* **Possible approaches to test whether 2 resume files are similar**

1. Direct Text Comparison: Read both files and compare them line by line.

**Limitation**: Formatting differences and synonyms can lead to mismatches.

1. Token-Based Similarity
   1. Jaccard Similarity: Compare the unique words in both resumes.
   2. NER Similarity: After extracting words using NER compare the words of all the words.
   3. Bag Of Words: Measures text similarity based on unique words
   4. TF-IDF Cosine Similarity: Measures text similarity based on word importance.
2. Semantic Similarity: Goes beyond word matching by capturing meaning.
   1. Embedding-Based Similarity (Using Sentence Transformers):
      1. Converts sentences into numerical vectors using pre-trained models (e.g., sentence-transformers like BERT, RoBERTa).
      2. Measures similarity using cosine similarity between the embeddings.

-----------------------------------------------------------------------------------------------------------------------------------------

* **Resume Clustering and Similarity Computation**

## 1. Data Preprocessing

* We load the **resource\_id, raw text, and languages** (generated by the resume parser) for each resume and save them in a CSV file for further preprocessing.
* The raw text is then cleaned by **removing special characters, URLs and StopWords** to ensure better quality for subsequent processing.

## 2. Explored Approaches

### **2.1 Token-Based Similarity Approach**

* Given the vast dataset of **over 1.5 lakh resumes** in more than **50 different languages**, relying on token-based similarity was deemed infeasible due to:

1. Failure to capture **semantic similarity**.
2. Requirement of a **dictionary of unique words**, which exceeded **3 crore words** due to an average of **200 words per resume**.

* Thus, we opted for an **embedding-based approach** that converts text into numerical representations **independently** while preserving semantic meaning.

### **2.2 Semantic Similarity Approach**

#### **2.2.1 Numerical Representation of Text**

* The cleaned text was converted into numerical embeddings using **Sentence Transformer (multilingual-MiniLM-L12-v2)**, chosen for its:

1. Multilingual capability.
2. Smaller size (~400 MB) compared to other embedding models.

#### **2.2.2 Efficient Similarity Computation using FAISS**

* Due to the large dataset, computing **cosine similarity for every resume pair** was computationally expensive. Instead, we:

1. Stored resume embeddings using the **FAISS** library.
2. Applied **K-Nearest Neighbors (KNN)** to find the **top K matching resumes**.
3. Experimented with **K values from 5 to 100** to optimize accuracy.

#### **2.2.3 Graph-Based Clustering**

* For large clusters (e.g., **500-1000 resumes**), efficiently identifying similar resumes was a challenge. We addressed this by:

1. Connecting resumes via **graph-based methods**, assigning the same **Cluster ID** to connected nodes.
2. Example: If **top 5 similar resumes of A** are {A, B, C, D, E} and **top 5 for W** are {W, B, X, Y, Z}, then B connects both groups, merging them into **one cluster**.

### **2.3 Addressing False Positives & Negatives in Clustering**

* Using only **sentence embeddings, KNN, and cosine similarity** was insufficient for effective clustering:

1. **High threshold (98%)** → Many similar resumes missed (**False Negatives**).
2. **Low threshold (90%)** → Unrelated resumes clustered together (**False Positives**).
3. Example: If **A & B = 91% similarity**, and **B & C = 91%**, then **A & C** might have **only 82% similarity**, ideally placing them in different clusters.
4. A low threshold (90%) led to nearly **10,000 resumes in a single cluster**, making it impractical.

## 3. Hybrid Approach

## Traditional token-based methods, such as Jaccard similarity or TF-IDF, struggle with large-scale datasets due to their inability to capture semantic meaning and their computational complexity when dealing with extensive vocabularies. On the other hand, embedding-based approaches using sentence transformers provide a more robust semantic understanding but can lead to false positives and false negatives in clustering.

## To address these limitations, we propose a hybrid approach that leverages both semantic embeddings and token-based refinement for improved resume similarity computation. Initially, resumes are transformed into numerical embeddings using a multilingual sentence transformer model. K-Nearest Neighbors (KNN) and FAISS indexing enable efficient similarity computation, identifying the top-k most similar resumes. However, to refine clusters and reduce misclassifications, token-based methods, such as TF-IDF, are applied within the identified subsets. This two-stage process ensures that semantically similar resumes are grouped while minimizing the impact of irrelevant matches.

### **3.1 Cluster Refinement using DBSCAN & Token-Based Methods**

1. To **avoid missing resumes**, we used a **lower similarity threshold (90%-95%)** initially to find top k similar resumes using **sentence embeddings, KNN, and cosine similarity (just like semantic similarity approach)**.
2. Then applied **DBSCAN** with **TF-IDF** and other token-based methods to refine clusters.
3. Token-based methods were feasible at this stage because a single cluster contained a maximum of 10,000 resumes, limiting the number of unique words.
4. **DBSCAN with sentence embeddings** and Agglomerative clustering with token-based methods were tested but often misclassified resumes as **noise** (**False Negatives**).
5. Refinement reduced the **number of resumes from ~25,000 to ~7,500**.

### **3.2 Enhancing Cluster Formation with TF-IDF Threshold**

1. An additional condition was introduced before creating edges: a TF-IDF threshold was applied.
2. Token-based methods were employed here (we have top k similar resumes, so we applied token-based method on this top k (max 100) number resume) because top-matching resumes were already identified using embedding-based similarity.
3. Among all token-based methods, TF-IDF performed the best due to its ability to capture word importance.

We determine resume similarity using both embedding-based similarity and TF-IDF similarity, ensuring semantic understanding and keyword importance. Instead of DBSCAN, which proved inefficient for breaking clusters.

Thus, our algorithm requires the finalization of three key hyperparameters: the value of K in KNN (which determines the number of top similar resumes retrieved), the embedding similarity threshold, and the TF-IDF similarity threshold. Optimizing these parameters ensures an optimal balance between accuracy and computational efficiency.

## 4. Final Parameter Optimization

Finalized algorithm requires the finalization of three key hyperparameters: the value of K in KNN (which determines the number of top similar resumes retrieved), the embedding similarity threshold, and the TF-IDF similarity threshold. Optimizing these parameters ensures an optimal balance between accuracy and computational efficiency.

### **4.1 Optimal Value of K**

1. The **choice of K** impacted the **TF-IDF threshold** and **corpus size**.
2. The **average cluster size was 3 resumes**.
3. Increasing **K (e.g., 10 to 100)** resulted in:
   1. More **irrelevant resumes**.
   2. Larger **corpus size** reducing **TF-IDF effectiveness**.
4. **Final Decision: K = 5**.

### **4.2 Embedding Threshold**

1. Initially set at **90%**.
2. Observed **False Negatives** where some resumes had **40%-60% similarity**.
3. Eliminated **fixed similarity threshold**.
4. Instead, relied on:
   1. **KNN-based filtering** using sentence embeddings (here we will get top k similar resume irrespective of similarity percentage).
   2. **Cluster refinement via TF-IDF thresholds**.

### **4.3 TF-IDF Threshold**

1. Experimented with values **0.5 to 0.9**.
2. **0.6 provided the best results**.

**Experimental Result:**

All the experiments are performed on 142059 number of resumes

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Embedding Threshold | TF-IDF Threshold | K | Total Similar Resume | Percentage of similar resume (%) | Total Number of Cluster | False Positive | False Negative |
| 0.9 | 0.6 | 5 | 10070 | 7.08 | 3203 | 0 | Many |
| 0.0 | 0.5 | 100 | 13971 | 9.83 | 3944 | 1508 | Very less |
| 0.0 | 0.6 | 100 | 12186 | 8.58 | 3660 | 628 | Many |
| 0.0 | 0.6 | 5 | 12463 | 8.77 | 3679 | 590 | Very Minimal |

## Synopsis

## First Result (0.9, 0.6, 5): The high embedding threshold (90%) led to many false negatives as resumes with 40-60% similarity were excluded.

## Second Result (0.0, 0.5, 100): Lowering the embedding threshold to 0.0 retrieved more resumes, but increasing K to 100 introduced more false positives due to irrelevant matches.

## Third Result (0.0, 0.6, 100): Increasing TF-IDF threshold to 0.6 improved filtering, reducing false positives but still leaving some false negatives.

## Final Result (0.0, 0.6, 5): Reducing K to 5 ensured that only the most relevant resumes were retrieved, minimizing both false positives and false negatives, leading to an optimal balance of accuracy and efficiency.

## 5. Conclusion

The final approach combines semantic embeddings for initial similarity detection with token-based refinement to achieve accurate resume clustering across multiple languages. This hybrid method effectively balances computational efficiency with clustering accuracy.