**Task 2 Summary: Yelp Business Review Recommendation System**

Review is an important type of textual data for understanding user evaluation and preference. High-quality review comments will give customers more useful guidance for decision making. As for yelp, the quality of reviews can be measured by “useful” voting option. While this online voting measurement is sometimes time-sensitive. The reviews published long time ago may be out of date and misleading, but it can still accumulate lots of votes; the new, fresh reviews should have higher time effectiveness, but it may not be voted yet, so people cannot “find” them immediately even they are more meaningful. Our second task is to predict the usefulness score and recommend most useful reviews to costumers.

**Implement Architecture**

**3.2 Yelp Business Review Recommendation System**

In this section, we applied topic-based model and sentiment-based model to predict usefulness. For topic-based method, we used LDA model, for sentiment based method, we used part of speech tagging and sentiment intensity analyzer provided by NLTK. We set useful global average as baseline to evaluate the predicting performance.

**3.2.1 Topic Based Model**

Topic model is a type of statistical model for discovering the abstract “topics” that occur in a collection of documents. This method has been frequently used in text-mining and topic discovery area [1].

One of the well-known topic mining model is LDA (Latent Dirichlet Allocation). It can be used as a cluster and dimension reduction method. We assume documents can cover only a small set of topics and that topics use only a small set of words frequently [2]. Traditionally, we use “bag of word” model to represent sentences and documents, but now, we can represent documents by the set of topics as well. The following Figure 1 gives a straight-forward expression about how LDA works to generate topics:

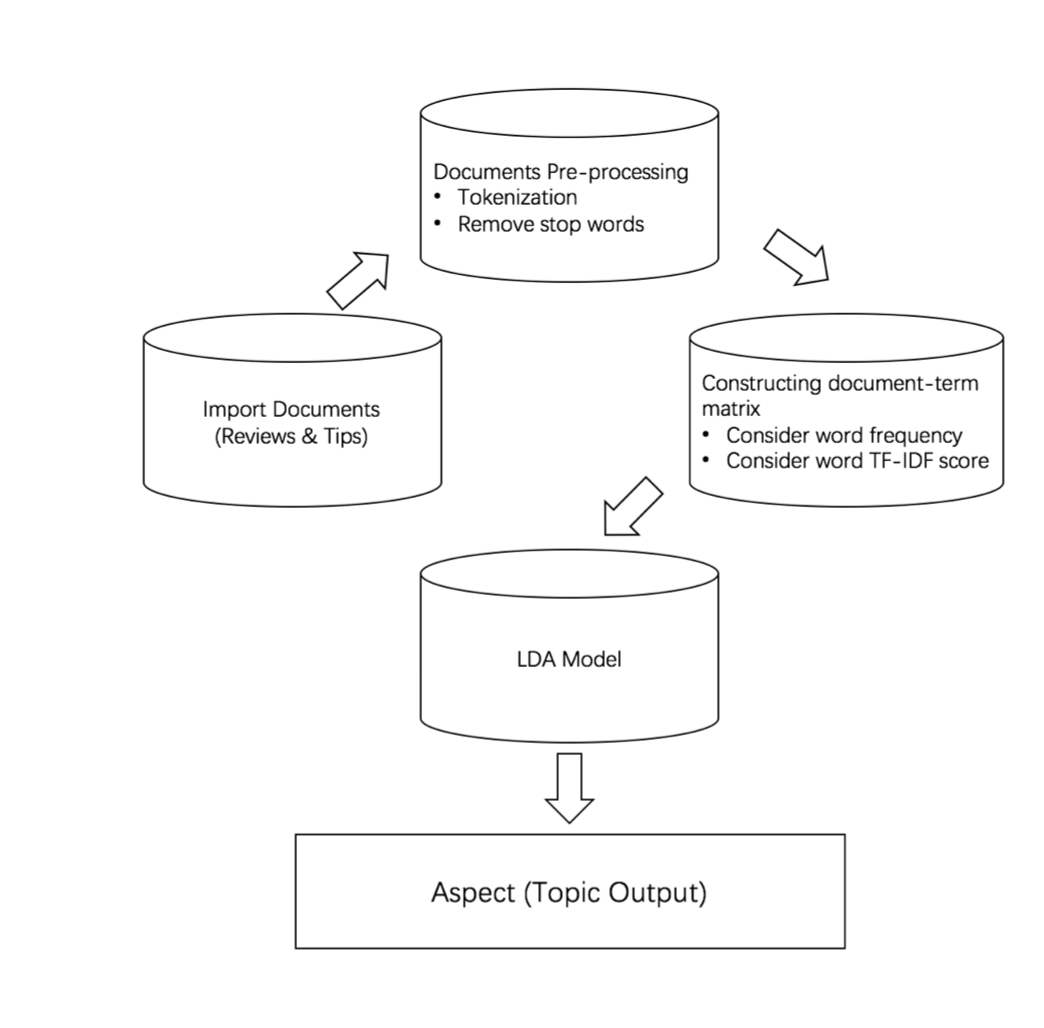


Figure 1. The general workflow of Latent Dirichlet Allocation model

There are some benefits for using LDA:

* Since it is an unsupervised algorithm, neither pre-defined aspect nor label is required. In other word, we don’t need to label each document manually.
* LDA has already been supported by Pyspark ML.

Our idea was to generate topics using LDA, and use the output topic distribution as input features to predict usefulness. The architecture of topic-based model is shown in Figure 2 (prediction part) and Figure 9 (recommendation part):



Figure 2. The prediction model architecture of topic-based method

For prediction part, the detail expression is given below:

* Data collection and import: The yelp review data is collected from MongoDB review collection. To import data into pyspark, we used Mongo-Spark-Connector and load the data into DataFrame.
* Tokenize: Tokenization is the process of taking text and breaking it into individual terms [3]. After tokenizer, we split the whole sentence into sequences of words. Pyspark ML library provide Tokenizer function to split sentence from raw text. The Figure 3 gives an example of output tokenized column:

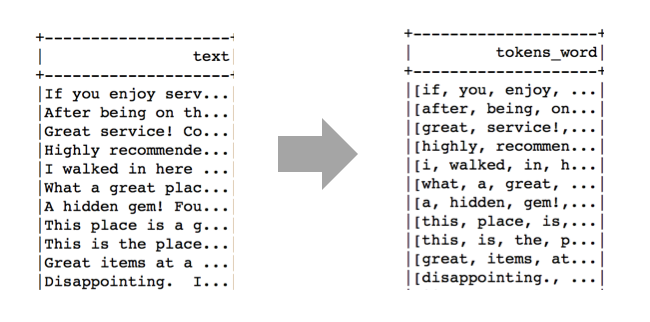


Figure 3. The sample for tokenized column of word

* Remove Stop Words: Stop words usually appear frequently but do not carry as much meaning [4]. Pyspark ML library provide StopWordsRemover function to remove stopwords from tokenized list of words. The Figure 4 gives an example of output filtered column after removing stop words:



Figure 4. The sample of filtered column of word

* Count Vector: Vectorization convert a collection of text documents to vectors of token counts. For each document, we used a frequent token vector to represent it. Pyspark ML library provide CountVectorizer function to convert list of words into frequency vector [5]. The Figure 5 gives an example of output vector column after vectorization:

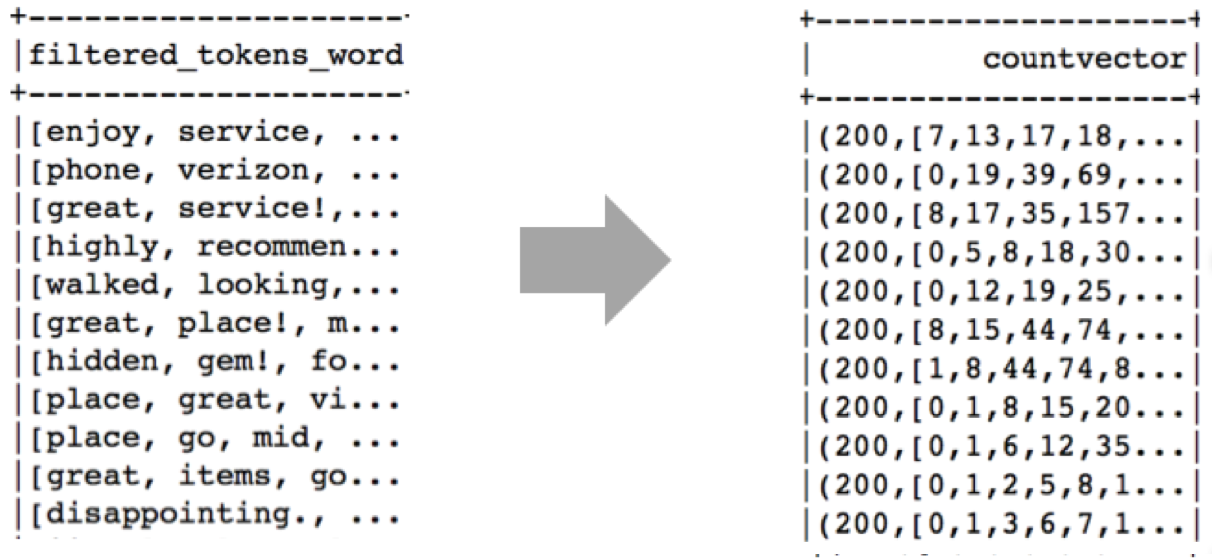


Figure 5. The sample of frequency count vector column

* IDF: IDF (inverse document frequency) down-weights words which appear frequently in a corpus [6]. The more frequent this word appears, the less important it will be. Pyspark ML library provide IDF function to calculate TF-IDF frequency. The Figure 6 gives an example of output TF-IDF column:

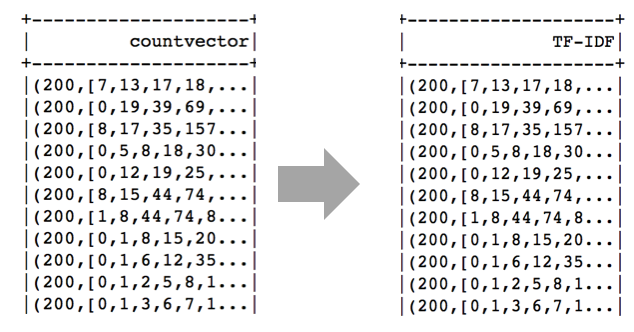


Figure 6. The sample of TF-IDF column

* LDA Model: LDA model use TF-IDF scores as input to generate topic distribution. The following Figure 7 and Figure 8 show a toy output example. We set number of topic to be 10, and 10 topics could be generated from a small set of raw text. For each document, there was a topic distribution showed that how much this document contributed, and for each topic, we could also output the corresponding term indices and term weights to show how this topic spread. Figure 7 shows the input TF –IDF features and the output topic distribution column, and Figure 8 shows the term indices and weight for each topic.

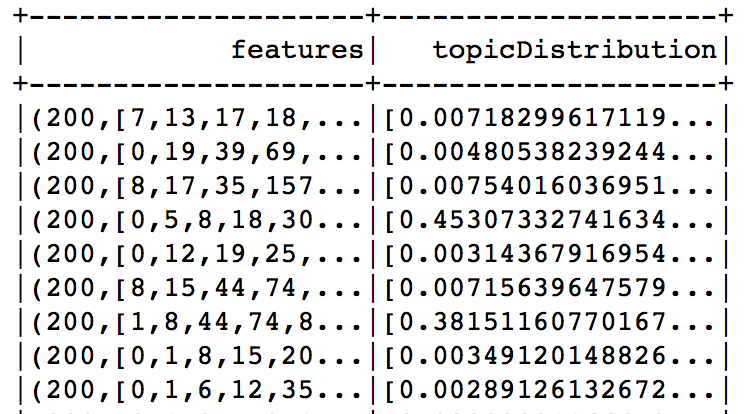


Figure 7. Input TF –IDF features and the output topic distribution

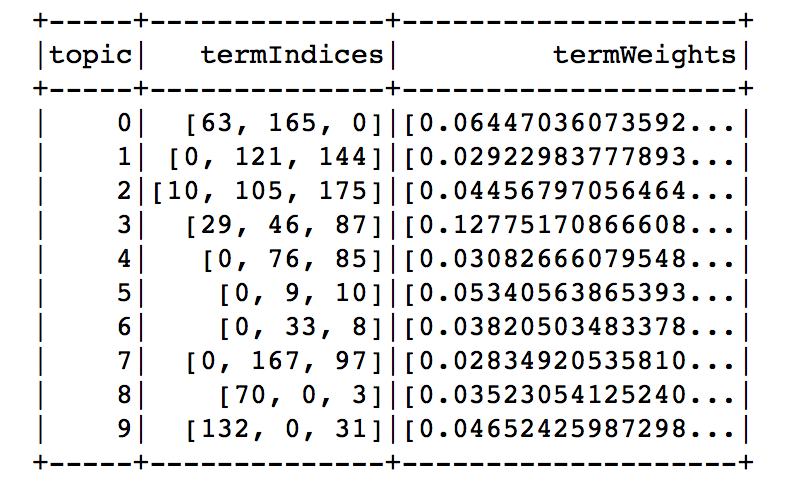


Figure 8. Term indices and weight for each topic

* Random Forest Regression Model: After pre-processing and topic extraction, we had the input feature for regression model. In topic-based model, we used random forest regression model to predict the usefulness for each review based on topic weight.

The recommend architecture is shown in Figure 9.

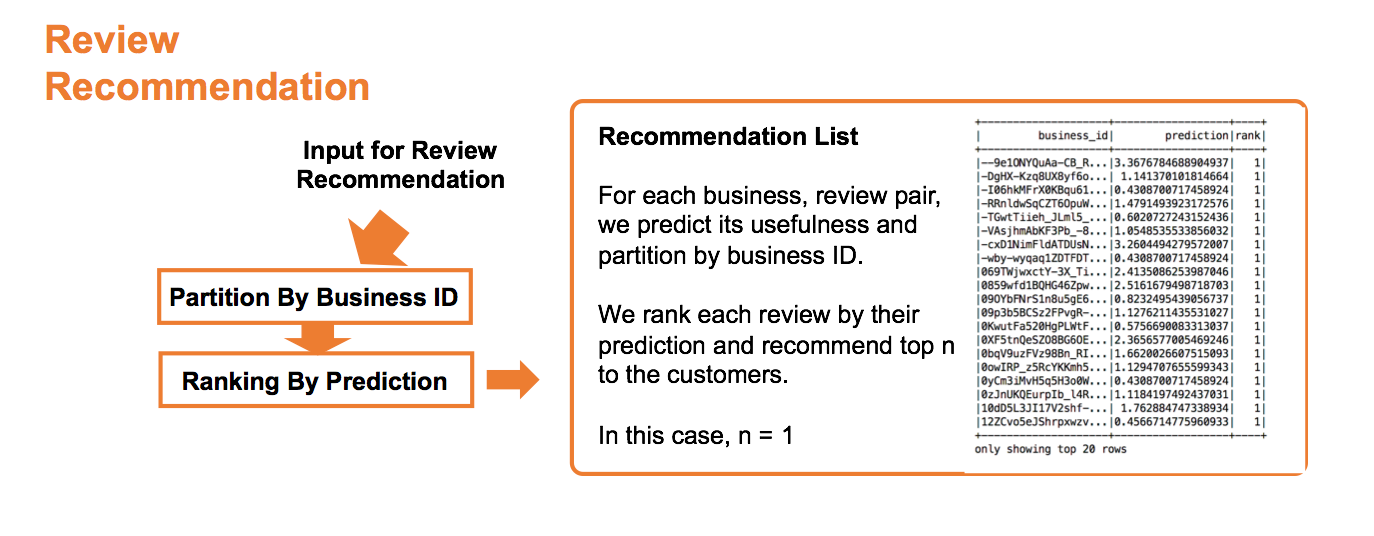


Figure 9. The recommendation model architecture of topic-based method

For recommendation part, we set a recommendation strategy based on prediction result:

* Ranking stage: After predict usefulness score, for each review, we first partitioned them by business\_d and ranked them by our prediction result.
* Recommendation stage: For each business, we recommended reviews with top n highest ranking for each business.

**3.2.2 Sentiment Based Model**

Sentiment analysis is also an important part of natural language processing. Sentiment analysis aims to determine the attitude of a speaker to a special item. In this project, we used NLTK sentiment intensity analyzer for sentiment detection.

The architecture of topic-based model is shown in Figure 10 (prediction part):



Figure 10. The prediction model architecture of sentiment-based method

For prediction part, the detail expression for each stage is given below:

- Data collection and import: We used Mongo-Spark-Connector to load the data into DataFrame.

* Data cleaning: Before applying sentiment analysis and part of speech tagging, we need to clean the raw text. For instance, remove non-ascii character, remove punctuation.
* Get word character list: After data cleaning, we used part of speech tagging in NLTK to assign part of speech for each single word, and computed the summation of words for different categories. Python NLTK package provide function nltk.pos\_tag to get the POS dictionary. The Figure 11 gives an example of output category list after POS tagging:

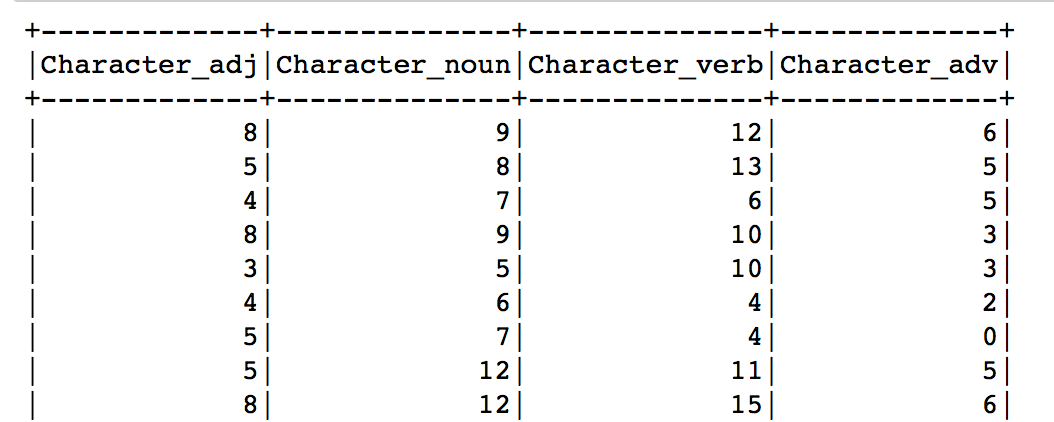


Figure 11. An example of output category list after POS tagging

- Get sentence sentiment scores: we used sentiment analysis tool in NLTK to give sentiment scores based on four different perspectives: positive, negative, neutral and compound. Python NLTK package provide function SentimentIntensityAnalyzer to get sentiment score result. The Figure 12 gives an example of output sentiment score list after POS tagging:

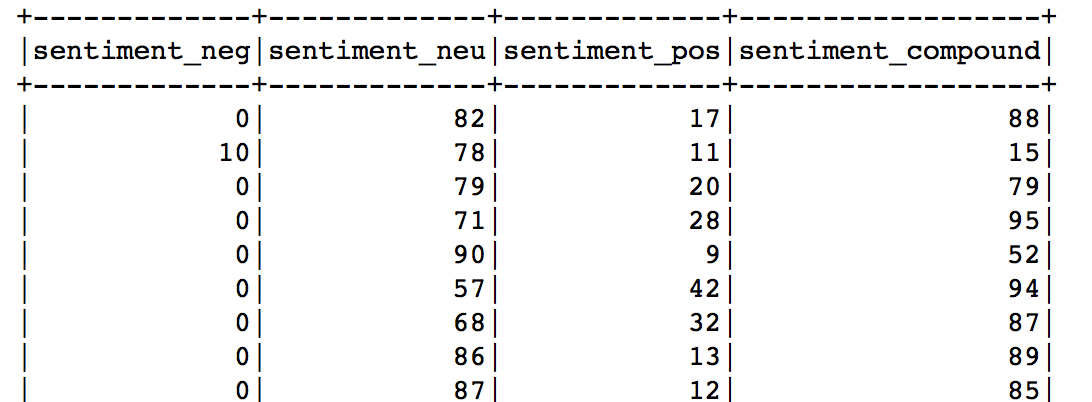


Figure 12. An example of output sentiment score list after sentiment analysis

-Feature assemble: After getting word character list and sentiment score list, we need to combine them as feature input column. We used user defined function to convert each row of list into vector, and assembled them together. The Figure 13 gives the output after feature assembling, the first element in each column is the number of words each text contains:

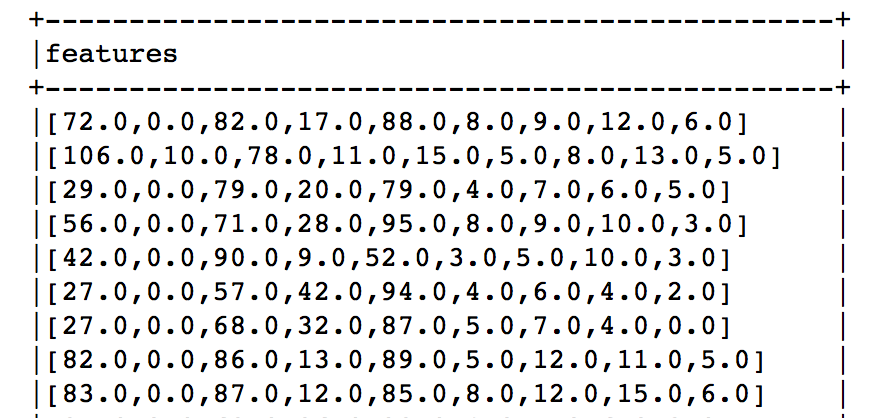


Figure 13. An example of output after feature assembling

For recommendation part, we used the same strategy which has been introduced in 3.2.1.

**Results**

**4.2 Yelp Business Review Recommendation System**

**4.2.1 Prediction Part**

Pyspark provide ParamGridBuilder and CrossValidator for tuning parameters. With this hyper-parameter tuning technology, we got the best performance for each model.

For prediction part, we used RMSE (Root mean absolute error) for evaluation, and our result is shown in Table 1:

Table 1. The result for topic-based and sentiment-based prediction models

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Baseline** | **Topic-Based Model** | **Sentiment-Based**  **Model** |
| **Prediction:**  **RMSE** | 2.77014 | 2.11462 | 2.54239 |

Obviously, topic-based model has best performance. The parameters achieve the lowest RMSE is listed in Table 2:

Table 2. The optimal parameters for topic-based model

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameters** | **Number of Topics** | **VocabSize** | **MinDF** |
| **Value** | 30 | 250 | 2 |

Where the number of topics is the topic number defined by LDA model, VocabSize is the maximum number of vocabulary the frequency count vector can keep, and MinDF is the corresponding lower bound for TF-IDF score.

**4.2.2 Recommendation Part**

For recommendation part, Figure 14 and Figure 15 list the output recommendation result. Each row represents a business-user review pair with ranking based on prediction score:

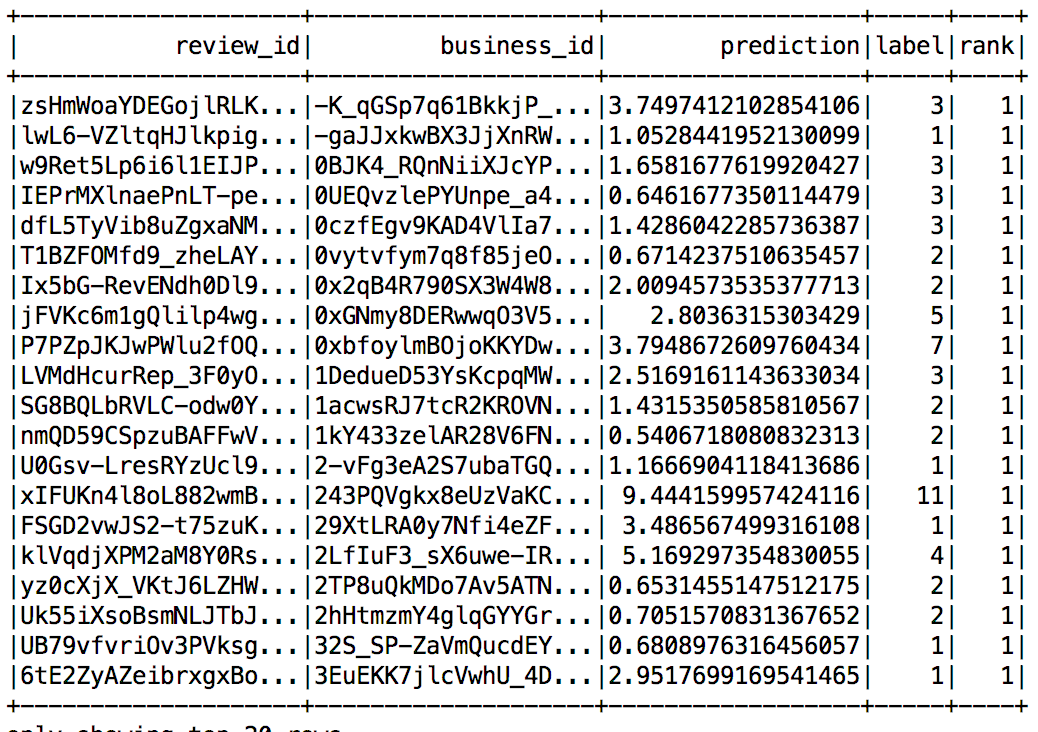


Figure 14: The output for topic-based recommendation system

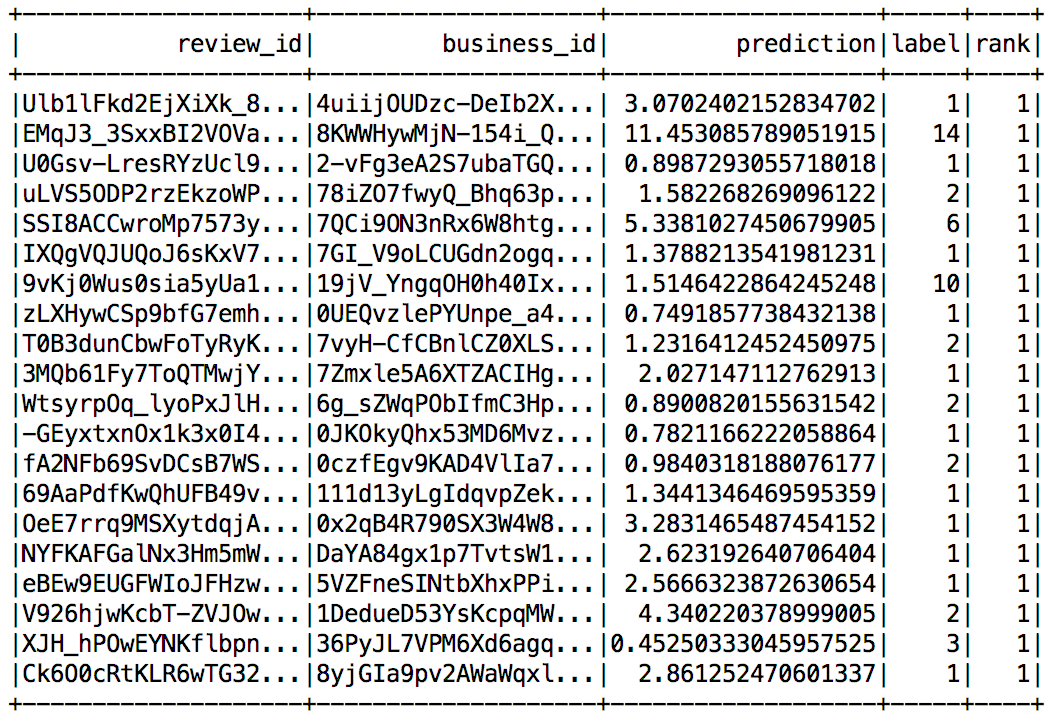


Figure 15. The output for sentiment-based recommendation system

In summary, we gave rank to each review based on predicting score, and recommended the reviews with highest rank to our customers for each business. Our prediction models successfully identify the candidate reviews which will be voted in the future.

**5. Benefits of Using Big Data Technology**

**MongoDB & Mongo-Spark-Connector**

MongoDB is a type of NoSQL database MongoDB using [JSON](https://en.wikipedia.org/wiki/JSON)-like documents with [schemas](https://en.wikipedia.org/wiki/Database_schema). It has user-friendly interface with sample statistical summary for each collection, which makes us easier to know our data. The Figure 16 shows the interface of MongoDB compass with some statistical data visualization.

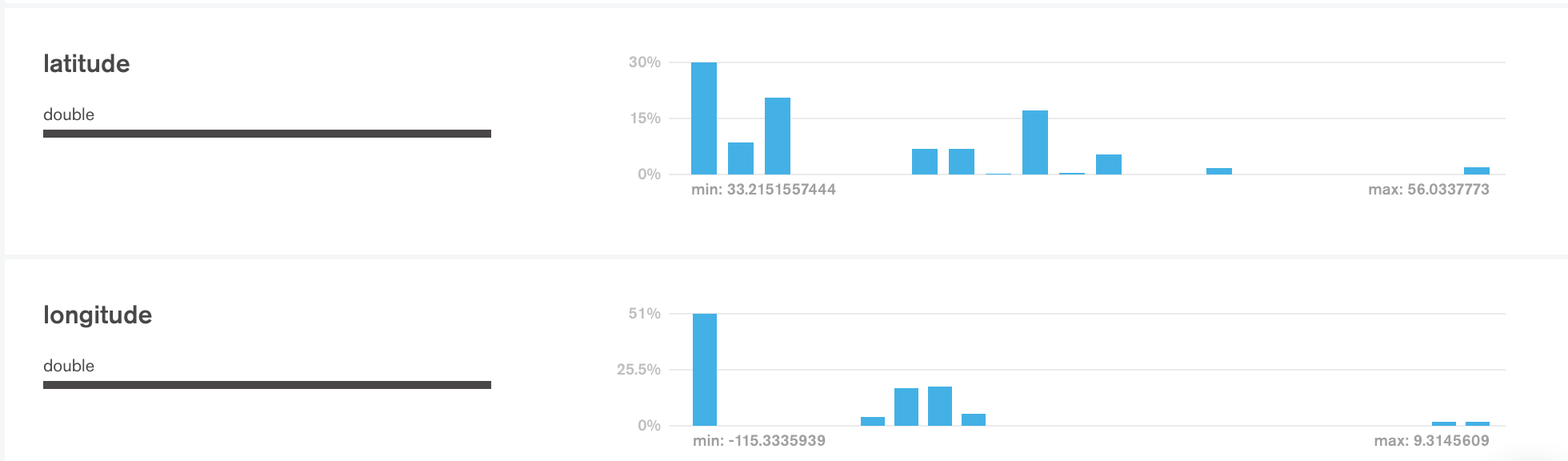


Figure 16. User interface of MongoDB Compass

Furthermore, MongoDB has API with spark, the Mongo-Spark-C0nnector. It allows users to load data from MongoDB into a DataFrame in spark, providing a faster data loading procedure. The Table 3 gives a brief comparison about loading data from json files and from MongoDB using connector.

Table 3. The comparison for loading data from json vs MongoDB

|  |  |  |
| --- | --- | --- |
| **Name** | **From json file** | **From MongoDB Collection** |
| Review | 8.759560108 | 0.485893965 |
| User | 3.069803953 | 0.542819977 |
| Business | 0.632100105 | 0.024842024 |
| Checkin | 0.308743 | 0.023883104 |
| Tip | 0.647626877 | 0.022135019 |

**Spark ML**

Spark Machine Learning Library collect common feature selection, extraction method and general learning algorithm, which makes practical machine learning scalable and easy. Besides, the DataFrame based library support pipeline structure, providing a good experience for us when tuning parameters and evaluating models.

**Reference**

[1]. Wikipedia, Topic model: https://en.wikipedia.org/wiki/Topic\_model#Topic\_mo dels\_for\_context\_informati.

[2]. Wikipedia, Latent Dirichlet Allocation: https://en.wikipedia.org/wiki/Latent\_D irichlet\_allocation.

[3]. Apache Spark, Machine Learning Features: https://spark.apache.org/docs/2.1. 0/ml-features.html#tokenizer.

[4]. Apache Spark, Machine Learning Features: https://spark.apache.org/docs/2.1. 0/ml-features.html#stopwordsremover.

[5]. Apache Spark, Machine Learning Features: <https://spark.apache.org/docs/2.1.0/ml->

features.html#countvectorizer

[6]. Apache Spark, Machine Learning Features: <https://spark.apache.org/docs/2.1>.

0/ml-features.html#tf-idf

说明：

1. 网页的参考格式我是按照网站名，网页名：网址来的，你觉得不合适可以看着改