Chapter 1

Introduction

Apart from the string matching algorithms, two major classes of string algorithms are discussed here

1. Approximate string matching (including sequence alignment)
2. Phonetic based

Python package **jellyfish** implements Levenshtein,Damareu-Levenshtein, Jaro, Jaro-Winkler for approximate string comparison and some Phonetic based algorithms.

In this source repo, I want to complement this algorithm with NeedleMan-Wunsch, Monge-Eklan, Soft-TFIDF, cosine similarity, N-gram similarity function.

This document describes the equations for the algorithms.

Note: The source code is in active development.

1.1 Jaro-Winkler

$$d_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & \text{otherwise} \end{cases} \tag{1.1}$$

where

- m is the number of matching characters
- t is half the number of transposition

The Jaro-winkler extension is

$$d_w = d_j + lp(1 - d_j) \tag{1.2}$$

where

- l length of common prefix
- p scaling factor p should not exceed 0.25.

1.2 TF-IDF

1.2.1 Term Frequency -TF

$$w_{t,d} = \begin{cases} 1 + \log_{10} tf_{t,d} & \text{if } tf_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{1.3}$$

where

- $tf_{t,d}$ is the number of time the term appear in document.

1.2.2 Inverse term frequency -IDF

$$idf_t = \log_{10} \frac{N}{df_t} \tag{1.4}$$

where

- N number of documents in the collection

1.2.3 TFIDF wiegths

$$w_{t,d} = (1 + \log_{10} tf_{t,d}). \log_{10} \frac{N}{df_t} \tag{1.5}$$

1.3 Cosine similarity

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta \tag{1.6}$$

$$similarity(\overrightarrow{q}, \overrightarrow{d}) = \cos(\overrightarrow{q}, \overrightarrow{d}) = \frac{\overrightarrow{q} \cdot \overrightarrow{d}}{\|q\|\|d\|} = \frac{\sum_{i=1}^{|v|} q_i \times d_i}{\sqrt{\sum_{i=1}^{|v|} (d_i)^2} \times \sqrt{\sum_{i=1}^{|v|} (d_i)^2}} \tag{1.7}$$

where

- $\overrightarrow{q}$ is the query vector representation with terms TFIDF scores.
- $\overrightarrow{d}$ is the document vector representation with term TFIDF scores.

1.4 Monge-Elkan

$$Sim_{MongeElkan}(x, y) = \frac{1}{|x|} \sum_{i=1}^{|x|} max_{j=1,|y|} sim'(x[i], y[j]) \tag{1.8}$$

where

- $|x|$ number of tokens in x
- sim' inner similarity function. Example: Jaro-Winkler distance.

1.5 Soft-TFIDF

$$Sim_{softTFIDF}(x,y) = \sum_{t \epsilon CLOSE(\theta, tok(x), tok(y))} V(t, tok(x)).V(t, tok(y)).N(t, tok(y)) \tag{1.9}$$

Where

- $V(t, tok(x))$ is TFIDF weight of term t in all token of x
- $N(t, tok(y)) = max_{v \epsilon T} sim^{'}(u,v)$. $sim^{'}$ similarity of best matching algorithm. Example: Jaro-Winkler