Natural Language Processing

Group 14:

Samuel Atkins,

Marie Ishimwe,

Rohith Sothilingam,

Dichia Zhang,

Yifan Zhang

Presentation Layout

1. Text Pre-Processing and Word Embedding Techniques
2. Using the Twitter API to Collect Tweet Data
3. Extracting data from RSS feeds
4. Text pre-processing applied to news articles

Text-Processing: Tokenization

“Don't try to fool us with fake reviews.: It's glaringly obvious that all of the glowing reviews have been written by the same person, …”

```
nltk.tokenize.TreebankWordTokenizer()
```

['Don', "'", 't', 'try', 'to', 'fool', 'us', 'with', 'fake', 'reviews', '.:', 'It', "'", 's', 'glaringly', 'obvious', 'that', 'all', 'of', 'the', 'glowing', 'reviews', 'have', 'been', 'written', 'by', 'the', 'same', 'person’, …]

Text Processing: Stemming & Lemmatizing

```
Original:
["persons", "feet", "foot", "apples", "trying", "fries", "geese", "women"]


Stemming:
["person", "feet", "feet", "appl", "tri", "fri", "gees", "women"]

Lemmatization:
["person", "foot", "foot", "apple", "trying", "fry", "goose", "woman"]
```

Word Embeddings: One-Hot Encoding

Goal:

Transform each word in vocabulary into a real-numbered vector that typical models can understand

One-Hot Encoding Problems:

1. Dimensionality increases with number of words in vocabulary

2. Token similarity is not captured

```
{'I', 'apple', ',', 'bought', 'pear',
'banana', 'orange', '.', 'and’}

‘I’ = [1, 0, 0, 0, 0, 0, 0, 0, 0]
‘apple’ = [0, 1, 0, 0, 0, 0, 0, 0, 0]
‘,’ = [0, 0, 1, 0, 0, 0, 0, 0, 0]
```

Word Embeddings: GloVe Vectors

What are GloVe Embeddings?

GloVe embeddings are pre-trained embeddings that translate English words into GloVe vectors

- ➢ GloVe vectors can be of a chosen pre-determined dimension
- ➢ Token similarity is captured through cosine similarity and Euclidean distance

```
glove = torchtext.vocab.GloVe(name="6B",
            dim=50)
print(glove["daughter"])
```

GloVe Embedding of “daughter”:

```
[ 0.3765, 1.2426, -0.3974, -0.5318, 1.1870,
1.5091, -0.8417, 0.6788, -0.2581, -0.4798,
0.1782, 0.7467, -0.1347, -0.9236, 0.9562,
0.2057, -1.2239, -0.0550, 0.5618, 0.7808, -
0.0441, 1.5692, -0.0668, 0.2514, 1.0403, -
2.1412, -0.3199, -0.7717, -0.0292, 0.0471,
1.4145, -0.2327, -0.3443, 0.2270, 0.8857, -
0.2018, -0.1517, 0.3621, 0.6495, -0.6872, -
0.0682, 0.5360, -0.1529, -0.9016, 0.3896, -
0.5230, -0.3219, -2.4262, 0.3005, 0.3389]
```

Word Embeddings: Word2Vec

Train your own embedding using Word2Vec:

```
Vocabulary:
['Even', 'Mommy', 'ha', 'fun', 'with',
'this', 'one', '!’, ...]
```

```
word2vec = Word2Vec(sentences=[words], window=5
            , min_count=1, workers=4)
word2vec_vocab = word2vec.wv.vocab
print("Word2Vec Vocab:\n\n{}“
            .format(word2vec_vocab))
```

```
Word2Vec Custom Word Embeddings:
{'Even':
<gensim.models.keyedvectors.Vocab
object at 0x0000026B36F16908>,
'Mommy':
<gensim.models.keyedvectors.Vocab
object at 0x0000026B36F16A08>, 'ha':
<gensim.models.keyedvectors.Vocab
object at 0x0000026B36F16A48>,
'fun’:...}
```

Python Twitter Module Selection

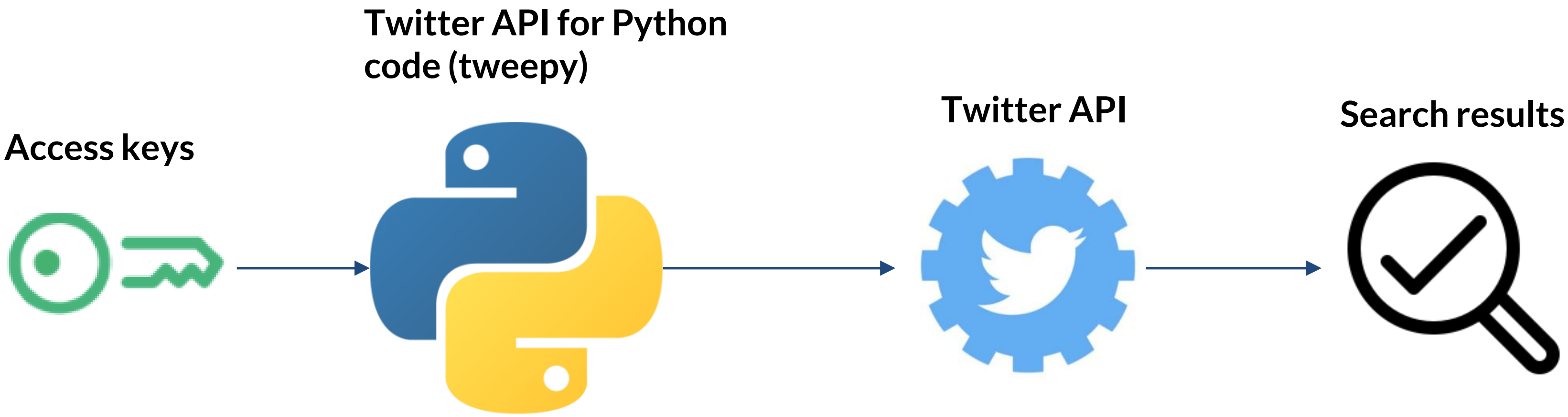
- ❑ Twitter data typically pulled using the JSON data structure
- ❑ We will select the tweepy module because it is reliable and easy to use

Python modules	Maturity
tweepy	~8.5 years
Python Twitter Tools	7 years
python-twitter	~ 5 years
twython	NA
TwitterAPI	~ 4.5 years
TwitterSearch	~ 4.5 years

Using Python's Twitter API for Tweet Analysis

- Downloading tweets
- Searching for tweets by hashtag
- Extracting metadata from tweets (retweets, user data, etc.)

API Authentication



Querying Tweets by Hashtag

```
<tweepy.cursor.ItemIterator object at 0x000002128FCDB6A0>
1323291868550078464 RT @i_ameztoy: Do you want a scary #Halloween? Here you go two months of CO evolution; Look at the tongues crossing o
ceans! 🧐

@CopernicusE…
1323291275320205312 #Wine country, fire country https://t.co/wtvGt998ON from @sfchronicle #winecountry #wildfires
1323290275586936832 RT @Alex_Bernhardt: A key reason for the increase in #wildfires is forest management – is the solution biomass? Learn
more from Stan Parton…
1323287849899225093 RT @ClimateSignals: Climate change is causing bigger, more frequent #wildfires to burn hotter and spread faster. Scie
ntists have identified…
1323287814193238016 RT @MarineGOfficial: #SavePantanal: The Pantanal is a terrestrial ecoregion of South America belonging to the prairie
and flooded savannah…
```

Extracting Metadata From Tweets

Tweet which was extracted

```
Are you a coding fanatic who wants to work with us and learn new technologies? 👨‍💻👩‍💻
Well then, we are looking just for you!

Register for our SDE Hiring Challenge right now!
https://t.co/Zg08gHhT0W

#hiring #challenge #coding #programming https://t.co/1N7gXaH9eA
```

Metadata Extraction Example


```
Out[49]: {'created_at': 'Thu May 28 06:14:48 +0000 2020',
 'id': 1265889240300257280,
 'id_str': '1265889240300257280',
 'full_text': 'Are you a coding fanatic who wants to work with us and learn new technologies? 👨\u200d💻👩\u200d💻\nWell then, we are l
ooking just for you!\n\nRegister for our SDE Hiring Challenge right now!\nhttps://t.co/Zg08gHhT0W  \n\n#hiring #challenge #coding #progra
mming https://t.co/1N7gXaH9eA',
 'truncated': False,
 'display_text_range': [0, 242],
 'entities': {'hashtags': [{'text': 'hiring', 'indices': [203, 210]},
   {'text': 'challenge', 'indices': [211, 221]},
   {'text': 'coding', 'indices': [222, 229]},
   {'text': 'programming', 'indices': [230, 242]}],
  'symbols': [],
  'user_mentions': [],
  'urls': [{'url': 'https://t.co/Zg08gHhT0W',
    'expanded_url': 'https://practice.geeksforgeeks.org/contest/hiring-challenge-sde',
    'display_url': 'practice.geeksforgeeks.org/contest/hiring…',
    'indices': [176, 199]}],
  'media': [{'id': 1265887151016812546,
    'id_str': '1265887151016812546',
    'indices': [243, 266],
    'media_url': 'http://pbs.twimg.com/media/EZFVqCoWoAILfq5.jpg',
    'media_url_https': 'https://pbs.twimg.com/media/EZFVqCoWoAILfq5.jpg',
    'url': 'https://t.co/1N7gXaH9eA',
    'display_url': 'pic.twitter.com/1N7gXaH9eA',
    'expanded_url': 'https://twitter.com/geeksforgeeks/status/1265889240300257280/photo/1',
    'type': 'photo',
    'sizes': {'medium': {'w': 1200, 'h': 1200, 'resize': 'fit'},
     'thumb': {'w': 150, 'h': 150, 'resize': 'crop'},
     'large': {'w': 1200, 'h': 1200, 'resize': 'fit'},
```

Extracting Metadata Elements

```
Out[42]: dict_keys(['created_at', 'id', 'id_str', 'full_text', 'truncated', 'display_text_range', 'entities', 'extended_entities', 'source', 'in_r
eply_to_status_id', 'in_reply_to_status_id_str', 'in_reply_to_user_id', 'in_reply_to_user_id_str', 'in_reply_to_screen_name', 'user', 'ge
o', 'coordinates', 'place', 'contributors', 'is_quote_status', 'retweet_count', 'favorite_count', 'favorited', 'retweeted', 'possibly_sen
sitive', 'possibly_sensitive_appealable', 'lang'])
```

```
The tweet was created at Thu May 28 06:14:48 +0000 2020 by the user GeeksforGeeks from India
This user has 20776 followers and 22 friends
```

Basic Text Pre-Processing on Tweets from .csv File

A tweet may contain:

- URL's https://
- Mentions @
- Hashtags #
- Emojis 😃
- Smileys ☺ :)
- Spefic words etc…

tweet-preprocessor

- Remove URL
- Remove Hashtag
- Remove Emojis
- convert Emojis to text

Then we can apply normal text preprocessing like:

- Lowercasing
- Punctuation Removal
- Replace extra white spaces
- Stop words removal
- etc.

csv file: the last 100 tweets made using the hashtag "#trump"

	text	text_embed_Preprocessor	text_after_preprocess
4	b'"I Love Covid-." - Donald Trump \\\\xba\\\\\\\\xef\\ #MAGA # #Trump # # https://t.co/'	b'"I Love Covid-." - Donald Trump \\\\xba\\\\\\\\xef\\ '	i love covid donald trump maga trump https t co
5	b'@murray_nyc @realDonaldTrump A vote for #Trump is a vote for bigotry, for #racism, and for intolerance.'	b' A vote for is a vote for bigotry, for , and for intolerance.'	murray_nyc realdonaldtrump a vote trump vote bigotry racism intolerance
6	b'RT @Dakotadamus: $, cash give giveaway: If either Presidential Candidate gets or more electoral votes (Obama had in) in #\\\'	b'RT : $, cash give giveaway: If either Presidential Candidate gets or more electoral votes (Obama had in) in \\\'	dakotadamus cash giveaway if presidential candidate gets electoral votes obama
7	b'RT @_Nico_Piro_: Una #NewYork mai vista cos\\xac: transenne, tavole di legno sulle vetrine dei negozi (non solo sulla quinta, non solo negozi d\\\'	b'RT : Una mai vista cos\\xac: transenne, tavole di legno sulle vetrine dei negozi (non solo sulla quinta, non solo negozi d\\\'	_nico_piro_ una newyork mai vista cos transenne tavole di legno sulle vetrine dei negozi non solo sulla quinta non solo negozi d
8	b'When Biden wins tonight, what is that fucking maniac Trump going to do?\n\n# #Trump #Biden\\\ https://t.co/'	b'When Biden wins tonight, what is that fucking maniac Trump going to do?\n\n \\\ '	when biden wins tonight fucking maniac trump going do n n trump biden https t co
9	b"Thru all my years of voting, I've always seen yard signs for candidates\n\nBut NOT this year\\\\\n\nI believe people ar\\\ https://t.co/"	b"Thru all my years of voting, I've always seen yard signs for candidates\n\nBut NOT this year\\\\\n\nI believe people ar\\\ "	thru years voting i ve seen yard signs candidates n nbut not year n ni believe people ar https t co

Mentions

Emojis

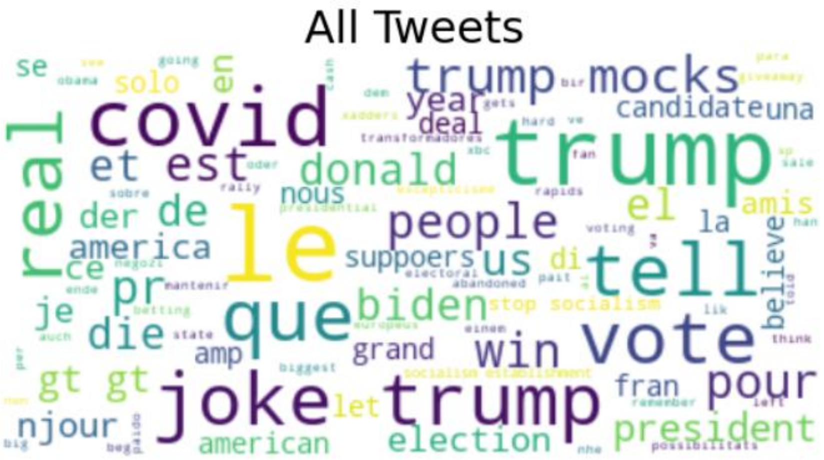
Hashtags

URL's

Basic Feature Extraction

Extract numerical features from text, namely:

- **Tokenizing** strings and giving an integer id for each possible token.
- **Counting** the occurrences of tokens in each document. (Could count #word, #URL, #Hashtag, #Emoji in each string)
- **Normalizing** and weighting with diminishing importance tokens that occur in the majority of samples



	text	word_count	Tweet_tokenized
5	a vote vote bigotry intolerance	6	[, a, vote, vote, bigotry, intolerance]
6	cash giveaway if presidential candidate gets electoral votes obama	13	[, cash, giveaway, if, presidential, candidate, gets, electoral, votes, obama]
7	una mai vista cos transenne tavole di legno sulle vetrine dei negozi non solo sulla quinta non solo negozi d	24	[, una, mai, vista, cos, transenne, tavole, di, legno, sulle, vetrine, dei, negozi, non, solo, sulla, quinta, non, solo, negozi, d]
8	when biden wins tonight fucking maniac trump going do n n	12	[, when, biden, wins, tonight, fucking, maniac, trump, going, do, n, n]

RSS Feed Data Extraction: NY Times Website

1. Get Metadata Fields:

```
# ny_feed: dictionary containing metadata (title, date, link, author, ...)
ny_feed = feedparser.parse('https://rss.nytimes.com/services/xml/rss/nyt/Arts.xml')
```

2. Request Text Data for Each Article:

```
html = rqst.urlopen(article['link'])
bs = BeautifulSoup(html,
                   features='html.parser')
targets = bs.select('.css-158dogj')
```

3. Export to .csv File:

```
ny.to_csv('./ny.csv', index=True)
```

Libraries Used:
feedparser, BeautifulSoup, request

	title	date	link	author	text
0	A Golden Team	Mon, 02 Nov 2020	https://www.nytimes.cor	Jesse Greer	How do you top “Wes
1	Johnny Depp L	Mon, 02 Nov 2020	https://www.nytimes.cor	Alex Marsha	LONDON — Johnny I
2	With New Show	Mon, 02 Nov 2020	https://www.nytimes.cor	Robin Pogre	NEW HAVEN — It wa
3	Dancing on Gra	Mon, 02 Nov 2020	https://www.nytimes.cor	Gia Kourlas	When it comes to dig
4	Dementia ‘Took	Mon, 02 Nov 2020	https://www.nytimes.cor	Sarah Bahr	Sean Connery, the ac

Advanced Text Processing: NY Times Example

Bag of Word Representation: TF & TF-IDF

1. ***Data Pre-Processing:***
   removing punctuation and uppercase

```python
def preprocessing(text):
     # Lowercase
    text=text.lower()
    # remove special characters and digits
    text=re.sub("(\\d|\\W)+"," ",text)
```

```
johnny depp loses court case against newspaper that called hi
gainst a british newspaper that called him a wife beater and
```

2. ***Article Vector Representation:***
   use TfidVectorizer() to get TF & TF-IDF representations

```python
def text_processing(text, is_tfidf):
    vectorizer = TfidfVectorizer(stop_words = 'english',
                                 use_idf = is_tfidf,
                                 max_features = 300,
                                 ngram_range = (2,3)).fit(text)
    return vectorizer
```

```python
train_text_rep = vectorizer.transform(df_train['title_text'])
test_text_rep = vectorizer.transform(df_test['title_text'])
```

3. ***Insert Representation Into Dataset***

```python
#Replace the article's text data (title and text) with their vector representations
X_train_tf = pd.concat([df_train.drop(['title', 'text', 'title_text'], axis=1),
          pd.DataFrame(data = train_text_rep.toarray(), columns = vectorizer.get_feature_names())], axis = 1)
```

	date	link	author	agatha christie	age thomas	aged robin	aghdashloo festival	aghdashloo work	albert broccoli	amber heard	...	worry dolls	year old
0	Mon, 02 Nov 2020 20:08:17 +0000	https://www.nytimes.com/2020/11/02/theater/a-p...	Jesse Green	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.447214

Word Embedding Techniques - Word2Vec

Word2Vec – Google News Text Processing Pipeline:

1. *Pre-process data*
2. *Get word embeddings for the news articles using the model*
3. *Adapt the words vector representation to be used by scikit-learn algorithms*

Pre-Processing the Data

1. Get word tokens for each article
2. Basic preprocessing including removing stopwords, punctuation
3. Remove any words that are not in the pretrained model vocabulary

```
#Get word tokens for the article
article = word_tokenize(text)
```

```
#remove stop words
article = [word for word in article if word not in stop_words]

#remove punctuation
article = [word for word in article if word.isalpha()]
```

```
#remove words that are not in the pre-trained model vocabulary
article = [word for word in article if word in model.vocab]
```

```
['johnny', 'depp', 'loses', 'court', 'case', 'newspaper', 'called', 'wife', 'beater', 'london', 'johnny', 'depp', 'monday', 'lo
st', 'court', 'case', 'british', 'newspaper', 'called', 'wife', 'beater', 'claimed', 'overwhelming', 'evidence', 'assaulted',
'actress', 'amber', 'heard', 'repeatedly', 'marriage', 'depp', 'assaults', 'included', 'heard', 'repeatedly', 'hitting', 'teari
ng', 'clumps', 'hair', 'wrote', 'andrew', 'british', 'judge', 'heard', 'case', 'ruling', 'issued', 'online', 'monday', 'dismiss
ing', 'case', 'said', 'defendants', 'shown', 'published', 'substantially', 'assaults', 'must', 'terrifying', 'judge', 'wrote',
'regarding', 'incidents', 'march', 'australia', 'heard', 'said', 'depp', 'assaulted', 'several', 'times', 'including', 'smashin
g', 'telephone', 'beside', 'face', 'accept', 'depp', 'put', 'fear', 'life', 'judge', 'wrote', 'judge', 'also', 'acknowledged',
'risk', 'heard', 'taken', 'speaking', 'violence', 'also', 'accept', 'heard', 'allegations', 'negative', 'effect', 'career', 'ac
tor', 'activist', 'said', 'women', 'rights', 'groups', 'britain', 'praised', 'decision', 'important', 'ruling', 'one', 'hope',
```

Word Embeddings

Use pre-trained model to get each article's representation. For each article:

```
#Example of the vector representation returned by the model
example_embedding = model[collection[1]]
```

A (*number of words in article x 300*) vector is returned:

```
Article size (Number of words):  626
Dimensions of returned vector representation:  (626, 300)
```

Adapt for scikit-learn Algorithms

- *For each article, average over the article words to get a 300 long vector representation*
- *Combine all articles into a number of articles by 300 array*
- *Merge it with the rest of the original dataset*

```
#To get a one dimensional vector representation of the article, average over the words
final_representation = np.mean(example_embedding, axis=0)
```

```
Final article vector representation:  (300,)
[ 0.00100749  0.02137169 -0.00101197  0.02886536 -0.02495218  0.00751302
  0.05842185 -0.05776473  0.12973578  0.06650877 -0.01028166 -0.11123077
```

```
for article in collection:
    #average over the word vectors of the curren
    X.append(np.mean(model[article], axis=0))
```

```
#Replave the title and text features with their vector representations
X = pd.concat([df.drop(['title', 'text', 'title_text'], axis=1),
          pd.DataFrame(data = X)], axis = 1)
```

	date	link	author	0	1	2	3	4	5	6	...	290
0	Mon, 02 Nov 2020 20:08:17 +0000	https://www.nytimes.com/2020/11/02/theater/a-p...	Jesse Green	0.049907	0.034578	0.006905	0.064221	-0.027185	-0.016675	0.040625	...	-0.068019
1	Mon, 02 Nov 2020 17:01:15 +0000	https://www.nytimes.com/2020/11/02/arts/johnny...	Alex Marshall	0.001007	0.021372	-0.001012	0.028865	-0.024952	0.007513	0.058422	...	-0.027191