Name Matching New Approach

**Old logic Documentation-**

https://bharatpe.atlassian.net/wiki/spaces/DS/pages/13598735/DS-KYC-Name+Matching+Service+Technical+Document

* Manually Data Validation (40K) and Prepared Balanced Data (20k) Still Preparing more data for re-training of model

STEP:1

* **Fuzzy Wuzzy**

**fuzzy\_similarity = (fuzzy\_ratio + fuzzy\_partial\_ratio + fuzzy\_token\_sort\_ratio + fuzzy\_token\_set\_ratio) / 4.0**

STEP:2

* **Preprocessing and Embedding**

**Ds new preprocessing steps**

**(Performing Data Pre-Processing )**

* Convert\_to\_lower (name):
* Replace\_adjacent\_duplicates (value):
* Replace\_bigrams (name):

replacements = {'ph': 'f', 'gh': 'g', 'th': 't', 'kh': 'k', 'dh': 'd', 'ch': 'c', 'sh': 's', 'au': 'o, 'bh': 'b', 'ks': 'x', 'ck': 'k', 'ah': 'h'}

* Replace\_characters\_unigram (name):

replacements = {'e': 'i', 'j': 'z', 'v': 'w', 'q': 'k'}

* Remove\_consonant\_a (name):

consonants = 'bcdfghjklmnpqrstvwxyz'

* Remove\_salutations (text):
* Remove\_relationship\_markers (name):
* Remove\_special\_characters (text):
* Remove\_stop\_words (text):
* SPECIAL\_CHAR\_DOT\_REGEX
* SPECIAL\_CHARS\_REGEX
* SALUTATION\_REGEX
* PARENT\_SPOUSE\_NAME\_REGEX COMMON\_MUSLIM\_SALUTATIONS\_MOHAMMAD\_REGEX
* LAST\_NAMES\_AGARWAL\_VARIANTS\_REGEX

**Convert to Lowercase**:

* + Standardize the case to ensure consistent matching.

**Replace Adjacent Duplicates**:

* + Example: Replace double letters or sequences that may lead to mismatch in similarity (e.g., "ff" to "f").

**Character Replacements**:

* + **Bigram Replacements**: Substitute common bigrams (e.g., {'ph': 'f', 'gh': 'g', ...}).
  + **Unigram Replacements**: Replace individual characters as per common linguistic variants (e.g., {'e': 'i', 'v': 'w'}).

**Remove Salutations & Relationship Markers**:

* + Salutations (Mr., Mrs., etc.) and relationship markers (parent/spouse indicators) are removed to avoid misleading similarity.

**Remove Special Characters**:

* + Use regex to eliminate unwanted characters that don’t contribute to name matching (e.g., punctuation, symbols).

**Stop Words Removal**:

* + Remove common words that don’t add value in matching names.

**Special Character Regex:**

* + **Remove dots or other special characters.**

**Salutation Regex:**

* + **Identify and remove common salutations.**

**Relationship Markers Regex:**

* + **Remove parent/spouse name indicators.**

**Name Variants Regex:**

* + **Cover variants specific to culture or common last name variations.**

STEP:3

**Then Using Embedding Model Over Pre-Processed Data**

Currently using this:- sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2

**Then Calculating Cosine Similarity**

calculate\_cosine\_similarity(embedding1, embedding2):

**Levenshtein Similarity**

def calculate\_levenshtein\_similarity(name1, name2):

**Phonetic Similarity**

def calculate\_phonetic\_similarity(name1, name2):

**Jaccard Similarity**

def calculate\_jaccard\_similarity(name1, name2):

**fuzzy\_similarity = (fuzzy\_ratio + fuzzy\_partial\_ratio + fuzzy\_token\_sort\_ratio + fuzzy\_token\_set\_ratio) / 4**

#### 

#### **Similarity Calculations**

1. **Embedding Model**:
   * Use sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2 or try variants like HF (all-MiniLM-L12-v2) to get embeddings, capturing semantic similarity.
2. **Fuzzy Similarity Measures**:
   * Combine multiple ratios for a comprehensive fuzzy similarity: \text{fuzzy\_similarity} = \frac{\text{fuzzy\\_ratio} + \text{fuzzy\\_partial\\_ratio} + \text{fuzzy\\_token\\_sort\\_ratio} + \text{fuzzy\\_token\\_set\\_ratio}}{4.0}
3. **Cosine Similarity**:
   * Compute cosine similarity on embeddings: cosine\_similarity=calculate\_cosine\_similarity(embedding1,embedding2)\text{cosine\\_similarity} = \text{calculate\\_cosine\\_similarity}(embedding1, embedding2)cosine\_similarity=calculate\_cosine\_similarity(embedding1,embedding2)
4. **Levenshtein Similarity**:
   * Measure the edit distance between names to calculate similarity.
5. **Phonetic Similarity**:
   * Use sound-based methods to capture phonetic resemblance.
6. **Jaccard Similarity**:
   * Calculate similarity by finding intersection over union of unique tokens.

STEP:4

**8- (Assigning Weight at Each Step)**

weight\_embedding = 0.4

weight\_levenshtein = 0.2

weight\_phonetic = 0.2

weight\_jaccard = 0.1

weight\_fuzzy = 0.2

**final\_score** = (

embedding\_similarity \* 0.3 +

levenshtein\_similarity \* 0.2 +

phonetic\_similarity \* 0.3 +

jaccard\_similarity \* 0.4 +

fuzzy\_similarity \* 0.6

)

return final\_score

STEP:5

* **Initials Matching Without Preprocessing**

**Tech Preprocessing used**

* Initials Letter Match

**Teach Preprocessing need to be Included**

**Example :**

**Name1-** Manu Kumar [M K]

**Name2-** Manu Kumar Enterprise [ M K E ]

**Result = False**

**Name1-** Manoj Kumar

**Name2-** Manoj K

**Result = True**

STEP:6

* **Fuzzy Wuzzy With Preprocessing**

**Example**

**Using only for False Positive Cases**

**Name1-** Manu kumar

**Name2-** Manu Kumar Tax

**Fuzzy Wuzzy - Will Pass with similarity above 80 then:**

**If pass :**

**Check keyword :**

**Result = False**

**What i Tried**

* Tech java Code Compiling for Inferencing on Locally
* Algorithm
  + - FuzzyWuzzy Ratio
    - FuzzyWuzzy Partial Ratio
    - FuzzyWuzzy Token Sort Ratio
    - FuzzyWuzzy Token Set Ratio
    - FuzzyWuzzy Similarity
    - Sequence Matcher Similarity
    - Levenshtein Similarity
    - HF(all-MiniLM-L6-v2)Similarity
    - Pretrained Siamese Similarity
    - HF(all-MiniLM-L12-v2)
    - Jaccard Similarity
    - HF(paraphrase-multilingual-MiniLM-L12-v2)
    - PollyFuzz
* Embedding model -

**Done:-**

* Bert Base
* Bert Large
* Siamese Bert Base
* Siamese Bert Large

**Need to Be Done:-**

* Siamese

**Ds Old Logic**

****

**Tech Logic On Java Code**

****

**New DS Logic**

**New Approach**

,

**FINAL**

,