A graph of a graph with a curved line

Description automatically generated with medium confidence

A graph of a graph with a line

Description automatically generated with medium confidence

A graph of a diagram

Description automatically generated with medium confidence

import pandas as pd

from sentence\_transformers import SentenceTransformer

from sklearn.metrics.pairwise import cosine\_similarity

from scipy.spatial.distance import euclidean, cityblock

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.metrics import jaccard\_score

import Levenshtein

# Load the dataset

file\_path = r"C:\Users\srini\Downloads\financial\_name\_matching\_corrected (1).csv"  # Update path if needed

df = pd.read\_csv(file\_path)

# Load the Sentence Transformer model

embedding\_model = SentenceTransformer("all-mpnet-base-v2", device="cpu")

# Function to calculate Jaccard similarity

def jaccard\_similarity(str1, str2):

    vectorizer = CountVectorizer(binary=True).fit([str1, str2])

    vecs = vectorizer.transform([str1, str2]).toarray()

    return jaccard\_score(vecs[0], vecs[1])

# Function to calculate Levenshtein similarity

def levenshtein\_similarity(str1, str2):

    return 1 - (Levenshtein.distance(str1, str2) / max(len(str1), len(str2)))

# Compute embeddings

df["Name\_Embedding"] = df["Name"].apply(lambda x: embedding\_model.encode(x))

df["Name\_Variant\_Embedding"] = df["Name\_Variant"].apply(lambda x: embedding\_model.encode(x))

# Compute cosine similarity

df["Cosine\_Similarity"] = df.apply(

    lambda row: cosine\_similarity(

        row["Name\_Embedding"].reshape(1, -1), row["Name\_Variant\_Embedding"].reshape(1, -1)

    )[0][0],

    axis=1

)

# Compute Euclidean distance

df["Euclidean\_Distance"] = df.apply(

    lambda row: euclidean(row["Name\_Embedding"], row["Name\_Variant\_Embedding"]),

    axis=1

)

# Compute Manhattan distance

df["Manhattan\_Distance"] = df.apply(

    lambda row: cityblock(row["Name\_Embedding"], row["Name\_Variant\_Embedding"]),

    axis=1

)

# Drop embeddings to keep the dataset clean

#df.drop(columns=["Name\_Embedding", "Name\_Variant\_Embedding"], inplace=True)

# Save the results

output\_path = "name\_matching\_with\_similarities.csv"