Finding similar items

Dzmitry Kurch, DSE

Algorithms for massive data project, June 2024

1. Introduction

The primary goal of this project is to detect similar pairs of records or identify a set of similar records of higher cardinality within a dataset. Specifically, the similarity detection is focused on the *job_summary* column containing LinkedIn position descriptions from the Kaggle dataset *Linkedin Jobs & Skills*. An essential application of such techniques for detecting similarities among documents is the deletion of duplicate records referring to the same entity when merging two or more data sources. This process is typically known as entity resolution or record linkage. Other terms often used in this context include entity disambiguation, entity linking, duplicate detection, deduplication, record matching, and etc. By identifying these similar records, we can ensure data integrity and avoid redundancy, which is crucial for maintaining clean and reliable datasets.

For this project, I utilized the Databricks community platform and Spark with the Python API (PySpark). To handle the text preprocessing tasks, I employed the spark-nlp library, which provided robust tools for processing the job summaries. For the clustering algorithms necessary to group similar records, I used Spark's MLlib library.

One of the significant advantages of using Spark for this project is its inherent scalability. Spark's distributed computing capabilities ensure that the solution can handle large datasets efficiently. All operations, from text preprocessing to clustering, are performed in a distributed manner, making the solution capable of scaling with respect to the dataset size out-of-the-box. This makes this approach not only effective but also highly efficient for large-scale data integration and entity resolution tasks.

2. Data organization and preprocessing

The data for this project was originally stored in a CSV file containing 1.4 million records with two columns: job link and job summary. Given the large size of the dataset and the limitations of the Databricks community edition cluster, I decided to work with a 10% sample of the data. This was necessary because the community edition lacks the computing power and number of nodes required to efficiently process the entire dataset within the available time constraints.

To start, I loaded the data directly into Databricks as a table and accessed it via Spark-SQL for preprocessing. Using Spark-SQL allowed me to efficiently query and manipulate the sample data before diving into more detailed text processing tasks.

For text preprocessing, I utilized the spark-nlp library, which offers a comprehensive set of tools for handling text data. The preprocessing pipeline included several key steps:

1. **Document Assembly**: Converting the raw text into a structured format suitable for further processing.
2. **Tokenization**: Splitting the text into individual words or tokens.
3. **Normalization**: Standardizing the text by converting it to lowercase and removing any special characters or punctuation.
4. **Stop Words Removal**: Eliminating common words that do not carry significant meaning (e.g., "and", "the").
5. **Stemming**: Reducing words to their root forms to ensure that different forms of the same word are treated equally (e.g., "running" becomes "run").
6. **Finishing**: Converting the processed tokens back into a format that can be used for further analysis.

These preprocessing techniques are essential for cleaning and standardizing the text data, which helps improve the accuracy and efficiency of subsequent analytical steps.

Example of top-10 most frequent tokens extracted: work, experi, manag, require, team, provid, include, service, show, care.

Following the text preprocessing, I applied two methods for transforming the text data into numerical features: Hashing Term Frequency (TF) and Inverse Document Frequency (IDF) using Spark's MLlib library. Hashing TF provides a bag-of-words (BOW) representation, where each document is represented by the frequencies of its tokens. It reduces the dimensionality of the feature space by mapping tokens to a fixed number of hash buckets, which helps mitigate the issue of high memory consumption and computational complexity associated with large vocabularies. IDF then scales these frequencies based on the importance of each token across the entire dataset, helping to highlight more informative words. As a result, I ended up with two possible ways of sentence embedding: TF-IDF and BOW. These embeddings served as the foundation for detecting similarities among the job summaries.

3. Finding similar pairs

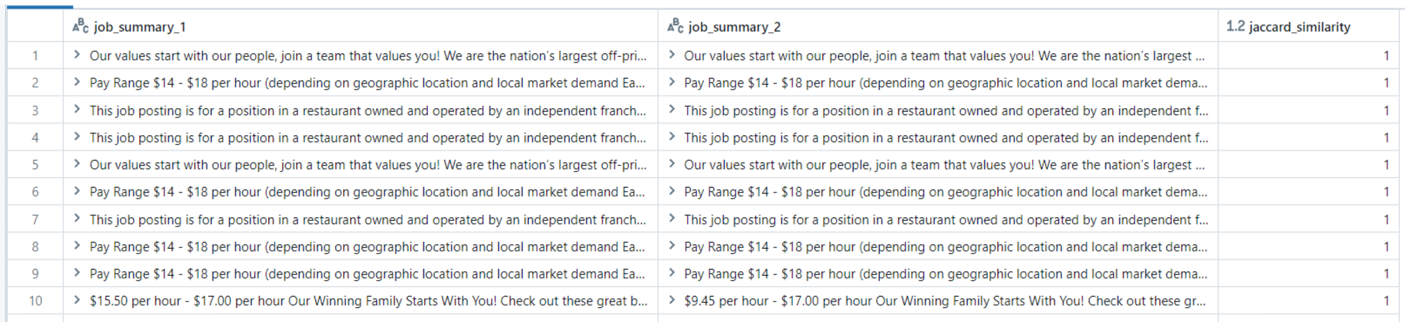
To identify similar pairs of records, I began by performing a cross join of the table with itself. This “brute-force” operation compares every record with every other record in the dataset. To avoid redundant comparisons, I filtered out the duplicated pairs, ensuring that each pair (A, B) and (B, A) was represented only once. This preparation step set the stage for row-wise similarity calculations.

For measuring similarity, I employed two custom User Defined Functions (UDFs): one for calculating Jaccard similarity and the other for computing cosine similarity.

Jaccard Similarity is a measure of similarity between two sets, defined as the size of the intersection divided by the size of the union of the sets. In the context of text data, this means comparing the sets of tokens (words) in each document. The formula for Jaccard similarity is:

$$Jaccard\ Similarity = \frac{|A \cap B|}{|A \cup B|}$$

where *A* and *B* are the sets of tokens from two documents. A higher Jaccard similarity indicates a greater overlap between the two sets, thus higher similarity.



	job_summary_1	job_summary_2	1.2 jaccard_similarity
1	Our values start with our people, join a team that values you! We are the nation's largest off-pri...	Our values start with our people, join a team that values you! We are the nation's largest ...	1
2	Pay Range $14 - $18 per hour (depending on geographic location and local market demand Ea...	Pay Range $14 - $18 per hour (depending on geographic location and local market dema...	1
3	This job posting is for a position in a restaurant owned and operated by an independent franch...	This job posting is for a position in a restaurant owned and operated by an independent f...	1
4	This job posting is for a position in a restaurant owned and operated by an independent franch...	This job posting is for a position in a restaurant owned and operated by an independent f...	1
5	Our values start with our people, join a team that values you! We are the nation's largest off-pri...	Our values start with our people, join a team that values you! We are the nation's largest ...	1
6	Pay Range $14 - $18 per hour (depending on geographic location and local market demand Ea...	Pay Range $14 - $18 per hour (depending on geographic location and local market dema...	1
7	This job posting is for a position in a restaurant owned and operated by an independent franch...	This job posting is for a position in a restaurant owned and operated by an independent f...	1
8	Pay Range $14 - $18 per hour (depending on geographic location and local market demand Ea...	Pay Range $14 - $18 per hour (depending on geographic location and local market dema...	1
9	Pay Range $14 - $18 per hour (depending on geographic location and local market demand Ea...	Pay Range $14 - $18 per hour (depending on geographic location and local market dema...	1
10	$15.50 per hour - $17.00 per hour Our Winning Family Starts With You! Check out these great b...	$9.45 per hour - $17.00 per hour Our Winning Family Starts With You! Check out these gr...	1

Table 1 - Top pairs of documents by Jaccard similarity

Cosine Similarity, on the other hand, measures the cosine of the angle between two non-zero vectors in a multidimensional space. When applied to text data, the vectors are typically the TF-IDF representations of the documents. The formula for cosine similarity is:

$$Cosine\ Similarity = \frac{\vec{A}\vec{B}}{\|\vec{A}\|\|\vec{B}\|}$$

where *A* and *B* are the TF-IDF vectors of two documents and $||A||$ and $||B||$ are their magnitudes. Cosine similarity ranges from -1 to 1, with 1 indicating identical direction (maximum similarity), 0 indicating orthogonality (no similarity), and -1 indicating opposite direction (maximum dissimilarity).

When it came to identifying 100% duplicates, both Jaccard similarity and cosine similarity yielded the same pairs of records. This consistency is since completely identical documents will have maximum similarity across both metrics. However, beyond perfect duplicates, the results varied because the metrics capture different aspects of similarity.

Jaccard Similarity focuses on the presence or absence of tokens, making it sensitive to the overlap of unique words between documents.

Cosine similarity considers the frequency and importance of words through TF-IDF, capturing the angle between document vectors in a high-dimensional space, which reflects the distribution of word importance.

These differences mean that Jaccard similarity might be more suitable for comparing documents with similar sets of words, while cosine similarity is often better at capturing the overall similarity in terms of word frequency and significance.

4. Finding sets of similar records

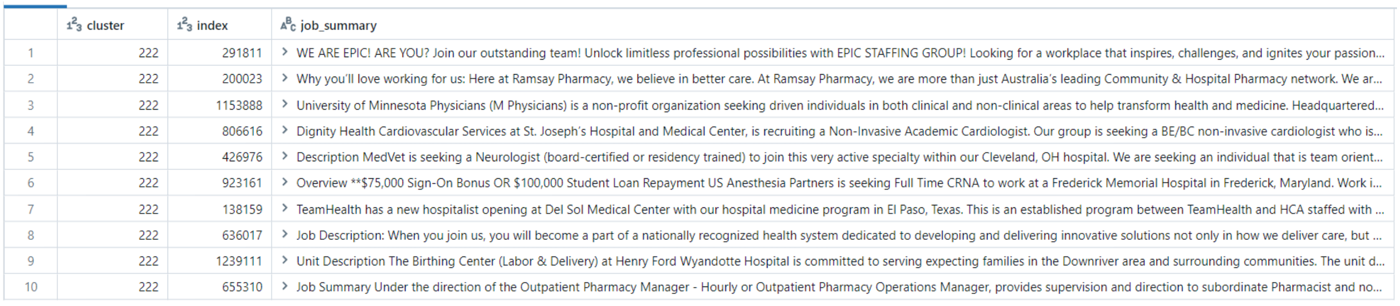
In the final step of the analysis, I employed the Bisecting K-Means clustering algorithm to identify sets of similar records of higher cardinality. Bisecting K-Means is a variant of the traditional K-Means algorithm that uses a hierarchical clustering approach.

The key difference between Bisecting K-Means and regular K-Means lies in the way clusters are formed. Instead of randomly initializing centroids and iteratively assigning points to the nearest centroid, Bisecting K-Means starts with a single cluster containing all data points and then recursively splits clusters into two child clusters until the desired number of clusters is reached. This approach tends to produce more balanced cluster sizes and can handle clusters of varying densities and shapes more effectively.

I used TF-IDF vectors as input to the Bisecting K-Means algorithm and experimented with different numbers of clusters. By analyzing the resulting groups manually, I found that clusters with too many records (100+ records) often contained heterogeneous job postings that were not truly similar. Conversely, smaller clusters (starting from 30-20 records and fewer) tended to contain very similar job postings.

The preference for smaller cluster sizes stems from the nature of the data and the clustering algorithm. Smaller clusters typically exhibit greater homogeneity, containing job postings that share more similar characteristics and keywords. This allows for more precise identification of specific themes or topics within the dataset. Additionally, smaller clusters are more interpretable and easier to analyze, making them preferable for understanding the underlying structure of the data.

It's important to note that Bisecting K-Means does not allow for direct control over the cluster size; instead, it is a result of the initial definition of *K*, the number of clusters. To ensure the emergence of smaller clusters, K must be chosen large enough to allow for sufficient splitting of clusters. By iteratively adjusting *K* and analysing the resulting clusters, I was able to identify an optimal number of clusters that produced meaningful and interpretable results. I found out that it should be about thousands, e.g. one thousand worked just fine.



	cluster	index	job_summary
1	222	291811	> WE ARE EPIC! ARE YOU? Join our outstanding team! Unlock limitless professional possibilities with EPIC STAFFING GROUP! Looking for a workplace that inspires, challenges, and ignites your passion...
2	222	200023	> Why you'll love working for us: Here at Ramsay Pharmacy, we believe in better care. At Ramsay Pharmacy, we are more than just Australia's leading Community & Hospital Pharmacy network. We ar...
3	222	1153888	> University of Minnesota Physicians (M Physicians) is a non-profit organization seeking driven individuals in both clinical and non-clinical areas to help transform health and medicine. Headquartered...
4	222	806616	> Dignity Health Cardiovascular Services at St. Joseph's Hospital and Medical Center, is recruiting a Non-Invasive Academic Cardiologist. Our group is seeking a BE/BC non-invasive cardiologist who is...
5	222	426976	> Description MedVet is seeking a Neurologist (board-certified or residency trained) to join this very active specialty within our Cleveland, OH hospital. We are seeking an individual that is team orient...
6	222	923161	> Overview **$75,000 Sign-On Bonus OR $100,000 Student Loan Repayment US Anesthesia Partners is seeking Full Time CRNA to work at a Frederick Memorial Hospital in Frederick, Maryland. Work i...
7	222	138159	> TeamHealth has a new hospitalist opening at Del Sol Medical Center with our hospital medicine program in El Paso, Texas. This is an established program between TeamHealth and HCA staffed with ...
8	222	636017	> Job Description: When you join us, you will become a part of a nationally recognized health system dedicated to developing and delivering innovative solutions not only in how we deliver care, but ...
9	222	1239111	> Unit Description The Birthing Center (Labor & Delivery) at Henry Ford Wyandotte Hospital is committed to serving expecting families in the Downriver area and surrounding communities. The unit d...
10	222	655310	> Job Summary Under the direction of the Outpatient Pharmacy Manager - Hourly or Outpatient Pharmacy Operations Manager, provides supervision and direction to subordinate Pharmacist and no...

Table 2 - Example of cluster size 32 with medical job postings, K-Means 1000

5. Conclusions and next steps

In conclusion, this project successfully employed PySpark and various NLP techniques to detect similar pairs and sets of records within a dataset of LinkedIn job summaries. By utilizing Spark's distributed computing capabilities and libraries such as spark-nlp and MLlib, I were able to efficiently preprocess the text data, compute similarity metrics, and perform clustering analysis. Through experimentation, I found that smaller clusters tended to contain more similar job postings, highlighting the importance of cluster size in capturing meaningful patterns in the data.

Moving forward, the next steps could involve further refinement of the preprocessing pipeline, exploration of alternative similarity metrics, and validation of clustering results through domain-specific analysis or manual verification. Additionally, deploying the developed solution on a larger cluster or cloud infrastructure could enable the analysis of the entire dataset and provide deeper insights into the relationships among job postings.