Architectural Decisions Document

1. Dataset

https://www.kaggle.com/new-york-city/new-york-city-current-job-postings

2. Use Case

To make a job recommender system using the dataset.

3. Architectural Choices

3.1 Apache Spark and Pandas for data collection, processing and modifications.

3.2 Matplotlib for data visualizations.

3.3 Natural Language Processing as Machine Learning Algorithm.

3.4 Scikit-Learn as Machine Learning Library.

3.5 Keras as Deep Learning Library.

4. Data Exploration

4.1 Found some unformatted and missing data. But they can't be dropped as it would reduce the dataset drastically.

4.2 Found certain columns with wrong data types. For example, salary should be float or integer type instead of string.

4.3 Data visualizations using bar plots and wordclouds. I can find the highest paid jobs as well as preferred skills and qualifications required for most of the jobs.

4.4 Correlation matrix allows us to see how relevant is a certain measurement.

5. Data Cleaning and Feature Engineering

5.1 Drop the column with unformatted and irrelevant data.

5.2 Modify some missing values such as full-time/part-time indicator.

5.3 Fix data types such as salary and number of positions.

5.4 Combine multiple columns into one and extract keywords to help perform one-hot encoding.

6. Model Training

6.1 Machine learning model – TfidVectorizer and Cosine Similarity Matrix => To weigh a keyword in the merged column and assign the importance to that keyword based on the number of times it appears in the column.

6.2 Deep Learning model – Keras Sequential => to create models layer- by-layer.

6.3 Evaluation metric – Accuracy => As the recommender is content based, the best way to evaluate the model performance is using the idea of relevance of information, namely, how relevant/similar two jobs are, in this example.

6.4 Model Improvement – remove punctuations and digits => As it is difficult extract keywords when having different types of punctuations. Also, digits in keywords provide no information for our model. Instead, we lose information during keywords extraction.