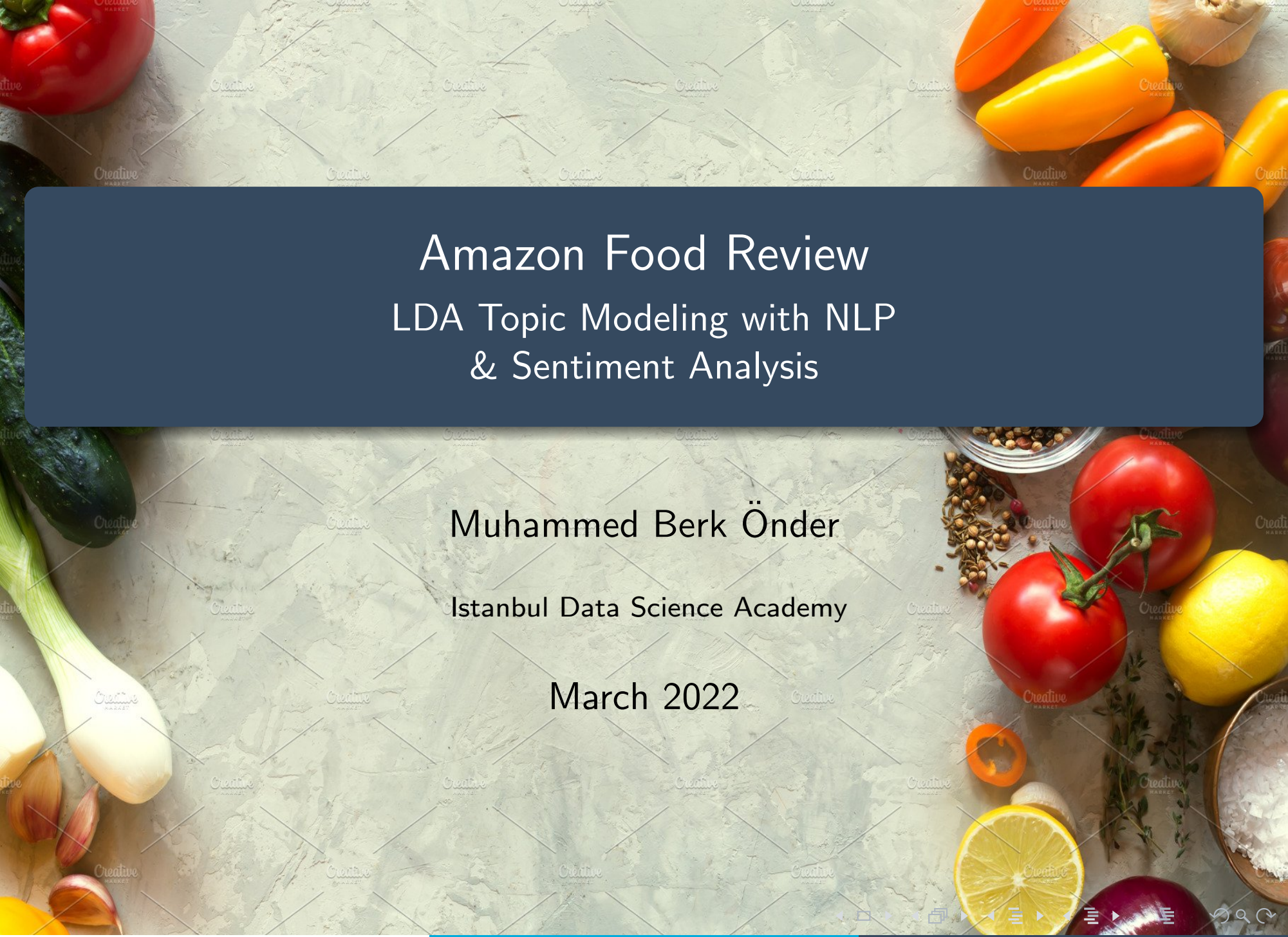


Amazon Food Review

LDA Topic Modeling with NLP & Sentiment Analysis

Muhammed Berk Önder

Istanbul Data Science Academy

March 2022

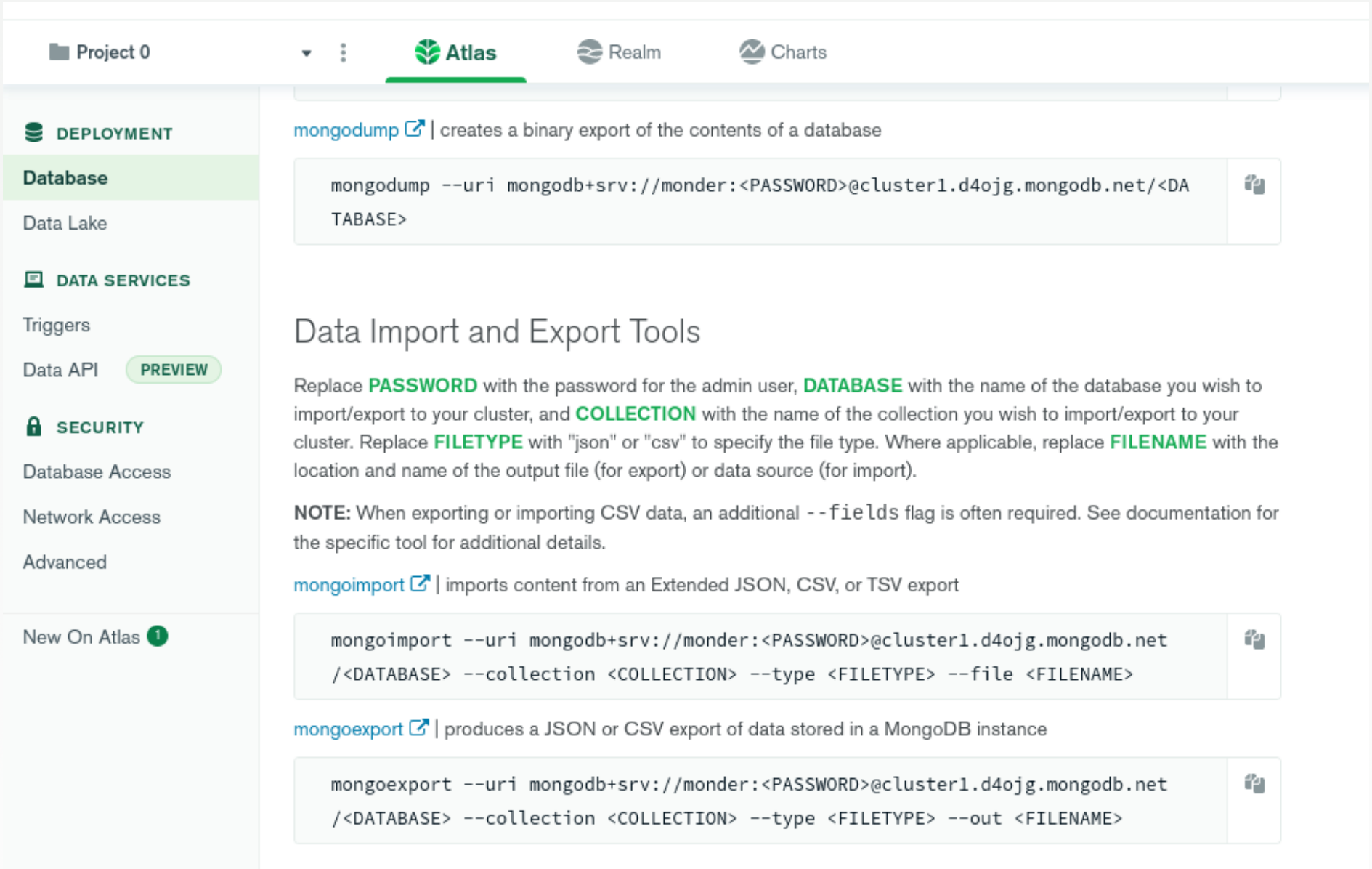
Table of Contents

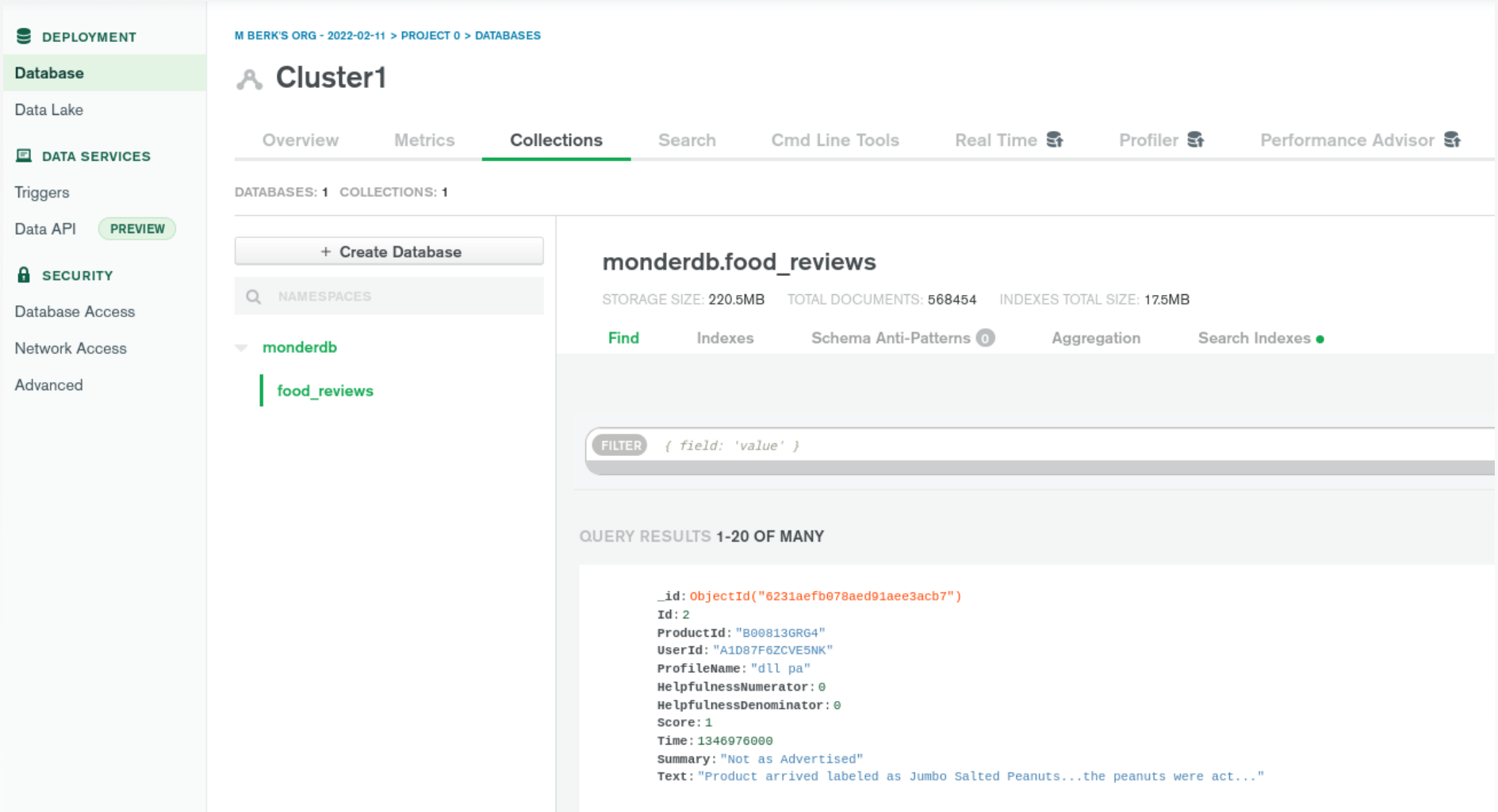
Table of Contents

Methodology

- Collect data from MongoDB with PyMongo
- Clean and Tokenize Data
- CV and TF-IDF
- Latent Dirichlet Allocation
- Model Results

Table of Contents


Project 0
Atlas
Realm
Charts
DEPLOYMENT
Database
Data Lake
DATA SERVICES
Triggers
Data API
PREVIEW
SECURITY
Database Access
Network Access
Advanced
New On Atlas
1
mongodump | creates a binary export of the contents of a database
mongodump --uri mongodb+srv://monder:<PASSWORD>@cluster1.d4ojg.mongodb.net/<DATABASE>
Data Import and Export Tools
Replace PASSWORD with the password for the admin user, DATABASE with the name of the database you wish to import/export to your cluster, and COLLECTION with the name of the collection you wish to import/export to your cluster. Replace FILETYPE with "json" or "csv" to specify the file type. Where applicable, replace FILENAME with the location and name of the output file (for export) or data source (for import).
NOTE: When exporting or importing CSV data, an additional --fields flag is often required. See documentation for the specific tool for additional details.
mongoimport | imports content from an Extended JSON, CSV, or TSV export
mongoimport --uri mongodb+srv://monder:<PASSWORD>@cluster1.d4ojg.mongodb.net/<DATABASE> --collection <COLLECTION> --type <FILETYPE> --file <FILENAME>
mongoexport | produces a JSON or CSV export of data stored in a MongoDB instance
mongoexport --uri mongodb+srv://monder:<PASSWORD>@cluster1.d4ojg.mongodb.net/<DATABASE> --collection <COLLECTION> --type <FILETYPE> --out <FILENAME>


DEPLOYMENT
Database
Data Lake
DATA SERVICES
Triggers
Data API
PREVIEW
SECURITY
Database Access
Network Access
Advanced
M BERK'S ORG - 2022-02-11 > PROJECT 0 > DATABASES
Cluster1
Overview
Metrics
Collections
Search
Cmd Line Tools
Real Time
Profiler
Performance Advisor
DATABASES: 1
COLLECTIONS: 1
+ Create Database
NAMESPACES
monderdb
food_reviews
monderdb.food_reviews
STORAGE SIZE: 220.5MB
TOTAL DOCUMENTS: 568454
INDEXES TOTAL SIZE: 17.5MB
Find
Indexes
Schema Anti-Patterns 0
Aggregation
Search Indexes
FILTER
{ field: 'value' }
QUERY RESULTS 1-20 OF MANY
_id: ObjectId("6231aefb078aed91aee3acb7")
Id: 2
ProductId: "B00813GRG4"
UserId: "A1D87F6ZCVE5NK"
ProfileName: "dll pa"
HelpfulnessNumerator: 0
HelpfulnessDenominator: 0
Score: 1
Time: 1346976000
Summary: "Not as Advertised"
Text: "Product arrived labeled as Jumbo Salted Peanuts...the peanuts were act..."

```
client = pymongo.MongoClient("mongodb+srv://monder:********@cluster1.d4ojg.mongodb.net/myFirstDatabase?retryWrites=true&w=majority")
db = client.monderdb
```

```
db.list_collection_names()
```

```
['food_reviews']
```

```
data = list(db.food_reviews.find())
```

```
download = False
if download:
    review_data = pd.DataFrame(data)
    review_data.to_csv('../data/raw/review_data.csv')
else:
    review_data = pd.read_csv('../data/raw/review_data.csv')

review_data = review_data.sample(5000, random_state=42)
```

```
review_data['Score'].value_counts(normalize=True)
```

```
5    0.6420
4    0.1404
1    0.0882
3    0.0776
2    0.0518
Name: Score, dtype: float64
```

Table of Contents

```
review_data = review_data[review_data['Score'] != 3].reset_index()
review_data['Sentiment'] = np.where(review_data['Score'] >= 4, 'positive', 'negative')
```

```
# 'Summary' and 'Score' features could be used for other projects in the future.
review_data = review_data.loc[:, ['Text', 'Summary', 'Sentiment']]
```

```
review_data.head()
```

	Text	Summary	Sentiment
0	Having tried a couple of other brands of glute...	Crunchy & Good Gluten-Free Sandwich Cookies!	positive
1	My cat loves these treats. If ever I can't fin...	great kitty treats	positive
2	I'm a fan of this brand and this one of my fav...	Great Coffee	positive
3	First there was Frosted Mini-Wheats, in origin...	So the Mini-Wheats were too big?	negative
4	This drink mix was a refreshing treat. It was...	Refreshingly tart and perfectly sweet	positive

```
def clean_text(text):
    clean_text = re.sub('[%s]'%re.escape(string.punctuation), '', text)
    clean_text = re.sub('\w*\d\w*', '', clean_text).lower()
    clean_text = ' '.join([word for word in clean_text.split() if len(word) > 2])
    return clean_text
```

```
def stemming(text):
    stemmer = LancasterStemmer()
    stemmed_text = ' '.join([stemmer.stem(word) for word in text.split()])
    return stemmed_text
```

```
review_data['Text'] = review_data['Text'].apply(clean_text)
```

```
review_data.head()
```

	Text	Summary	Sentiment
0	Having tried a couple of other brands of glute...	Crunchy & Good Gluten-Free Sandwich Cookies!	positive
1	My cat loves these treats. If ever I can't fin...	great kitty treats	positive
2	I'm a fan of this brand and this one of my fav...	Great Coffee	positive
3	First there was Frosted Mini-Wheats, in origin...	So the Mini-Wheats were too big?	negative
4	This drink mix was a refreshing treat. It was...	Refreshingly tart and perfectly sweet	positive

```
review_data.head()
```

	Text	Summary	Sentiment
0	having tried couple other brands glutenfree sa...	Crunchy & Good Gluten-Free Sandwich Cookies!	positive
1	cat loves these treats ever cant find her the ...	great kitty treats	positive
2	fan this brand and this one favorites theirs s...	Great Coffee	positive
3	first there was frosted miniwheats original si...	So the Mini-Wheats were too big?	negative
4	this drink mix was refreshing treat was just t...	Refreshingly tart and perfectly sweet	positive

Table of Contents


taste
one
product
flavor
love
good
coffee
tea
great
food
little
best
much
dont
bag
time
dog
drink
try
way
ive
think
use
find
now
buy
favorite
tried
snack
even
found
eat
cup
well
price
still
bought
nice
keep
delicious
water
sugar
order
better
make
used
day
really
will
amazon
cat
box
made
treat
brand
lot
first
organic
healthy
perfect
got
give
sweet
two
stuff
enough
add
mix
say
thing
know
work
many
using
need

```
y = review_data.Sentiment
X = review_data.Text
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, random_state=42)
```

```
cv1 = CountVectorizer(stop_words='english')

X_train_cv1 = cv1.fit_transform(X_train)
X_test_cv1 = cv1.transform(X_test)

pd.DataFrame(X_train_cv1.toarray(), columns=cv1.get_feature_names_out()).head()
```

	aachen	aafco	abandoned	ability	abit	able	abnormally	abomination	aboutbr	abr	...	ziti	ziwipeak	ziwipeakbr	zoey	zoloft	zone	zoo	zucchinis	zukes	zyrtec
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 14269 columns

```
cv2 = CountVectorizer(ngram_range=(1, 2), binary=True, stop_words='english')

X_train_cv2 = cv2.fit_transform(X_train)
X_test_cv2 = cv2.transform(X_test)

pd.DataFrame(X_train_cv2.toarray(), columns=cv2.get_feature_names_out()).head()
```

	aachen	aachen munich	aafco	aafco regulations	abandoned	abandoned itbr	ability	ability induce	ability make	ability soften	...
0	0	0	0	0	0	0	0	0	0	0	...
1	0	0	0	0	0	0	0	0	0	0	...
2	0	0	0	0	0	0	0	0	0	0	...
3	0	0	0	0	0	0	0	0	0	0	...
4	0	0	0	0	0	0	0	0	0	0	...

5 rows × 112313 columns

Table of Contents

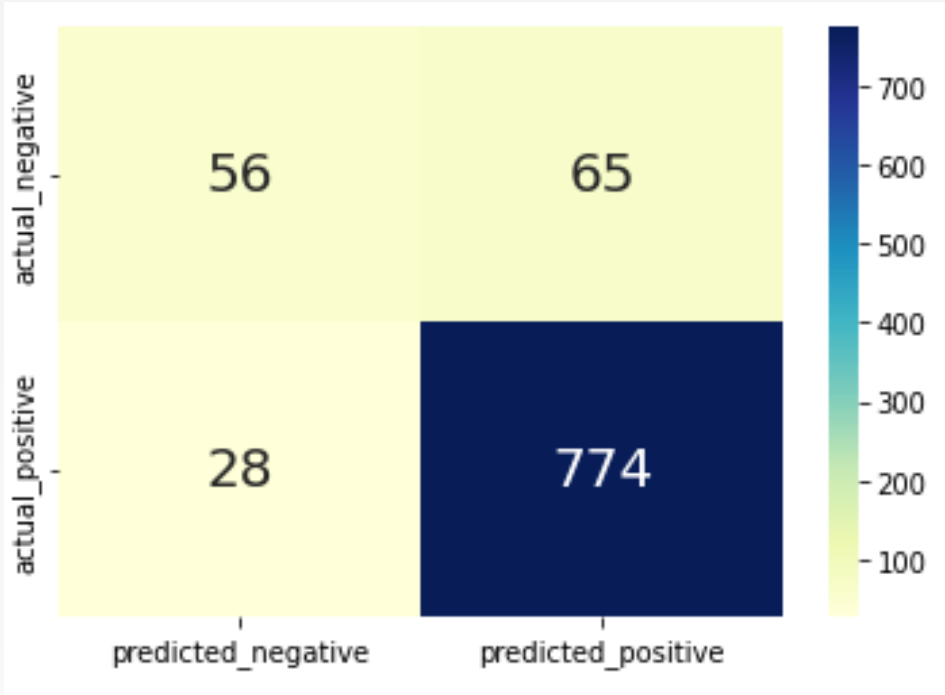
```
# Creating the object for LDA model using gensim library
LDA = gensim.models.ldamodel.LdaModel

num_topics=5

lda_model_filepath = os.path.join('../models/lda_model_prepared_'+str(num_topics))

# Build LDA model
prepare = True
if prepare:
    lda_model_prepared = LDA(corpus=doc_term_matrix, id2word=dictionary, num_topics=num_topics, random_state=42,
                             chunksize=1000, passes=50,iterations=100)
    with open(lda_model_filepath, 'wb') as f:
        pickle.dump(lda_model_prepared, f)

with open(lda_model_filepath, 'rb') as f:
    lda_model = pickle.load(f)
```

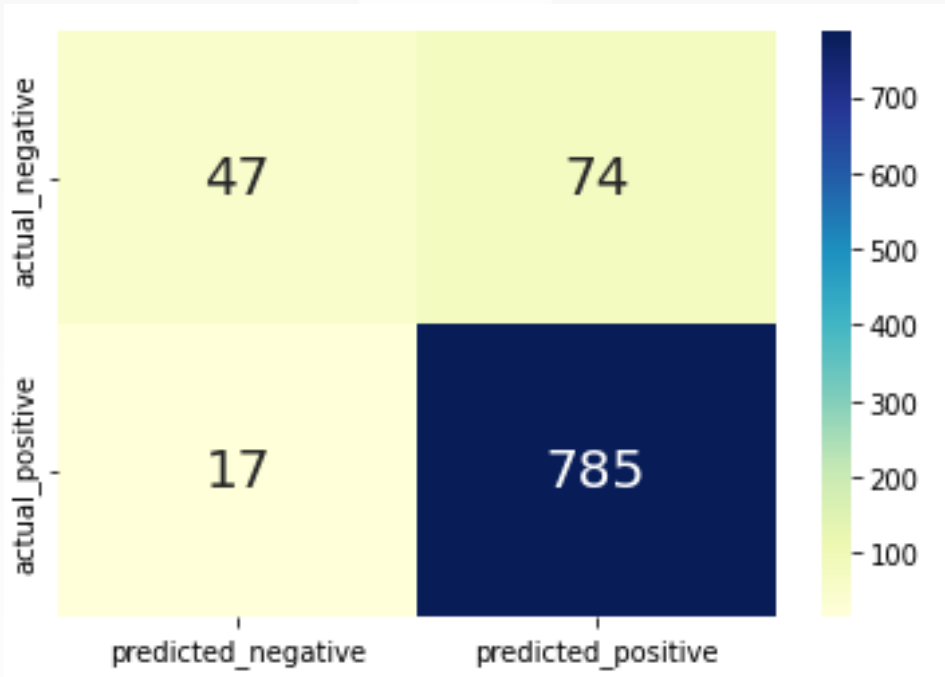
```
lda_model.print_topics()
```

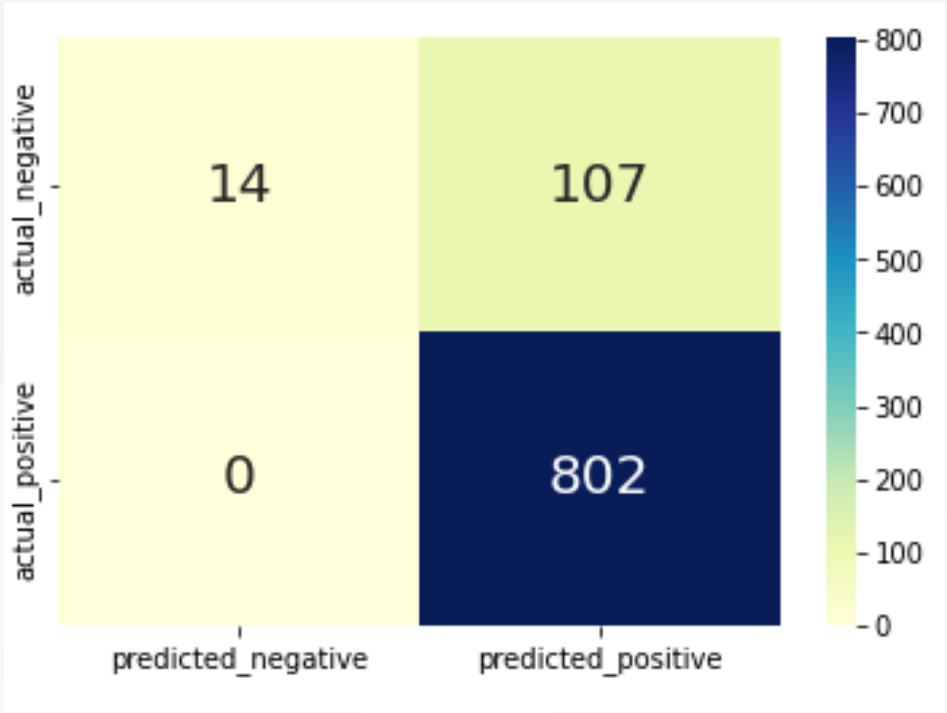
```
[(0,
  '0.049*"tea" + 0.020*"good" + 0.019*"product" + 0.017*"flavor" + 0.015*"water" + 0.014*"taste" + 0.012*"great"
n"'),
 (1,
  '0.029*"good" + 0.020*"flavor" + 0.020*"great" + 0.016*"snack" + 0.015*"chocolate" + 0.014*"bar" + 0.013*"cooki
ther"'),
 (2,
  '0.023*"product" + 0.013*"sauce" + 0.011*"item" + 0.011*"store" + 0.010*"great" + 0.009*"price" + 0.009*"box" +
y"'),
 (3,
  '0.058*"coffee" + 0.020*"cup" + 0.019*"good" + 0.016*"flavor" + 0.011*"strong" + 0.011*"great" + 0.010*"taste"
r"'),
 (4,
  '0.041*"food" + 0.027*"dog" + 0.014*"good" + 0.014*"cat" + 0.013*"product" + 0.010*"other" + 0.008*"great" + 0.
```

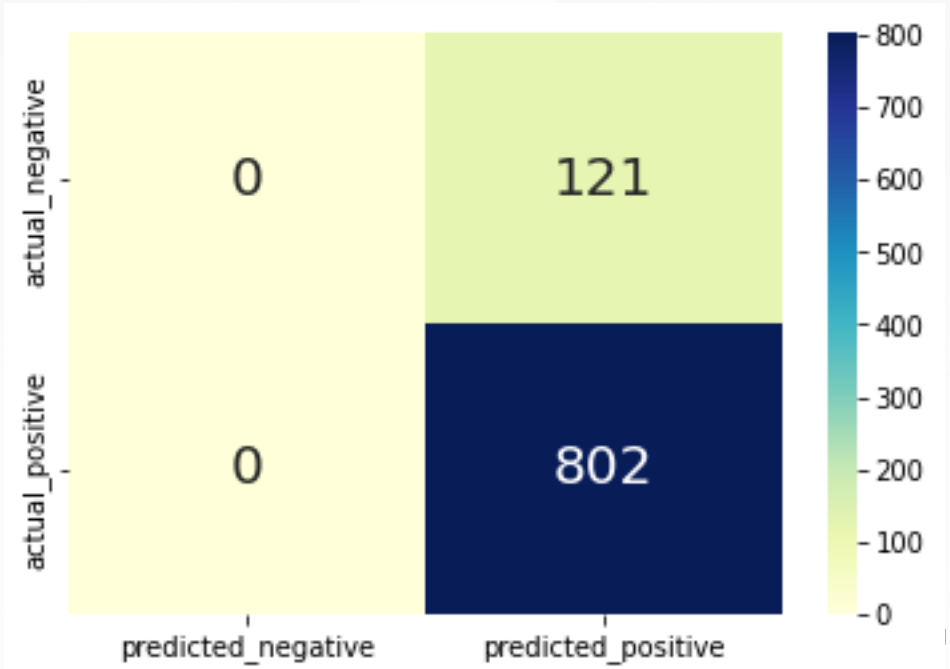
```
pyLDAvis.enable_notebook()
LDAvis_data_filepath = os.path.join('../reports/results/ldavis_prepared_'+str(num_topics))

prepare = True
if prepare:
    LDAvis_prepared = pyLDAvis.gensim_models.prepare(lda_model, doc_term_matrix, dictionary)
    with open(LDAvis_data_filepath, 'wb') as f:
        pickle.dump(LDAvis_prepared, f)# load the pre-prepared pyLDAvis data from disk

with open(LDAvis_data_filepath, 'rb') as f:
    LDAvis_prepared = pickle.load(f)

pyLDAvis.save_html(LDAvis_prepared, '../reports/results/ldavis_prepared_'+ str(num_topics) +'.html')

LDAvis_prepared
```

Table of Contents


56
65
28
774
actual_negative
actual_positive
predicted_negative
predicted_positive
700
600
500
400
300
200
100

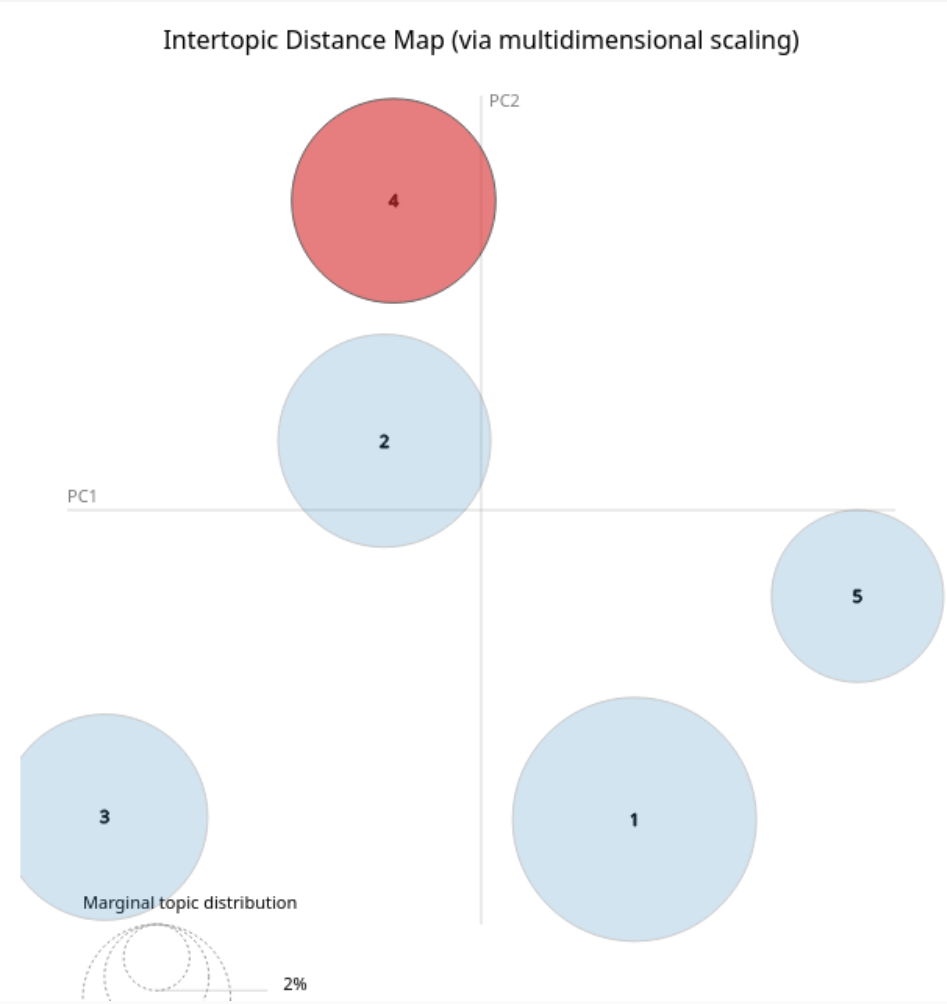

47
74
17
785
actual_negative
actual_positive
predicted_negative
predicted_positive
700
600
500
400
300
200
100

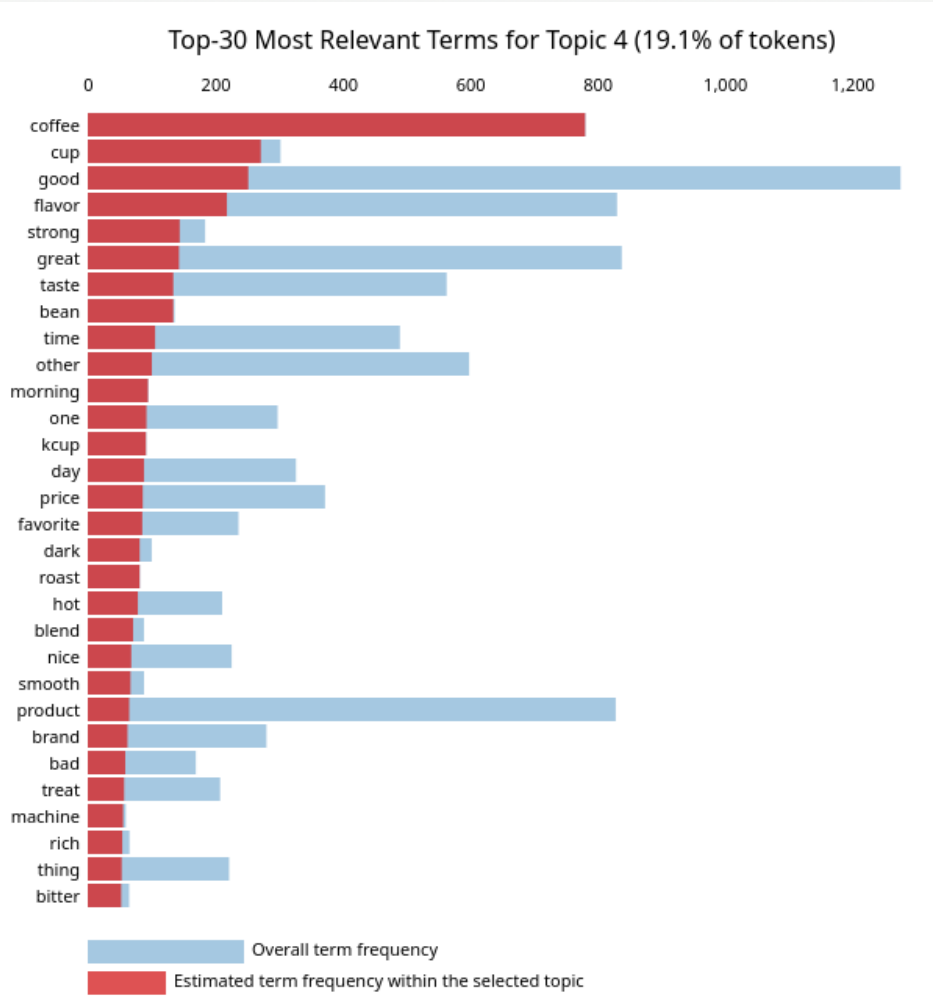

actual_negative
actual_positive
14
107
0
802
predicted_negative
predicted_positive
800
700
600
500
400
300
200
100
0

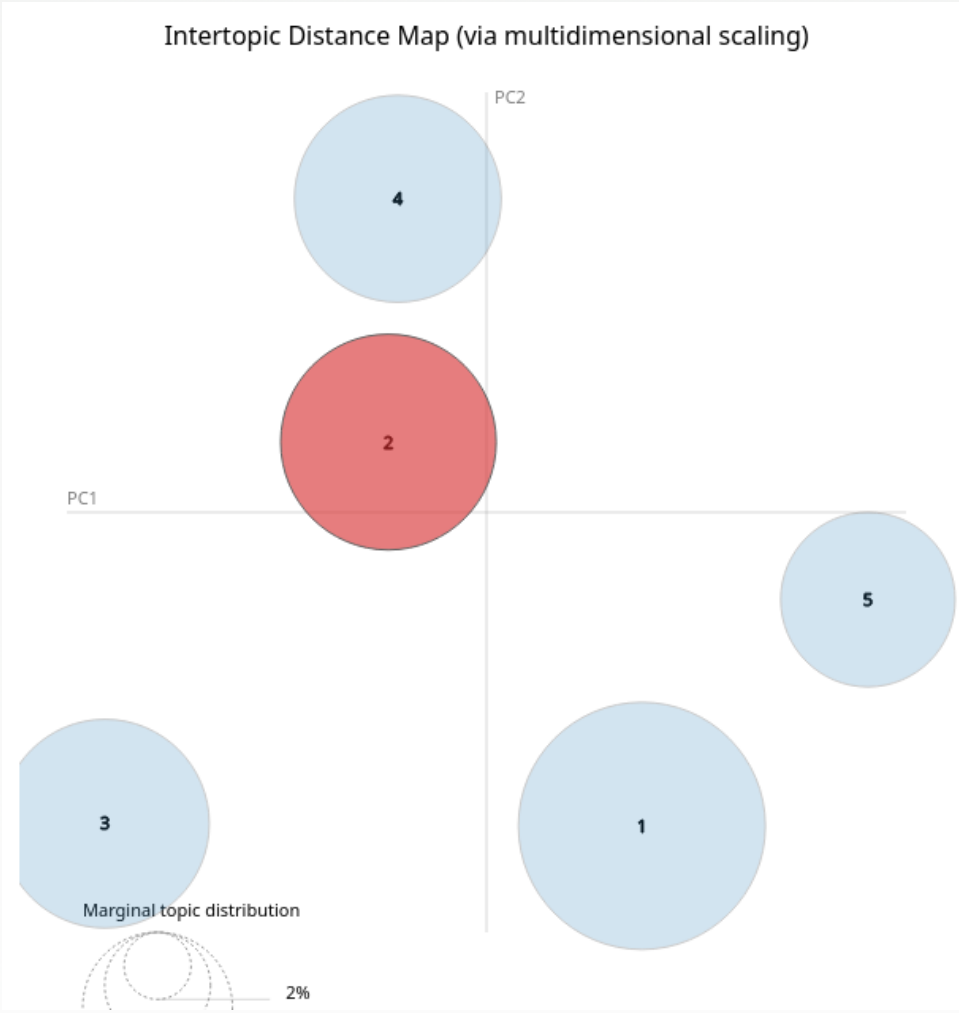

actual_negative
actual_positive
0
121
0
802
predicted_negative
predicted_positive
800
700
600
500
400
300
200
100
0

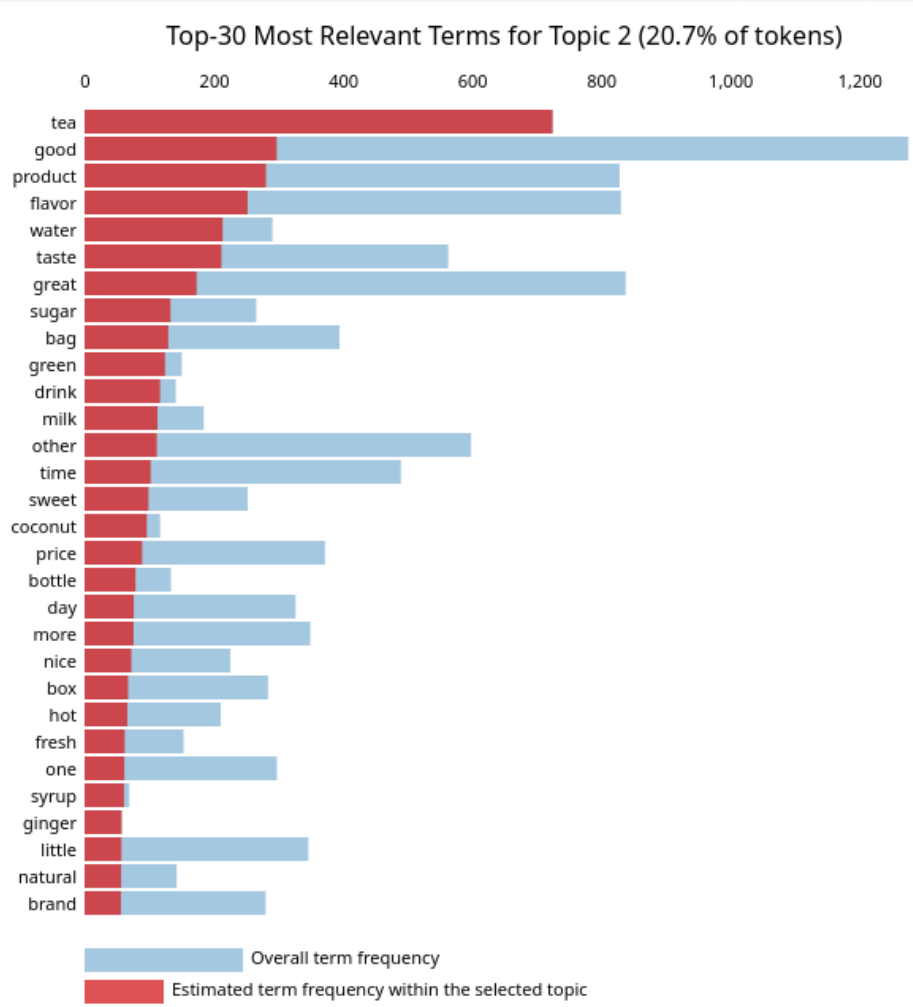
```
results = pd.DataFrame(list(zip(cm1, cm2)))
results = results.set_index([['Accuracy', 'Precision', 'Recall', 'F1 Score']])
results.columns = ['LogReg1', 'LogReg2']

results_nb = pd.DataFrame(list(zip(cm3, cm4)))
results_nb = results_nb.set_index([['Accuracy', 'Precision', 'Recall', 'F1 Score']])
results_nb.columns = ['NB1', 'NB2']
results_nb

results = pd.concat([results, results_nb], axis=1)

results_tf = pd.DataFrame(list(zip(cm5, cm6, cm7, cm8)))
results_tf = results_tf.set_index([['Accuracy', 'Precision', 'Recall', 'F1 Score']])
results_tf.columns = ['LR1-TFIDF', 'LR2-TFIDF', 'NB1-TFIDF', 'NB2-TFIDF']
results_tf

results = pd.concat([results, results_tf], axis=1)
results
```

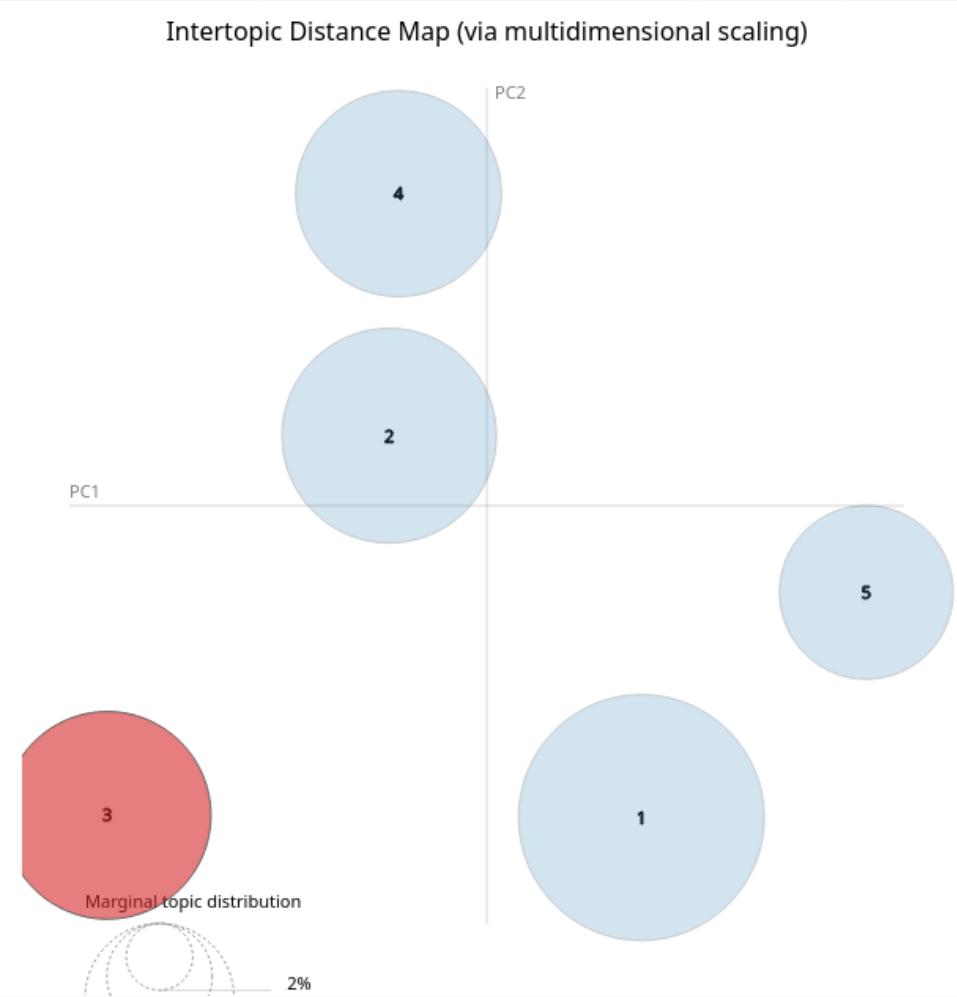
	LogReg1	LogReg2	NB1	NB2	LR1-TFIDF	LR2-TFIDF	NB1-TFIDF	NB2-TFIDF
Accuracy	0.899	0.901	0.888	0.869	0.884	0.869	0.869	0.869
Precision	0.923	0.914	0.892	0.869	0.882	0.869	0.869	0.869
Recall	0.965	0.979	0.991	1.000	1.000	1.000	1.000	1.000
F1 Score	0.944	0.945	0.939	0.930	0.937	0.930	0.930	0.930

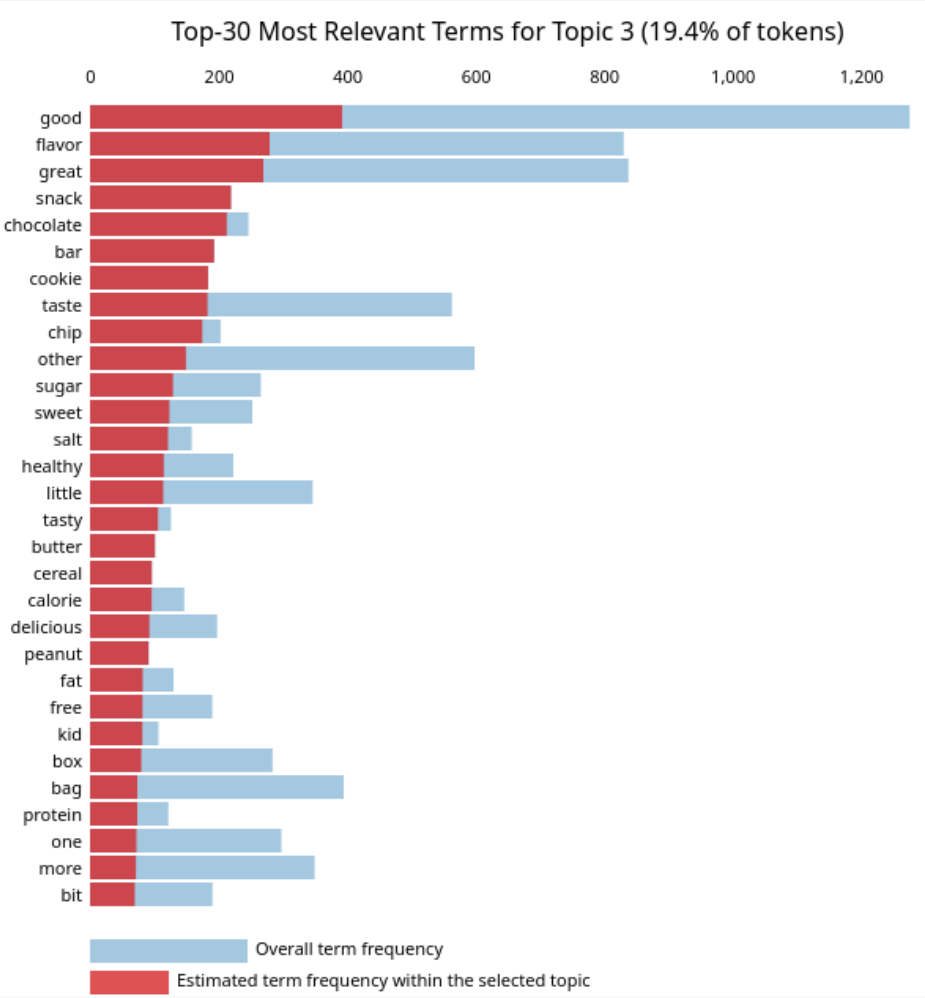

Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
4
2
5
3
1
Marginal topic distribution
2%

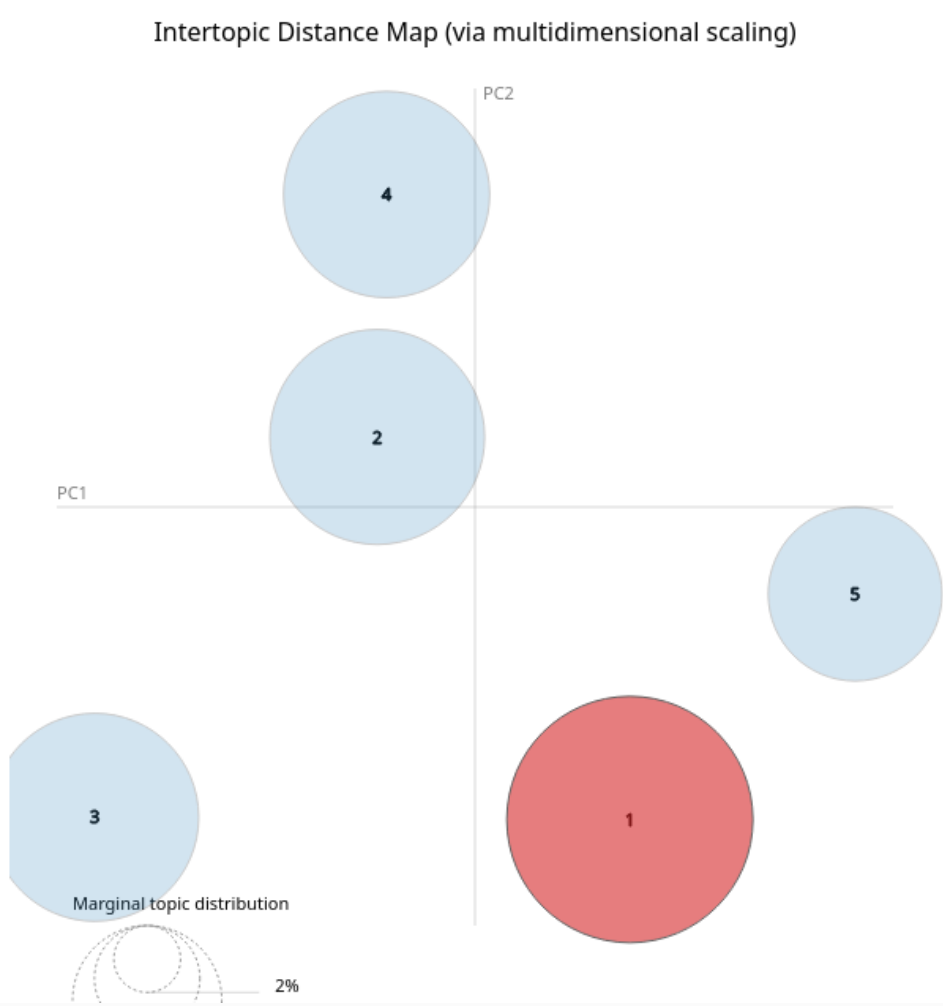

Top-30 Most Relevant Terms for Topic 4 (19.1% of tokens)
0
200
400
600
800
1,000
1,200
coffee
cup
good
flavor
strong
great
taste
bean
time
other
morning
one
kcup
day
price
favorite
dark
roast
hot
blend
nice
smooth
product
brand
bad
treat
machine
rich
thing
bitter
Overall term frequency
Estimated term frequency within the selected topic


Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
4
2
5
3
1
Marginal topic distribution
2%


Top-30 Most Relevant Terms for Topic 2 (20.7% of tokens)
0
200
400
600
800
1,000
1,200
tea
good
product
flavor
water
taste
great
sugar
bag
green
drink
milk
other
time
sweet
coconut
price
bottle
day
more
nice
box
hot
fresh
one
syrup
ginger
little
natural
brand
Overall term frequency
Estimated term frequency within the selected topic


Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
4
2
5
3
1
Marginal topic distribution
2%


Top-30 Most Relevant Terms for Topic 3 (19.4% of tokens)
0
200
400
600
800
1,000
1,200
good
flavor
great
snack
chocolate
bar
cookie
taste
chip
other
sugar
sweet
salt
healthy
little
tasty
butter
cereal
calorie
delicious
peanut
fat
free
kid
box
bag
protein
one
more
bit
Overall term frequency
Estimated term frequency within the selected topic


Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
4
2
5
3
1
Marginal topic distribution
2%

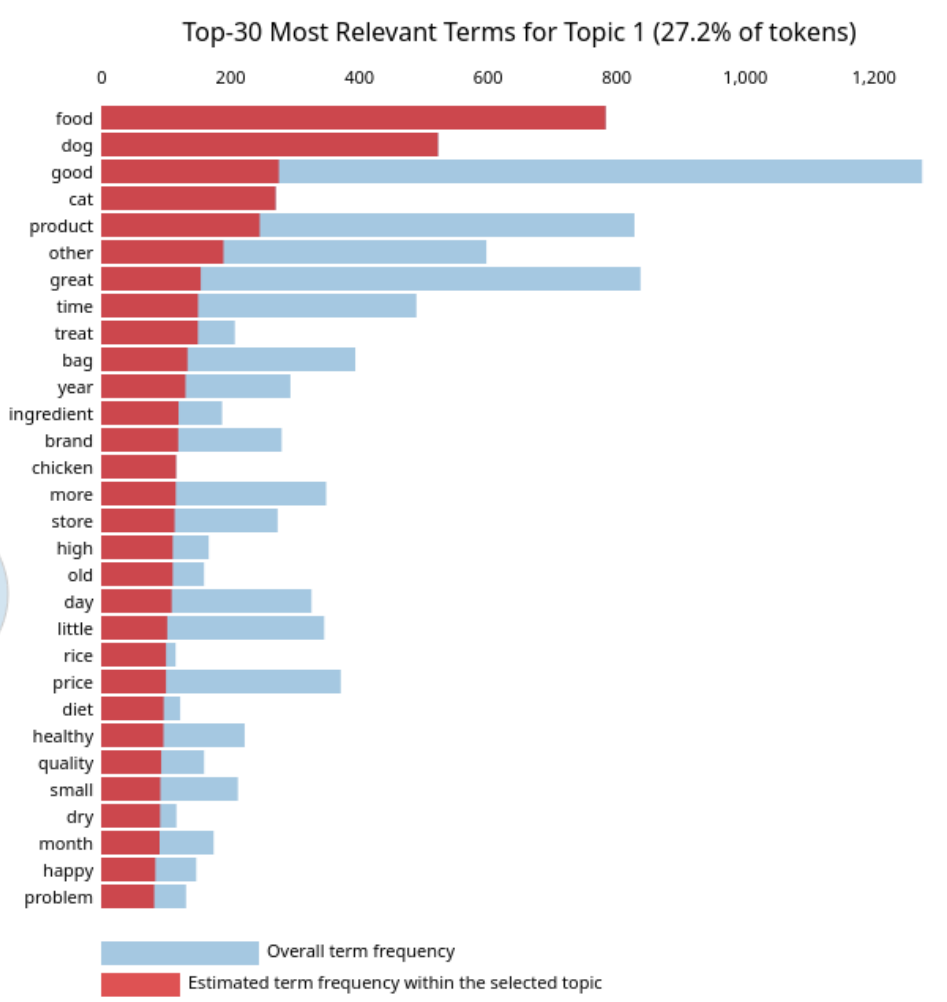

Top-30 Most Relevant Terms for Topic 1 (27.2% of tokens)
0
200
400
600
800
1,000
1,200
food
dog
good
cat
product
other
great
time
treat
bag
year
ingredient
brand
chicken
more
store
high
old
day
little
rice
price
diet
healthy
quality
small
dry
month
happy
problem
Overall term frequency
Estimated term frequency within the selected topic

Table of Contents

Future Work

- Process the whole data with Spark
- Deploy as an application with Flask
- Find the Optimal Number of Topics
- Combine the results from Sentiment Analysis and LDA

Thank you