UVA CS5001 Exploratory Text Analytics Final Project
Huilin Chang, hc5hq

Final Project-NLP applications

Huilin Chang
hc5hq@virginia.edu

I. INTRODUCTION

Natural Language Processing is everywhere even if we do not realize it. According to article [1], here are some examples of the most widely used NLP applications: Machine translation, Automatic summarization, Sentiment analysis, Text classification, Question answering, Automatic summarization, Chatbots, Market intelligence, Language modeling, Speech Recognition. From article [2], NLP application is widely applied in our daily life. For example:

1. Text classification
    a. Spam filtering, classifying email text as spam or not
    b. Language identification, classifying the language of the source text
    c. Genre classification, classifying the genre of a fictional story
2. Language modeling
    a. Generating new article headlines
    b. Generating new sentences, paragraphs or documents
    c. Generating suggested continuation of a sentence
3. Speech recognition
    a. Transcribing a speech
    b. Creating text captions for a movie or TV shows
    c. Issuing commands to the radio while driving
4. Machine translation
    a. Translating a text document from French to English
    b. Translating Spanish audio to German text
    c. Translating English text to Italian audio
5. Document Summarization
    a. Creating a heading for a document
    b. Creating an abstract.

The motivations of this report are using the skills learned from DS5001 for NLP possible applications: (1) sentimental analysis and polarity understanding using the twitter API to stream recent tweets (2) sentimental analysis machine learning using Amazon review data (3) Adopt N-gram language model to do auto-complete (4) Adopt gradient decent/back pass to predict analogy word(s) and word embeddings.

The models used in this report are further presented as follows:

UVA CS5001 Exploratory Text Analytics Final Project
Huilin Chang, hc5hq

Sentimental analysis: the sentiment analysis starts from analyzing a body of text for understanding the opinion expressed by it. In general, we quantify this sentiment with a positive or negative value called polarity. The overall sentiment is often inferred as positive, neutral or negative from the sign of the polarity score.

Sentiment analysis machine learning used in this study includes Logistic Regression, Support Vector Machine, Decision Tree and Random Forest. The evaluation matrix is considered for model selections. The description of algorithms is presented as follows:

- Logistic regression: the logistic regression model : the logistic regression model arises from the desire to model the posterior probabilities of the K classes via linear functions in x, while at the same time ensuring that they sum to one and remain in [0, 1].
- Support Vector Machine: Support Vector Machine is supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. An SVM is a discriminative classifier formally defined by a separating hyperplane. The algorithm outputs an optimal hyperplane which categories new examples. An SVM model is representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.
- Decision Tree: Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node holds a class label.
- Random Forest: Random Forests is a tree-based ensemble with each tree depending on a collection of random variables. We can consider each decision tree in the forest a random subset of features when forming questions and it only has access to a random set of the training data points.

The n-gram language model is used for auto-complete. Lastly, adopt word embeddings to find similar words.

- n-gram language model: n-gram is characterized as a sequence of n variables; we truncate the history to length n-1 $p(w_1, \dots \dots . . w_{i-1}) = p(w_i | w_{i-n+1}, \dots \dots . . , w_{i-1})$. We can represent n-gram in a tree structure using a V -ary branching tree structure for vocabulary size V. Each node in the tree is associated with a probability distribution for the V words in the vocabulary. The unigram is the root node; the V different bigrams are at the next level; and the trigrams are at the next. The tree could be extended further for higher order n-grams. The nodes further down the tree represent longer-distance histories.
- Word embeddings [1]: Word embedding is a real number, vector representation of a word. In general, words with similar meanings will have vector representations that are close together in the embedding space. When we encode words in a numeric form, we

can apply mathematical rules and do matrix operations to them.

In this work, I start with Twitter sentiment analysis and I use the twitter API to stream the topics to get the tweet data set. I stream the period of time (Jan 27th to July 27th, 2020). Further I use Amazon review data for Machine learning of sentiment analysis. Next, I use twitter data for Auto-complete. Lastly, I use US news for a word embedding study.

II. DATABASE AND PURPOSES

The generation or retraction of the database used in this study is described as follows:

1. Twitter sentiment analysis

In this activity, I register a Twitter API developer account, and then I'm able to stream current twitter data from the twitter API. I'm interested in the topics which are a reflection of popular queries. Therefore, I query the topics including "covid-19", "Donald Trump", "birthday", and "love". The purpose of this activity is to use streaming twitter data with chosen topics to predict polarity distributions.

Figures 1 and 2 show the example of streaming twitter data with the query words "Donald Trump" and then extracting the tweet text.

```
In [3]:
1 auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
2 auth.set_access_token(access_key, access_secret)
3 api = tweepy.API(auth, wait_on_rate_limit=True)
```

```
In [5]:
1 # Define the search term and the date_since date as variables
2 search_words = "#Donald Trump"
3 date_since = "2020-01-27"
```

Figure 1: Example of search_words = "#Donald Trump" to collect tweet.text

In [7]:
```
# Collect tweets
tweets = tweepy.Cursor(api.search,
                       q=search_words,
                       lang="en",
                       since=date_since).items(100)

# Iterate and print tweets
for tweet in tweets:
    print(tweet.text)
```

```
@charliekirk11 #Hitler knew that if he created an imaginary enemy it would help him mobilize fearful people so he c… https://t.
co/TAs7q84Kbh
@charliekirk11 #Hitler knew that if he created an imaginary enemy it would help him mobilize fearful people so he c… https://t.
co/9kdAXAxeLB
#DONALD TRUMP is a RUSSIAN ASSET, a Russian AGENT! #Putin'sBitch!
@realDonaldTrump #DONALD TRUMP is a RUSSIAN ASSET, a Russian AGENT!
Please don't #tweet anymore #Donald
No, seriously #NOMORE #Twitter #ترامب #Трамп 🚮#Trump
```

Figure 2: Example of collect tweets with keyword "Donald Trump"

2. Amazon review sentiment analysis

In this activity, I downloaded Amazon review data from the website: http://jmcauley.ucsd.edu/data/amazon/. Since I moved to California in 2019, I have been interested in the topic "Patio Lawn and Garden". I downloaded the data from the Amazon site. The purpose of this activity is to use machine learning to predict the sentiment of customers from customer review data.

3. Auto-complete

In this activity, I'm interested in US news. I downloaded "en_US.news.txt", and I used this for an auto-complete corpus. The purpose of this work is to predict the next words using the n-gram language model.

4. Word embeddings

In this activity, I'm interested in twitter data and I downloaded "en_US.twitter.txt". I used this for word embeddings. I implement gradient descent and use forward/backward pass to train word similarity. The purpose of this activity is to get word vectors and predict the word similarity.

RESULTS

III. TWITTER SENTIMENT ANALYSIS

Sentiment analysis is perhaps one of the most popular applications of NLP. This aspect of sentiment analysis is to analyze a body of text for understanding the opinion expressed by it. In general, sentiment is quantified with positive, neutral and negative values called polarity. We can

UVA CS5001 Exploratory Text Analytics Final Project
Huilin Chang, hc5hq
use a polarity score to determine the overall sentiment based on positive values, neutral values or negative values.

1. Twitter sentiment analysis

First, I collect the tweets and review the tweets. I search the word as a topic for sentiment analysis. The words I chose are “Covid-19”, “Donald-Trump”, “Love”, “Birthday”.
From the Tweet example, I can clearly check if my tweet collection is correct or not.

The example tweets relevant to each topic are as follows:

- The first tweet of **Covid-19** “What you need to know about COVID-19: Stop the LYING”.

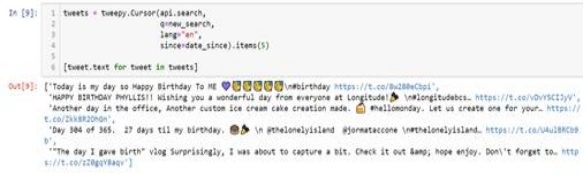
- The first tweet of **Donald Trump** “Hitler knew that if he created an imaginary enemy it would help mobilize the fearful people so he c..” .
- The first tweet of **love** is “I scratched my forehead, resigned. Arguing with a tight-ass was a pointless exercise. ‘At least tell me what you kn….
- The first tweet of **birthday** is “Today is my day so Happy Birthday To ME 💜🥳🥳🥳🥳🥳\n#birthday https://t.co/8w280eCbpi'”

Table 1 shows tweet example from each search word

COVID -19

Donald Trump

Love



Birthday

Then I use a table format to facilitate the following polarity analysis as shown in Table 2.

UVA CS5001 Exploratory Text Analytics Final Project
Huilin Chang, hc5hq

Table 2: Table format of tweets with search words "COVID-19", "Donald Trump", "Love" and "Birthday".

	user	text
0	_LoveBlondz	my mother ain't now say owen arthur passed awa...
1	amNewYork	Factbox: How does being fatter increase severe...
2	TheDailySun	Navarro County reported 19 new cases of COVID-...
3	XBLArrgh	@KittyPlays Because of COVID-19 there are a sh...
4	KStateRschExtn	On #AgToday:\n• The weekly cattle market updat...
...	...	...
495	QueensParkToday	We know Amazon is keeping employees in the dar...
496	MattZajechowski	How COVID-19 Could Change Saving Habits for Mi...
497	WLJReporter	Have questions about appearing in court in per...
498	BluDigitalLife	The report revealed that the net balance of co...
499	pontiacaholic	@pgammo Very intelligent.\nhttps://t.co/8cWxfl...

COVID -19

	user	text
0	PatrioticRight	@charliekirk11 #Hitler knew that if he created...
1	PatrioticRight	@charliekirk11 #Hitler knew that if he created...
2	ykhalim	#DONALD TRUMP is a RUSSIAN ASSET, a Russian AG...
3	ykhalim	@realDonaldTrump #DONALD TRUMP is a RUSSIAN AS...
4	RLCpsychology	Please don't #tweet anymore #Donald\nNo, serio...
...	...	...
205	ActorDeborah	@donwinslow @jack @TwitterSupport Everything I...
206	dailymvtv	President Donald Trump Speech At UPS Airport I...
207	Ronnieconnell7	#donald trump anyone else out there like me ju...
208	yetti420_69	Donald "Its not my Fault" Trump #donald #covid...
209	__peso___	Is Eric #TrUMp really #Donald's son? 🤔 Hmmm

Donald Trump

	user	text
0	RituSingh1131	👉Youtube Channel name :- Fashion Tips By Ritu ...
1	CBI_Rocks	#Love this #event for a #greatcause, get out t...
2	menitinsawant	Fear makes the wolf bigger than he is... 🐺🔥\n....
3	NoreenWise777	To save planet, path forward is ONLY buy #clim...
4	Affiliates_geek	https://t.co/pDN0DpRL4z #butterfly high heels ...
...	...	...
495	MERNiK6	Lineup of Al-Sadd football club enter their ma...
496	KimHyunJoongFan	In honor of #Taemin's upcoming comeback. Read ...
497	Euralia7	Yes, your poems had the exact metric to fill i...
498	stiffmidlefinga	Caterpillar day. https://t.co/svMnHdy40S #musi...
499	22nishtha	Hello Resties! ❤️\n\n#rzpicprompt155 #yqrestzo...

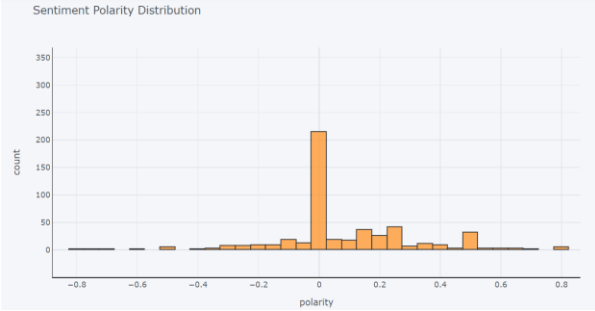
Love

	user	text
0	victoria2707	Today is my day so Happy Birthday To ME 💜😇😇😇😇😇...
1	longitudebcs	HAPPY BIRTHDAY PHYLLIS!! Wishing you a wonderf...
2	Marbleslabba	Another day in the office, Another custom ice ...
3	mummychelle	Day 304 of 365. 27 days til my birthday. 🍩🍌 \...
4	tati_Djuicy	"The day I gave birth" vlog Surprisingly, I wa...
...	...	...
495	simarp	Good Morning!\nMerae Desh Ki Dharti Sona Ugle,...
496	N_Confectionery	Happy Birthday. May each an every moment of yo...
497	stevenbrown87	Good day/night for dads bday 🍡👍 #dad #family #...
498	Waldmeisterei	A nice, quiet day. #birthday #44
499	raresince92	2nd Birthday post because I love her and she d...

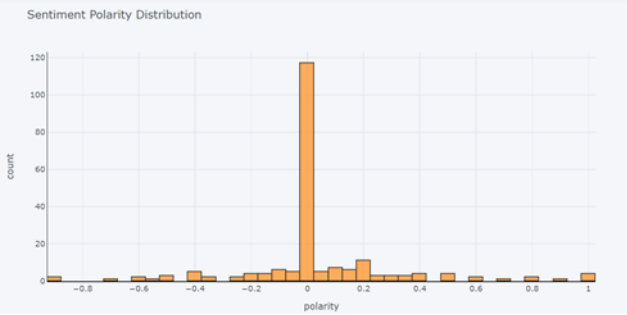
500 rows × 2 columns

Birthday

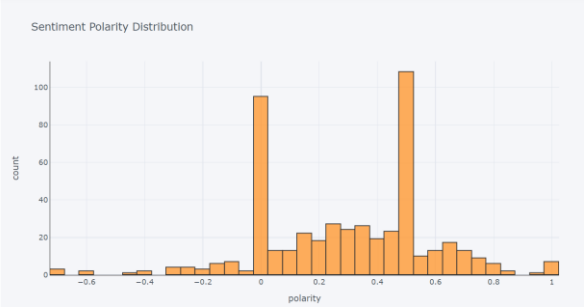
Polarity distribution plots for search words "Donald Trump", "Covid-19", "Birthday" and "Love" are shown in Figure 3. I used TextBlot sentiment analysis [1]
Obviously, we can see the search words are "love" and "birthday" show a higher polarity than the words "covid-19" and "Donald Trump".



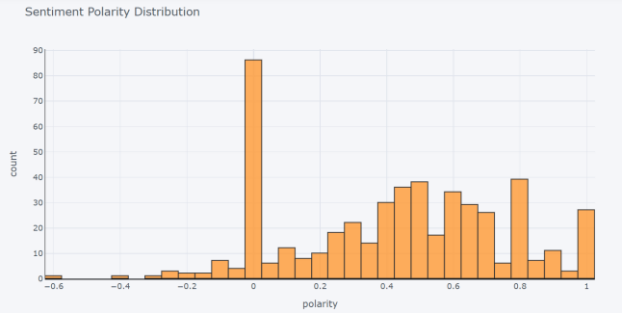
Tweets polarity related to search word "covid-19" polarity distribution



Tweets polarity related to search word "Donald Trump" polarity distribution



Tweets polarity related to search word "love" polarity distribution

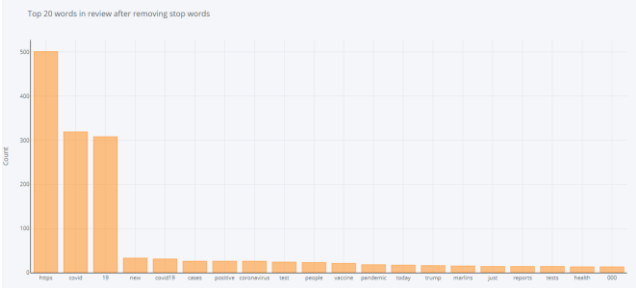


Tweets polarity related to search word "Birthday" polarity distribution

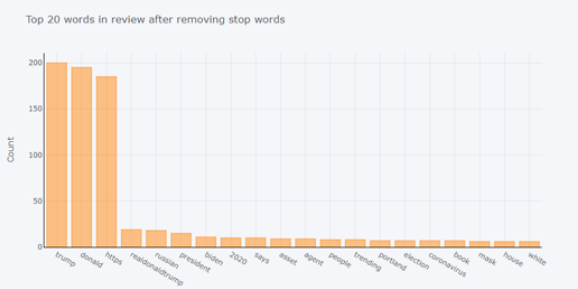
Figure 3 Tweets polarity distribution of each topic

The top 20 words for each topic after removing stop words are shown in Fig 4.

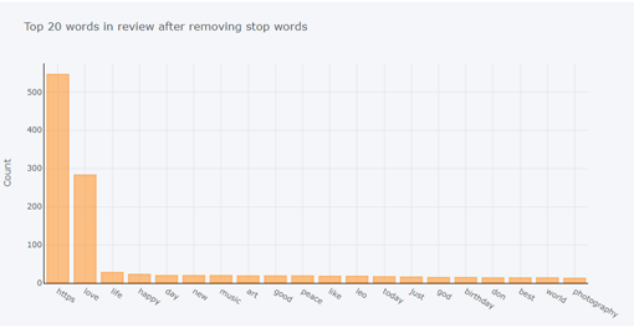
- The top 20 words for select topic is "Covid-19" are "covid, 19, new, covid19, cases, positive, cornavirus, test, people, vaccine, pandemic, today, trump, marlins, just, reports, tests, health"
- The top 20 words for select topic is "Donald Trump" are "trump, donald, realdonaldtrump, Russian, president, biden, 2020, says, asset, agent, people, trending, electron, coronavirus, book, mask, house, white"
- The top 20 words for select topic is "love" are "love, life, happy, day, new, music, art, good, peace, like, leo, today, just, god, birthday, don, best, world, photography "
- The top 20 words for select topic is "birthday" are " birthday, day, happy, today, hope, special, wishing, wish, great, years, best, love, old, happybirthday, celebrate, returns, like, lovely, amazing "



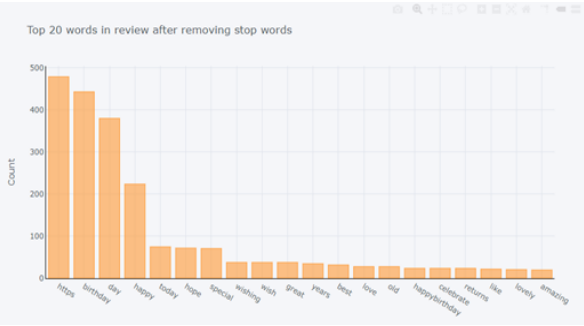
Top 20 words related to search word "covid-19" polarity distribution



Top 20 words related to search word "Donald Trump" polarity distribution



Top 20 words related to search word "love" polarity distribution



Top 20 words related to search word "Birthday" polarity distribution

Figure 4 Top 20 words of each topic

Clearly, the top 20 words for select topics of "love" and "birthday" show more positive words than the topics of "covid-19" and "Donald Trump".

We can further apply n-gram language models to see the top words for each topic. I use trigrams. Trigrams can bring more messages for each topic. Figure 5 shows the top 20 trigrams of each topic.

Top 20 trigrams for search word "covid-19" are

covid 19 https 40, positive covid 19 18, covid 19 cases 15, covid 19 vaccine 14, covid 19 pandemic 10
tests positive covid 8, test positive covid 8, feel safe travelling 8, safe travelling plane 8
travelling plane summer 8, plane summer covid 8, summer covid 19 8, new covid 19 7
remote freelance roles 7, freelance roles commonly 7, roles commonly hired 7
commonly hired today 7, hired today amid 7, today amid covid19 7, amid covid19 https 7

Top 20 trigrams for search word "Donald Trump" are

donald trump russian 9, trump russian asset 9, russian asset russian 9, asset russian agent 9
president donald trump 8, realdonaldtrump donald trump 6
donald trump real 5, trump real donald 5,real donald trump 5, donald trump https 4,
donald trump 2020 4, donald trump officially 4, trump officially textbook 4

UVA CS5001 Exploratory Text Analytics Final Project
Huilin Chang, hc5hq

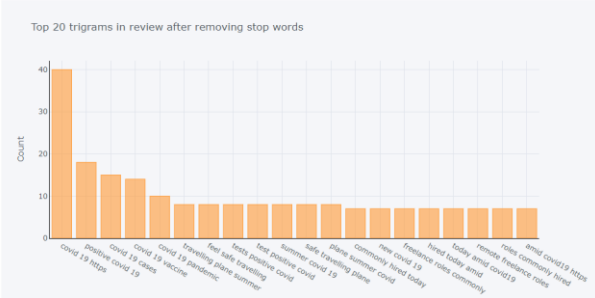
donald trump best 3, trump 2020 election 3, 2020 election david 3, election david horowitz 3
david horowitz thelatestnow 3, says donald trump 3, cognitive test donald 3

Top 20 trigrams for search word "birthday" are

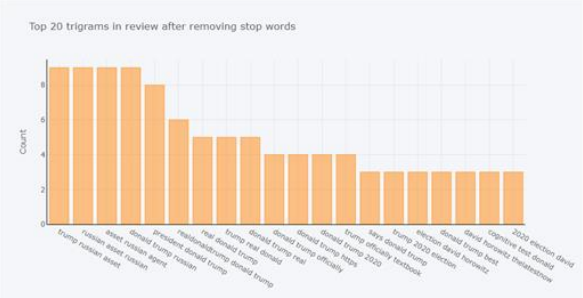
happy birthday leo 9, birthday leo leoseason 9, leo leoseason love 9
leoseason love tarotreading 9, love tarotreading empresscup 9
tarotreading empresscup leo 9, bestlife riders rideon 4
riders rideon like4like 4, rideon like4like harleydavidsonmotorcycle 3
centerofhope la losangeles 3, cats catsofinstagram cat 2, catstagram instagram https 2
travel travelblogger https 2, cartoon art drawing 2
art drawing illustration 2, drawing illustration digitalart 2
illustration digitalart artist 2, fuckme gay gayporn 2
gay gayporn trade 2, gayporn trade suck 2

Top 20 trigrams for search word "love" are

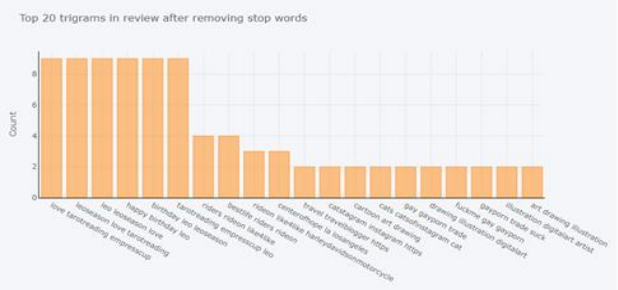
happy returns day 20, hope lovely day 14, happy birthday regular 12, birthday regular guests 12
bigspenwilding hope lovely 12, regular guests bigspenwilding 11, guests bigspenwilding hope 11,
lovely day spencerwilding 11, day spencerwilding https 11, 리나 birthday 리나데이 7
birthday 리나데이 lina_day 7, 리나데이 lina_day 리나_생일축하해 7
day happy birthday 6, excited share latest 6, share latest addition 6, latest addition etsy 6
addition etsy shop 6, best special day 6, hope wonderful day 6, hope great day 6

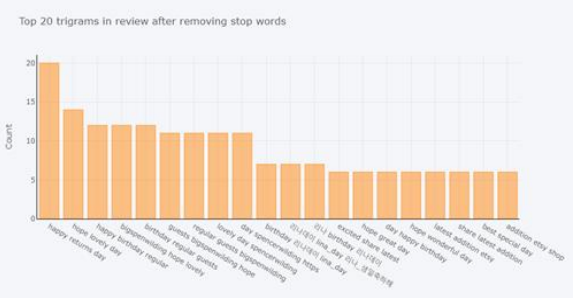


Top 20 trigrams related to search word "covid-19" polarity distribution



Top 20 trigrams related to search word "Donald Trump" polarity distribution





Top 20 trigrams related to search word "love" polarity distribution	Top 20 trigrams related to search word "Birthday" polarity distribution

Figure 5 Top 20 trigrams of each topic

IV. AMAZON REVIEW MACHINE LEARNING

The motivation to study Amazon review data is that I have done a lot of online purchasing from Amazon; therefore, I'm interested in Amazon review and applied machine learning to process sentiment analysis. Amazon review has 5 rating scores: 1, 2, 3, 4, 5. From "Music" review data, you can see there are a total of 10261 reviews (Table 3). The overall rating is the score for each review. If I further check the distribution of the scores, the review has a higher percentage of ratings that are 4 and 5 than 1 and 2 (Fig. 6). This means that Amazon receives more positive reviews than negative reviews. Since score 3 is neither positive or negative, I label this score as neutral. I can further group score 1 and 2 as negative reviews, group scores 4 and 5 as positive reviews. This also implies the data is imbalanced because the weight of positives is higher than the negatives. The model matrix consideration for imbalanced data is taken into consideration. I used machine learning including logistic regression, Support Vector Machine, Decision Tree, and random forest for model selections.

The machine learning procedures are as follows: (1) Pre-processing (2) Machine learning and model selections (3) Evaluation metrics.

UVA CS5001 Exploratory Text Analytics Final Project
Huilin Chang, hc5hq

```
In [45]: 1 data
```

Out[45]:

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime
0	A2IBPI20UZIR0U	1384719342	cassandra tu "Yeah, well, that's just like, u...	[0, 0]	Not much to write about here, but it does exac...	5	good	1393545600	02 28, 2014
1	A14VAT5EAX3D9S	1384719342	Jake	[13, 14]	The product does exactly as it should and is q...	5	Jake	1363392000	03 16, 2013
2	A195EZSQDW3E21	1384719342	Rick Bennette "Rick Bennette"	[1, 1]	The primary job of this device is to block the...	5	It Does The Job Well	1377648000	08 28, 2013
3	A2C00NNG1ZQQG2	1384719342	RustyBill "Sunday Rocker"	[0, 0]	Nice windscreen protects my MXL mic and preven...	5	GOOD WINDSCREEN FOR THE MONEY	1392336000	02 14, 2014
4	A94QU4C90B1AX	1384719342	SEAN MASLANKA	[0, 0]	This pop filter is great. It looks and perform...	5	No more pops when I record my vocals.	1392940800	02 21, 2014
...	...	...	...	...	...	...	...	...	...
10256	A14B2YH83ZXMPP	B00JBIVXGC	Lonnie M. Adams	[0, 0]	Great, just as expected. Thank to all.	5	Five Stars	1405814400	07 20, 2014
10257	A1RPTVW5VEOSI	B00JBIVXGC	Michael J. Edelman	[0, 0]	I've been thinking about trying the Nanoweb st...	5	Long life, and for some players, a good econom...	1404259200	07 2, 2014
10258	AWCJ12KBO5VII	B00JBIVXGC	Michael L. Knapp	[0, 0]	I have tried coated strings in the past (incl...	4	Good for coated.	1405987200	07 22, 2014
10259	A2Z7S8B5U4PAKJ	B00JBIVXGC	Rick Langdon "Scriptor"	[0, 0]	Well, MADE by Elixir and DEVELOPED with Taylor...	4	Taylor Made	1404172800	07 1, 2014
10260	A2WA8TDCTGUADI	B00JBIVXGC	TheTerrorBeyond	[0, 0]	These strings are really quite good, but I wou...	4	These strings are really quite good, but I wou...	1405468800	07 16, 2014

10261 rows × 9 columns

Table 3 Amazon review dataset (patio)

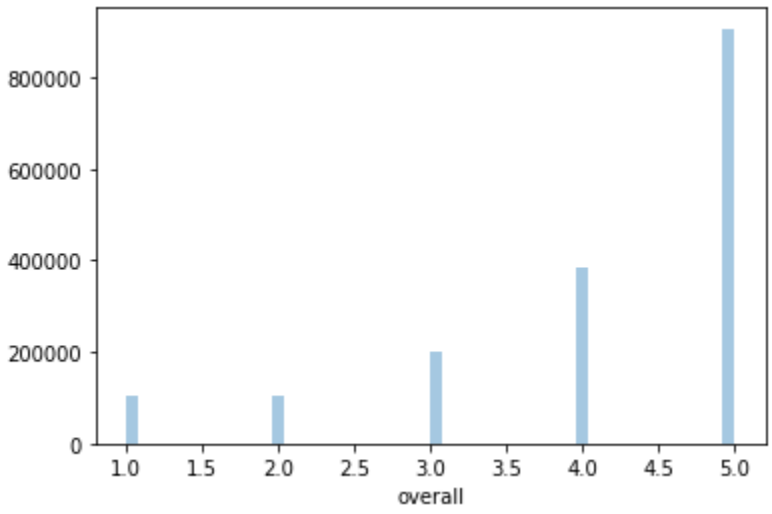


Figure 6 overall rating for Amazon review (patio)

Since the dataset is imbalanced, I adopt F1-score, precision and recall to process model evaluations. The definition of F1-score, precision and recall are as follows:

UVA CS5001 Exploratory Text Analytics Final Project
Huilin Chang, hc5hq

- Precision: TP/TP+ FP
- Recall: TP/TP+FN
- F1 score: 2*(Recall + Precision)/(Recall + Precision)

Table 4 shows the model matrix using Logistic Regression, Support Vector Machine, Decision tree and Random Forest.

	Classification Report				Confusion Metrics
Logistic Regression		precision	recall	f1-score	support	[[1 2 90] [0 7 148] [1 2 1802]]
	Negative	0.50	0.01	0.02	93	
	Neutral	0.64	0.05	0.08	155	
	Positive	0.88	1.00	0.94	1805	
	accuracy			0.88	2053	
	macro avg	0.67	0.35	0.35	2053	
	weighted avg	0.85	0.88	0.83	2053	
Support Vector Machine		precision	recall	f1-score	support	[[13 13 67] [6 18 131] [8 14 1783]]
	Negative	0.48	0.14	0.22	93	
	Neutral	0.40	0.12	0.18	155	
	Positive	0.90	0.99	0.94	1805	
	accuracy			0.88	2053	
	macro avg	0.59	0.41	0.45	2053	
	weighted avg	0.84	0.88	0.85	2053	
Decision Tree		precision	recall	f1-score	support	[[11 13 69] [13 20 122] [43 109 1653]]
	Negative	0.16	0.12	0.14	93	
	Neutral	0.14	0.13	0.13	155	
	Positive	0.90	0.92	0.91	1805	
	accuracy			0.82	2053	
	macro avg	0.40	0.39	0.39	2053	
	weighted avg	0.81	0.82	0.81	2053	
Random Forest		precision	recall	f1-score	support	[[0 1 92] [0 1 154] [0 0 1805]]
	Negative	0.00	0.00	0.00	93	
	Neutral	0.50	0.01	0.01	155	
	Positive	0.88	1.00	0.94	1805	
	accuracy			0.88	2053	
	macro avg	0.46	0.34	0.32	2053	
	weighted avg	0.81	0.88	0.82	2053	

The summarized evaluation matrix for each model is shown in Table 5

Table 5 Summary table of model performance

	Positive F1 score	Accuracy
Logistic Regression	0.94	0.88
Suport Vector Machine	0.94	0.88
Decision Tree	0.91	0.82
Random Forest	0.94	0.88

Since I'm more interested in how the model accurately labels the positive class, the precision and recall values are based on the positive class. The positive F1 score scores for Logistic Regression, Support Vector Machine, Decision Tree and Random Forest are 0.94, 0.94, 0.91, and 0.94, respectively. We can see Logistic Regression, Support Vector Machine, and Random Forest showing comparable F1 score. The decision tree preforms the worst among all. The model accuracy for Logistic Regression, Support Vector Machine, Decision Tree and Random Forest are 0.88, 0.88, 0.82, and 0.88, respectively. The model accuracy for Logistic Regression, Support Vector Machine, and Random Forest are 0.88. We can see decision tree shows the worst accuracy among all.

V. AUTO-COMPLETE

In this activity, I'm motivated because I'm using an auto-complete system everyday. When I google something, I get a suggestion(s) to help me to complete my search. When I write an email, I get suggestions telling me possible endings to my sentences. A key building block for an auto-complete system is a language model. A language model assigns the probability to a sequence of words, in a way that more likely sequences receive higher scores. For example "*I like CS5001*" has a higher probability than "*I am CS5001*". I use an N-grams method for language modeling, and the procedures are as follows:

- Preprocess data
    - Load and tokenize data
    - Split the sentences into train and test sets
    - Replace words with a low frequency
- Develop N-gram based language models
    - Compute the count of n-grams from a given dataset

    - Estimate the conditional probability of a next word with k-smoothing
- Evaluate the N-gram models by computing the perplexity score
- Use your own model to suggest an upcoming word give your sentence

The outcome of trained models show the predicted word based on the precious tokens.

- When I input "I want to go ", the predicted next word is "to"
- When I input "Hey how are", the predicted next word is "the, you, depaul, depaul"
- When I input "What is your", the predicted next word is "doctor, dog, depaul"
- When I input "My name is", the predicted next word is "doing, david, depaul".

The implemented auto-complete sometimes makes sense but sometimes it is inaccurate. The possible reason is the corpus for training models. The auto-complete implementation is able to predict the next words using an n-gram language model.

The result of this activity can be seen from Table 6

Table 6 Auto-complete python implementation

In [50]:
```
previous_tokens = ["i", "want", "to", "go"]
tmp_suggest5 = get_suggestions(previous_tokens, n_gram_counts_list, vocabulary, k=1.0)

print(f"The previous words are {previous_tokens}, the suggestions are:")
display(tmp_suggest5)
```

```
The previous words are ['i', 'want', 'to', 'go'], the suggestions are:

[('to', 0.0211940999470284),
 ('to', 0.006951171542467314),
 ('to', 0.00044599217986314763),
 ('to', 7.958128003428117e-05)]
```

In [52]:
```
previous_tokens = ["hey", "how", "are"]
tmp_suggest6 = get_suggestions(previous_tokens, n_gram_counts_list, vocabulary, k=1.0)

print(f"The previous words are {previous_tokens}, the suggestions are:")
display(tmp_suggest6)
```

```
The previous words are ['hey', 'how', 'are'], the suggestions are:

[('the', 0.013606988601305492),
 ('you', 0.00031208885353241746),
 ('depaul', 6.123886218194066e-06),
 ('depaul', 6.123886218194066e-06)]
```

UVA CS5001 Exploratory Text Analytics Final Project
Huilin Chang, hc5hq

VI. WORD EMBEDDINGS

In this activity, I compute word embeddings and use them for sentiment analysis. The first step is to implement sentiment analysis and count the number of positive words and negative words.

Since word vectorization is the way to represent each word numerically, word to vector is the next step. When the word can be represented in a vector, the vector could then represent syntactic (i. e. parts of speech) and semantic (i.e meaning) structures. The continuous bag of words (CBOW) model is implemented. The procedures for this task include training word vectors from scratch, creating batches of data, backpropagation and visualizing learned word vectors.

The architecture for this task is as follows:

$$h = W_1 X + b_1$$

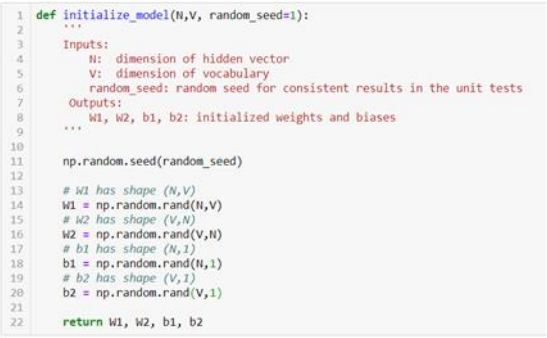
$$a = ReLU(h)$$

$$z = W_2 a + b_2$$

$$\hat{y} = softmax()$$

Continuous Bag-of-Words(CBOW) learns an embedding by predicting the current words based on the context. The context is determined by the surrounding words.
In python, the implementation is shown in Table 7.

Table 7 Python implementation for word embeddings

Initialization the model: initialize two matrices and two vectors	

```
def initialize_model(N,V, random_seed=1):
    '''
    Inputs:
        N:  dimension of hidden vector
        V:  dimension of vocabulary
        random_seed: random seed for consistent results in the unit tests
     Outputs:
        W1, W2, b1, b2: initialized weights and biases
    '''

    np.random.seed(random_seed)

    # W1 has shape (N,V)
    W1 = np.random.rand(N,V)
    # W2 has shape (V,N)
    W2 = np.random.rand(V,N)
    # b1 has shape (N,1)
    b1 = np.random.rand(N,1)
    # b2 has shape (V,1)
    b2 = np.random.rand(V,1)

    return W1, W2, b1, b2
```

UVA CS5001 Exploratory Text Analytics Final Project
Huilin Chang, hc5hq

Implement softmax function

```
# UNQ_C2 (UNIQUE CELL IDENTIFIER, DO NOT EDIT)
# softmax
def softmax(z):
    '''
    Inputs:
        z: output scores from the hidden layer
    Outputs:
        yhat: prediction (estimate of y)
    '''


    e_z = np.exp(z)
    yhat = e_z/np.sum(e_z,axis=0)


    return yhat
```

Implement cross-entropy cost function

```
# compute_cost: cross-entropy cost functioN
def compute_cost(y, yhat, batch_size):
    # cost function
    logprobs = np.multiply(np.log(yhat),y) + np.multiply(np.log(1 - yhat), 1 - y)
    cost = - 1/batch_size * np.sum(logprobs)
    cost = np.squeeze(cost)
    return cost
```

Training the model-Backpropagation: compute the gradients to backpropagate the errors

```
# back_prop
def back_prop(x, yhat, y, h, W1, W2, b1, b2, batch_size):
    '''
    Inputs:
        x:  average one hot vector for the context
        yhat: prediction (estimate of y)
        y:  target vector
        h:  hidden vector (see eq. 1)
        W1, W2, b1, b2:  matrices and biases
        batch_size: batch size
    Outputs:
        grad_W1, grad_W2, grad_b1, grad_b2:  gradients of matrices and biases
    '''

    # Compute l1 as W2^T (Yhat - Y)
    # Re-use it whenever you see W2^T (Yhat - Y) used to compute a gradient
    l1 = np.dot(W2.T,(yhat-y))
    # Apply relu to l1
    l1 = np.maximum(0,l1)
    # Compute the gradient of W1
    grad_W1 = (1/batch_size)*np.dot(l1,x.T)    #1/m * relu(w2.T(yhat-y)) . xT
    # Compute the gradient of W2
    grad_W2 = (1/batch_size)*np.dot(yhat-y,h.T)
    # Compute the gradient of b1
    grad_b1 = np.sum((1/batch_size)*np.dot(l1,x.T),axis=1,keepdims=True)
    # Compute the gradient of b2
    grad_b2 = np.sum((1/batch_size)*np.dot(yhat-y,h.T),axis=1,keepdims=True)

    return grad_W1, grad_W2, grad_b1, grad_b2
```

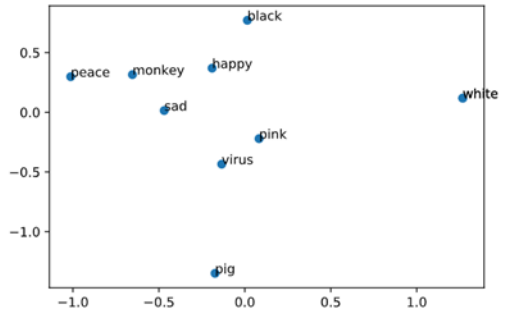
Gradient Decent : implement batch gradient descent over training set

```
# gradient_descent
def gradient_descent(data, word2Ind, N, V, num_iters, alpha=0.03):
    W1, W2, b1, b2 = initialize_model(N,V, random_seed=282)
    batch_size = 128
    iters = 0
    C = 2
    for x, y in get_batches(data, word2Ind, V, C, batch_size):
        # Get z and h
        z, h = forward_prop(x, W1, W2, b1, b2)
        # Get yhat
        yhat = softmax(z)
        # Get cost
        cost = compute_cost(y, yhat, batch_size)
        if ( (iters+1) % 10 == 0):
            print(f"iters: {iters + 1} cost: {cost:.6f}")
        # Get gradients
        grad_W1, grad_W2, grad_b1, grad_b2 = back_prop(x, yhat, y, h, W1, W2, b1, b2, batch_size)

        # Update weights and biases
        W1 -= alpha*grad_W1
        W2 -= alpha*grad_W2
        b1 -= alpha*grad_b1
        b2 -= alpha*grad_b2


        iters += 1
        if iters == num_iters:
            break
        if iters % 100 == 0:
            alpha *= 0.66

    return W1, W2, b1, b2
```

UVA CS5001 Exploratory Text Analytics Final Project
Huilin Chang, hc5hq

Visualizing the word vectors

```
result= compute_pca(X, 2)
pyplot.scatter(result[:, 0], result[:, 1])
for i, word in enumerate(words):
    pyplot.annotate(word, xy=(result[i, 0], result[i, 1]))
pyplot.show()
```



VII. CONCLUSIONS

1. Twitter sentiment analysis starts from a Twitter API developer account application. I can stream the data using query words. In this exercise, I use the query words “COVID-19”, “Donald Trump”, “Love”, “Birthday” to get tweets and analyze the polarity tweets from different query words. The polarity for tweets from query words “Love” and “Birthday” are more positive compared to “COVID-19” and “Donald Trump”.

2. Amazon review sentiment analysis machine learning achieves a F1 positive score of 94% for Logistic regression, Support Vector Machine (SVM), Random Forrest while 91% for

Decision Tree. The dataset is imbalanced with 85% positive compared to 15% negative. The model accuracy achieves 88% for Logistic regression, Support Vector Machine, Random Forrest while 82% for Decision Tree. The F1 negative score are 2%, 22%, 14%, 0% for Logistic regression, Support Vector Machine, Random Forest, respectively. The SVM and decision tree algorithms can classify relative well in the negative class. The future work is to improve the model performance: the consideration of bi-grams or tri-grams text tokenization are options. The token words are essential to the change probability of model prediction. The optimized n-gram as text tokenization is worth evaluating.

3. Auto-complete python implementation adopts n-gram language modeling to predict the next word. The chosen corpus is US-new. The Auto-complete is retrieved based on modeling and the next word is predicted correctly when the syntax is shown in the corpus.

4. Word embeddings adopt gradient decent and back pass algorithms to construct word vectors. The model is able to predict the word vector based on the chosen words and the word similarity.

REFERENCES

[1] (2015, June 07). Retrieved July 27, 2020, from https://planspace.org/20150607-textblob_sentiment/

[2] (n.d.). Retrieved July 28, 2020, from https://www.citationmachine.net/bibliographies/54035991-e566-408c-a9fc-eda3cd9fd032

[3] Heidenreich, H. (2018, August 16). Introduction to Word Embeddings. Retrieved July 28, 2020, from https://towardsdatascience.com/introduction-to-word-embeddings-4cf857b12edc