NLP Sentiment Analysis

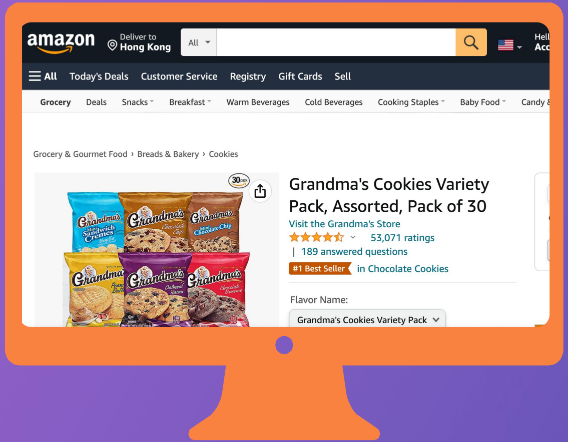
Ching and Chloe

TABLE OF CONTENTS

01

Business Question

Business Objectives



Boost Sales by better marketing tatics

By analysing customers' sentiment, company can know the trend and new feature or products

Improve company's products & services

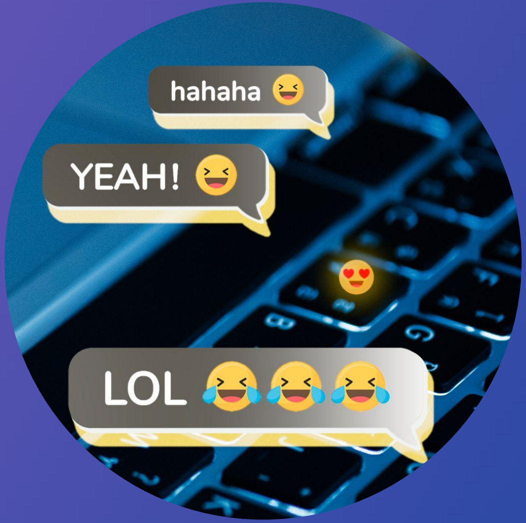
Customers may reviews some issues within the products or certain problems when using the services. Company could address those bugs and provide better one.

Business Objectives

Track sentiments in real-time

Retaining customers could generate more than half of the total revenue.

Hence by tracking real-time, company could quickly identify if sudden changes happened in customers’ feeling. They could react promptly to retain the existing customers.



Maintain brand preception

Customers value brand image. Yet, it requires long time to build the reputation. But at the same time,, it could be ruined within a single day. It is important to monitor the customers’ feeling towards the products.

02

Data Collection

Data Collection

kaggle

STANFORD NETWORK ANALYSIS PROJECT · UPDATED 5 YEARS AGO

1876 | New Notebook | Download (254 MiB)

Amazon Fine Food Reviews

Analyze ~500,000 food reviews from Amazon

https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews

03

Data Preprocessing

Data Preprocessing

Original Dataset

```
df.head()
```

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 568454 entries, 0 to 568453
Data columns (total 10 columns):
 #   Column                  Non-Null Count   Dtype 
---  ------                  --------------   ----- 
 0   Id                      568454 non-null  int64 
 1   ProductId               568454 non-null  object
 2   UserId                  568454 non-null  object
 3   ProfileName             568438 non-null  object
 4   HelpfulnessNumerator    568454 non-null  int64 
 5   HelpfulnessDenominator  568454 non-null  int64 
 6   Score                   568454 non-null  int64 
 7   Time                    568454 non-null  int64 
 8   Summary                 568427 non-null  object
 9   Text                    568454 non-null  object
dtypes: int64(5), object(5)
memory usage: 43.4+ MB
```

Data Preprocessing

1. Create a binary column and define score > 3 is positive, O.W. negative.

```
df['Positive'] = df['Score'].apply(lambda x: 1 if x > 3 else 0)
```

2. Drop duplicate rows by checking “UserID”, “ProfileName”, “Time” and “Text”.

```
clean_df = df.drop_duplicates(subset={"UserId","ProfileName","Time","Text"}, keep='first')
clean_df.shape
```

```
(393933, 11)
```

3. Only extract two columns (review, positive/negative)

```
df_text = clean_df[['Positive','Text']]
```

Text Preprocessing

1. Clean contractions

```
def clean(text):
    text = re.sub(r"n\'t", " not", text)
    text = re.sub(r"\'re", " are", text)
    text = re.sub(r"\'s", " is", text)
    text = re.sub(r"\'d", " would", text)
    text = re.sub(r"\'ll", " will", text)
    text = re.sub(r"\'t", " not", text)
    text = re.sub(r"\'ve", " have", text)
    text = re.sub(r"\'m", " am", text)
    return text
```

2. Define customized stopword list

```
#customized stopword list
stop_words = set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself',
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those',
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of',
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're',
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't",
            'won', "won't", 'wouldn', "wouldn't","'s","..."])
```

3. Import SnowballStemmer

```
from nltk.stem.snowball import SnowballStemmer
snow = SnowballStemmer('english')
```

Text Preprocessing

1. Use regex to clean:
   (i) website links; (ii) words with numbers (e.g. 200ounces, 18yo);
   (iii) words with repeated letters (e.g. yummmmmyyyyy, ahhhhh)
2. Convert the texts into lower case and tokenize into words
3. Exclude stopwords
4. Stem the word tokens by Snowball Stemmer

```
def preprocessing(string):
    word_sent = re.sub(r"http\S+", "", string)
    word_sent = clean(word_sent)
    word_sent = re.sub("\S*\d\S*", "", word_sent).strip() #removing words with numerical digits
    word_sent = re.sub('[^A-Za-z]+', ' ', word_sent)   # removing non-word characters
    word_sent = re.sub(r'(\w)\1{2,}',r'\1',word_sent) #removing words of repeated letters
    word_sent = word_tokenize(word_sent.lower())
    word_sent = [word for word in word_sent if word not in stop_words]
    word_sent= ' '.join([snow.stem(word) for word in word_sent])
    word_sent = ' '.join([w for w in word_sent.split() if len(w)>1])
    return word_sent
```

```
df_text['ProcessedText'] = df_text['Text'].astype(str).apply(preprocessing)
```

Data Preprocessing

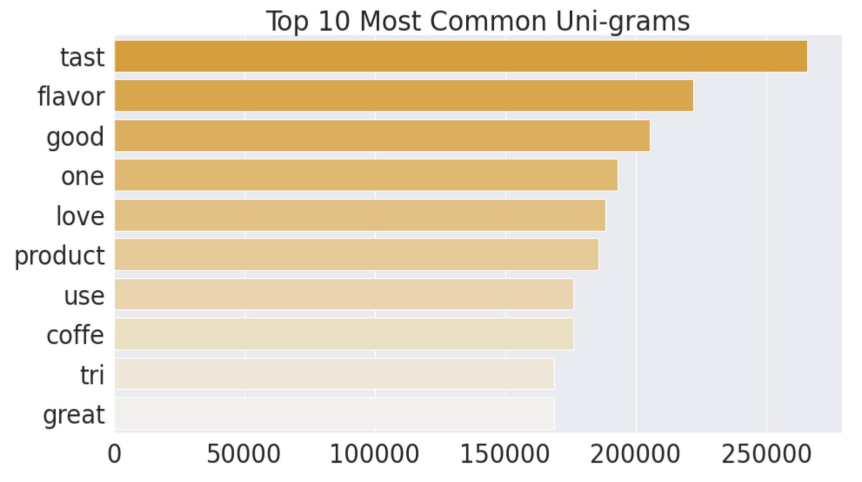
Final Dataframe

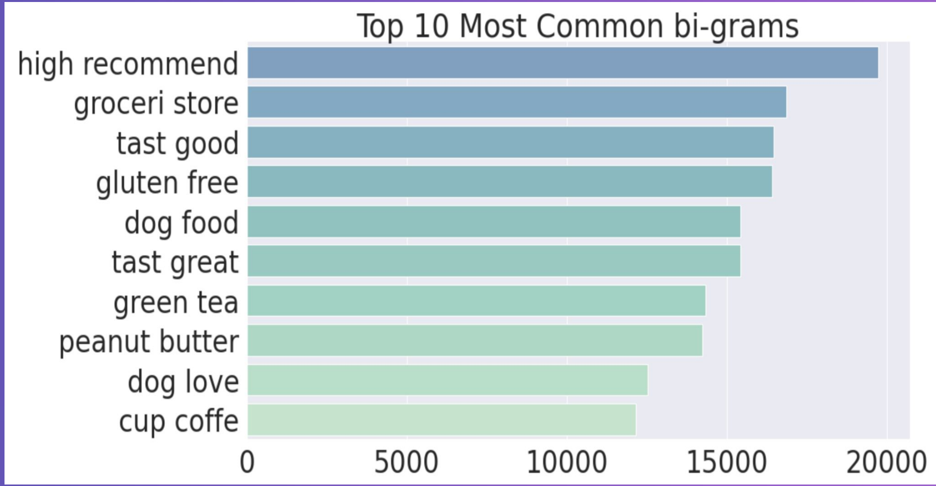
	Positive	Text	ProcessedText
0	1	I have bought several of the Vitality canned d...	bought sever vital can dog food product found ...
1	0	Product arrived labeled as Jumbo Salted Peanut...	product arriv label jumbo salt peanut peanut a...
2	1	This is a confection that has been around a fe...	confect around centuri light pillowi citrus ge...
3	0	If you are looking for the secret ingredient i...	look secret ingredi robitussin believ found go...
4	1	Great taffy at a great price. There was a wid...	great taffi great price wide assort yummi taff...
...	...	...	...
393928	1	Great for sesame chicken..this is a good if no...	great sesam chicken good not better restur eat...
393929	0	I'm disappointed with the flavor. The chocolat...	disappoint flavor chocol note especi weak milk...
393930	1	These stars are small, so you can give 10-15 o...	star small give one train session tri train do...
393931	1	These are the BEST treats for training and rew...	best treat train reward dog good groom lower c...
393932	1	I am very satisfied ,product is as advertised,...	satisfi product advertis use cereal raw vinega...

393933 rows × 3 columns

Data Preprocessing

Finding the most frequent unigrams and bigrams





generally positive

Data Preprocessing

Visualize the most frequent words using wordcloud

positive reviews (Score 4-5)

negative reviews (Score 1-3)

04

Models Creation

Models Creation

Stratified K-fold Cross Validation

```python
from sklearn.model_selection import StratifiedKFold
skf = StratifiedKFold(n_splits=4, shuffle=True, random_state=1)
lst_accu_stratified = []
```

Create a Naive Bayes model

```python
from sklearn import naive_bayes
Naive = naive_bayes.MultinomialNB()
```

```python
for train_index, test_index in skf.split(X, y):
    X_train_fold, X_test_fold = X[train_index], X[test_index]
    y_train_fold, y_test_fold = y[train_index], y[test_index]

    Tfidf_vect = TfidfVectorizer(stop_words='english', max_df=0.8, dtype= np.float32)
    Tfidf_vect.fit(X_train_fold)

    X_train_Tfidf = Tfidf_vect.transform(X_train_fold)
    X_test_Tfidf = Tfidf_vect.transform(X_test_fold)
    Naive.fit(X_train_Tfidf, y_train_fold)
    y_pred = Naive.predict(X_test_Tfidf)
    lst_accu_stratified.append(f1_score(y_test_fold, y_pred))
```

Models Creation

Create a model of SGDClassifier

(Suitable for large and sparse dataset)

```
from sklearn.linear_model import SGDClassifier
sgd = SGDClassifier()
```

```
lst_accu_stratified_sgd = []
for train_index, test_index in skf.split(X, y):
    X_train_fold, X_test_fold = X[train_index], X[test_index]
    y_train_fold, y_test_fold = y[train_index], y[test_index]

    Tfidf_vect = TfidfVectorizer(stop_words='english', max_df=0.8, dtype= np.float32)
    Tfidf_vect.fit(X_train_fold)

    X_train_Tfidf = Tfidf_vect.transform(X_train_fold)
    X_test_Tfidf = Tfidf_vect.transform(X_test_fold)
    sgd.fit(X_train_Tfidf, y_train_fold)

    y_pred = sgd.predict(X_test_Tfidf)
    lst_accu_stratified_sgd.append(f1_score(y_test_fold, y_pred))
```

Models Creation

F1 Score of the Naive Bayes Model after Cross Validation

```
List of possible F1 score: [0.8914590332213512, 0.8911640381307757, 0.8911468261602047, 0.8909237153781592]

Maximum F1 score That can be obtained from this model is: 89.14590332213513 %

Minimum F1 score: 89.09237153781592 %

Overall F1 score: 89.11734032226228 %

Standard Deviation is: 0.00021963800599970014
```

Small standard deviation → robust models

F1 Score of the SGDClassifier Model after Cross Validation

```
List of possible F1 score: [0.918450211579385, 0.9183829511325337, 0.9182471484052476, 0.917640304642314]

Maximum F1 score That can be obtained from this model is: 91.8450211579385 %

Minimum F1 score: 91.76403046423141 %

Overall F1 score: 91.81801539398701 %

Standard Deviation is: 0.00036967704485381517
```

Models Creation

Dimensionality Reduction

Baseline SGDClassifier model:
73780 features ;
F1 Score: 91.81802%

TruncatedSVD transformer is often used on count/tf-idf matrices, known as latent semantic analysis (LSA).

```
from sklearn.decomposition import TruncatedSVD
svd = TruncatedSVD(n_components = 100) # n_components = 100, as recommended in sklearn documentation

X_train_svd = svd.fit_transform(X_train_Tfidf_svd, y_train)
X_test_svd = svd.transform(X_test_Tfidf_svd)
```

100 components:
F1 Score: 89.38094%

```
svd1 = TruncatedSVD(n_components = 500) # try to increase the number of components

X_train_svd1 = svd1.fit_transform(X_train_Tfidf_svd, y_train)
X_test_svd1 = svd1.transform(X_test_Tfidf_svd)
```

500 components:
F1 Score: 90.95921%

Findings

(Unigram TD-IDF)

	Features	Importance
1	great	3.987900
2	best	3.344341
3	love	3.066141
4	delici	3.026104
5	perfect	2.716438
6	excel	2.489259
7	amaz	2.154049
8	good	2.134706
9	awesom	1.851652
10	favorit	1.830844

	Features	Importance
1	disappoint	-4.488774
2	worst	-3.522368
3	ok	-3.468223
4	return	-3.306934
5	aw	-3.040941
6	unfortun	-2.956783
7	terribl	-2.927215
8	horribl	-2.865595
9	okay	-2.714790
10	bland	-2.639161

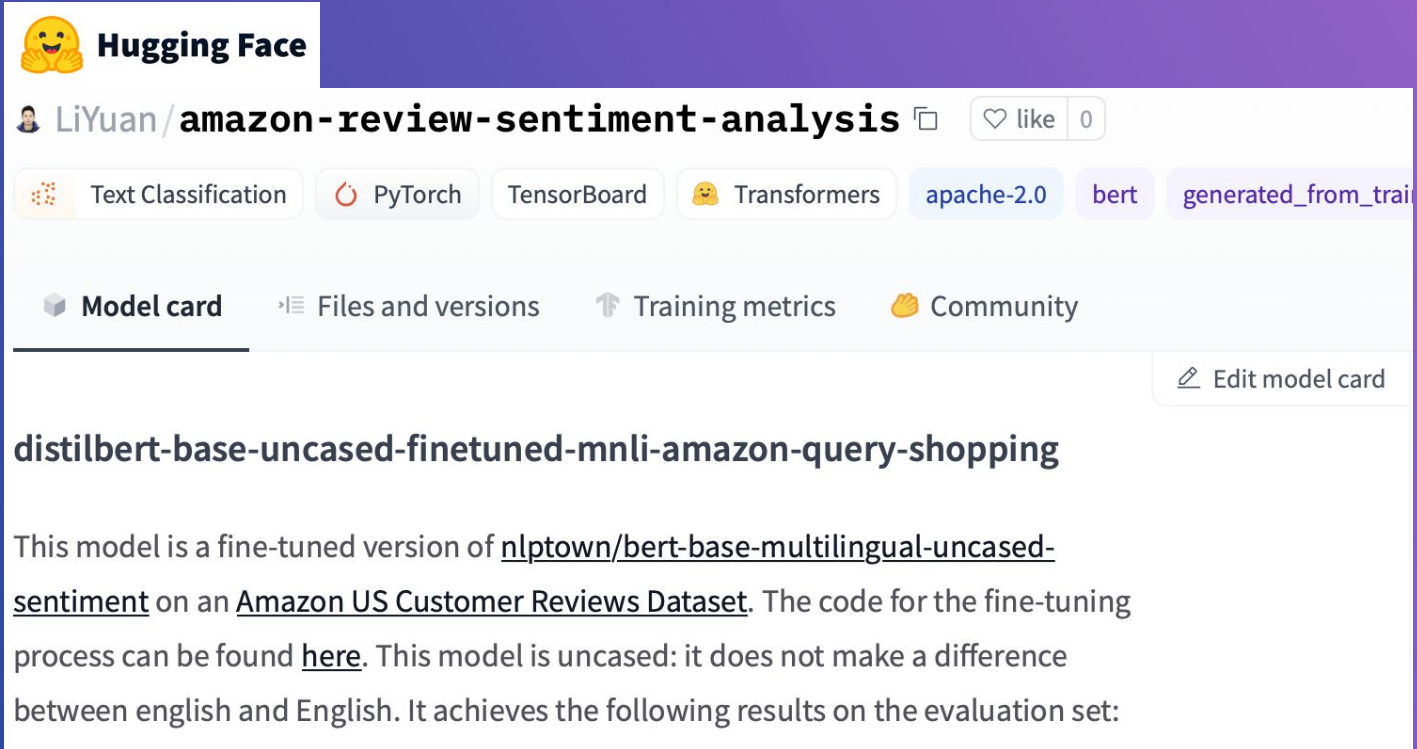
Findings

(Bigram TD-IDF)

	Features	Importance
1	high recommend	0.690828
2	pleasant surpis	0.428914
3	tast great	0.350020
4	great product	0.344050
5	definit buy	0.335500
6	great tast	0.319084
7	love stuff	0.310013
8	bast tast	0.300389
9	realli good	0.299654
10	far best	0.292919

	Features	Importance
1	wast money	-5.883018
2	wo buy	-2.909699
3	disappoint product	-1.438679
4	threw away	-1.405127
5	buyer bewar	-1.340235
6	bad batch	-1.300711
7	wo order	-1.261163
8	throw away	-1.220730
9	tast bad	-1.210144
10	tast ok	-1.192638

Models Creation



Hugging Face

LiYuan / amazon-review-sentiment-analysis

like 0

Text Classification | PyTorch | TensorBoard | Transformers | apache-2.0 | bert | generated_from_trai

Model card | Files and versions | Training metrics | Community

Edit model card

distilbert-base-uncased-finetuned-mnli-amazon-query-shopping

This model is a fine-tuned version of nlptown/bert-base-multilingual-uncased-sentiment on an Amazon US Customer Reviews Dataset. The code for the fine-tuning process can be found here. This model is uncased: it does not make a difference between english and English. It achieves the following results on the evaluation set:

Models Creation

```python
from transformers import AutoTokenizer, AutoModelForSequenceClassification

tokenizer = AutoTokenizer.from_pretrained("LiYuan/amazon-review-sentiment-analysis")

model = AutoModelForSequenceClassification.from_pretrained("LiYuan/amazon-review-sentiment-analysis")
```

```python
for index, entry in enumerate(corpus['text']):
  batch = tokenizer(entry, padding = True, truncation = True, max_length = 512, return_tensors = "pt")
  with torch.no_grad():
    outputs = model(**batch)
    predictions = F.softmax(outputs.logits, dim =1 )
    labels = torch.argmax(predictions, dim = 1)
    labels = [model.config.id2label[label_id] for label_id in labels.tolist()]
    corpus.loc[index,'labels'] = labels
```

Models Creation

	label	label_hf	text
0	5	5 stars	I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better. My Labr...
1	1	1 star	Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actually small sized unsalted. Not sure if this was an error or if the vendor intended to represent the product as "Jumbo".
2	4	5 stars	This is a confection that has been around a few centuries. It is a light, pillowy citrus gelatin with nuts – in this case Filberts. And it is cut into tiny squares and then liberally coated with ...
3	2	5 stars	If you are looking for the secret ingredient in Robitussin I believe I have found it. I got this in addition to the Root Beer Extract I ordered (which was good) and made some cherry soda. The fl...
4	5	5 stars	Great taffy at a great price. There was a wide assortment of yummy taffy. Delivery was very quick. If your a taffy lover, this is a deal.

Models Creation

```
mean_absolute_error(corpus['label'], corpus['label_hf'])
```

```
0.3682
```

```
f1_score(corpus['label'], corpus['label_hf'], average = None)
```

```
array([0.69375    , 0.38857143, 0.41991925, 0.38023451, 0.88294314])
```

```
f1_score(corpus['label'], corpus['label_hf'], average = 'micro')
```

```
0.7444
```

05

Conclusion & Future Improvements

Conclusion & Future Improvements

Limitations

- Dataset is pretty large, requires long time to process
- No real-time tracking as Amazon.com cannot be scrapped directly

Future Improvements

- Apply bagging/ big data techniques
- Look for other data sources which updates regularly for real-time tracking

Next Steps

1

Better Visualization

Using BI tools e.g. Tableau to create a dashboard that make it more user-friendly

2

Compare with Competitors

Conduct similar sentiment analysis on competitors to know the opportunities and threats

THANKS!