Amazon Review Analysis & Application

Group 2: Ahjeong Yeom, Allen Dong, Kenji Laurens, Kenneth Jin

Agenda

01 Team Introduction

02 Executive Summary

03 Business Problem

04 Data

- Data Profile
- Implementation Tools
- EDA

05 Business Problem 1: Helpfulness Prediction

06 Business Problem 2: Product Recommendations

07 Business Recommendations

08 Lessons & Future Considerations

Amazon Team

Ahjeong Yeom	Kenji Laurens	Allen Dong	Kenneth Jin
			
Data Scientist	Data Scientist	Data Scientist	Data Scientist

Executive Summary

Overall Business Objective: To help amazon optimize and improve its overall ecommerce experience through the scope of customer and seller

Project Approach 1
Helpful reviews result in better sales performance, less return, and a higher conversion rate.

Our Solution
Predict helpful reviews (0,1) before the review receives any traction and customers' votes on its helpfulness (classification).

Project Approach 2
The recommendation engine encourages purchases by creating personalized shopping experience for customers.

Our Solution
Build collaborative filtering and content-based recommendation using customers' reviews and star ratings.

Business Implementation:
Amazon can better understand how to rank their reviews.
Sellers can identify helpful reviews and use them to improve selling strategies and products.
Amazon can provide better-informed recommendations to customers.

Data Profile

Dataset

1. 60GB TSV Format Data
2. **Description:** Amazon products across major categories with customer reviews, star ratings and helpful votes

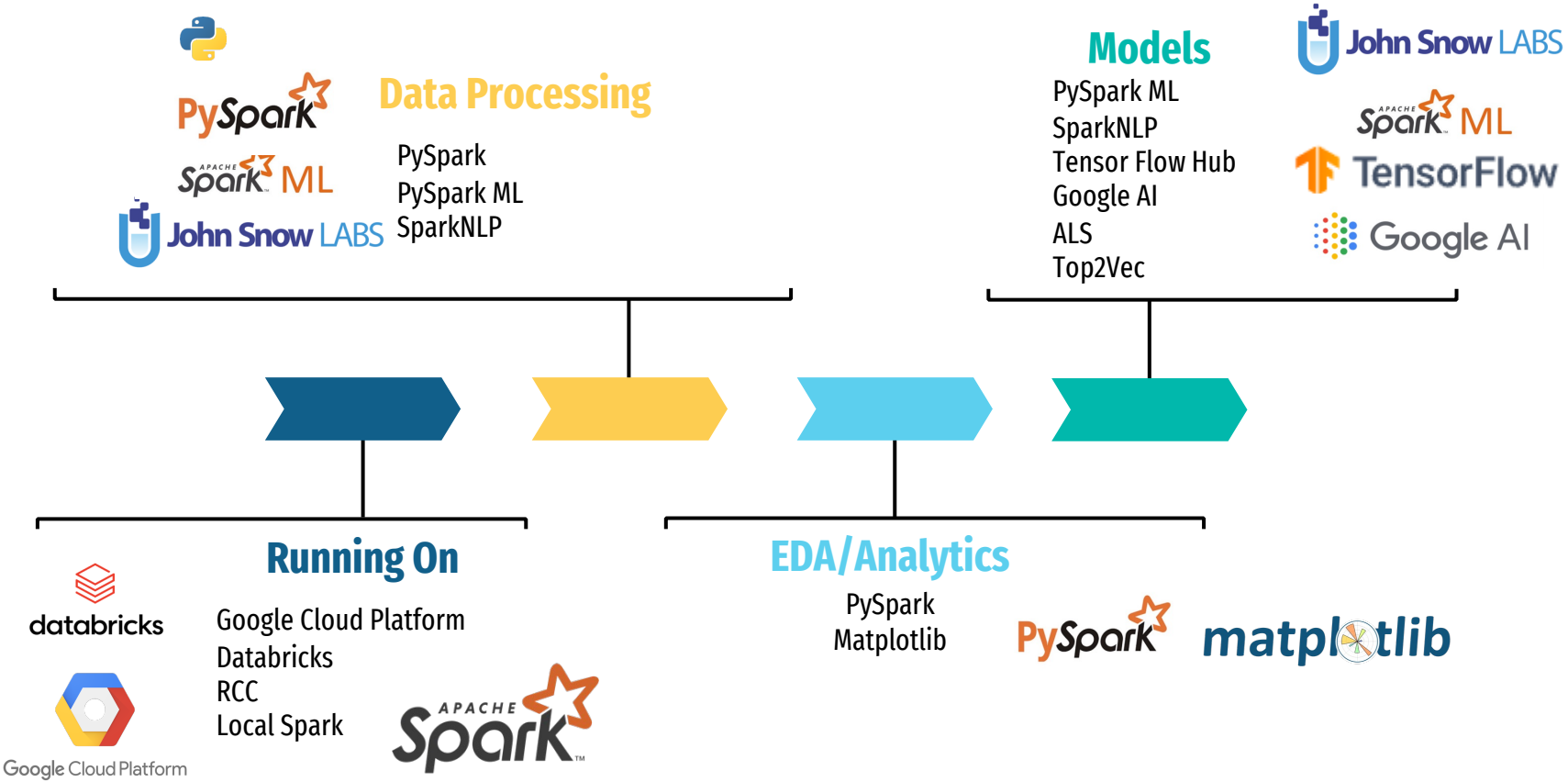
Original Variables

1. marketplace
2. customer_id
3. review_id
4. product_id
5. product_parent
6. product_title
7. product_category
8. star_rating
9. helpful_votes
10. total_votes
11. vine
12. verified_purchase
13. review_headline
14. review_body
15. review_date

Used Variables

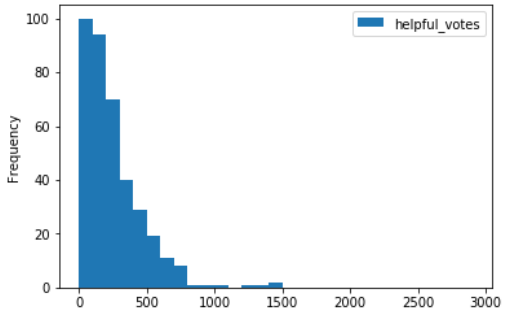
1. review_headline
2. review_id
3. review_body
4. customer_id
5. product_id
6. star_rating
7. helpful_votes
8. total_votes

Implementation Tools

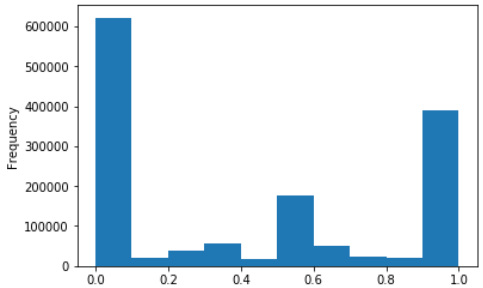


Explanatory Data Analysis

Histogram of Overall Helpful Votes



Histogram of Helpful Ratio

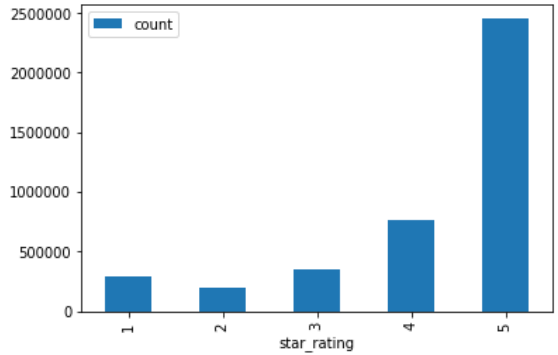


Null Values in Reviews

```
print("No review body:", vid.filter(col('review_body').isNull()).count())
print("No review headline:", vid.filter(col('review_headline').isNull()).count())
print("Total:", vid.count())
```

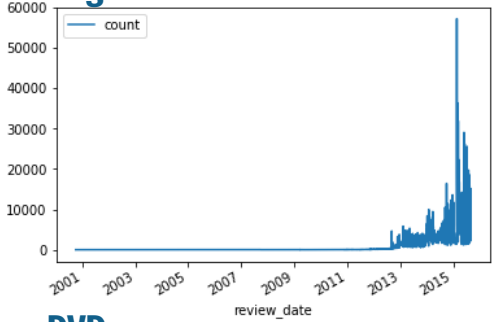
```
No review body: 616
No review headline: 368
Total: 4057147
```

Start Rating Frequency Bar Graph

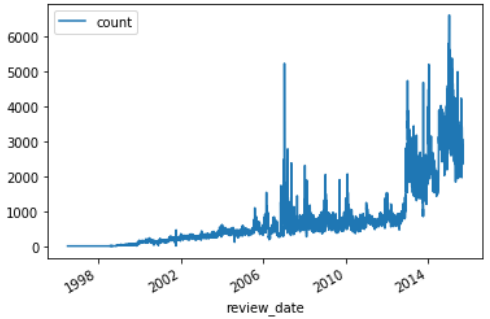


Review Counts Over Time

Digital Video



DVD



Explanatory Data Analysis

Reviews have rich, untapped information for analytics

Understanding what reviews are about

- The breadth and depth of information in reviews is hard to capture
- Cleaned data with **SparkNLP**

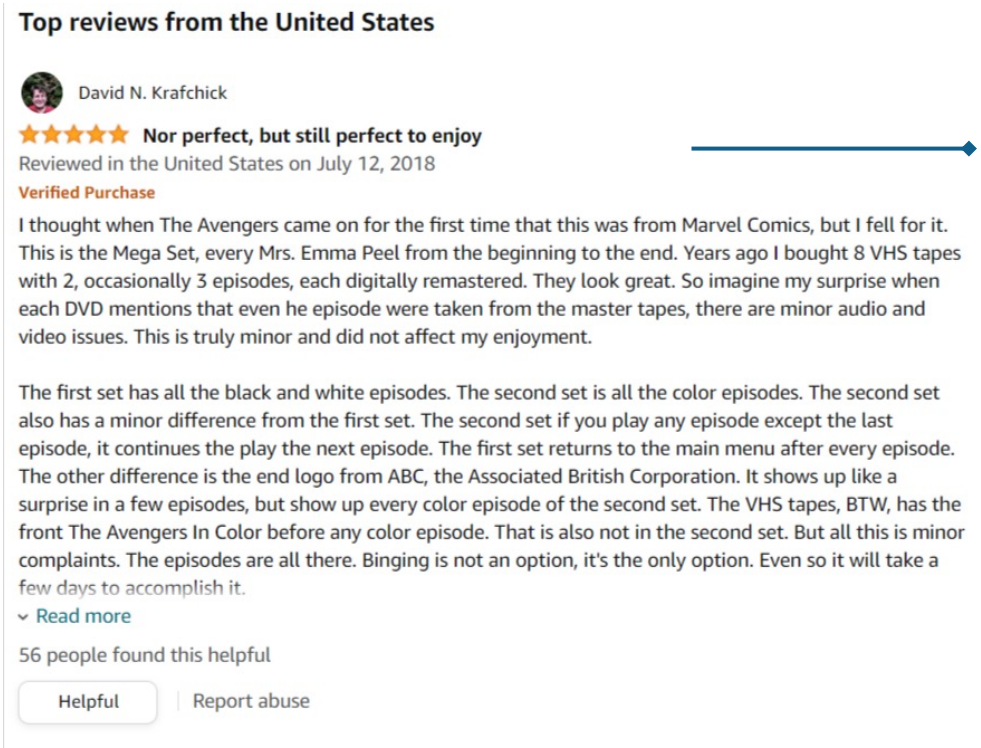
```
docPatternRemoverPipeline = \
  Pipeline() \
    .setStages([
        documentAssembler,
        documentNormalizer,
        sentenceDetector,
        tokenizer,
        stopwords_cleaner,
        finisher])
```

```
cleanUpPatterns = ["<[^>]*>"]
```

Two topic word clouds from digital video dataset

- Ran a topical modelling algorithm, **Top2Vec**, on clean data to check whether we can extract useful information from reviews
- **Top2Vec** returns the number of topics found in text and keywords per topic.
- 191 topics from just digital video dataset and topics are clustered with meaningful keywords per topic
- **Reviews indeed have meaningful information for analysis**

Star Ratings & Helpful Vote Defined



Amazon as of today

Star ratings & helpful votes significantly contribute to making top reviews

Understanding what reviews are helpful

- Customers rely on reviews to purchase products
- There can be many reasons why review is helpful.
- Sellers might have hard time figuring out what reviews are helpful but it is essential to increase sales
- Hence, build an NLP model to classify reviews into helpful(1) or not(0) based on given information: star ratings, review text

Data Preparation: Data Cleanup

Removing NaN

```
df2 = dvd.select([count(when(col(c).contains('None') | \
                             col(c).contains('NULL') | \
                             (col(c) == '' ) | \
                             col(c).isNull() | \
                             isnan(c), c
                            )).alias(c)
                  for c in dvd.columns])
df2.count()
df2.show(5, vertical = True)
```

```
-RECORD 0---------------
 product_id       | 19
 star_rating      | 0
 product_category | 0
 review_headline  | 600
 review_body      | 4460
 helpful_votes    | 0
 total_votes      | 0
```

Get Products With Total Votes > 10

```
dvd = dvd.filter(col('total_votes') > 10)
dvd.count()
```

```
62710
```

Combine Review Headline And Review Body & Create Review Text

```
dvd = dvd.fillna("", "review_body")
dvd = dvd.fillna("", "review_headline")
```

```
dvd = dvd.withColumn('review_text', F.concat('review_headline', F.lit(" "), 'review_body'))
dvd.show(1, vertical = True, truncate = False)
```

```
-RECORD 0-------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-----------------------------------------------
 product_id       | B01489L5LQ
 star_rating      | 4
 product_category | Digital_Video_Download
 review_headline  | Charming movie
 review_body      | This movie isn't perfect, but it gets a lot of things right. Yes, the librari
setting, and the likability of the characters overcome this flaw. The quote at the end brought te
entertaining and thoughtful.
 helpful_votes    | 17
 total_votes      | 18
 review_text      | Charming movie This movie isn't perfect, but it gets a lot of things right. Y
the beautiful setting, and the likability of the characters overcome this flaw. The quote at the
try. It is is entertaining and thoughtful.
only showing top 1 row
```

1. **Review Text:** Combine both headline and review body text to create review text and minimize data loss
2. **Threshold:** Use products that have total_votes > 10 only

Data Preparation: Data Cleanup

Create Helpful Ratio & Helpful

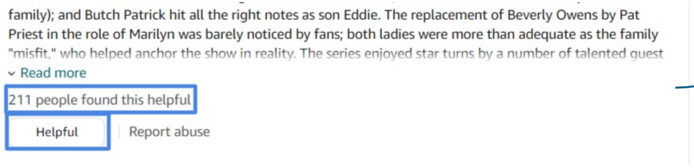
```
dvd = dvd.withColumn('helpful_ratio', F.col('helpful_votes') / F.col('total_votes'))
dvd.show(1, vertical = True, truncate = False)
```

```
-RECORD 0-----------------------------------------------------------------------------
-------------------------------------------------------------------------------------
--------------------------------------------
 product_id       | B01489L5LQ
 star_rating      | 4
 product_category | Digital_Video_Download
 review_headline  | Charming movie
 review_body      | This movie isn't perfect, but it gets a lot of things right. Yes,
setting, and the likability of the characters overcome this flaw. The quote at the en
entertaining and thoughtful.
 helpful_votes    | 17
 total_votes      | 18
 review_text      | Charming movie This movie isn't perfect, but it gets a lot of thi
the beautiful setting, and the likability of the characters overcome this flaw. The q
try. It is is entertaining and thoughtful.
 helpful_ratio    | 0.9444444444444444
only showing top 1 row
```

```
dvd.filter(col('helpful_ratio') < 0).count()
```

```
0
```

```
dvd = dvd.withColumn('helpful', when(col("helpful_ratio") < 0.5, 0).otherwise(1))
```



Further cleanup to remove html tags and irrelevant words

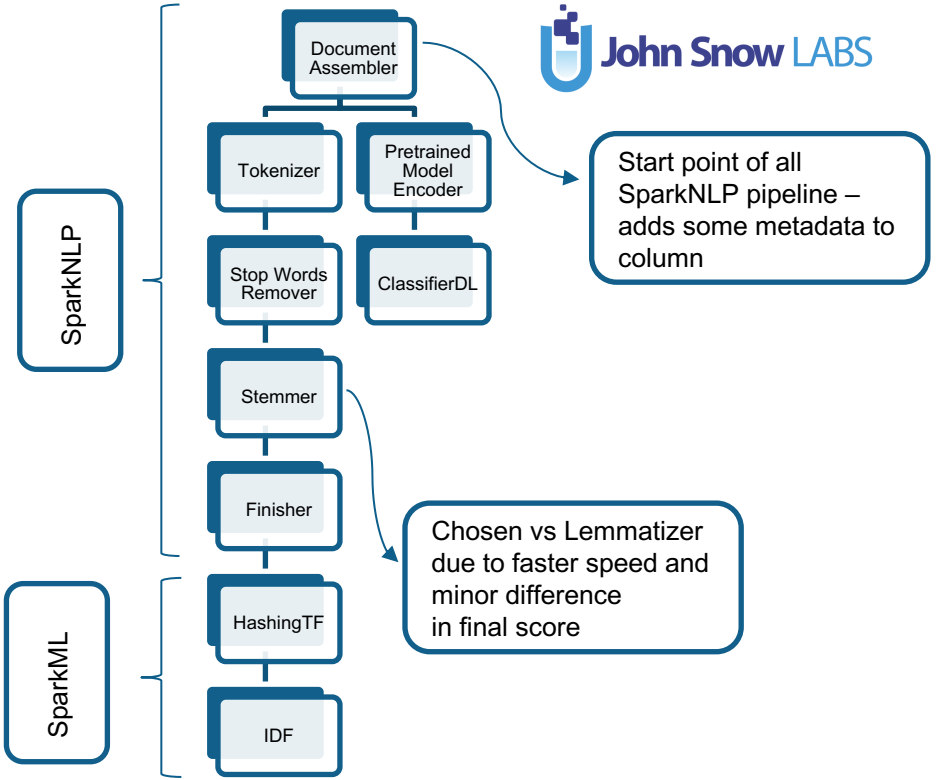
```
from pyspark.sql.functions import col, lower, regexp_replace, split

def clean_text(c):
    c = lower(c)
    c = regexp_replace(c, "\"", "")
    c = regexp_replace(c, "^rt ", "")
    c = regexp_replace(c, "(https?\://)\S+", "")
    c = regexp_replace(c, "[^a-zA-Z0-9\\s]", "")
    c = regexp_replace(c, "<.*?>|&([a-z0-9]+|#[0-9]{1,6}|#x[0-9a-f]{1,6});", "")
    return c
```

1. **Helpful Ratio:** helpful_votes/ total votes
2. **Created Column Helpful:**

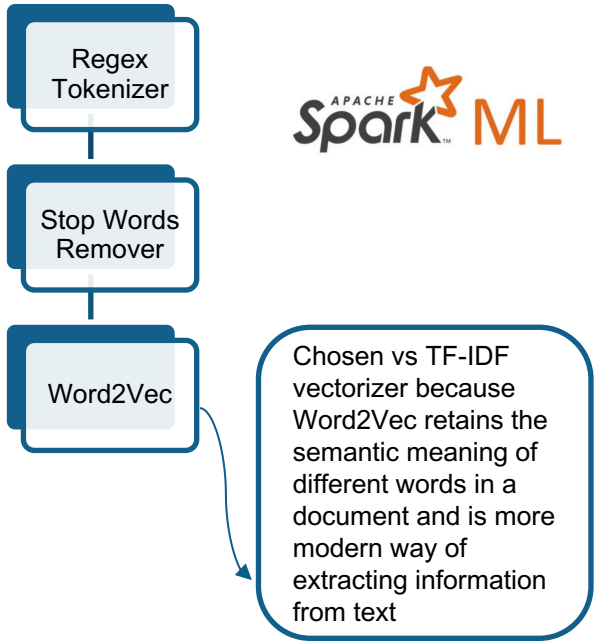
   helpful_ratio < 0.5: not helpful = 0
   helpful_ratio > 0.5: helpful = 1

NLP Pipeline Design & Considerations

Helpfulness Pipeline



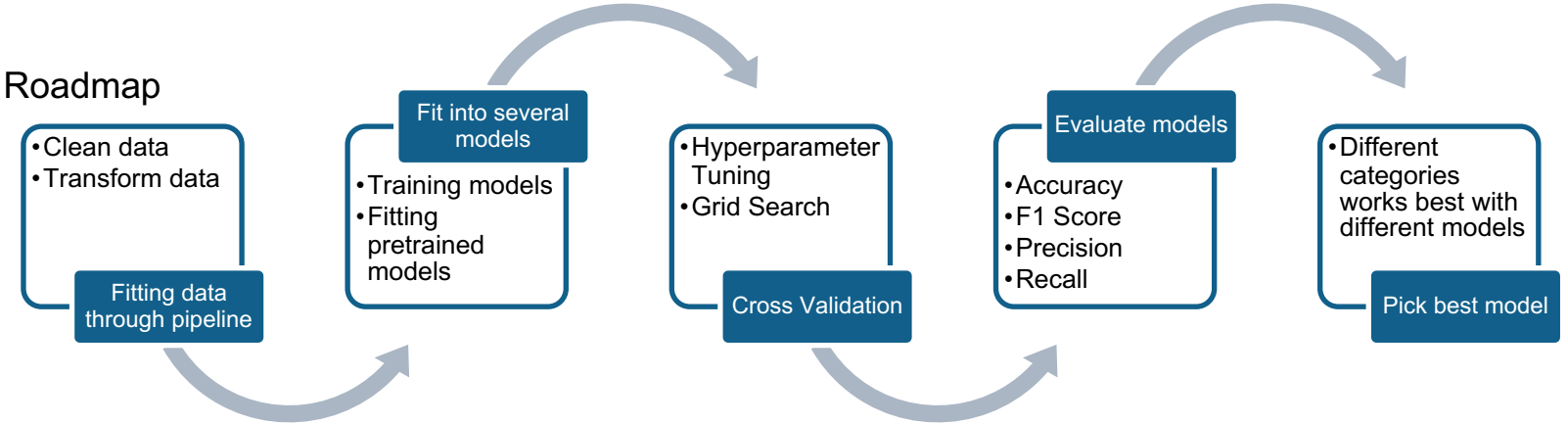
Content-based Recommendation Pipeline



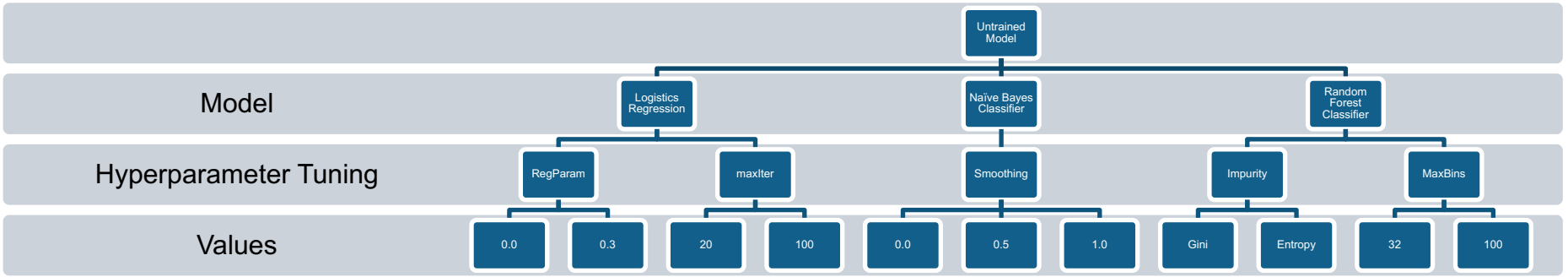
Business Problem 1: Helpfulness Prediction

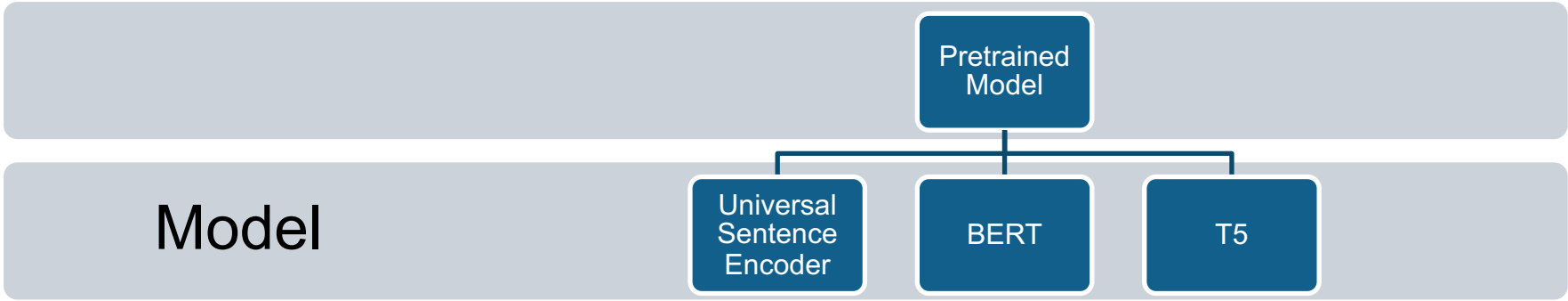
- Amazon reviews that are voted as helpful are assumed to be reviews that people will naturally want to read more of
- Reviews are an important part of the consumer's purchasing decisions. They are just as, if not more important than star ratings that are not descriptive enough
- However, reviews may not have enough helpfulness score if it is new and has yet to gain any traction. Lack of participation and feedback from other purchasers can also lead to insufficient scoring
- Therefore, prediction of review helpfulness before other users even see them can help Amazon push these reviews up and provide a better consumer buying experience
- Classification problems = classifier models; Text data = natural language processing

Roadmap



Models Training





Logistic Regression (1 = Helpful, 0 = Not Helpful)

```
%%time
lr = LogisticRegression(featuresCol = 'features', labelCol='helpful')

paramGrid = (ParamGridBuilder().addGrid(lr.regParam, [0.3, 0.0]).addGrid(lr.elasticNetParam, [0.0]).addGrid(lr.maxIter, [20, 100]).build())

#Evaluator
evaluator = MulticlassClassificationEvaluator(labelCol='helpful', predictionCol="predict

# Create 3-fold CrossValidator
cv = CrossValidator(estimator=lr,estimatorParamMaps=paramGrid,evaluator=evaluator,numFol

cvModel = cv.fit(train)

predictions = cvModel.transform(train)

print(evaluator.evaluate(predictions, {evaluator.metricName: "accuracy"}))
print(evaluator.evaluate(predictions, {evaluator.metricName: "f1"}))
print(evaluator.evaluate(predictions, {evaluator.metricName: "weightedPrecision"}))
print(evaluator.evaluate(predictions, {evaluator.metricName: "weightedRecall"}))
```

```
0.7792051866970825
0.7718823505867335
```

Naive Bayes Classifier (1 = Helpful, 0 = Not Helpful)

```
%%time
nb = NaiveBayes(featuresCol='features', labelCol='helpful')

paramGrid = (ParamGridBuilder().addGrid(nb.smoothing, [0.1, 0.5, 1.0]).build())

#Evaluator
evaluator = MulticlassClassificationEvaluator(labelCol='helpful', predictionCol="prediction")

# Create 3-fold CrossValidator
cv = CrossValidator(estimator=nb,estimatorParamMaps=paramGrid,evaluator=evaluator,numFolds=3)

cvModel = cv.fit(train)

predictions = cvModel.transform(train)

print(evaluator.evaluate(predictions, {evaluator.metricName: "accuracy"}))
print(evaluator.evaluate(predictions, {evaluator.metricName: "f1"}))
print(evaluator.evaluate(predictions, {evaluator.metricName: "weightedPrecision"}))
print(evaluator.evaluate(predictions, {evaluator.metricName: "weightedRecall"}))
```

```
0.7585144275023012
0.7542496802270101
0.7650004125607055
0.7585144275023012
```

```
#Document & Tokenize
document_assembler = DocumentAssembler().setInputCol("review_text_l").se
tokenizer = Tokenizer().setInputCols(["document"]).setOutputCol("review_

#Cleaning Tokens
remover           = StopWordsCleaner().setInputCols("review_words").setO
#lemmatizer        = Lemmatizer().setInputCols(["review_words_stop"]).se
stemmer           = Stemmer().setInputCols(["review_words_stop"]).setOut
finisher          = Finisher().setInputCols(["review_words_lemstem"]).se
#hashingTF         = HashingTF(inputCol="token_features", outputCol="raw
#idf               = IDF(inputCol="rawFeatures", outputCol="features")

pipeline_stem = Pipeline(stages=[document_assembler,tokenizer,remover,stemmer,finisher])#,hashingTF,idf])
```

Running NLP pipeline ¶

```
%%time
df_clean_nlp = pipeline_stem.fit(df_clean).transform(df_clean)
```

```
Wall time: 290 ms
```

Training Model Results

	Models Train/Test Accuracy Scores		
Product Category	Logistic Regression	Naïve Bayes	Random Forest
Books	**0.85 / 0.84**	0.74 / 0.74	0.83 / 0.83
E-Books	0.98 / 0.71	0.81 / 0.71	**0.77 / 0.78**
Music	**0.85 / 0.83**	0.74 / 0.74	0.72 / 0.72
Digital Music	**0.82 / 0.74**	0.74 / 0.66	0.67 / 0.68
DVD	**0.80 / 0.78**	0.70 / 0.70	0.68 / 0.68
Digital Video	0.78 / 0.72	**0.76 / 0.72**	0.66 / 0.65
Software	0.94 / 0.82	0.78 / 0.74	**0.86 / 0.86**
Digital Software	0.99 / 0.84	0.91 / 0.83	**0.82 / 0.83**
Toys	0.99 / 0.84	0.90 / 0.81	**0.89 / 0.89**
Digital Video Games	0.88 / 0.70	0.78 / 0.66	**0.69 / 0.69**
Average	0.89 / 0.72	0.79 / 0.73	**0.83 / 0.76**

Pretrained Model Experiments

- USE
- BERT
- T5

Sentence Encoders & Deep Learning Classifiers

Why are we using sentence encoders?

- Situations when words combined into a sentence holds more valuable information rather than just looking at the importance of one single word.
- In this case, different from word embedding, a sentence embedder uses composition function (mathematical process) of combining multiple words into a single vector
- In our case, we will be using syntactic composition function, rather than treating the corpus with an unordered composition function (bag of words)

Pretrained Universal Sentence Encoder

- From Spark NLP model through TensorFlowHub (Creator = Google)
- Used the version that was trained through DAN (Deep Averaging Network)
- Trained on multiple data sources, can be fed into multiple models for text classification, etc

Pretrained BERT (Bidirectional Encoder Representations from Transformers) Sentence Embedding

- Created by Google Research, highly used today by other companies like Hugging Face to tune and evolve
- Similar to USE but encoding math and trained material is different
- Trained on Wikipedia and BookCorpus

ClassifierDL Approach

- Very first multi-class text classifier in Spark NLP
- Built on the basis of DNN (Deep Learning Model), works like Lego bricks or like a human brain
- In this case, ClassifierDL uses DNN to help us classify embeddings based on the given labels

Universal Sentence Encoder/ Classifier Deep Learning

USE Pipeline

```
#USE (Universal Sentence Encoder) Sentence Embedding
document = DocumentAssembler()\
    .setInputCol("review_text")\
    .setOutputCol("document")

embeddingsSentence = UniversalSentenceEncoder.load('tfhub_use_en_2.4.0_2.4_1587136330099') \
 .setInputCols(["document"])\
 .setOutputCol("sentence_embeddings")

classifierdl = ClassifierDLApproach()\
  .setInputCols(["sentence_embeddings"])\
  .setOutputCol("prediction")\
  .setLabelColumn("helpful")\
  .setMaxEpochs(5)\
  .setEnableOutputLogs(True)

use_clf_pipeline = Pipeline(
    stages = [
        document,
        embeddingsSentence,
        classifierdl
    ])
```

Inserting Review through USE

Running Embedding through ClassiferDL

Train & Test Split

```
train, test = df.randomSplit([0.8, 0.2], seed=12345)
```

```
%%time
pipeline2 = use_clf_pipeline.fit(train)
```

```
Wall time: 33.5 s
```

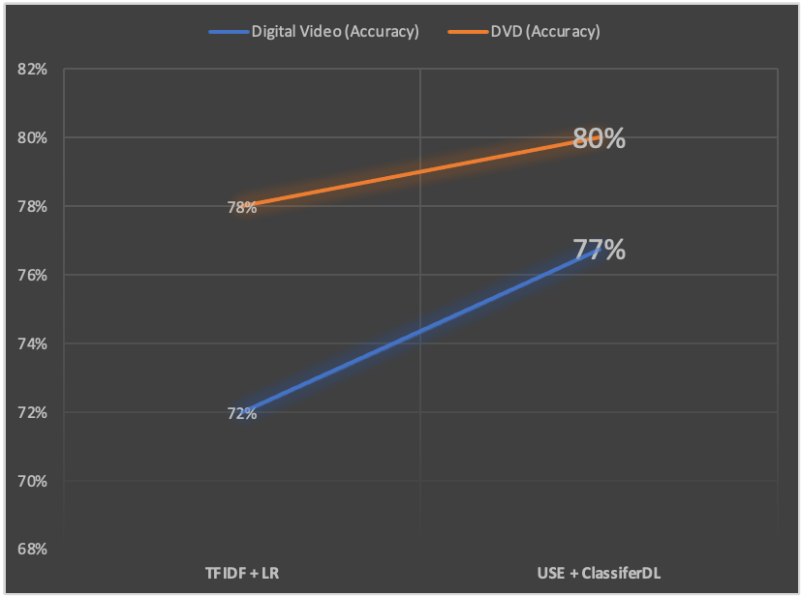
Training & Testing ClassiferDL

```
%%time
final = pipeline2.transform(test)
```

```
Wall time: 47.5 ms
```

```
metrics = final.select('helpful','product_id',"prediction.result")
```

Results



BERT Sentence Embedding/ Classifier Deep Learning

BERT_uncased Embedding Pipeline

```
#BERT Sentence Embedding
document = DocumentAssembler()\
    .setInputCol("review_text")\
    .setOutputCol("document")

bert_cmlm = BertSentenceEmbeddings.load('sent_bert_base_uncased_en_2.6.0_2.4_1598346203624')\
.setInputCols(["document"])\
.setOutputCol("sentence_embeddings")
```

Inserting Review through BERT

```
classifierdl = ClassifierDLApproach()\
  .setInputCols(["sentence_embeddings"])\
  .setOutputCol("prediction")\
  .setLabelColumn("helpful")\
  .setMaxEpochs(5)\
  .setEnableOutputLogs(True)

use_clf_pipeline = Pipeline(
    stages = [
        document,
        embeddingsSentence,
        classifierdl
    ])
```

Running Embedding through ClassiferDL

```
%%time
pipeline2 = use_clf_pipeline.fit(train)
```

```
Wall time: 33.3 s
```

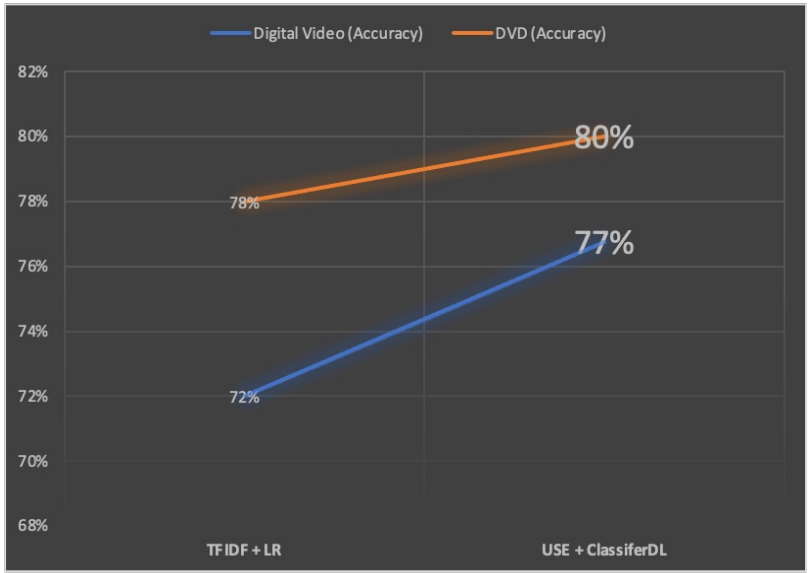
Training & Testing ClassiferDL

```
%%time
final = pipeline2.transform(test)
```

```
Wall time: 36.5 ms
```

```
metrics = final.select('helpful','product_id',"prediction.result")
```

Results



Spark NLP (Light Model) Inference Testing

“Spark NLP LightPipelines are Spark ML pipelines converted into a single machine but the multi-threaded task, becoming more than 10x times faster for smaller amounts of data.” (Veysel Kocaman)

Digital Video

USE Inference

```
In [27]: light_model = LightPipeline(pipeline2)
         #Using a review that was stated Helpful on Amazon
         text="The show is smart and awkwardly, yet deliciously, inappropriate. Miss,
         light_model.annotate(text)['prediction'][0]
Out[27]: '1'

In [52]: #Using a review that has not beed stated Helpful on Amazon YET
         text="I tossed it in the trash. It smelled so bad."
         light_model.annotate(text)['prediction'][0]
Out[52]: '0'
```

BERT Inference

```
In [48]: light_model = LightPipeline(pipeline2)
         #Using a review that was stated Helpful on Amazon
         text="The show is smart and awkwardly, yet deliciously, inappropriate. Miss,
         light_model.annotate(text)['prediction'][0]
Out[48]: '1'

In [53]: #Using a review that has not beed stated Helpful on Amazon YET
         text="I tossed it in the trash. It smelled so bad."
         light_model.annotate(text)['prediction'][0]
Out[53]: '0'
```

DVD

USE Inference

```
In [26]: light_model = LightPipeline(pipeline2)
         #Using a review that was stated Helpful on Amazon
         text="I learned 10 dances on the first disc. Easy dances to follow. Instru
         light_model.annotate(text)['prediction'][0]
Out[26]: '1'

In [37]: #Using a review that has not beed stated Helpful on Amazon YET
         text="No complaints, was good, could be better, not gonna buy again"
         light_model.annotate(text)['prediction'][0]
Out[37]: '0'
```

BERT Inference

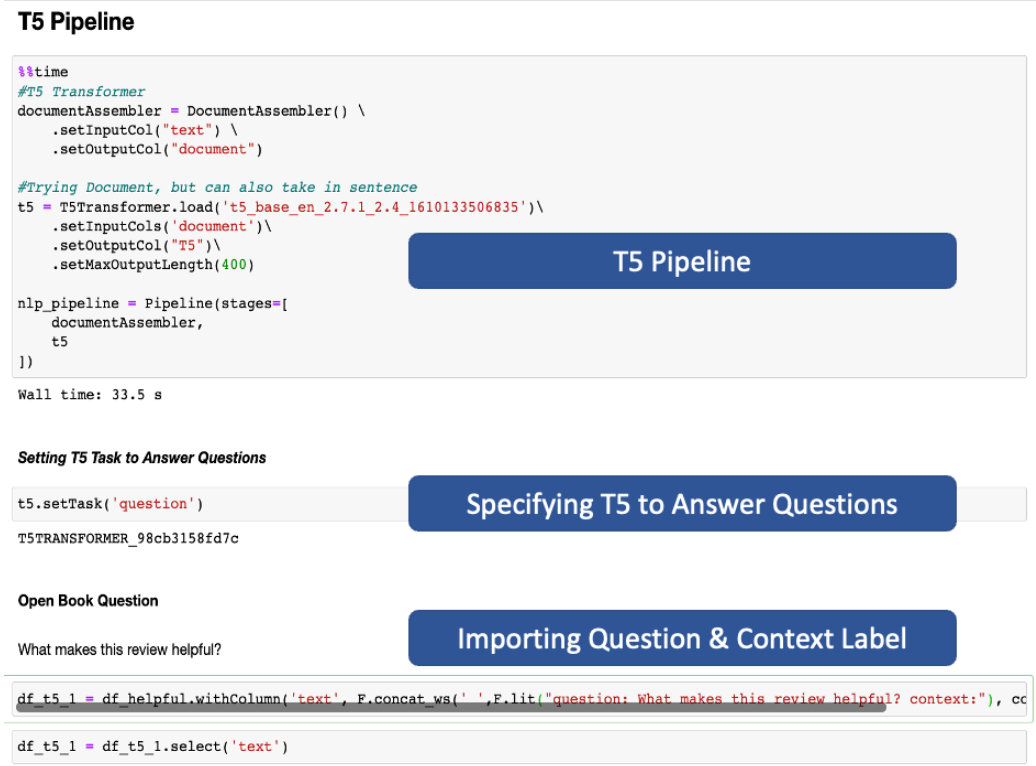
```
In [34]: light_model = LightPipeline(pipeline2)
         #Using a review that was stated Helpful on Amazon
         text="I learned 10 dances on the first disc. Easy dances to follow. Instructor was very
         light_model.annotate(text)['prediction'][0]
Out[34]: '1'

In [38]: #Using a review that has not beed stated Helpful on Amazon YET
         text="No complaints, was good, could be better, not gonna buy again"
         light_model.annotate(text)['prediction'][0]
Out[38]: '0'
```

T5 Encoder-Decoder Model

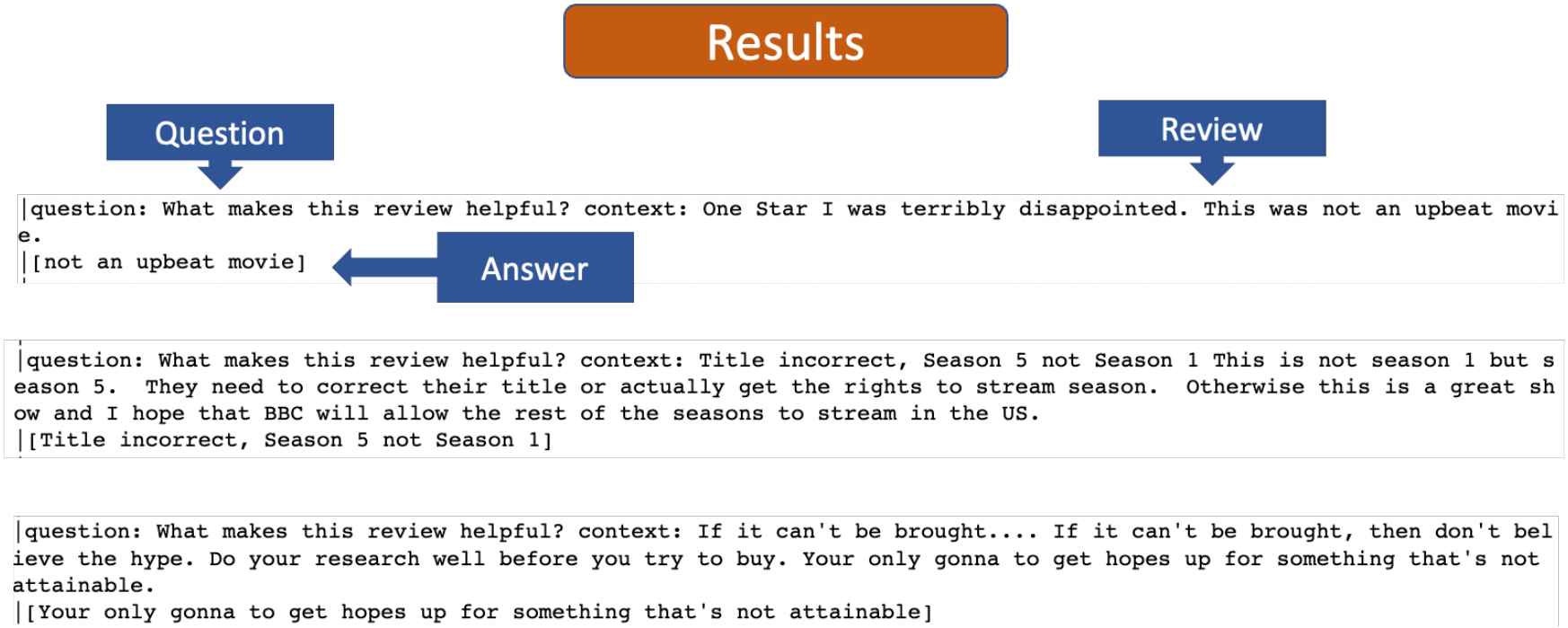
Since we now have a better idea of if a review is helpful or not thus…

New Question!
What makes this review helpful?

- T5 is an NLP model that has the ability to translate, summarize, **in our case we used it to ask questions on the reviews to grab insights for WordClouds**
- "T5 is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks and for which each task is converted into a text-to-text format."
- T5 model is trained on Colossal Clean Crawled Corpus (C4), cleaned version of common crawl, and two magnitudes larger than Wiki
- Created by Google Research

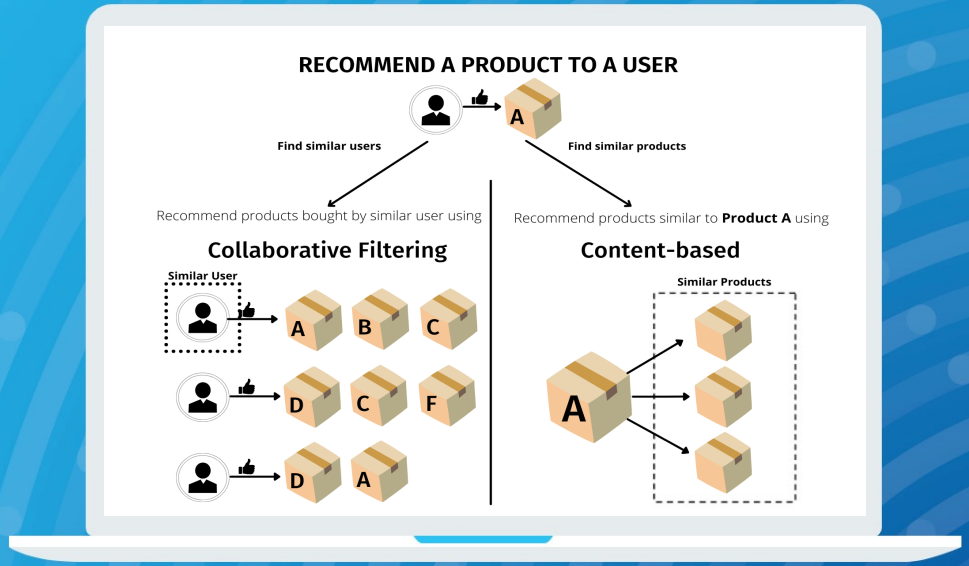


T5 Open-Book Question Answering



Business Problem 2: Product Recommendation

A solid recommendation system will enable Amazon to provide better personalized user experience and promote more relevant products to user through one-to-one marketing



Content Based Recommendation System

Implementation

1. Clean and transform the features with NLP pipeline into vector form
2. Define the similarity function Cosine, Pearson etc.
3. Define function to generate product recommendation for users
4. Find products similar with the products user previously reviewed on
5. Recommend top 10 products excluding previously reviewed products

Advantages

- Depends on product's meta data so doesn't require strong user-base to build recommendations

Disadvantages

- Computationally expensive
- Ignores potentially important factors (ex. Star ratings, user feedbacks)
- No serendipity: only make recommendations based on existing interests of the user

Create recommendations based on products user reviewed before

```
#Find products for a user '52895410'
ProductRecoms(52895410)
```

```
User previously reviewed the following products:
```

	product_id	product_title
0	B01489L5LQ	After Words

```
Recommended products based on user's previous behavior:
```

	product_id	score	product_title
0	B00JGMIU4G	0.904484	In The Blood
1	B00TTRBYGY	0.899473	A Girl Walks Home Alone At Night (English Subt...
2	B00932AA5G	0.890527	Headhunters
3	B00T9DO6RC	0.887819	The Voices
4	B00T9DLXYQ	0.887094	Jupiter Ascending
5	B009O8JSSO	0.884902	28 Hotel Rooms
6	B00LTMIHVW	0.880586	Under the Skin
7	B003E48UMY	0.873963	The Republic of Love
8	B00M7VB4OG	0.873450	Saving Grace
9	B00SY9MEX0	0.871544	Suburban Gothic

Collaborative Filtering Recommendation System

To address some of the limitations of content-based filtering, collaborative filtering uses similarities between users and items simultaneously to provide recommendations.

Option considered:

User based collaborative filtering

- Scalability issue due to the dynamic nature of users
- Require constantly re-training

Item based collaborative filtering

- Popularity bias: recommender is prone to recommender popular items
- Cold-start: new or less-known items have none or very little interactions

Model based collaborative filtering

- Matrix factorization can solve sparse matrix problem
- Recommend less-known movies through explicit and implicit information

Collaborative Filtering Recommendation System

Model based collaborative filtering using Alternating Least Square (ALS)

User – Product – Rating matrix

↓

Matrix factorization – ALS

↓

Fit and train the model

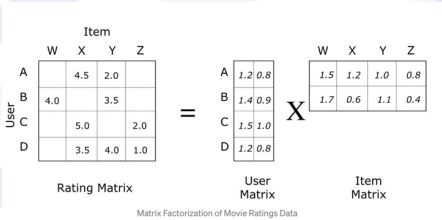
↓

Generate prediction

↓

Evaluation

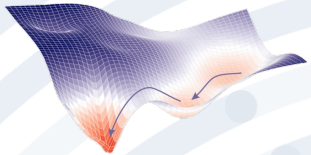
```
+-----------+-------------+-----------+
|customer_id|productIDint|star_rating|
+-----------+-------------+-----------+
|   12190288|          411|          5|
|   30549954|         1203|          5|
|   52895410|        58539|          4|
|   27072354|          291|          5|
|   26939022|        11079|          5|
+-----------+-------------+-----------+
```



Parameter tuning with cross validator

- latent factor, regularization, coldstart

$$\min_{p,q} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2)$$



Stochastic Gradient Descent (SGD) minimizes the objective function

```
als = ALS(userCol="customer_id", itemCol="productIDint", ratingCol="star_rating",coldStartS

param_grid = ParamGridBuilder().addGrid(
    als.rank,
    [10, 30, 50],
).addGrid(
    als.regParam,
    [0.1, 0.5, 1],
).build()

evaluator = RegressionEvaluator(metricName="rmse", labelCol="star_rating",
                                predictionCol="prediction")
cv = CrossValidator(estimator=als, estimatorParamMaps=param_grid, evaluator=evaluator, numF
cv_als_model = cv.fit(train)
```

```
+-----------+--------------------+
|customer_id|     recommendations|
+-----------+--------------------+
|      61051|[[61817, 4.383597...|
|      69637|[[140578, 5.88298...|
|      76493|[[140578, 4.95292...|
|      78120|[[140578, 5.49418...|
|      91446|[[140578, 6.05820...|
|     102960|[[140578, 5.08396...|
|     109613|[[153328, 5.95451...|
|     134748|[[140578, 6.27201...|
+-----------+--------------------+
```

Product Recommendation

- Cross Validator RMSE 1.42 (Rank:50; RegParam: 0.5)

Business Recommendations

- Sellers can identify helpful reviews in advance and use the insights from reviews to optimize their selling strategies by endorsing helpful reviews and use them as part of marketing campaigns.
- Sellers can filter helpful reviews and analyze at a massive scale and derive key insights to improve products and understand target customers.
- Amazon can identify unhelpful reviews, helping them better rank their reviews to improve overall customer satisfaction
- Amazon can provide enhanced recommendations with elements from reviews and ratings combined to buyers to increase sales.

Lessons Learned	Future Improvements
• Differences between Word Embedding NLP vs Sentence Embedding NLP • Created custom transformer for NLP pipelines • Understanding the usage and impact of different methods of text normalization like stop words removal, stemming, and lemmatization • The difference between self-trained NLP pipelines vs pretrained NLP pipelines • The implementation of using ClassiferDL & DNN to predict a binary classification problem • Infrastructures on PySpark setups in GCP, Databricks, RCC, and local devices • Understanding the usage of GPU's on model running • The importance of acknowledgement of optimization and tuning to make overall processes to run more efficiently • Understanding parameters of models and better feature selection for different business problems • Understand the difference of content based and collaborative filtering recommendation system and their implementation • Considerations: bots upvoting reviews can mess up the system	• Being able to obtain stronger computational machines to output results of T5 into a Word Cloud • Instead of using pretrained NLP pipelines for BERT & T5, train these models ourselves to better define final prediction results • Apply topical modelling into content-based recommendation • To predict additional information within reviews, such as it being fake or intentionally malicious • Before training for helpfulness prediction, improve the quality of the reviews by weeding out fake reviews • Build a hybrid recommendation system

Thank you

References

< Data Sources >
Amazon: https://www.kaggle.com/cynthiarempel/amazon-us-customer-reviews-dataset?select=amazon_reviews_us_Wireless_v1_00.tsv

< Tools Used >
PySpark: https://spark.apache.org/docs/latest/api/python/
SparkNLP: https://nlp.johnsnowlabs.com/
- **Universal Sentence Encoder:** https://nlp.johnsnowlabs.com/2020/04/17/tfhub_use.html
- **BERT:** https://nlp.johnsnowlabs.com/docs/en/transformers#bertsentenceembeddings
- **T5:** https://nlp.johnsnowlabs.com/2020/12/21/t5_small_en.html
- https://medium.com/geekculture/understanding-and-analyzing-deep-neural-networks-a2a7ef737511#:~:text=Conclusion%3A%20DNNs%20are%20powerful%20algorithms,output%20of%20the%20previous%20layer.
- https://medium.com/tech-that-works/deep-averaging-network-in-universal-sentence-encoder-465655874a04
- https://towardsdatascience.com/named-entity-recognition-ner-with-bert-in-spark-nlp-874df20d1d77
- https://github.com/JohnSnowLabs/spark-nlp-workshop/blob/master/tutorials/blogposts/3.NER_with_BERT.ipynb
- https://towardsdatascience.com/transfer-learning-in-nlp-fecc59f546e4
- https://ahmetemin-tek.medium.com/text-classification-implementation-in-sparknlp-by-using-universal-sentence-encoder-fd244d175c5b
- https://deepnote.com/@oliver-ngo-1647/Spark-NLP-for-Natural-Language-Processing-with-Disaster-Tweets-kZCXazF9TdKFoa6ocQmhCA

SparkML: https://spark.apache.org/docs/1.2.2/ml-guide.html
Top2Vec: https://github.com/ddangelov/Top2Vec
ALS:https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-2-alternating-least-square-als-matrix-4a76c58714a1
Content Based: https://github.com/Samimust/my-yelper