Project Proposal By – Siddiq Syed

TOPIC MODELING ON AIRBNB REVIEWS IN SPARK

Abstract:

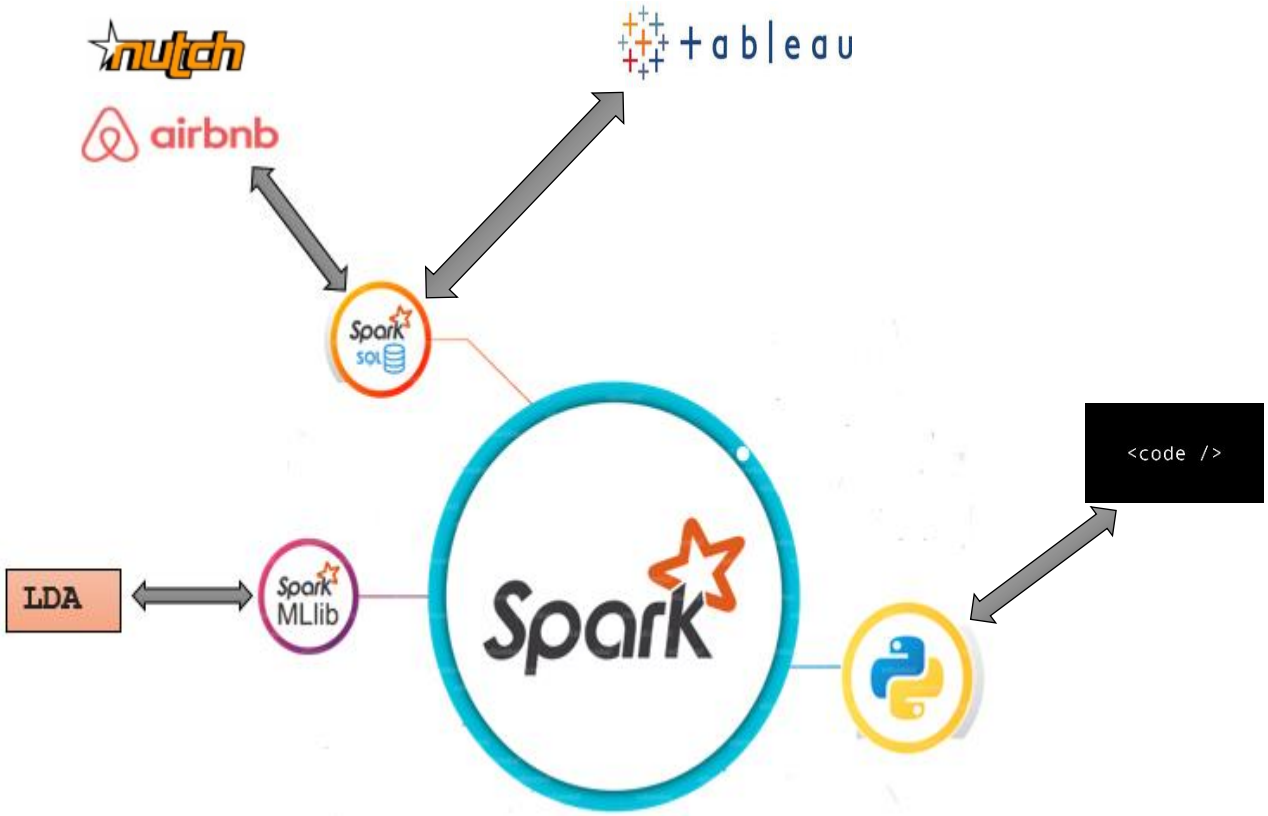
To answer why topic modeling on Airbnb reviews is that the Amazon website already has this feature available. This model will be useful for any dataset that has words. When a user wants to read the reviews of something, here it is a house, the user gets to have some keywords as suggestions on the available reviews. This suggestion will help the user to save some time and awards the hosts in providing excellent customer service. For small datasets, this will be modeled in python or R. But If a particular site has a large number of customers and reviews gets added every hour/minute Spark serves the best. To model, the data will be processed using LDA (Latent Dirichlet Distribution) modeling from the mllib library available in spark.

Problem Statement:

- Suppose we take a house on Airbnb for which we are interested in renting. We first see the images and then see the reviews before renting.
- Now let's take a home that has over 1000 reviews. A user will not be able to get to read all the reviews in some time constraints.
- Also, a user wants to know about the landlord or the neighborhood specifically.
- In such case, if the site can display all the possible topics that a customer is looking for it will help the Airbnb site to have more bookings.
- For this to build topic modeling on Airbnb reviews, reviews need to be collected, and the same will be collected through web scrapping using Sparkler API.
- For the collected reviews data cleaning needs to be performed and will be feed to the mllib – LDA.train function.
- The function returns the list of topics and keywords under the respective topics. Co-occurring keywords can be removed depending on the number of times they have occurred in topics.
- The above-obtained words will be used to display in the Airbnb site. The same will help the user/customers to review a house easily and quickly.

- Furthermore, on the Airbnb site, there are two types of landlords – Hosts and Super hosts. The Super hosts are always given preference, and for them, particular topics can be suggested while for hosts general suggestion can be made.

Methodology and Pipeline Architecture Diagram:



SPARK ECOSYSTEM:

Our Spark ecosystem has five important components: Spark core, Spark SQL, MLlib, Taleau integration, and Python integration.

1. **Spark Core**
   Spark Core is the base engine for large-scale parallel and distributed data processing. Further, additional libraries which are built on the top of the core allows diverse workloads for streaming, SQL, and machine learning.

2. **Spark SQL**
   Spark SQL is a new module in Spark which integrates relational processing with Spark's functional programming API. We will be processing the pandas data-frame in spark SQL. We have extracted the data from AirBnb website using we scraping and store the data using Spar SQL

3. **MLlib (MachineLearning)**
   MLlib stands for Machine Learning Library. Spark MLlib is used to perform machine learning in Apache Spark. We have used the LDA (Latent Dirichlet Allocation) package in the clustering submodule of the library. We installed the package using pyspark.mllib.clustering.LDA command.

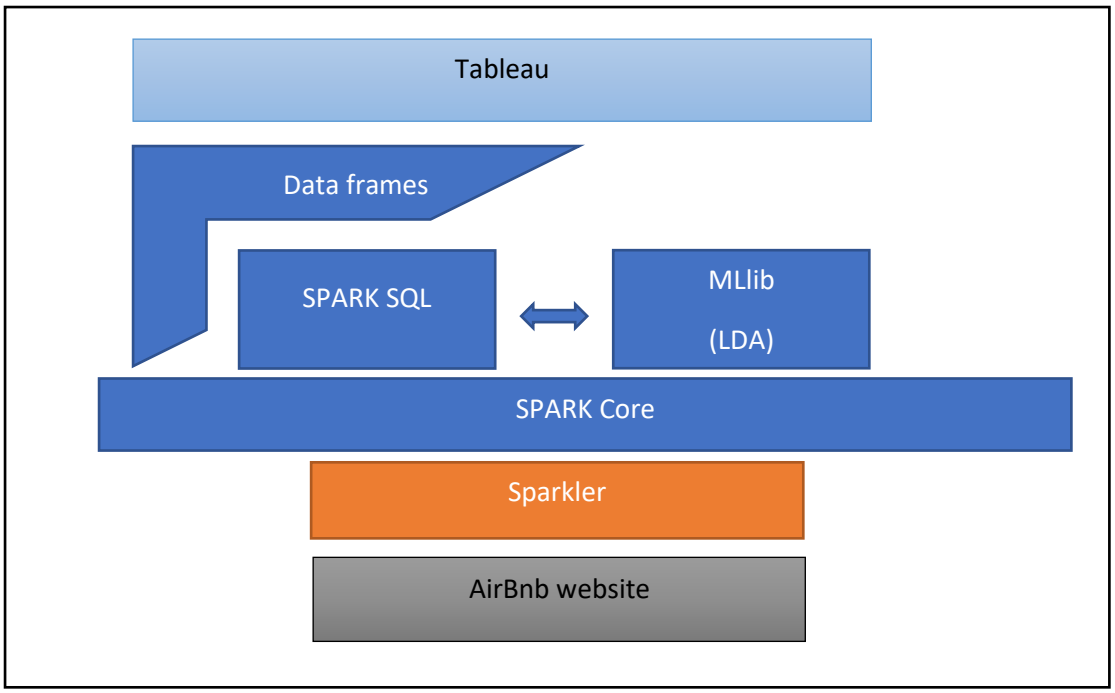
4. **API- SparklerSparkler—Crawler on Apache Spark**

We will be using a Sparkler – an open source web crawler to extract the data from AirBnb website. It provides streamed crawl content through apache kafka and parses everything using apache tika.

5. **Tableau integration:**

Integration with Tableau provides new capabilities - users can visually analyze data without writing Spark SQL code. That's a big deal because creating a visual interface to your data expands the Spark technology beyond data scientists and data engineers to all business users.

AirBnb Topic modeling block-diagram



Spark Architecture:

At the core of our architecture is spark core component. It interacts with Sparkler which is an open source web crawler. Sparkler fetches data from the AirBnb website in batches processed using Apache Kafka and parsed using Apache Tika. Data is stored using SparkSQL and then we perform the data cleaning and model building using LDA package and text cleaning packages of MLlib. For visualizing our output we use Tableau's connector to spark which provide an efficient data engine to visualize the data without writing a Spark query.

Data:

We are using Airbnb reviews by users in their website. To fetch the data we are using Apache Sparkler. Sparkler is a distributed crawler that can scale horizontally. It is easy to deploy and use. The output from the Sparkler API looks like below:

listing_id	id	date	reviewer_	reviewer_	comments
3781	37776825	7/10/2018	36059247	Greg	The apartment was as advertised and Frank was incredibly helpful through the entire process. I would definitely recommend this place
3781	41842494	8/9/2018	10459388	Tai	It was a pleasure to stay at Frank's place. The place has everything you need to make it a home base to visit or work downtown. It is clo

For a listing we have all the review comments. The size of the data is 175 GB for all the US cities listings.

Solution:

From the available four options (Scala, Java, Python and R) this project works good with Pyspark as it has required libraries built in for data cleaning.

```
>>> import pandas as pd

>>> import pyspark
>>> from pyspark.sql import SQLContext
>>> sqlContext = SQLContext(sc)
>>> from nltk.corpus import stopwords
>>> import re as re
>>> from pyspark.ml.feature import CountVectorizer , IDF
>>> from pyspark.mllib.linalg import Vector, Vectors
>>> from pyspark.mllib.clustering import LDA, LDAModel
>>> data = sqlContext.read.format("csv") \
...     .options(header='true', inferschema='true') \
...     .load(os.path.realpath("/home/cse587/Downloads/reviews.csv"))
>>> reviews = data.rdd.map(lambda x : x['Review Text']).filter(lambda x: x is not None)
>>> StopWords = stopwords.words("english")
>>> tokens = reviews                                                    \
...      .map( lambda document: document.strip().lower())              \
...      .map( lambda document: re.split(" ", document))          \
...      .map( lambda word: [x for x in word if x.isalpha()])          \
...      .map( lambda word: [x for x in word if len(x) > 3] )          \
...      .map( lambda word: [x for x in word if x not in StopWords])    \
...      .zipWithIndex()
```

In the above code it is seen that libraries like pandas, pyspark, SQLContext, nltk.corpus, re, pyspark.ml.feature(CountVectorizer , IDF) , pyspark.mllib.linalg (Vector, Vectors) , pyspark.mllib.clustering (LDA, LDAModel) are used.

Data is loaded into data variable from csv which contains all reviews in one column. Each review is then cleaned and tokenized.

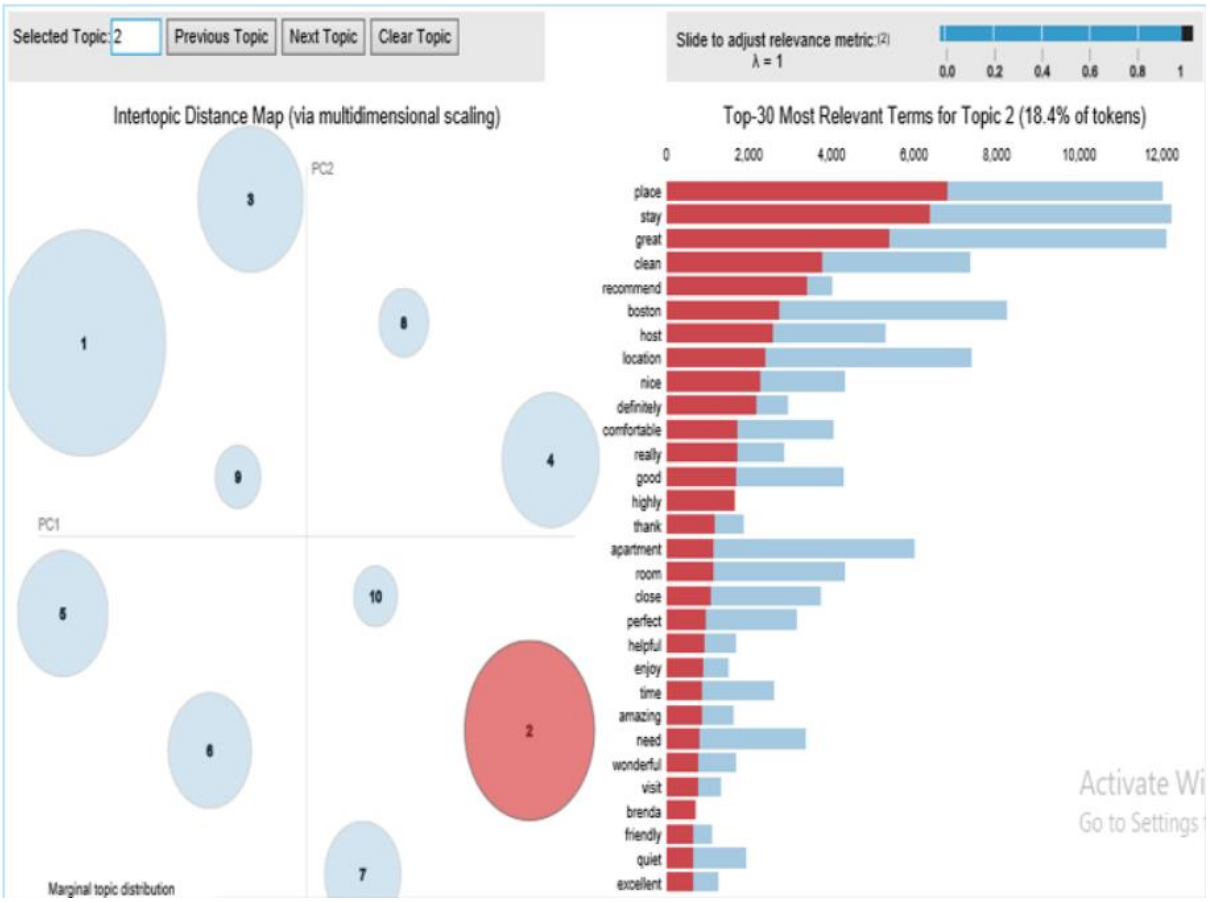
```
>>> df_txts = sqlContext.createDataFrame(tokens, ["list_of_words",'index'])
>>> cv = CountVectorizer(inputCol="list_of_words", outputCol="raw_features", vocabSize=5000, minDF=10.0)
>>> cvmodel = cv.fit(df_txts)
>>> lda_model = LDA.train(result_tfidf[['index','features']].map(list), k=num_topics, maxIterations=max_iterations)
```

Above where LDA.train function is used to build the model for which the inputs are provided from TF-IDF matrix which is obtained from the tokens in the above code.

From the output i.e. lda_model we can look at the topics and their top keywords. Further reference for the code is provided below in references.

Outcomes and Visualizations:

Possible outcome from LDA model can be visualized using Tableau. Below is screenshot of the same.



In the above screenshot explains about the LDA output that has 10 topics. Under these 10 topics there are keywords that are ranked according to their occurrence in the reviews.

Summary:

Reviews are collected from API and review content is stored in a csv in one column. Data is loaded to Spark SQL data frames and using python packages same is cleaned for stop words and other symbols. The data is then transformed to TF-IDF matrix which is provided as input to the mllib LDA function. The function returns topics and list of words which can further be used to display on the website.

References:

Amazon link where the product has topics suggested above reviews - https://www.amazon.com/gp/product/B075QJSQM5?pf_rd_p=f3acc539-5d5f-49a3-89ea-768a917d5900&pf_rd_r=ZX7ASNJWR598XKZ83E38&th=1

Airbnb link where for the reviews there is no suggestions available - https://www.airbnb.com/rooms/24847816?location=Boston%2C%20MA%2C%20United%20States&adults=2&guests=1&toddlers=0&check_in=2019-05-17&check_out=2019-05-19&source_impression_id=p3_1557087432_DsfqmUpAchIKkc3E

Sparkler- Crawler on Apache Spark: https://www.slideshare.net/SparkSummit/sparklercrawler-on-apache-spark-spark-summit-east-talk-by-karanjeet-singh-and-thamme-gowda-narayanaswamy

Topic modeling code reference for python - https://towardsdatascience.com/topic-modelling-in-python-with-nltk-and-gensim-4ef03213cd21

Topic modeling code reference for spark - https://medium.com/@connectwithghosh/topic-modelling-with-latent-dirichlet-allocation-lda-in-pyspark-2cb3ebd5678e

Help with Spark SQL and dataframes - https://spark.apache.org/docs/2.2.0/sql-programming-guide.html

Tableau & Spark SQL: Big data just got even more supercharged:

https://www.tableau.com/about/blog/2014/10/tableau-spark-sql-big-data-just-got-even-more-supercharged-33799

pyspark.mllib package: https://spark.apache.org/docs/latest/api/python/pyspark.mllib.html#pyspark.mllib.clustering.LDA