraduate studies to graduate pursuits. The model suggests a focus on professional fields (like 'nursing' and 'pharmacy') and the application process ('applying', 'apply'), with an emphasis on the immediacy ('current') and the aspirational goal ('master') of advancing to higher education.

Academic Stress & Relief

Explanation

The terms 'swe', 'mayor', 'entrepreneurial', 'colgate', 'son', 'msu', 'worthwhile', 'adp', 'vet', and 'robot' represent a broad spectrum of stress-related topics discussed within academic communities, pointing to stressors outside of exam contexts.

The list suggests that conversations about stress relief may involve diverse strategies, including engaging with technology ('robot'), focusing on specific career paths, and navigating the personal dimensions of student life.

Social Life

Explanation

The model associates 'social' and 'university' with terms like 'president', 'penn', 'gross', 'event', 'scene', 'coronavirus', 'club'.

These terms can be seen as facets of a vibrant campus life: 'president' might relate to student government, 'gross' could refer to a popular location or a term specific to a community, while 'event', 'club', and 'scene' speak to the various social gatherings, student organizations, and the overall campus atmosphere.

'Coronavirus' appearing in this context indicates its significant impact on the social fabric of university life.

Highschool to College

Transition from High School to College:

```
[ ]  1  model.wv.most_similar(positive=['college', 'highschool'], negative=['job'], topn=10)

[('livinglearning', 0.6982067823410034),
 ('tomorrowworld', 0.6740970611572266),
 ('regular', 0.6600842475891113),
 ('bergen', 0.6512401700019836),
 ('colloquium', 0.645980715751648),
 ('miracle', 0.6221482753753662),
 ('easton', 0.6211614012718201),
 ('church', 0.6209285259246826),
 ('cord', 0.6105048656463623),
 ('blvd', 0.6065792441368103)]
```

Explanation

The model highlights the transition from high school to college as a blend of new academic endeavors ('livinglearning', 'colloquium'), ('tomorrowworld'), and the adaptation to a new social and geographical landscape ('bergen', 'easton', 'blvd').

Easton assumed to be easton ave at Rutgers.

Career vs Academic

Career Focus vs. Academic Focus:

```
1  model.wv.most_similar(positive=['career', 'education'], negative=['fun'], topn=10)
```

```
[('healthcare', 0.7187228202819824),
 ('information', 0.7116981148719788),
 ('master', 0.6934827566146851),
 ('certificate', 0.6678239703178406),
 ('phd', 0.664107620716095),
 ('tech', 0.6639496088027954),
 ('leadership', 0.6598132252693176),
 ('ahmed', 0.6398521065711975),
 ('concentration', 0.63780677318573),
 ('medicine', 0.6344700455665588)]
```

Explanation

The model contrasts career-oriented discourse against educational content devoid of leisure, highlighting terms like 'healthcare', 'information', 'master', 'certificate', 'phd', 'tech', 'leadership', 'ahmed', 'concentration', and 'medicine'.

This implies a focus on professional qualifications, sectors, and personal advancement within academic conversations, emphasizing preparation for specific fields and leadership roles rather than general education or entertainment.

Online Learning

Impact of Online Learning:

```
[ ]  1  model.wv.most_similar(positive=['online', 'classroom'], negative=['campus'], topn=10)
```

```
[('feigenson', 0.7976371049880981),
 ('bender', 0.7900770902633667),
 ('planet', 0.7805723547935486),
 ('miller', 0.7685720920562744),
 ('synchronous', 0.7587992548942566),
 ('kalef', 0.7567599415779114),
 ('earth', 0.756706714630127),
 ('lepre', 0.7546864151954651),
 ('myth', 0.7528048753738403),
 ('appreciation', 0.7516663074493408)]
```

Explanation

The model associates 'online' learning, in contrast to traditional 'classroom' settings, while excluding 'campus', with terms such as 'feigenson', 'bender', 'planet', 'miller', 'synchronous', 'kalef', 'earth', 'lepre', 'myth', and 'appreciation'.

This suggests that the impact of online learning extends into various realms, including the necessity for synchronous communication and perhaps a broader, more global or universal reach of education ('planet', 'earth').

Work & Study

Balancing Work and Study:

```
[ ]  1  model.wv.most_similar(positive=['work', 'study'], negative=['vacation'], topn=10)
```

```
[('labor', 0.5876917243003845),
 ('abroad', 0.53315269947052),
 ('guide', 0.513145923614502),
 ('studying', 0.4976927936077118),
 ('opportunity', 0.47777122259140015),
 ('resource', 0.44318392872810364),
 ('learning', 0.4399336576461792),
 ('research', 0.43641868233680725),
 ('done', 0.4348618984222412),
 ('relation', 0.42972394824028015)]
```

Explanation

The terms 'labor', 'abroad', 'guide', 'studying', 'opportunity', 'resource', 'learning', 'research', 'done', and 'relation' suggest a multifaceted conversation around work-study life that includes practical aspects of working ('labor', 'resource'), the pursuit of education ('studying', 'learning'), the expansion of experiences ('abroad', 'opportunity'), and the culmination of efforts ('done').

Tech in Education

Technology's Role in Education:

```
[ ]  1  model.wv.most_similar(positive=['technology', 'education'], negative=['textbook'], topn=10)

[('informatics', 0.7492423057556152),
 ('certificate', 0.7130184173583984),
 ('tech', 0.7125326991081238),
 ('analyst', 0.7040334939956665),
 ('security', 0.6996744871139526),
 ('science', 0.6922192573547363),
 ('medicine', 0.6915063261985779),
 ('development', 0.6900174617767334),
 ('entry', 0.685135006904602),
 ('global', 0.6837186217308044)]
```

Explanation

The Word2Vec model output suggests a technology-centric educational dialogue, excluding traditional 'textbook' references. Terms like 'informatics', 'certificate', 'tech', 'analytics', 'security', 'science', 'medicine', 'development', 'entry', and 'global' reflect a shift towards digital fluency in education, with a focus on fields that intersect with technology, such as data analysis, cybersecurity, and global digital infrastructures.

Extracurricular Activities

Extracurricular Activities' Impact:

```
1  model.wv.most_similar(positive=['extracurricular', 'student'], negative=['class'], topn=10)
```

```
[('alumnus', 0.7753589749336243),
 ('philomathean', 0.735614538192749),
 ('rocket', 0.724502682685852),
 ('former', 0.7229083180427551),
 ('alliance', 0.7196483016014099),
 ('alum', 0.7187314629554749),
 ('rescue', 0.713726282119751),
 ('nonprofit', 0.7093974947929382),
 ('impacting', 0.7075549364089966),
 ('hup', 0.7064353227615356)]
```

Explanation

The terms 'alumnus', 'philomathean', 'rocket', 'former', 'alliance', 'alum', 'rescue', 'nonprofit', 'impacting', and 'hup' highlight a range of activities and roles that students might engage with outside the classroom. This includes involvement with alumni networks, participation in clubs or societies (like 'philomathean'), interest in projects or competitions ('rocket'), and contributions to causes ('rescue', 'nonprofit'), reflecting how extracurriculars shape the student experience and potentially impact future endeavors.

Athletics

Atheletics:

```
[ ]  # Athlete experience at Rutgers
     rutgers_athlete_analogy = model.wv.most_similar(positive=['athlete', 'rutgers'], negative=['student'], topn=10)

[ ]  # Athlete experience at UPenn
     upenn_athlete_analogy = model.wv.most_similar(positive=['athlete', 'upenn'], negative=['student'], topn=10)

     # Execute the athlete experience analogy for Rutgers
     print("Words associated with athletes at Rutgers as compared to the general student body:")
     for word, similarity in rutgers_athlete_analogy:
         print(f"{word}: {similarity:.4f}")

     # Execute the athlete experience analogy for UPenn
     print("\nWords associated with athletes at UPenn as compared to the general student body:")
     for word, similarity in upenn_athlete_analogy:
         print(f"{word}: {similarity:.4f}")

Words associated with athletes at Rutgers as compared to the general student body:
banning: 0.7180
listserv: 0.7178
actively: 0.7117
ruhungry: 0.7025
orgs: 0.6991
association: 0.6969
kelley: 0.6965
capability: 0.6959
fired: 0.6950
body: 0.6932

Words associated with athletes at UPenn as compared to the general student body:
upenns: 0.7691
stance: 0.7584
upennwharton: 0.7569
msds: 0.7557
simulation: 0.7552
kelley: 0.7542
gamma: 0.7533
population: 0.7528
founded: 0.7509
ruhungry: 0.7475
```

Explanation

At Rutgers, athlete discussions emphasize organizational involvement and active participation, with terms like 'banning' and 'listserv' indicating specific athletic-related issues and communications.

UPenn athlete conversations align more with academic integration and institutional policy, with references to prestigious programs like 'upennwharton' and broader concepts such as 'population'.

Rutgers vs UPenn Analogy

Rutgers vs. UPenn Experience Analogy:

```
[ ]  1  # Rutgers experience versus UPenn
     2  rutgers_upenn_analogy = model.wv.most_similar(positive=['rutgers', 'upenn'], negative=['student'], topn=10)
```

```
     1  # Execute the Rutgers vs. UPenn experience analogy
     2  print("\nWords that distinguish the Rutgers experience from the UPenn experience:")
     3  for word, similarity in rutgers_upenn_analogy:
     4      print(f"{word}: {similarity:.4f}")
     5
```

```
Words that distinguish the Rutgers experience from the UPenn experience:
penn: 0.7255
pennsylvania: 0.5822
server: 0.5807
listserv: 0.5620
kelley: 0.5497
fellow: 0.5428
minecraft: 0.5399
created: 0.5255
upenns: 0.5253
competitive: 0.5240
```

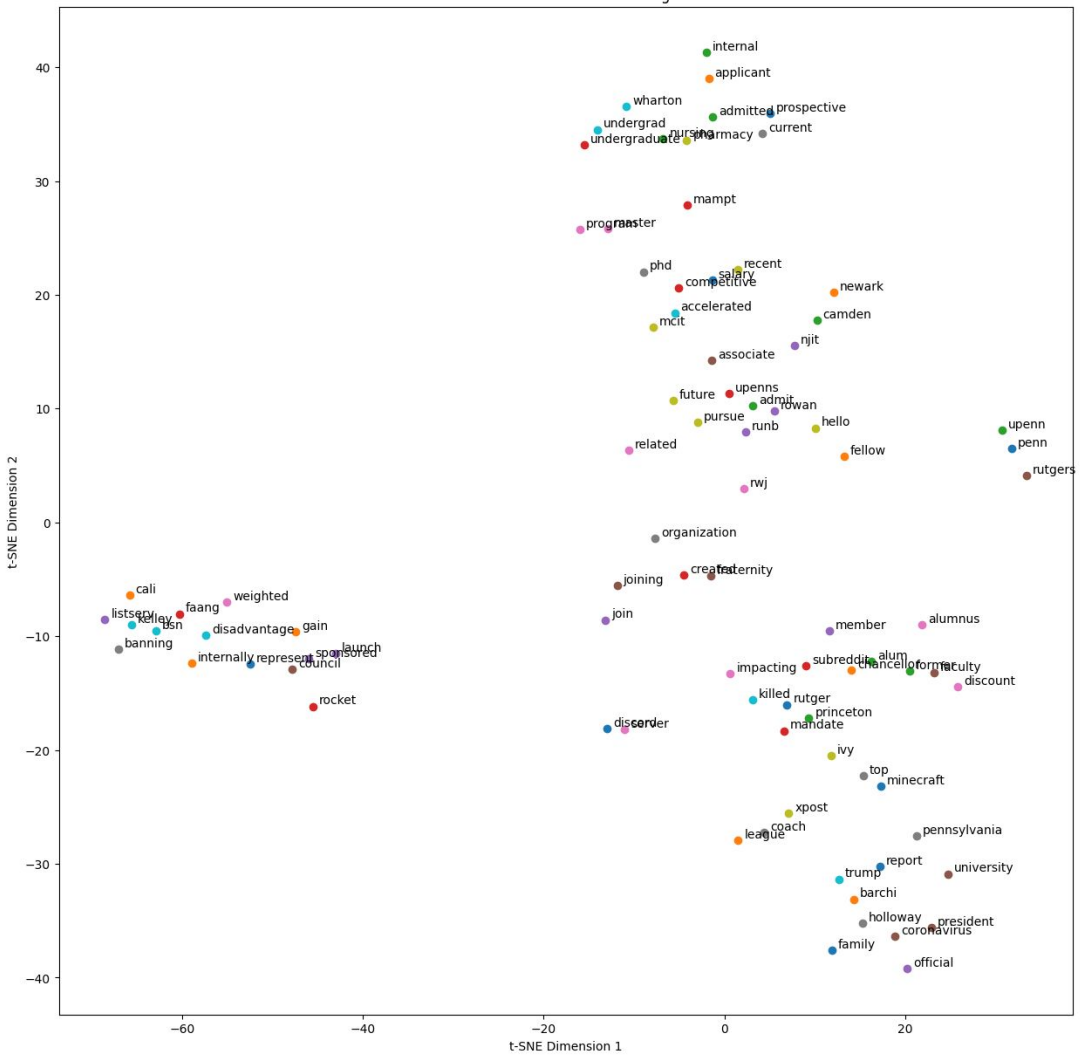
Explanation

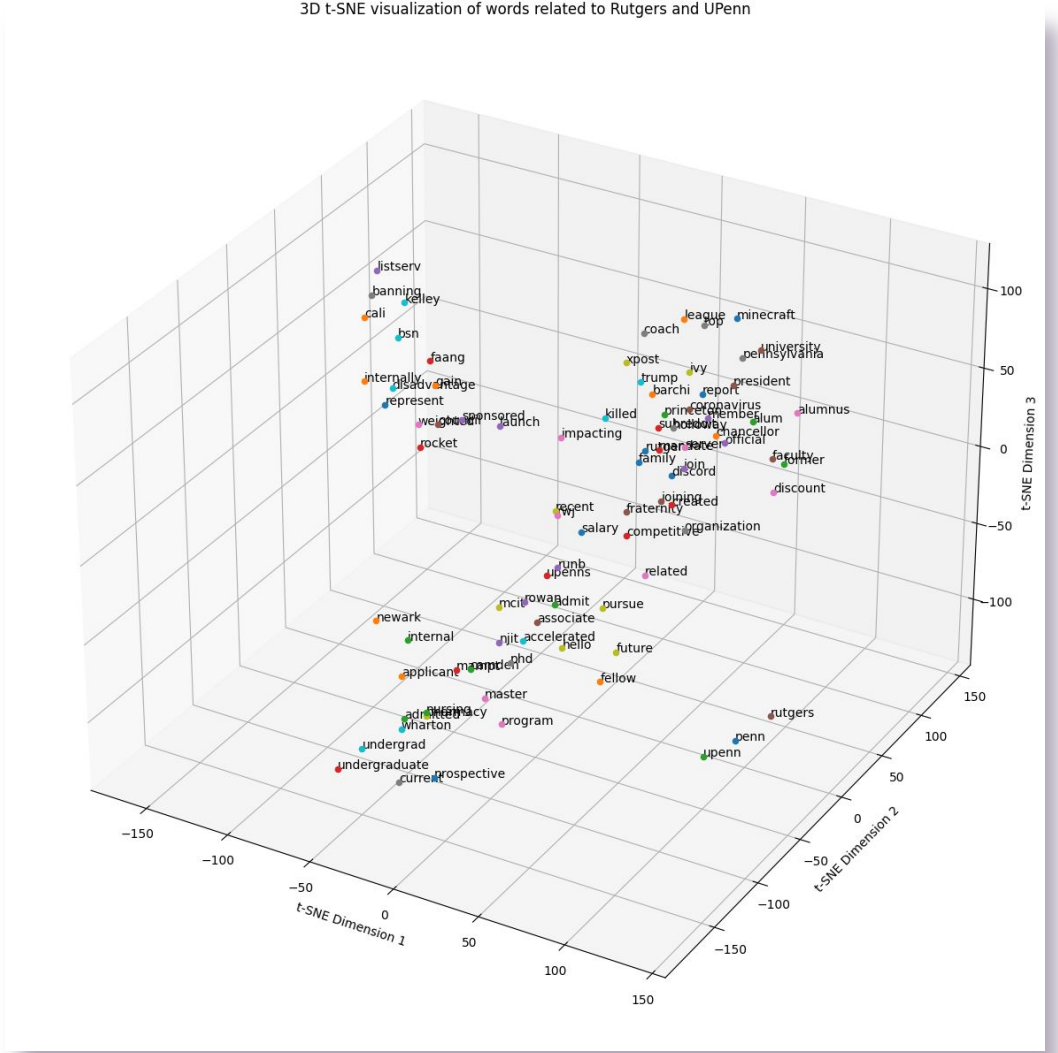
The Word2Vec analysis differentiates the Rutgers experience from UPenn's by highlighting terms like 'penn', 'pennsylvania', 'server', 'listserv', 'kelley', 'fellow', 'minecraft', 'created', 'upenns', and 'competitive'. This suggests Rutgers discussions may emphasize state and regional identity ('pennsylvania'), digital communication platforms ('server', 'listserv'), community and academic recognition ('fellow', 'kelley'), and aspects of creativity and competition within the student experience.

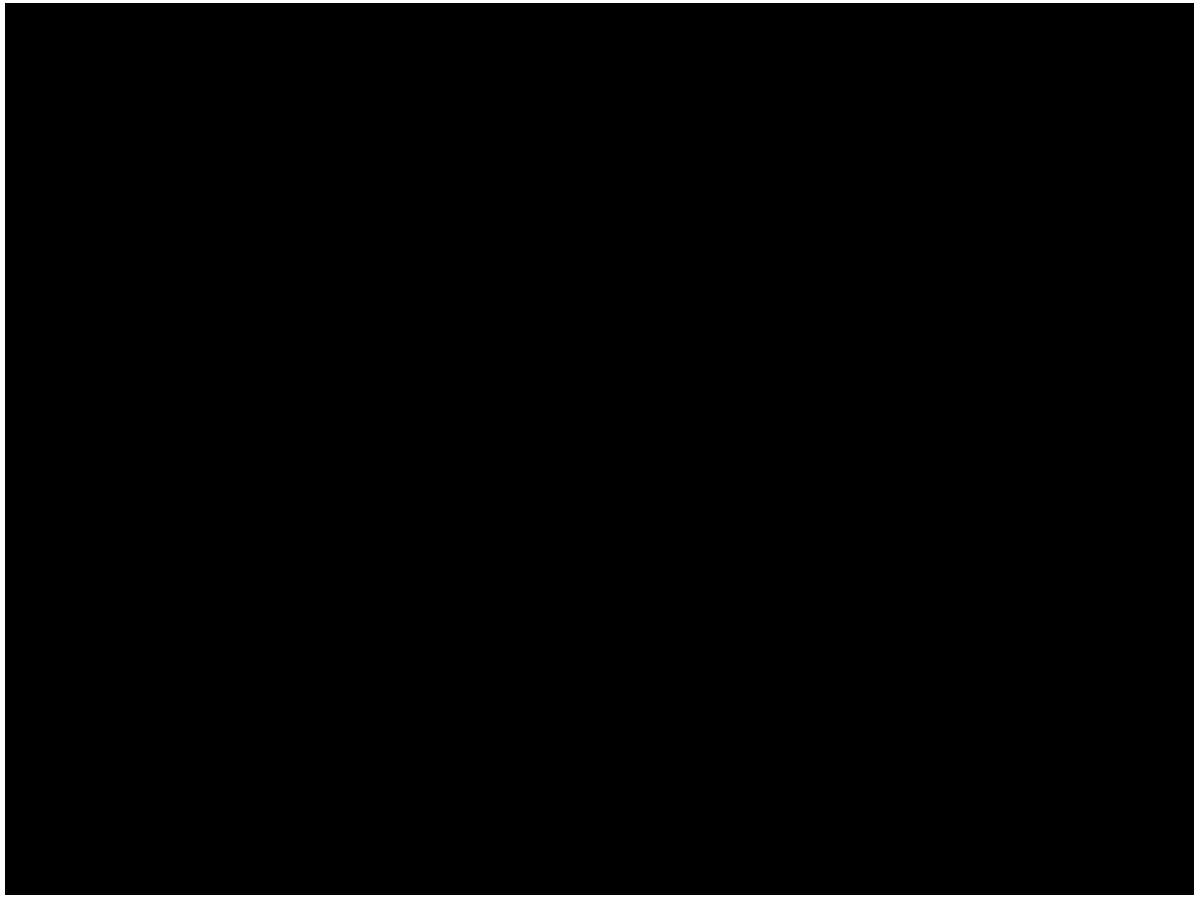
4

Visualizing t-SNE

Visualizing how the topics group together.


t-SNE visualization of words related to Rutgers and UPenn
t-SNE Dimension 1
t-SNE Dimension 2
internal
applicant
wharton
admitted
prospective
undergrad
undergraduate
nursing
pharmacy
current
mampt
phd
recent
salary
competitive
newark
accelerated
mcit
camden
njit
associate
future
upenns
admit
rowan
pursue
runb
hello
related
fellow
upenn
penn
rutgers
rwj
organization
joining
join
member
alumnus
cali
weighted
faang
listserv
kelley
usn
disadvantage
gain
banning
internally
council
launch
rocket
impacting
subreddit
alum
chancellor
former
faculty
discount
killed
rutger
princeton
mandate
ivy
top
minecraft
xpost
coach
league
pennsylvania
report
trump
university
barchi
holloway
president
coronavirus
family
official


3D t-SNE visualization of words related to Rutgers and UPenn
t-SNE Dimension 1
t-SNE Dimension 2
t-SNE Dimension 3
listserv
banning
kelley
cali
bsn
faang
internally
represent
rocket
impacting
killed
coach
league
minecraft
university
pennsylvania
xpost
ivy
trump
president
barchi
report
coronavirus
alumnus
alum
chancellor
faculty
former
discord
discount
recent
fraternity
salary
competitive
organization
related
upenns
admit
pursue
associate
newark
internal
njit
accelerated
hello
future
phd
applicant
fellow
master
program
wharton
undergrad
undergraduate
current
prospective
rutgers
penn
upenn

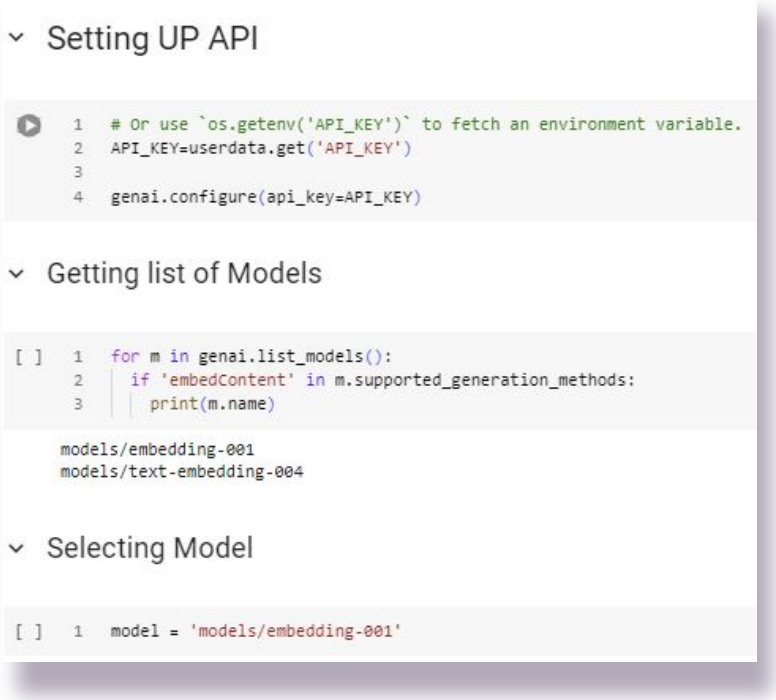


5

Gemini Model

Building an Experimental Model using Gemini API for creating embeddings

Setting Up the API



- Our Dataset has been loaded initially
- We will setup Gemini API and load the available embeddings that we have
- We then select the model/embeddings-001

Cleaning Up the Dataset

```python
import pandas as pd
import re
import string
import emoji
import nltk

# Ensure necessary NLTK resources are available
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')

# Data Cleaning and Preprocessing
def clean_text(text):
    text = emoji.replace_emoji(text, replace='')
    text = re.sub(r'https?://\S+|www\.\S+', '', text)
    text = re.sub(r'[\t\n\r]+', ' ', text)
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = re.sub(r'\w*\d\w*', '', text)
    text = re.sub(r'\s+[a-zA-Z]\s+', ' ', text)
    text = text.strip()
    # Filter out any strings that have become empty after cleaning
    return text if text else None

def process_text(text):
    if text is None:  # If text is None, skip processing
        return None
    stopwords = set(nltk.corpus.stopwords.words("english")) - set(["not"])
    lemmatizer = nltk.stem.WordNetLemmatizer()
    tokens = nltk.word_tokenize(text)
    processed_text = [lemmatizer.lemmatize(word) for word in tokens if word.lower() not in stopwords]
    # Create a combined string and filter out any that would be empty
    processed = " ".join(processed_text)
    return processed if processed else None

# Apply cleaning and processing to the dataset
filtered_data['cleaned_title'] = filtered_data['title'].apply(clean_text)
filtered_data['processed_title'] = filtered_data['cleaned_title'].apply(process_text)

# Drop any rows where 'processed_title' is None
filtered_data = filtered_data.dropna(subset=['processed_title'])

print("Data after cleaning and processing:", filtered_data.head())
```

- Our Dataset has been loaded initially
- We then follow the same process of cleaning the text and processing it and creating a new df of the clean data

Cleaning Up the Dataset

```python
import pandas as pd
import re
import string
import emoji
import nltk

# Ensure necessary NLTK resources are available
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')

# Data Cleaning and Preprocessing
def clean_text(text):
    text = emoji.replace_emoji(text, replace='')
    text = re.sub(r'https?://\S+|www\.\S+', '', text)
    text = re.sub(r'[\t\n\r]+', ' ', text)
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = re.sub(r'\w*\d\w*', '', text)
    text = re.sub(r'\s+[a-zA-Z]\s+', ' ', text)
    text = text.strip()
    # Filter out any strings that have become empty after cleaning
    return text if text else None

def process_text(text):
    if text is None:  # If text is None, skip processing
        return None
    stopwords = set(nltk.corpus.stopwords.words("english")) - set(["not"])
    lemmatizer = nltk.stem.WordNetLemmatizer()
    tokens = nltk.word_tokenize(text)
    processed_text = [lemmatizer.lemmatize(word) for word in tokens if word.lower() not in stopwords]
    # Create a combined string and filter out any that would be empty
    processed = " ".join(processed_text)
    return processed if processed else None

# Apply cleaning and processing to the dataset
filtered_data['cleaned_title'] = filtered_data['title'].apply(clean_text)
filtered_data['processed_title'] = filtered_data['cleaned_title'].apply(process_text)

# Drop any rows where 'processed_title' is None
filtered_data = filtered_data.dropna(subset=['processed_title'])

print("Data after cleaning and processing:", filtered_data.head())
```

- Our Dataset has been loaded initially
- We then follow the same process of cleaning the text and processing it and creating a new df of the clean data

Random Sampling

```
[ ]  1  # Random sampling
     2  sampled_data = df.groupby('subreddit').sample(n=2000, random_state=1)
```

```
     1  sampled_data.value_counts("processed_title")
```

```
processed_title
CIS                          7
Housing                      4
summer                       3
Friends                      3
V                            3
                            ..
MAE Technical Electives      1
MATH Help                    1
MATH Ideas Mathematics       1
MATH MATH                    1
…                            1
Name: count, Length: 3943, dtype: int64
```

- Our Dataset has a lot of rows of titles but to deal with the computational issues I have proceeded to downsample it to 2000 from each subreddit

Random Sampling

```
[ ]  1  # Random sampling
     2  sampled_data = df.groupby('subreddit').sample(n=2000, random_state=1)
```

```
     1  sampled_data.value_counts("processed_title")
```

```
processed_title
CIS                          7
Housing                      4
summer                       3
Friends                      3
V                            3
                            ..
MAE Technical Electives      1
MATH Help                    1
MATH Ideas Mathematics       1
MATH MATH                    1
…                            1
Name: count, Length: 3943, dtype: int64
```

- Our Dataset has a lot of rows of titles but to deal with the computational issues I have proceeded to downsample it to 2000 from each subreddit

Encoding and Splitting the Data

```python
import pandas as pd
from sklearn.model_selection import train_test_split

def sample_and_encode_data(df, num_samples=None):
    # Sample data if num_samples is specified
    if num_samples:
        df = df.groupby('subreddit', as_index=False).apply(lambda x: x.sample(num_samples, replace=True, random_state=42)).reset_index(drop=True)

    # Encode the 'color' column as categorical codes
    df['subreddit'] = df['subreddit'].astype('category')
    df['Encoded Label'] = df['subreddit'].cat.codes

    return df

# Assuming 'df' is your full DataFrame
# Splitting the data into train and test sets
train_size = 0.70
df_train, df_test = train_test_split(df, stratify=df['subreddit'], train_size=train_size, test_size=0.30, random_state=42)

# Optionally balance classes in training data
BALANCE_SAMPLES = 1000  # We will be downsampeling our dataset
df_train = sample_and_encode_data(df_train, BALANCE_SAMPLES)
df_test = sample_and_encode_data(df_test)  # No balancing for test data, just encoding

# Check the distribution of classes and the head of the datasets to confirm setup
print(df_train.value_counts('subreddit'))
print(df_test.value_counts('subreddit'))
print(df_train.head())
print(df_test.head())
```

```
subreddit
Rutgers    1000
UPenn      1000
Name: count, dtype: int64
subreddit
Rutgers    10951
UPenn       3608
Name: count, dtype: int64
   subreddit                                    processed_title  Encoded Label
0   Rutgers  girl currently not oncampus housing would like...              0
1   Rutgers                 Livingston Dining Hall Guest Swipe              0
2   Rutgers                                        bus route                0
3   Rutgers  Hypothetical question allowed store gallon liq...              0
4   Rutgers           get another covid relief fund spring                  0
      subreddit                                  processed_title  \
142320     UPenn                freshman take course first semester
145166     UPenn       UPenn Masters Biotechnology v Bioengineering
106086   Rutgers                                        fun class
108742   Rutgers  ' Social Media amp Work KerrMccurry credit cou...
95253    Rutgers                                  ticket Football game

        Encoded Label
142320              1
145166              1
106086              0
108742              0
95253               0
```

- We now move with encoding the subreddit class as 0 and 1
- The training data has also been balanced with each class of data to be 100 rows of rutgers and upenn each

Encoding and Splitting the Data

```python
import pandas as pd
from sklearn.model_selection import train_test_split

def sample_and_encode_data(df, num_samples=None):
    # Sample data if num_samples is specified
    if num_samples:
        df = df.groupby('subreddit', as_index=False).apply(lambda x: x.sample(num_samples, replace=True, random_state=42)).reset_index(drop=True)

    # Encode the 'color' column as categorical codes
    df['subreddit'] = df['subreddit'].astype('category')
    df['Encoded Label'] = df['subreddit'].cat.codes

    return df

# Assuming 'df' is your full DataFrame
# Splitting the data into train and test sets
train_size = 0.70
df_train, df_test = train_test_split(df, stratify=df['subreddit'], train_size=train_size, test_size=0.30, random_state=42)

# Optionally balance classes in training data
BALANCE_SAMPLES = 1000  # We will be downsampeling our dataset
df_train = sample_and_encode_data(df_train, BALANCE_SAMPLES)
df_test = sample_and_encode_data(df_test)  # No balancing for test data, just encoding

# Check the distribution of classes and the head of the datasets to confirm setup
print(df_train.value_counts('subreddit'))
print(df_test.value_counts('subreddit'))
print(df_train.head())
print(df_test.head())
```

```
subreddit
Rutgers    1000
UPenn      1000
Name: count, dtype: int64
subreddit
Rutgers    10951
UPenn       3608
Name: count, dtype: int64
   subreddit                                    processed_title  Encoded Label
0   Rutgers  girl currently not oncampus housing would like...              0
1   Rutgers                 Livingston Dining Hall Guest Swipe              0
2   Rutgers                                        bus route                0
3   Rutgers  Hypothetical question allowed store gallon liq...              0
4   Rutgers           get another covid relief fund spring                  0
        subreddit                                processed_title  \
142320      UPenn                freshman take course first semester
145166      UPenn       UPenn Masters Biotechnology v Bioengineering
106086    Rutgers                                       fun class
108742    Rutgers  ' Social Media amp Work KerrMccurry credit cou...
95253     Rutgers                                ticket football game

        Encoded Label
142320              1
145166              1
106086              0
108742              0
95253               0
```

- We now move with encoding the subreddit class as 0 and 1
- {'UPenn': 1, 'Rutgers': 0}
- The training data has also been balanced with each class of data to be 100 rows of rutgers and upenn each

Generating Embeddings

Generating Embeddings

```python
from tqdm.auto import tqdm
tqdm.pandas()

from google.api_core import retry

def make_embed_text_fn(model):

  @retry.Retry(timeout=300.0)
  def embed_fn(text: str) -> list[float]:
    # Set the task_type to CLASSIFICATION.
    embedding = genai.embed_content(model=model,
                                    content=text,
                                    task_type="classification")
    return embedding['embedding']

  return embed_fn

def create_embeddings(model, df):
  df['Embeddings'] = df['processed_title'].progress_apply(make_embed_text_fn(model))
  return df
```

- Now we make the API calls and pass each title to generate vector embeddings for them

Train Embeddings Eg:

```
[ ]  1  df_train
```

	Unnamed: 0	subreddit	processed_title	Encoded Label	Embeddings
0	0	Rutgers	girl currently not oncampus housing would like...	0	[0.017395863, -0.03309162, -0.026439212, -0.07...
1	1	Rutgers	Livingston Dining Hall Guest Swipe	0	[0.04196943, -0.011381745, -0.057876058, -0.05...
2	2	Rutgers	bus route	0	[0.016273398, -0.030588085, -0.05447116, -0.04...
3	3	Rutgers	Hypothetical question allowed store gallon liq...	0	[-0.013889403, -0.06625572, -0.052370287, -0.0...
4	4	Rutgers	get another covid relief fund spring	0	[0.003015446, -0.0065499432, -0.0581211, -0.04...
...	...	...	...	...	...
1995	1995	UPenn	schedule vaccine schedule one dose	1	[0.004068121, -0.06631477, -0.033852454, -0.04...
1996	1996	UPenn	everyone get preorientation program acceptance...	1	[0.021544827, -0.039928205, -0.019442463, -0.0...
1997	1997	UPenn	prof better	1	[0.013047099, -0.016865127, 0.027865369, -0.02...
1998	1998	UPenn	Penn building still closed	1	[0.03347769, -0.01859094, -0.026078595, -0.057...
1999	1999	UPenn	Admitted master program bigger dream admit PhD...	1	[-0.018892385, -0.023043964, -0.056339025, -0....

2000 rows × 5 columns

Building simple Classifier

#Lets build a simple classification model

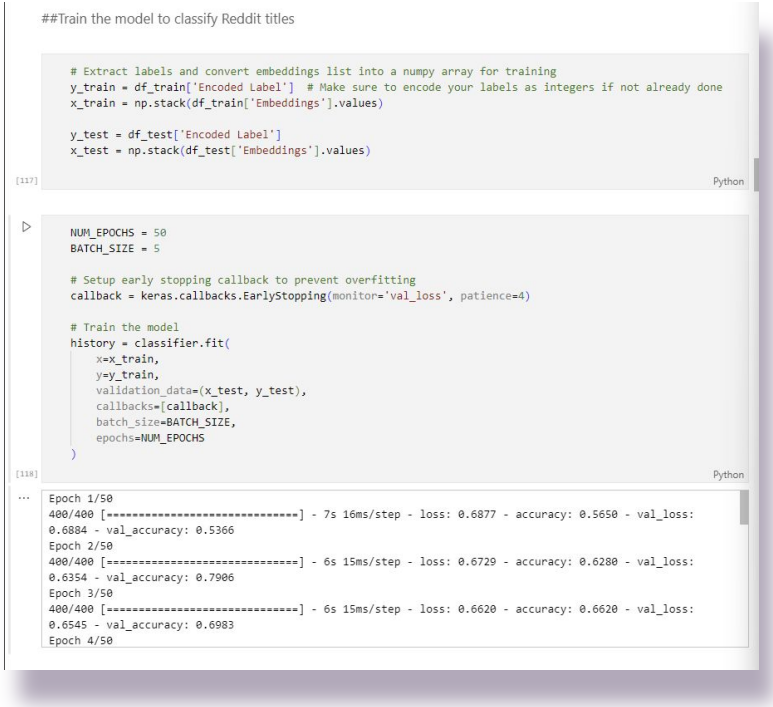
```
import ast
df_train['Embeddings'] = df_train['Embeddings'].apply(ast.literal_eval)
df_test['Embeddings'] = df_test['Embeddings'].apply(ast.literal_eval)
```

```
from keras import Model, layers, Input

def build_single_neuron_model(input_size):
    input_layer = Input(shape=(input_size,))
    output_layer = layers.Dense(1, activation='sigmoid')(input_layer)
    return Model(inputs=input_layer, outputs=output_layer)

# Assuming num_classes is 1 because it's a binary classification with a single neuron
input_size = 768  # Ensure this matches the size of your embeddings
classifier = build_single_neuron_model(input_size)

# Compile the model
classifier.compile(loss='binary_crossentropy',  # Use binary_crossentropy for binary classification
                   optimizer='adam',
                   metrics=['accuracy'])

# Display the model structure
classifier.summary()
```

```
Model: "model_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_7 (InputLayer)        [(None, 768)]             0

 dense_10 (Dense)            (None, 1)                 769

=================================================================
Total params: 769 (3.00 KB)
```

- Now our model is a basic model with a sigmoid activation and is a binary classifier
- The Model also has a single layer and uses the adam optimizer

Training the Model



##Train the model to classify Reddit titles

```
# Extract labels and convert embeddings list into a numpy array for training
y_train = df_train['Encoded Label']  # Make sure to encode your labels as integers if not already done
x_train = np.stack(df_train['Embeddings'].values)

y_test = df_test['Encoded Label']
x_test = np.stack(df_test['Embeddings'].values)
```

[117] Python

```
NUM_EPOCHS = 50
BATCH_SIZE = 5

# Setup early stopping callback to prevent overfitting
callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=4)

# Train the model
history = classifier.fit(
    x=x_train,
    y=y_train,
    validation_data=(x_test, y_test),
    callbacks=[callback],
    batch_size=BATCH_SIZE,
    epochs=NUM_EPOCHS
)
```

[118] Python

```
Epoch 1/50
400/400 [==============================] - 7s 16ms/step - loss: 0.6877 - accuracy: 0.5650 - val_loss:
0.6884 - val_accuracy: 0.5366
Epoch 2/50
400/400 [==============================] - 6s 15ms/step - loss: 0.6729 - accuracy: 0.6280 - val_loss:
0.6354 - val_accuracy: 0.7906
Epoch 3/50
400/400 [==============================] - 6s 15ms/step - loss: 0.6620 - accuracy: 0.6620 - val_loss:
0.6545 - val_accuracy: 0.6983
Epoch 4/50
```

- We can start with training the model with Early stopping that monitors our val_loss and stops once the model reaches 3 same results

Training the Model

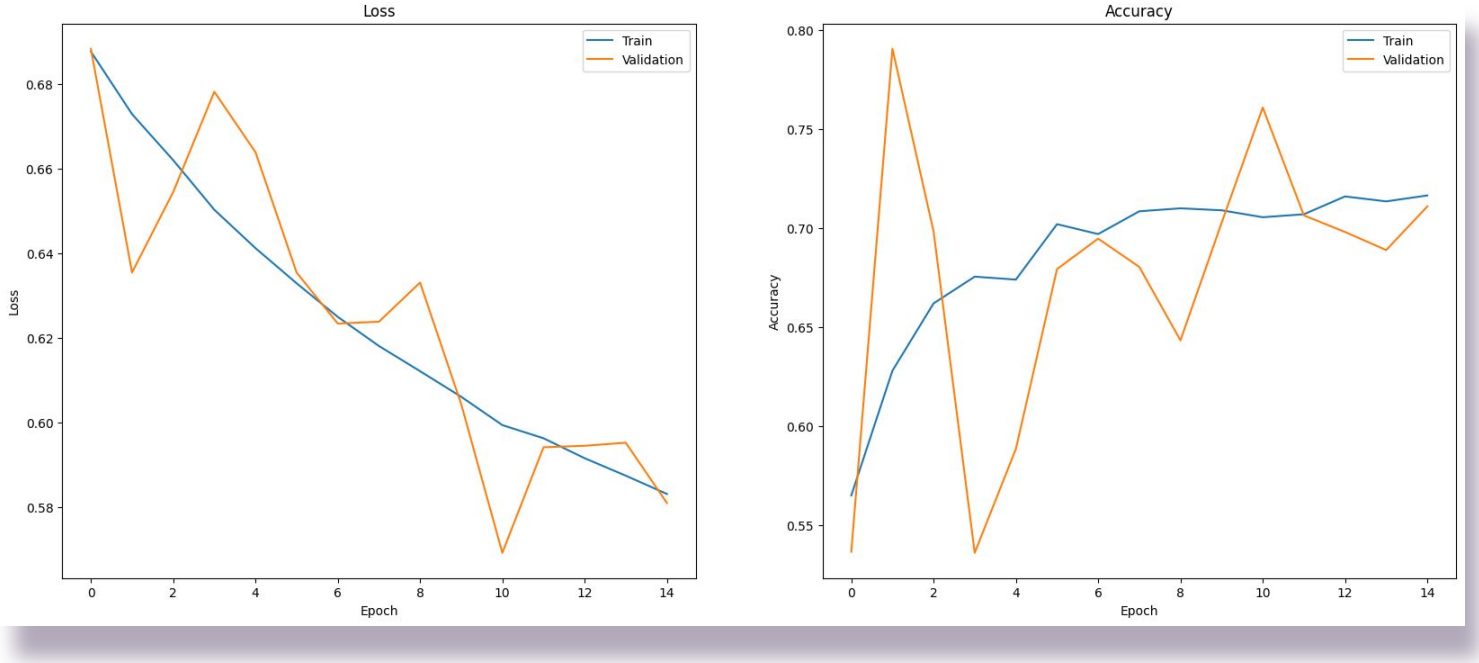
```
results = classifier.evaluate(x=x_val, y=y_val, return_dict=True)
print("Evaluation results:", results)
```

[141] Python

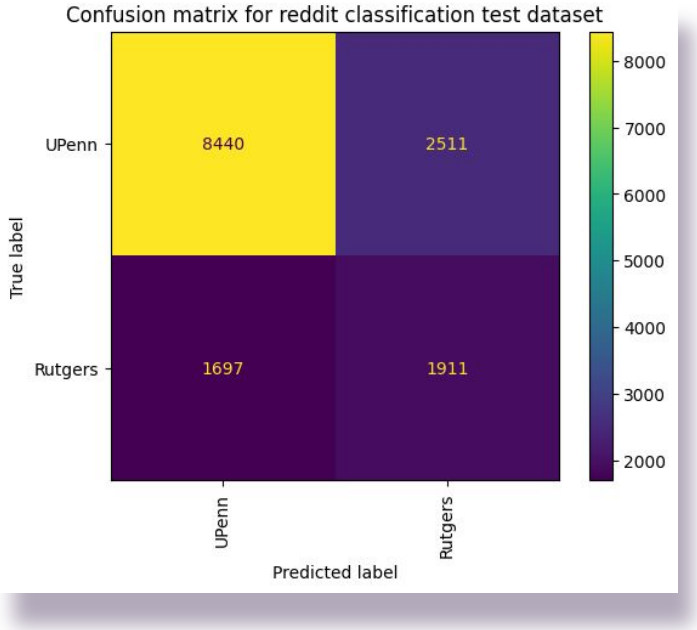
```
455/455 [==============================] - 1s 2ms/step - loss: 0.5809 - accuracy: 0.7110
Evaluation results: {'loss': 0.5809255242347717, 'accuracy': 0.7109691500663757}
```

- Lets evaluate our model we can see our accuracy is 71%
- We also noticed that the loss is around 58%

Loss & Accuracy vs Epoch

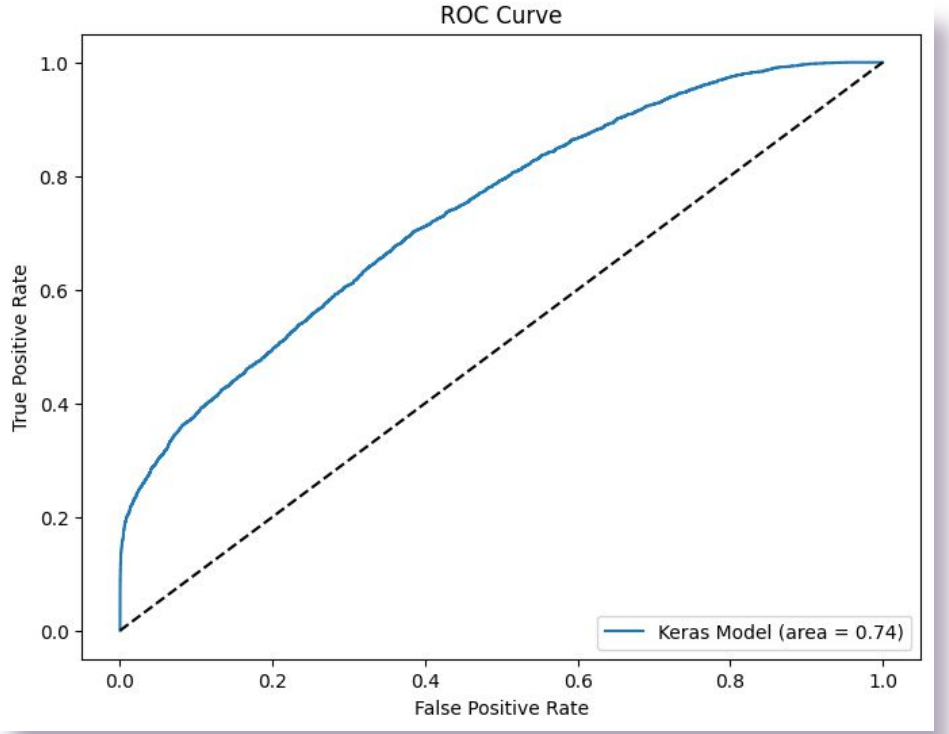


Confusion Matrix



- Despite balanced training data, the model predicted UPenn posts more accurately, with 8,440 true positives, but it confused 1,697 Rutgers posts as UPenn, likely due to overlapping features between subreddits. The test imbalance influenced the predictive outcome, showing a tendency toward UPenn classifications.

ROC Curve



Our confusion matrix was bad but the ROC curve looks good and i think the reason could be:

Class Imbalance: If there is a class imbalance, the ROC curve can be misleading. It considers the rates of true and false positives, which may not reflect the model's ability to predict the minority class well. This can make the model appear to perform better on the ROC curve than it actually does when considering precision and recall, as shown in the confusion matrix.

6

Treditional ML Model

Since the gemini model was definitely not upto the mark lets also build a traditional model

Sampling the Data

Traditional ML

```
def sample_and_encode_data(df, num_samples=None):
    # Sample data if num_samples is specified
    if num_samples:
        df = df.groupby('subreddit', as_index=False).apply(lambda x: x.sample
        (num_samples, replace=True, random_state=42)).reset_index(drop=True)

    # Encode the 'subreddit' column as categorical codes
    df['subreddit'] = df['subreddit'].astype('category')
    df['Encoded Label'] = df['subreddit'].cat.codes

    return df

# Now call this function on your DataFrame
df_encoded = sample_and_encode_data(sampled_data)
```

```
df_encoded
```

	subreddit	processed_title	Encoded Label
88842	Rutgers	Intro Environmental Science Francisco	0
117664	Rutgers	Rutgers invest money Computer Science	0
99275	Rutgers	get football ticket	0
120847	Rutgers	Schedule change help	0
100947	Rutgers	feel like ' behind prebusiness reqs first year...	0
...	...	...	...
145793	UPenn	Picking book course reserve	1
141527	UPenn	Reliable Summer Storage Options near Penn	1
141252	UPenn	Anyone Percec Orgo previously	1
150549	UPenn	Anyone going Boston fall break	1
148731	UPenn	Penn summer	1

4000 rows × 3 columns

Let's start by sampling the same dataset again and encoding it as 0 = rutgers and 1 as upenn

TF-IDF

```
[ ]  1  from sklearn.feature_extraction.text import TfidfVectorizer
     2
     3  # Vectorize the processed titles
     4  vectorizer = TfidfVectorizer(max_features=5000)  # Adjust the number of max_features as needed
     5  X = vectorizer.fit_transform(df_encoded['processed_title']).toarray()
     6
     7  # The labels are already encoded in your 'Encoded Label' column
     8  y = df_encoded['Encoded Label'].values
     9
    10  # Use the same train-test split you used for the Keras model
    11  train_size = 0.70
    12  X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, train_size=train_size, test_size=0.30, random_state=42)
```

The script uses TfidfVectorizer to convert text data into numerical TF-IDF features, considering the 5,000 most frequent terms, to prepare for text classification.

Labels for prediction are extracted from the 'Encoded Label' column, and the dataset is split into a 70:30 ratio for training and testing, maintaining class proportionality.

The specified random_state ensures consistent dataset splits across different runs, which is crucial for replicable results in machine learning experiments.

Setting up the Models

```python
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.neighbors import KNeighborsClassifier

# List of models
models = [
    DecisionTreeClassifier(),
    LogisticRegression(max_iter=1000),
    SVC(),
    RandomForestClassifier(),
    BernoulliNB(),
    KNeighborsClassifier()
]

model_names = ["Decision Tree", "Logistic Regression", "SVC", "Random Forest", "Naive Bayes", "K-Neighbors"]

# Initialize dictionary to store accuracies
model_accuracies = {}

# Training and Evaluating Models
for model, name in zip(models, model_names):
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)
    model_accuracies[name] = accuracy  # Store the accuracy in the dictionary
    print(f"{name} Test Accuracy: {accuracy:.2f}")
    print("Confusion Matrix:\n", confusion_matrix(y_test, predictions))
    print("Classification Report:\n", classification_report(y_test, predictions))

# Identify the best performing model
best_model_name = max(model_accuracies, key=model_accuracies.get)
best_accuracy = model_accuracies[best_model_name]
print(f"The best performing model is: {best_model_name} with an accuracy of {best_accuracy:.2f}")

# Select the best model
best_model = models[model_names.index(best_model_name)]
```

We now trains and evaluate multiple machine learning models, including Decision Tree, Logistic Regression, Support Vector Classifier (SVC), Random Forest, Naive Bayes, and K-Neighbors Classifier, using scikit-learn.

Each model's performance is assessed using a confusion matrix and classification report for precision, recall, and F1 scores; accuracy scores are stored for comparison.

We then save the best model

Decision Tree

```
Decision Tree Test Accuracy: 0.68
Confusion Matrix:
 [[445 155]
 [234 366]]
Classification Report:
              precision    recall  f1-score   support

           0       0.66      0.74      0.70       600
           1       0.70      0.61      0.65       600

    accuracy                           0.68      1200
   macro avg       0.68      0.68      0.67      1200
weighted avg       0.68      0.68      0.67      1200
```

Logistic Regression

```
Logistic Regression Test Accuracy: 0.71
Confusion Matrix:
 [[420 180]
 [162 438]]
Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.70      0.71       600
           1       0.71      0.73      0.72       600

    accuracy                           0.71      1200
   macro avg       0.72      0.71      0.71      1200
weighted avg       0.72      0.71      0.71      1200
```

SVC

```
SVC Test Accuracy: 0.72
Confusion Matrix:
 [[439 161]
 [170 430]]
Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.73      0.73       600
           1       0.73      0.72      0.72       600

    accuracy                           0.72      1200
   macro avg       0.72      0.72      0.72      1200
weighted avg       0.72      0.72      0.72      1200
```

Random Forests

```
Random Forest Test Accuracy: 0.70
Confusion Matrix:
 [[368 232]
 [130 470]]
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.61      0.67       600
           1       0.67      0.78      0.72       600

    accuracy                           0.70      1200
   macro avg       0.70      0.70      0.70      1200
weighted avg       0.70      0.70      0.70      1200
```

Naive Bayes

```
Naive Bayes Test Accuracy: 0.73
Confusion Matrix:
 [[399 201]
 [119 481]]
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.67      0.71       600
           1       0.71      0.80      0.75       600

    accuracy                           0.73      1200
   macro avg       0.74      0.73      0.73      1200
weighted avg       0.74      0.73      0.73      1200
```

K-Neighbors

K-Neighbors Test Accuracy: 0.68
Confusion Matrix:
 [[428 172]
 [210 390]]
Classification Report:

	precision	recall	f1-score	support
0	0.67	0.71	0.69	600
1	0.69	0.65	0.67	600
accuracy			0.68	1200
macro avg	0.68	0.68	0.68	1200
weighted avg	0.68	0.68	0.68	1200

Best Model

```
The best performing model is: Naive Bayes with an accuracy of 0.73
```

Grid Search on Best Model

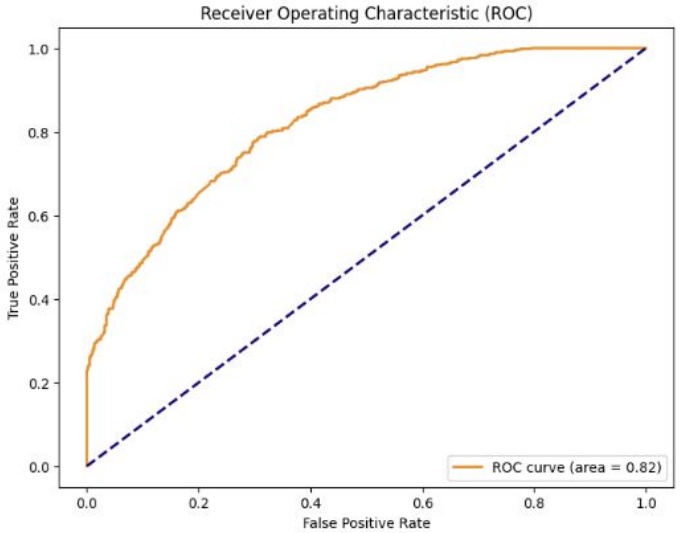
```
Fitting 5 folds for each of 3 candidates, totalling 15 fits
Best Grid Search Parameters: {'alpha': 1.0}
Best Grid Search Score: 0.707857142857143
Grid Search Test Accuracy: 0.7333333333333333

Confusion Matrix:
 [[399 201]
 [119 481]]

Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.67      0.71       600
           1       0.71      0.80      0.75       600

    accuracy                           0.73      1200
   macro avg       0.74      0.73      0.73      1200
weighted avg       0.74      0.73      0.73      1200
```

Model ROC

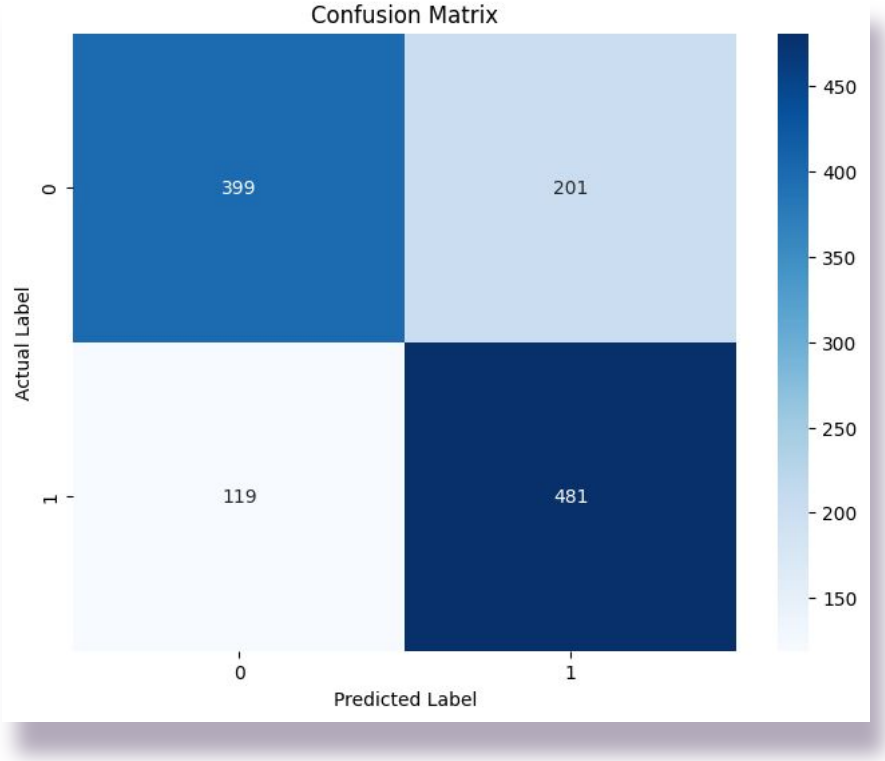
Evaluate Model Performance

```
from sklearn.metrics import roc_curve, auc

# Predict probabilities for the test data
y_scores = best_model.predict_proba(X_test)[:, 1]

# Calculate ROC metrics
fpr, tpr, thresholds = roc_curve(y_test, y_scores)
roc_auc = auc(fpr, tpr)
```

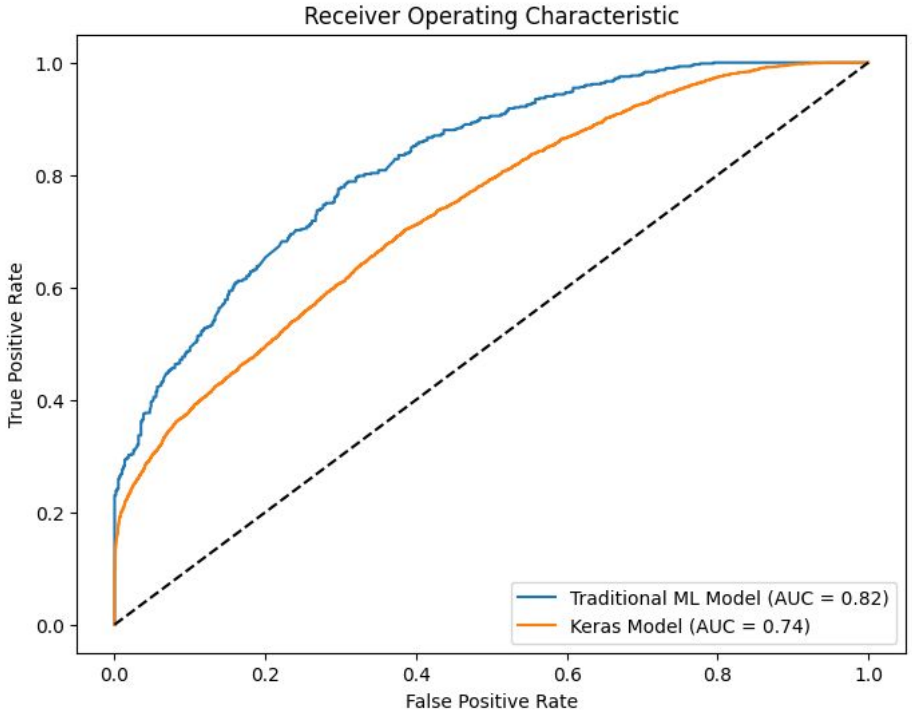
```
# Start plotting
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")
plt.show()
```



Confusion Matrix



Keras Model VS NB



5

Summary

Insights and recommendations.

Insights

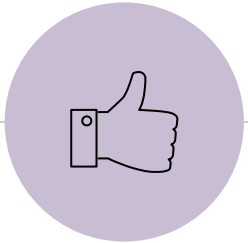
- We displayed the use of various NLP techniques starting with basic EDA and then slowly building up to a Word2Vec Model
- We used this to get insights to the rutgers and upenn subreddits especially how active the students are during the semester and their common topics discussed

Insights

- We then compared two models by also using Google's Gemini API to create embeddings and used it in the Neural Network Model and plotted various evaluation curves for it
- Finally we went on to build a ML model and use a grid search for the ideal parameters for the best model
- We finally compared the models and saw the NB model performance be better

Recommendations

- Go with Naive Bayes Model
- If you want to build a better Neural Network Model you might need to add more factors to it and tweek the learning rate and other parameters
- On a larger scale considering a laarge dataset and spending more time training might be better
- A BERT model finetuned will definitely land better results (still training on all the data) Especially to get better insights about the attention mechanism



Thanks!

Any questions?