K-Means Based Recommendation Engine

Matthew Scanlan

DSC680

Project 3

Project Overview

- Document Database and Pre-processing
- Reccomendation engine
  - Pyspark - Pipelines
  - RegexTokenizer
  - StopWordRemover
  - Word2Vec
  - K-means Model

Project Objective

- Create a Basic Recommendation Engine Using a K-Means Model
    1. Intake Document From User
    2. Process Text
    3. Predict Cluster Label with K-Means Model
    4. Return a Set of Similar Documents

Project Description

- Aggregate data
  - Light Novel EPUBs > Text Files
- Build Pyspark Pipeline Model and Prediction Database
  1. Arrange Pipeline Stages
  2. Fit Pipeline to Document Database to Train Model
- Transform Document Database with Pipeline Model to Pair Sample Data and Predictions
- Intake Documents From User
- Return Recommended documents

Document Database

- The Document Database used in this project was sourced from Light Novel Crawler a Python program that retrieves Light Novels from Light Novel Aggregator Websites
- The Documents started out in EPUB format and were converted into text files
  1. Unzip Epub files into HTML files
  2. BeautifulSoup to extract text within <p> tags
  3. Preprocess Text to remove symbols
  4. Read Documents into a Pyspark DF

Create Pipeline

- Pyspark Pipeline
  - Pyspark.ml.Pipeline
- Pull Text Into a Format for Processing
  - Pyspark.ml.feature.RegexTokenizer
- Remove Uninformative Words
  - Pyspark.ml.feature.StopWordRemover
- Change Tokens into Computer processable Vectors
  - Pyspark.ml.feature.Word2Vec
- Process Document Vectors to Predict Similar Documents
  - Pyspark.ml.clustering.KMeans

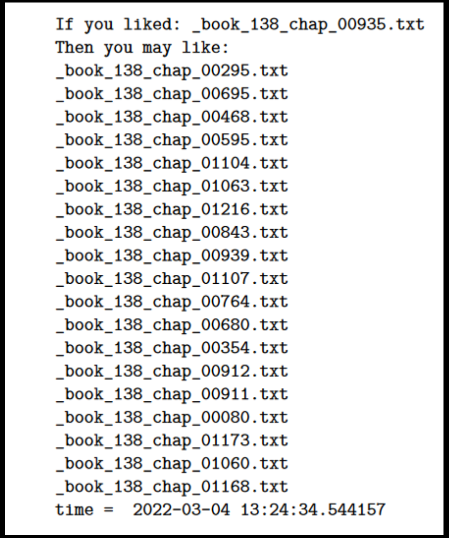
Train and Fit Pyspark PipelineModel

- Once the Document Database and Pipeline are created the Document Database Must First be .fit() to the Pipeline To Train the Model
  - Unsupervised Model
- To Record the Predictions of Similar Documents (Predicted Cluster Labels) the Document Database must be .transform()ed with the new PipelineModel

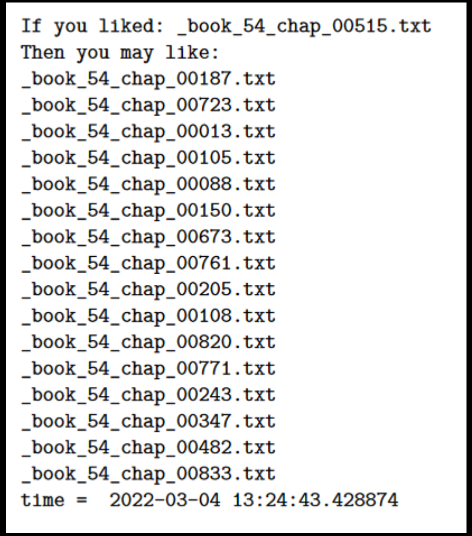
Use Model to Recommend Documents

- Once the Model Has Been Fit and the Original Document Database Has Been Classified, It Can Then Be Used to Recommend Similar Documents
- User's Document is Submitted
- The Existing Pipeline is used to .transform() the User's Document
- The Transformation Returns a Cluster Label
- Filter Original Document Database to Return Documents Predicted to be in the Same Cluster

Validity of Model



```
If you liked: _book_138_chap_00935.txt
Then you may like:
_book_138_chap_00295.txt
_book_138_chap_00695.txt
_book_138_chap_00468.txt
_book_138_chap_00595.txt
_book_138_chap_01104.txt
_book_138_chap_01063.txt
_book_138_chap_01216.txt
_book_138_chap_00843.txt
_book_138_chap_00939.txt
_book_138_chap_01107.txt
_book_138_chap_00764.txt
_book_138_chap_00680.txt
_book_138_chap_00354.txt
_book_138_chap_00912.txt
_book_138_chap_00911.txt
_book_138_chap_00080.txt
_book_138_chap_01173.txt
_book_138_chap_01060.txt
_book_138_chap_01168.txt
time =  2022-03-04 13:24:34.544157
```



```
If you liked: _book_54_chap_00515.txt
Then you may like:
_book_54_chap_00187.txt
_book_54_chap_00723.txt
_book_54_chap_00013.txt
_book_54_chap_00105.txt
_book_54_chap_00088.txt
_book_54_chap_00150.txt
_book_54_chap_00673.txt
_book_54_chap_00761.txt
_book_54_chap_00205.txt
_book_54_chap_00108.txt
_book_54_chap_00820.txt
_book_54_chap_00771.txt
_book_54_chap_00243.txt
_book_54_chap_00347.txt
_book_54_chap_00482.txt
_book_54_chap_00833.txt
time =  2022-03-04 13:24:43.428874
```

- How can we tell if the model's predictions are valid?
  - We cannot definitively say this
- We can **infer** the accuracy of the model by looking at the predicted data
- Same author, same book > prediction is probably valid