Getting Sentimental

Utilizing Tweets to Create a Sentiment Analyzer

Siddhant Chauhan, Samir Epili, Callie Gilmore,
Brett Nesfeder & Ali Sayyed

TEXAS McCombs
The University of Texas at Austin
McCombs School of Business

TABLE OF CONTENTS

Project Goal

Problem Statement

- Businesses must have an online presence and conduct social media marketing in order to remain relevant and compete
- It can be difficult to understand how their online platforms are being perceived by users and therefore affecting their brand image
- Positive and negative online interactions can result in an increased brand awareness through going ‘viral’ or can devastate profits through being ‘cancelled’
- Understanding their online perception could allow companies to keep track and protect their brand images and ultimately impact profits

Goals and Importance

Goals

- Identify which words and phrases in tweets display the overall sentiment of the tweet
- Obtain insights that help us determine the correlation between keywords and phrases and their predicted sentiments

Importance

- By understanding these sentiments and key words/ phrases, businesses can do a better job of monitoring their online presence
- Businesses can better understand user's reactions to their brand image through response tweets or tweet mentions
- They can better understand and improve their own tweets to portray a more 'positive' sentiment

Exploratory Analysis

Twitter Dataset

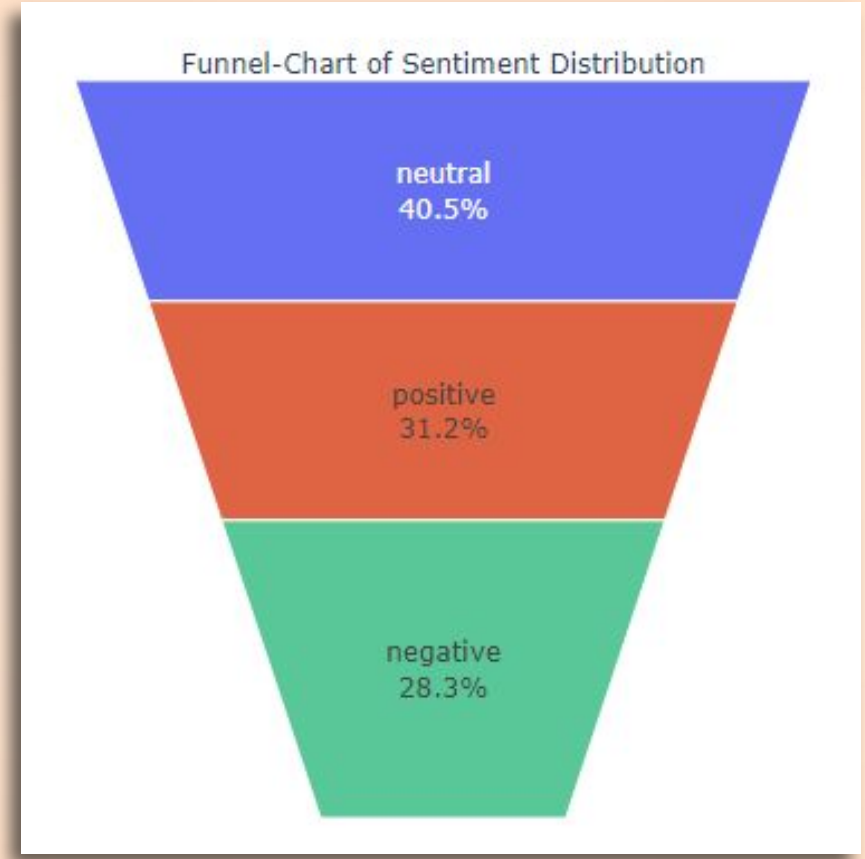
Train
4 Columns
27482 Rows

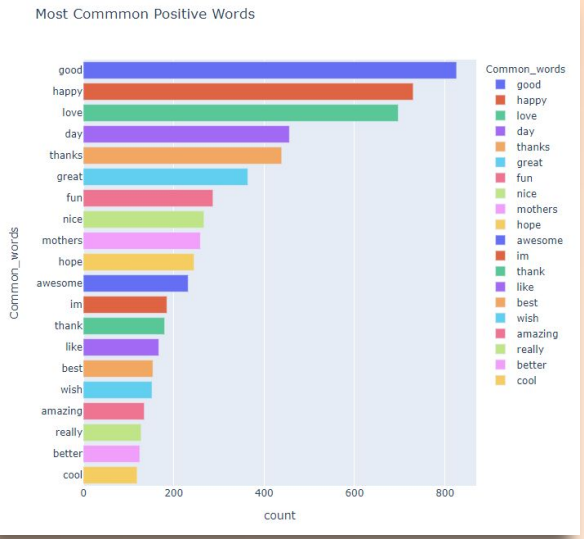
Test
3 Columns
3535 Rows

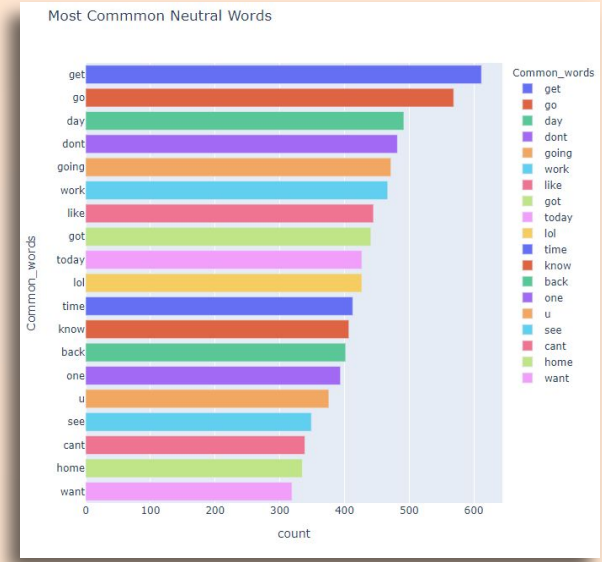
Sentiments
Positive
Negative
Neutral

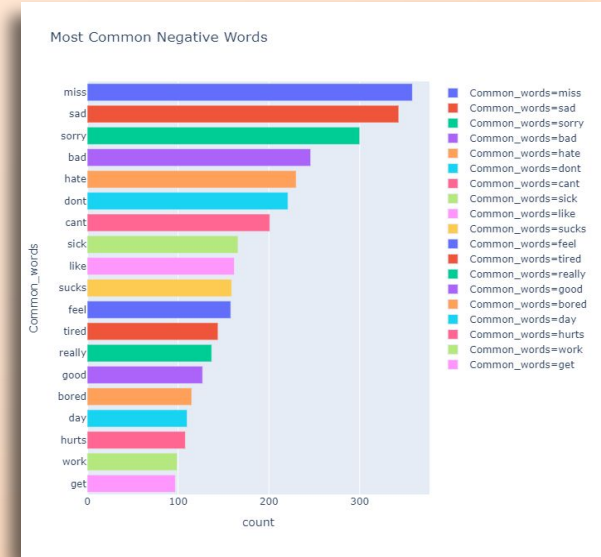
Selected Text
The text that portrays the sentiments

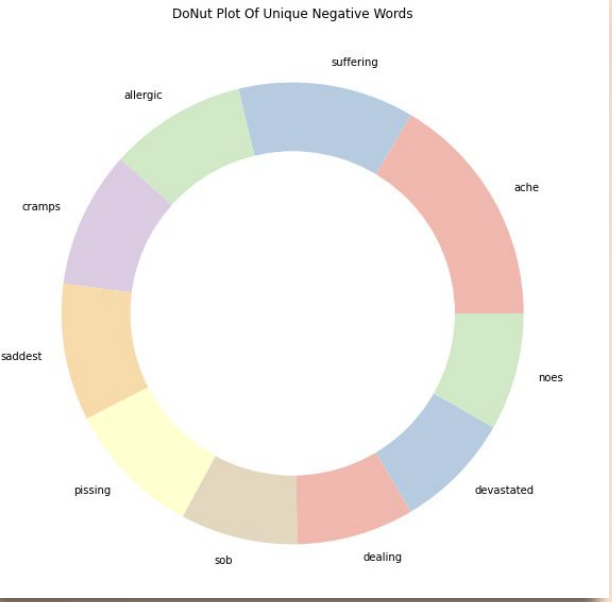
Goal
The goal of our project is to predict the text that portrays the sentiment and compare that to the actual selected text

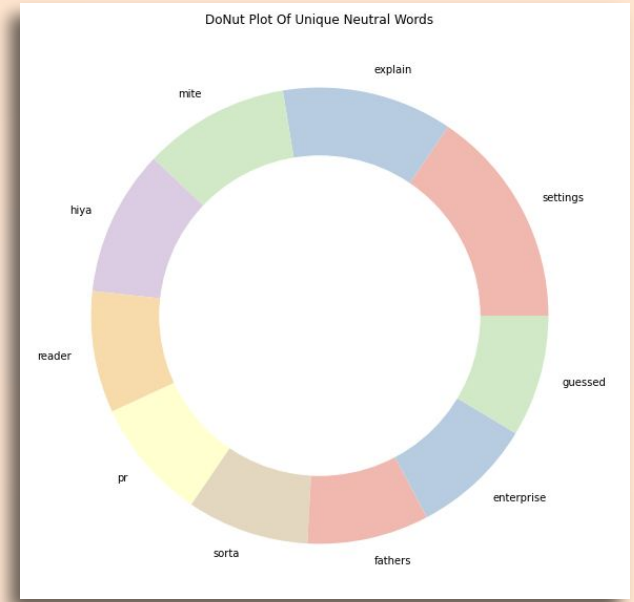

Funnel-Chart of Sentiment Distribution
neutral
40.5%
positive
31.2%
negative
28.3%

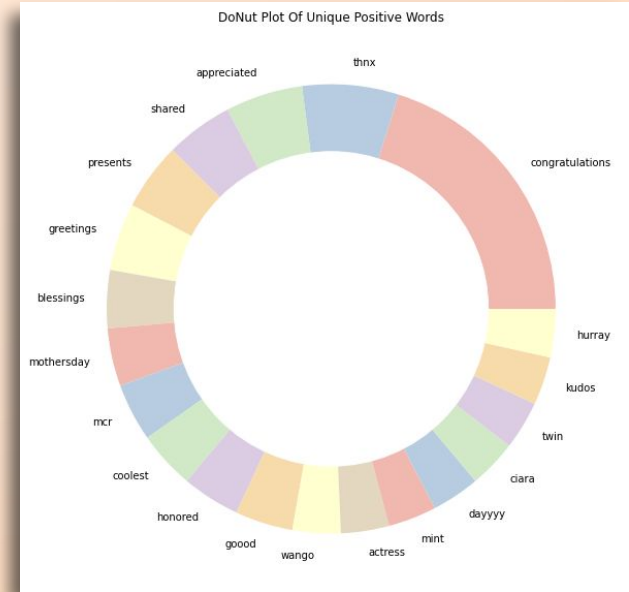

Most Commmon Positive Words
good
happy
love
day
thanks
great
fun
nice
mothers
hope
awesome
im
thank
like
best
wish
amazing
really
better
cool
Common_words
0
200
400
600
800
count
Common_words


Most Commmon Neutral Words
get
go
day
dont
going
work
like
got
today
lol
time
know
back
one
u
see
cant
home
want
Common_words
0
100
200
300
400
500
600
count
Common_words


Most Common Negative Words
miss
sad
sorry
bad
hate
dont
cant
sick
like
sucks
feel
tired
really
good
bored
day
hurts
work
get
Common_words=miss
Common_words=sad
Common_words=sorry
Common_words=bad
Common_words=hate
Common_words=dont
Common_words=cant
Common_words=sick
Common_words=like
Common_words=sucks
Common_words=feel
Common_words=tired
Common_words=really
Common_words=good
Common_words=bored
Common_words=day
Common_words=hurts
Common_words=work
Common_words=get
0
100
200
300
count
Common_words


DoNut Plot Of Unique Negative Words
suffering
allergic
ache
cramps
saddest
noes
pissing
devastated
sob
dealing


DoNut Plot Of Unique Neutral Words
explain
mite
settings
hiya
reader
guessed
pr
enterprise
sorta
fathers


DoNut Plot Of Unique Positive Words
thnx
appreciated
shared
congratulations
presents
greetings
blessings
hurray
mothersday
kudos
mcr
twin
coolest
ciara
honored
dayyyy
goood
mint
wango
actress

Neutral Words

Negative Words

Positive Words

Count Vectorizer

General Idea

- **Using only word counts with CountVectorizer to make predictions**
- Find weights of the words used most often in certain kinds of tweets
- Search all subsets of tweets and calculate the score based on the weights
- For positive or negative tweets, the predicted selected text is the most highly weighted subset
- For neutral tweets, return entire tweets

Remove Null Values, Clean, Lemmatize & Remove Stopwords

```
train[train['text'].isna()]
```

	textID	text	selected_text	sentiment
314	fdb77c3752	NaN	NaN	neutral

```
[11] import re
     from nltk.stem import WordNetLemmatizer

     words = nltk.corpus.stopwords.words('english')
     words.remove('no')
     words.remove('not')
     wn = WordNetLemmatizer()

     a = ['i','to','the','a','my','you','and','it','is','in','for','im','of','me','on','so','have','that','be','its','with','day','at','was']

     i=1


     def clean_txt(txt):
         new = ' '.join([w for w in txt.split() if len(w)>2])
         dup = re.sub(r'\b(\w+)( \1\b)+', r'\1', new)
         unw = " ".join([u for u in dup.split() if u.lower() not in a])
         txt = " ".join([wn.lemmatize(word) for word in unw.split()])
         alp = " ".join([d for d in txt.split() if d.lower() in words or not d.isalpha()])
         return alp

     X_train['selected_text'] = X_train['selected_text'].apply(lambda x: clean_txt(x))
```

```
train['text'] = train['text'].apply(lambda x: x.lower())
test['text'] = test['text'].apply(lambda x: x.lower())

import re

# remove '\\n'
train['text'] = train['text'].map(lambda x: re.sub('\\n',' ',str(x)))

# remove any text starting with User...
train['text'] = train['text'].map(lambda x: re.sub("\[\[User.*",'',str(x)))

# remove IP addresses or user IDs
train['text'] = train['text'].map(lambda x: re.sub("\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}",'',str(x)))

#remove http links in the text
train['text'] = train['text'].map(lambda x: re.sub("(http://.*?\s)|(http://.*)",'',str(x)))
```

Create Positive, Neutral & Negative DFs

```
pos_train.head(5)
```

	textID	text	selected_text	sentiment
13623	2a69c2fa17	my ex-colleagues from shaanxi kept boasting o...	I`m	positive
20298	5efd224f4e	happy mothers day to all im off to spend the...		positive
10645	77c01f0add	yes, that does sound like a distinct advantag...	Fortunately,	positive
20662	057b982878	really good concepts at #mozconcept. i *really...	#mozconcept. *really* mine,	positive
8495	43c2747b8e	going to bed early... got a lot of important w...	BEAR!	positive

```
neg_train.head(5)
```

	textID	text	selected_text	sentiment
14935	1990d1f25b	had to give my 5 year old golden retriever awa...	#sad	negative
518	b41f647910	i miss her alot and its only been one day		negative
25339	f360cc52c2	cupcake!! you can call me anything but 'bitc...	can but 'bitch'	negative
10926	a67e879d1f	already did... index finger, left hand... sti...	bad.	negative
471	7a4a97fba9	the birds are out,, oh man... that`s not cool ...	That`s	negative

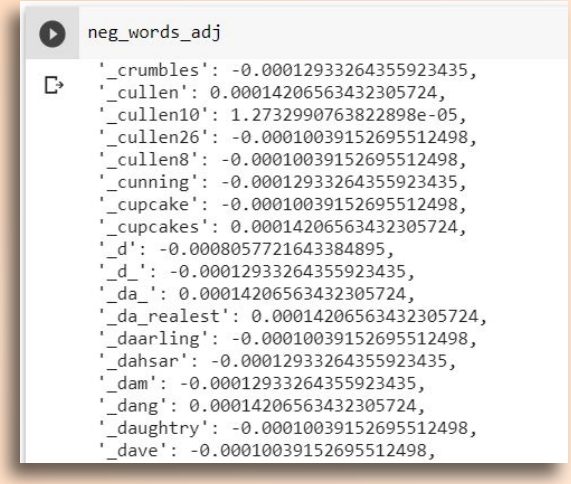
```
neutral_train.head(5)
```

	textID	text	selected_text	sentiment
9384	d4eea7e328	sad day. i was lucky enough to realize it be...	day. before out lot.	neutral
26276	b4410b1922	lol. no new diet plan...ran late for wrk so d...	lol. plan...ran didn`t lunch. when hm!	neutral
20655	03efda0e7e	i`m a minute in and i want to strangle the us...	I`m doing presentation. guy?	neutral
8038	57a8f2f255	back from mountains	from	neutral
2827	ede4721177	no credit left on my mobile sorry brit! *pounc...	Brit! *pounces back*	neutral

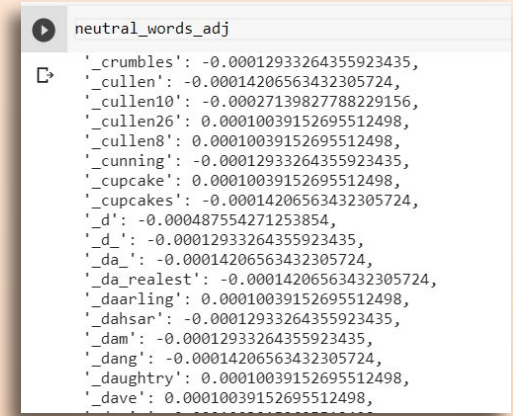
CountVectorizer

```
cv = CountVectorizer(analyzer='word',max_df=0.90, min_df=0.00,stop_words='english')

X_train_cv = cv.fit_transform(X_train['text'])

X_pos = cv.transform(pos_train['text'])
X_neutral = cv.transform(neutral_train['text'])
X_neg = cv.transform(neg_train['text'])


pos_count_df = pd.DataFrame(X_pos.toarray(), columns=cv.get_feature_names())
neutral_count_df = pd.DataFrame(X_neutral.toarray(), columns=cv.get_feature_names())
neg_count_df = pd.DataFrame(X_neg.toarray(), columns=cv.get_feature_names())
```

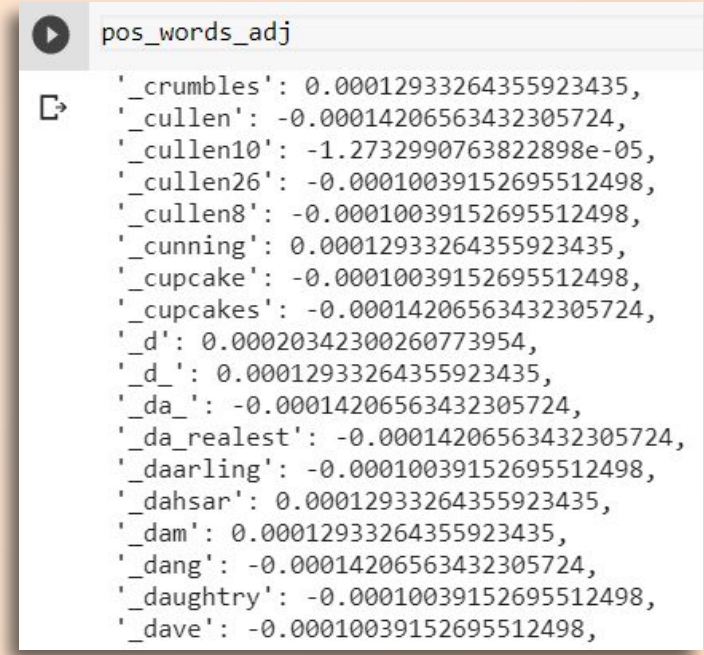
Dictionaries of Positive, Neutral & Negative Words



```
neg_words_adj
'_crumbles': -0.00012933264355923435,
'_cullen': 0.00014206563432305724,
'_cullen10': 1.2732990763822898e-05,
'_cullen26': -0.00010039152695512498,
'_cullen8': -0.00010039152695512498,
'_cunning': -0.00012933264355923435,
'_cupcake': -0.00010039152695512498,
'_cupcakes': 0.00014206563432305724,
'_d': -0.0008057721643384895,
'_d_': -0.00012933264355923435,
'_da_': 0.00014206563432305724,
'_da_realest': 0.00014206563432305724,
'_daarling': -0.00010039152695512498,
'_dahsar': -0.00012933264355923435,
'_dam': -0.00012933264355923435,
'_dang': 0.00014206563432305724,
'_daughtry': -0.00010039152695512498,
'_dave': -0.00010039152695512498,
```



```
neutral_words_adj
'_crumbles': -0.00012933264355923435,
'_cullen': -0.00014206563432305724,
'_cullen10': -0.00027139827788229156,
'_cullen26': 0.00010039152695512498,
'_cullen8': 0.00010039152695512498,
'_cunning': -0.00012933264355923435,
'_cupcake': 0.00010039152695512498,
'_cupcakes': -0.00014206563432305724,
'_d': -0.000487554271253854,
'_d_': -0.00012933264355923435,
'_da_': -0.00014206563432305724,
'_da_realest': -0.00014206563432305724,
'_daarling': 0.00010039152695512498,
'_dahsar': -0.00012933264355923435,
'_dam': -0.00012933264355923435,
'_dang': -0.00014206563432305724,
'_daughtry': 0.00010039152695512498,
'_dave': 0.00010039152695512498,
```



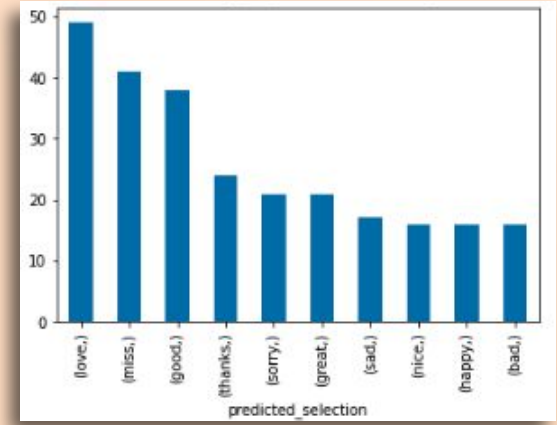
```
pos_words_adj
'_crumbles': 0.00012933264355923435,
'_cullen': -0.00014206563432305724,
'_cullen10': -1.2732990763822898e-05,
'_cullen26': -0.00010039152695512498,
'_cullen8': -0.00010039152695512498,
'_cunning': 0.00012933264355923435,
'_cupcake': -0.00010039152695512498,
'_cupcakes': -0.00014206563432305724,
'_d': 0.00020342300260773954,
'_d_': 0.00012933264355923435,
'_da_': -0.00014206563432305724,
'_da_realest': -0.00014206563432305724,
'_daarling': -0.00010039152695512498,
'_dahsar': 0.00012933264355923435,
'_dam': 0.00012933264355923435,
'_dang': -0.00014206563432305724,
'_daughtry': -0.00010039152695512498,
'_dave': -0.00010039152695512498,
```

Predict the Text that Displays Sentiment in Tweet

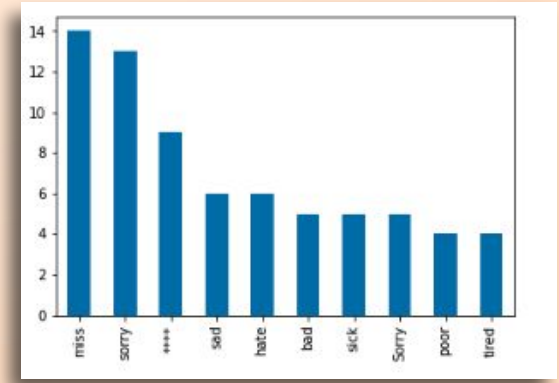
	textID	text	selected_text	sentiment	predicted_selection
20673	3391184efc	loves the nice weather and 7:30 exams	loves	positive	loves the nice
12581	b35daf9677	okay, this weather isn`t 'cute sundress' frien...	Okay, this weather isn`t 'cute sundress' frien...	neutral	okay, this weather isn`t 'cute sundress' frien...
13136	06e5249859	woo hoo!! congratulations	Congratulations	positive	congratulations
14013	3cd4960670	thanks got a hold of someone there who knew...	thanks	positive	thanks
25030	92b75314ca	got back and putting in the laundry. we got in...	Got back and putting in the laundry. We got in...	neutral	got back and putting in the laundry. we got in...

Jaccard Similarity Score

	textID	text	selected_text	sentiment	predicted_selection	jaccard
20673	3391184efc	loves the nice weather and 7:30 exams	loves	positive	loves the nice	0.333333
12581	b35daf9677	okay, this weather isn`t 'cute sundress' frien...	Okay, this weather isn`t 'cute sundress' frien...	neutral	okay, this weather isn`t 'cute sundress' frien...	1.000000
13136	06e5249859	woo hoo!! congratulations	Congratulations	positive	congratulations	1.000000
14013	3cd4960670	thanks got a hold of someone there who knew...	thanks	positive	thanks	1.000000
25030	92b75314ca	got back and putting in the laundry. we got in...	Got back and putting in the laundry. We got in...	neutral	got back and putting in the laundry. we got in...	1.000000
19380	7438c9c09a	congrats on graduating college!	CONGRATS	positive	congrats	1.000000
17226	fa042d9ad5	can you get me a sub from subway when ur on yo...	Can you get me a sub from subway when ur on yo...	neutral	can you get me a sub from subway when ur on yo...	1.000000
10078	a4ede54987	it`s the weekend but 9 year old is grounded wh...	it`s the weekend but 9 year old is grounded wh...	neutral	it`s the weekend but 9 year old is grounded wh...	1.000000




```
The jaccard similarity score is: 0.6625123273340392
```



BiDirectional Encoder

BERT

BERT

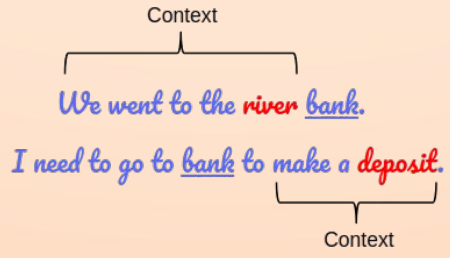
- Designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers

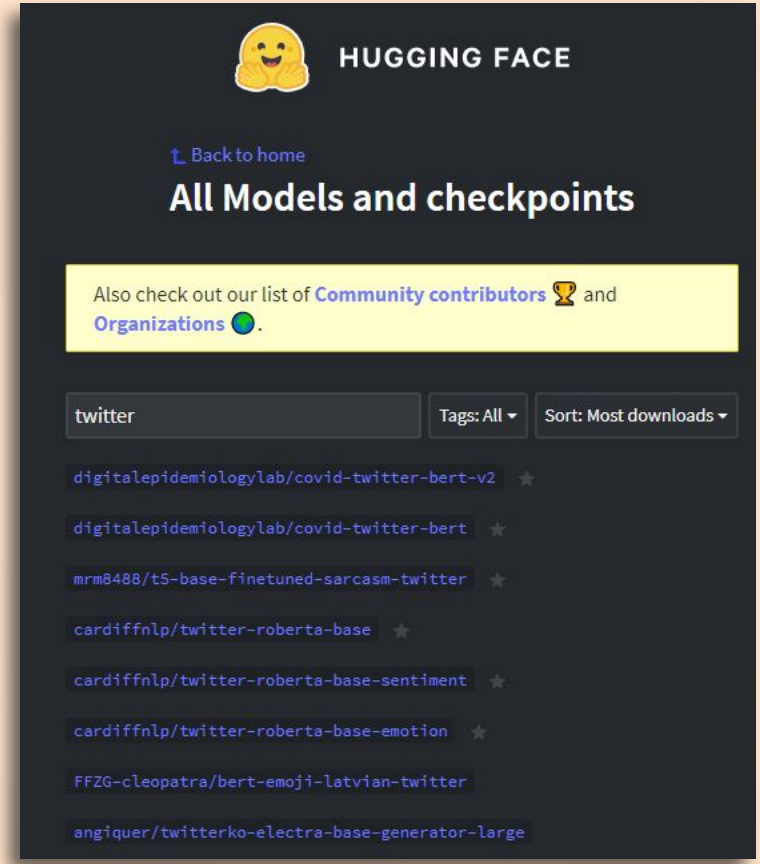
Pre-training Process

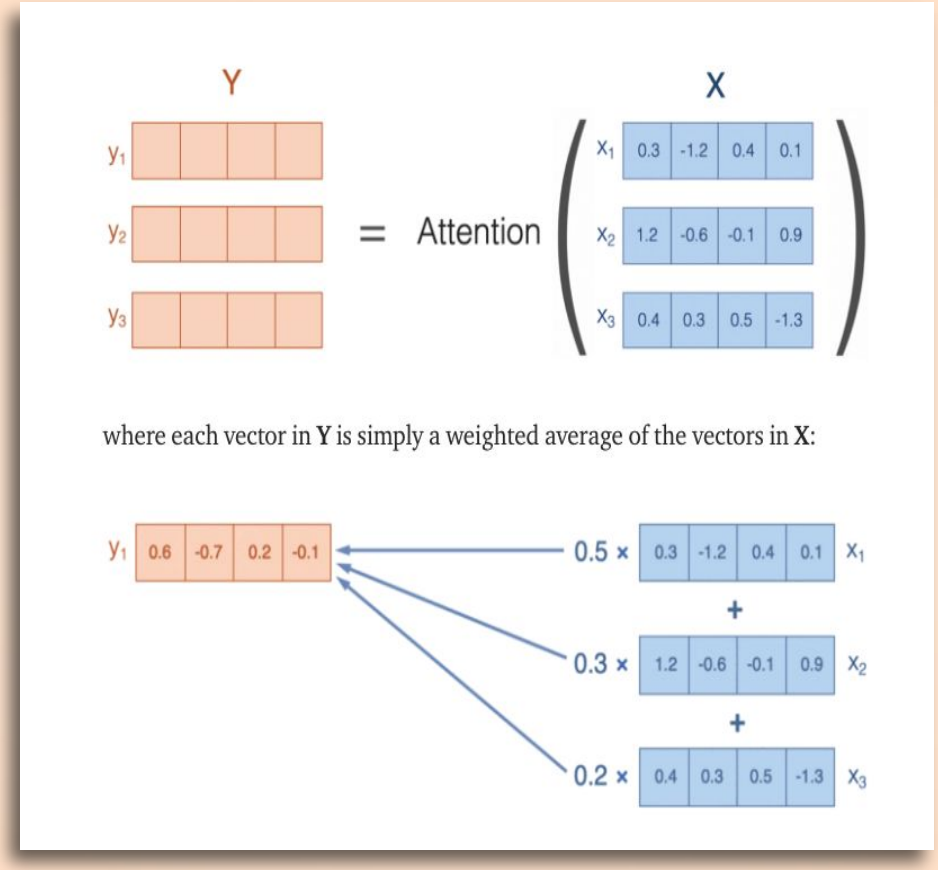
- Masked Language Model (MLM)
- Next Sentence Prediction (NSP)

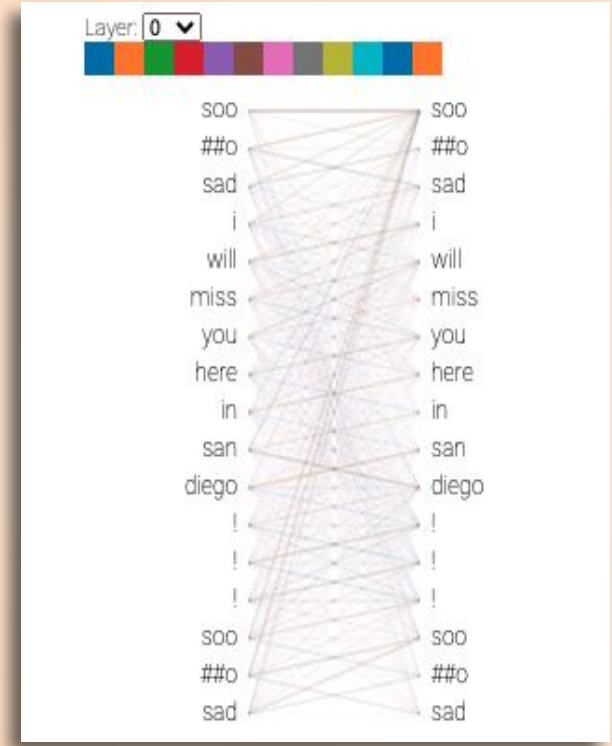
Tokenization

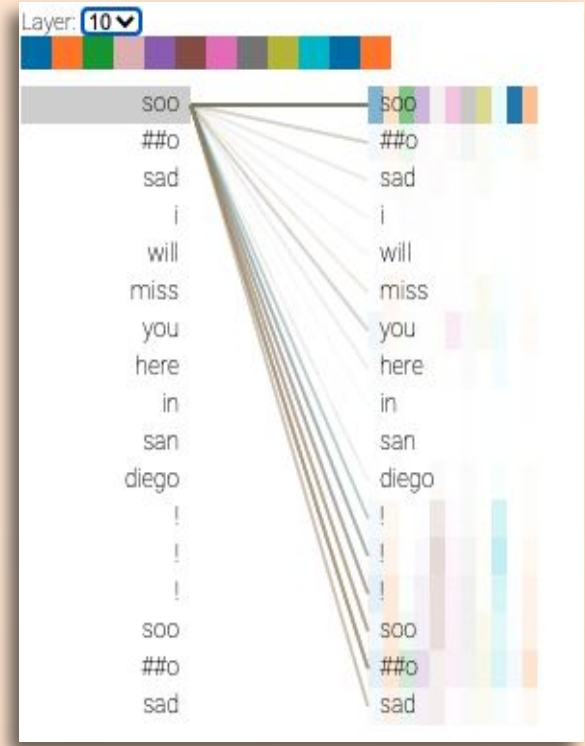
- Looks at WordPieces
  - A word can be broken down into more than one sub-words




HUGGING FACE
Back to home
All Models and checkpoints
Also check out our list of Community contributors and Organizations.
twitter
Tags: All
Sort: Most downloads
digitalepidemiologylab/covid-twitter-bert-v2
digitalepidemiologylab/covid-twitter-bert
mrm8488/t5-base-finetuned-sarcasm-twitter
cardiffnlp/twitter-roberta-base
cardiffnlp/twitter-roberta-base-sentiment
cardiffnlp/twitter-roberta-base-emotion
FFZG-cleopatra/bert-emoji-latvian-twitter
angiquer/twitterko-electra-base-generator-large


Y
X
y1
y2
y3
=
Attention
x1 0.3 -1.2 0.4 0.1
x2 1.2 -0.6 -0.1 0.9
x3 0.4 0.3 0.5 -1.3
where each vector in Y is simply a weighted average of the vectors in X:
y1 0.6 -0.7 0.2 -0.1
0.5 × 0.3 -1.2 0.4 0.1 x1
+
0.3 × 1.2 -0.6 -0.1 0.9 x2
+
0.2 × 0.4 0.3 0.5 -1.3 x3


Layer: 0
soo
##o
sad
i
will
miss
you
here
in
san
diego
!
!
!
soo
##o
sad
soo
##o
sad
i
will
miss
you
here
in
san
diego
!
!
!
soo
##o
sad


Layer: 10
soo
##o
sad
i
will
miss
you
here
in
san
diego
!
!
!
soo
##o
sad
soo
##o
sad
i
will
miss
you
here
in
san
diego
!
!
!
soo
##o
sad

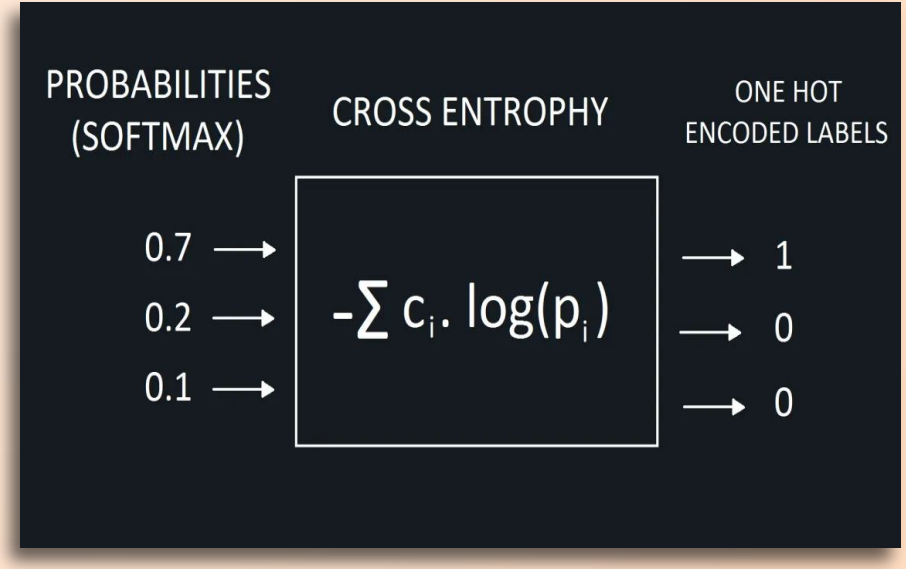
Tokenization

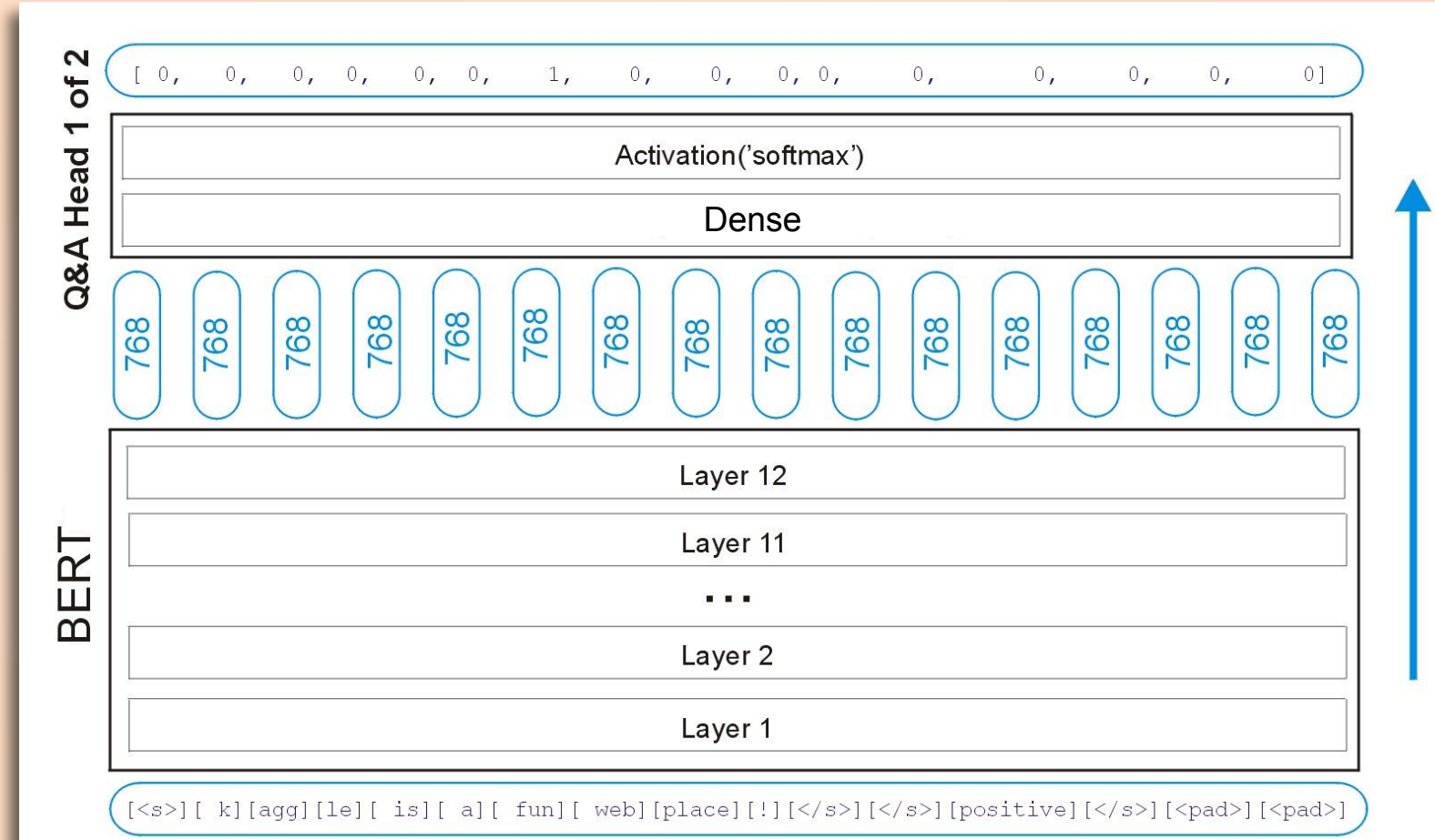
Text = “Kaggle is a fun webplace!”, Selected_text = “fun webplace!”, Sentiment = positive

tokens =	[<s>][k][agg][le][is][a][fun][web][place][!][</s>][</s>][positive][</s>][<pad>][<pad>]
input_ids =	[0, 449,7165,459, 16, 10, 1531, 3748, 6406, 328, 2, 2, 1313, 2, 1, 1]
attention_mask =	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0]
start_token =	[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
end_token =	[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]

Implementation

- Bert-base Uncased:
  - 12-layer, 768-hidden, 12-heads, 110M parameters
  - Trained on lower-cased English text
- Question-Answering Approach
  - Question -> Sentiment
  - Context -> Text
  - Answer -> Selected Text
- TensorFlow Hyperparameters
  - Learning Rate: 3e-5
  - Epochs: 3
  - Batch size: 64
- CrossEntropyLoss combines softmax and cross-entropy




Q&A Head 1 of 2
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Activation('softmax')
Dense
768
768
768
768
768
768
768
768
768
768
768
768
768
768
768
768
BERT
Layer 12
Layer 11
...
Layer 2
Layer 1
[<s>][k][agg][le][is][a][fun][web][place][!][</s>][</s>][positive][</s>][<pad>][<pad>]

```
########################
### FOLD 5
########################
Train on 21985 samples, validate on 5496 samples
Epoch 1/3
21952/21985 [============================>.] - ETA: 0s - loss: 2.6614 - activation_loss: 1.3
259 - activation_1_loss: 1.3355
Epoch 00001: val_loss improved from inf to 2.01513, saving model to v0-bert-4.h5
21985/21985 [==============================] - 357s 16ms/sample - loss: 2.6600 - activation_
loss: 1.3247 - activation_1_loss: 1.3341 - val_loss: 2.0151 - val_activation_loss: 1.0306 -
val_activation_1_loss: 0.9840
Epoch 2/3
21952/21985 [============================>.] - ETA: 0s - loss: 1.9939 - activation_loss: 1.0
006 - activation_1_loss: 0.9933
Epoch 00002: val_loss improved from 2.01513 to 1.89673, saving model to v0-bert-4.h5
21985/21985 [==============================] - 333s 15ms/sample - loss: 1.9935 - activation_
loss: 1.0005 - activation_1_loss: 0.9926 - val_loss: 1.8967 - val_activation_loss: 0.9892 -
val_activation_1_loss: 0.9070
Epoch 3/3
21952/21985 [============================>.] - ETA: 0s - loss: 1.7087 - activation_loss: 0.8
684 - activation_1_loss: 0.8403
Epoch 00003: val_loss improved from 1.89673 to 1.86479, saving model to v0-bert-4.h5
21985/21985 [==============================] - 333s 15ms/sample - loss: 1.7084 - activation_
loss: 0.8682 - activation_1_loss: 0.8398 - val_loss: 1.8648 - val_activation_loss: 0.9861 -
val_activation_1_loss: 0.8781
Loading model...
Predicting OOF...
5496/5496 [==============================] - 32s 6ms/sample
Predicting Test...
3534/3534 [==============================] - 19s 5ms/sample
>>>> FOLD 5 Jaccard = 0.6020019500321347
```

```
Model: "model"
__________________________________________________________________________________________________
______
Layer (type)                 Output Shape              Param #     Connected to
==================================================================================================
======
input_1 (InputLayer)         [(None, 128)]             0
__________________________________________________________________________________________________
______
input_2 (InputLayer)         [(None, 128)]             0
__________________________________________________________________________________________________
______
input_3 (InputLayer)         [(None, 128)]             0
__________________________________________________________________________________________________
______
tf_bert_model (TFBertModel)  ((None, 128, 768), ( 109482240   input_1[0][0]
__________________________________________________________________________________________________
______
dropout_37 (Dropout)         (None, 128, 768)          0           tf_bert_model[0][0]
__________________________________________________________________________________________________
______
dropout_38 (Dropout)         (None, 128, 768)          0           tf_bert_model[0][0]
__________________________________________________________________________________________________
______
dense (Dense)                (None, 128, 1)            769         dropout_37[0][0]
__________________________________________________________________________________________________
______
dense_1 (Dense)              (None, 128, 1)            769         dropout_38[0][0]
__________________________________________________________________________________________________
______
flatten (Flatten)            (None, 128)               0           dense[0][0]
__________________________________________________________________________________________________
______
flatten_1 (Flatten)          (None, 128)               0           dense_1[0][0]
__________________________________________________________________________________________________
______
activation (Activation)      (None, 128)               0           flatten[0][0]
__________________________________________________________________________________________________
______
activation_1 (Activation)    (None, 128)               0           flatten_1[0][0]
==================================================================================================
======
Total params: 109,483,778
Trainable params: 109,483,778
Non-trainable params: 0
__________________________________________________________________________________________________
```

BiDirectional Encoders

roBERTa

roBERTa Builds Upon BERT

Stands for Robustly Optimized BERT Pretraining Approach

RoBERTa performs better than BERT by applying the following adjustments:

1. Bigger training data (16G vs 161G)
2. Using dynamic masking pattern (BERT use static masking pattern)
3. Replacing the next sentence prediction training objective
4. Training on longer sequences

RoBERTa further tuned BERT by increasing data size and hyper-parameters only.

Implementation

- RoBERTa Large
  - RoBERTa using the BERT-base architecture
  - 24-layers, 1024-hidden, 16-heads, 355M parameters
  - Employed Adam optimizer for SGD
  - Learning Rate: 3e-5
  - Epochs: 3
  - Batch size: 32

```
print('>>>> OVERALL 5Fold CV Jaccard =',np.mean(jac))
```

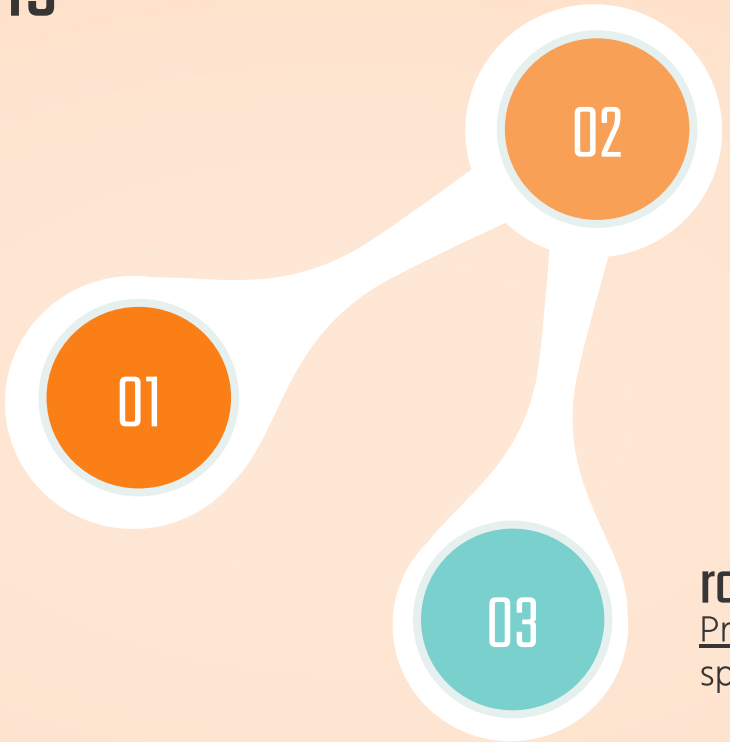
```
>>>> OVERALL 5Fold CV Jaccard = 0.6865174437287186
```

How are Companies Utilizing BERT Models?

- Utilized within Google Search circa 2019
- Facebook
  - Inadvertently created roBERTa for content moderation
  - Their purpose was to create an algorithm that builds up a statistical image of what hate speech or bullying looks like in any language
- Wayfair
  - Extract information from product reviews, return comments and various surveys to learn more about customer experience

Pros & Cons of Each Model

Pros & Cons



Count Vectorizer

Pros Relatively high Jaccard Score, created dictionaries of Positive, Negative & Neutral Words with relative scores

Cons Very manual process, simple

BERT

Pros Older, more versatile and pretrained models are highly accessible

Cons Specialized in MLM and NSP so must adjust layers manually for QA tasks, longer training time

roBERTa

Pros More training data, highly specialized, and faster training

Cons Bulkier model - More storage and computing power required, smaller batch sizes

Conclusions

07

Model Scores

01

Count Vectorizer

Jaccard Score: .6625

02

BERT

Jaccard Score: .6020

03

roBERTa

Jaccard Score: .6865

Goals and Next Steps

Goals

- Identify which words and phrases in tweets display the overall sentiment of the tweet
  - Our models accomplished this
- Obtain insights that help us determine the correlation between keywords and phrases and their predicted sentiments
  - Our Count Vectorizer Model achieved this with the manual dictionary we created

Next Steps

- We recognize our models is most useful when sentiment is already identified, so next steps could be extract our own twitter data, using roberta to classify tweets and then identify sentimentally charged phrases

Referenced Material

- https://www.notion.so/Breaking-BERT-Down-Towards-Data-Science-b7851ec713b449128149263d81378c3d
- https://www.kaggle.com/c/tweet-sentiment-extraction/overview
- https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270
- https://huggingface.co/transformers/pretrained_models.html
- https://arxiv.org/pdf/1810.04805.pdf
- https://arxiv.org/pdf/1907.11692.pdf