ISSN PRINT 2319 1775 Online 2320 7876

Research paper  UGC CARE Listed (Group -I) Journal Volume 8, Issue 3, 2019

# Implementation of String Similarity Metrics for Document Clustering

**N. SreeRam**

Department of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, India, sriramnimmagadda@gmail.com

**Abstract**

Document clustering is an important task in information retrieval of text mining. The proximity measures such as similarity or dissimilarity measures are used in any kind of clustering. However, in document clustering there is need of string similarity measures. String similarity measure is the basic task in document clustering, sentimental analysis, information retrieval of text mining and Natural Language Processing. There exists variety of string similarity metrics that are using for sentimental analysis such as Jaccard coefficient, Edit distance, Cosine similarity, Soundex algorithm, Hamming distance, Levensntein distance. In this paper we are presenting the overview of various string similarity metrics with their implementation in machine learning software

**Keywords:** Document clustering, text mining, proximity measures, string similarity metrics.

## 1. Introduction

Document grouping [2] is one of the important tasks in text mining. The process of grouping documents is generally based on proximity among them. So, document segregation process involves the measuring of either similarity or dissimilarity among them. Proximity of documents usually represent with a value from 0 to 1. Proximity value 0 indicates completely dissimilar whereas 1 indicates completes similar. This document proximity can measure by using string similarity metrics. String Similarity Search is a fundamental query that has been widely used for DNA sequencing, error-tolerant query auto completion, and data cleaning needed in database, data warehouse, and data mining. In this paper, we study string similarity search based on edit distance that is supported by many database management systems such as Oracle and PostgreSQL. Given the edit distance, ed(s,t), between two strings, s and t, the

ISSN PRINT 2319 1775 Online 2320 7876

Research paper  UGC CARE Listed (Group -I) Journal Volume 8, Issue 3, 2019

string similarity search is to find every string t in a string database D which is similar to a query string s such that ed(s,t)<= ĩ for a given threshold ĩ. In the literature, most existing work takes a filter-and-verify approach, where the filter step is introduced to reduce the high verification cost of two strings by utilizing an index built offline for D. The two up-to-date approaches are prefix filtering and local filtering. In this paper, we study string similarity search where strings can be either short or long. This approach can support long strings, which are not well supported by the existing approaches due to the size of the index built and the time to build such index. We propose two new hash-based labelling techniques, named OX label and XX label, for string similarity search. We assign a hash-label, Hs, to a string s, and prune the dissimilar strings by comparing two hash-labels, Hs and Ht, for two strings s and t in the filter step. The key idea is to take the dissimilar bit-patterns between two hash-labels. The hash-based approaches address their pruning power and give the algorithms. This hash-based approaches achieve high efficiency and keep its index size and index construction time one order of magnitude smaller than the existing approaches in our experiment at the same time. STRINGS are widely used to represent a variety of textual data including DNA sequences, messages, emails, product reviews, and documents. And there are a large number of string datasets collected from various data sources in real applications. Due to the fact that string data from different sources may be inconsistent caused by the typing mistakes or the differences in data formats, as one of the most important fundamental tasks, string similarity search has been extensively studied which checks whether two strings are similar enough for data cleaning purposes in databases, data warehousing, and data mining systems.

The applications that need string similarity search include fuzzy search, query auto-completion, and DNA sequencing. In the literature, there are two categories to measure string similarity. One is token-based similarity metric including overlap, Jaccard, cosine and dice, and the other is the character-based similarity metric including edit distance. The edit distance, ed(s,t), between two strings, s and t, is the minimum number of operations (substitution, insertion, and deletion) required to transform one string to another string. In this paper, we focus on string similarity search based on edit distance, which is supported in many database management systems such as Oracle, PostgreSQL and Lucene, and is used in detecting spammers and DNA sequence alignment. The challenge of the string similarity search is to design an effective index that can achieve high efficiency for query processing

ISSN PRINT 2319 1775 Online 2320 7876

Research paper  UGC CARE Listed (Group -I) Journal Volume 8, Issue 3, 2019

with small overhead in the index size and the indexing time. It becomes more challenge in the age of big data since the string datasets become increasingly large from two aspects, the lengths of the strings and the number of strings insides. The existing approaches need to build a large index when the strings are longer in a dataset. The larger the index is, the more time it requires to process queries. Accordingly, the performance decreases. In addition, such performance will be affected by a large number of strings in a dataset. In this paper, we focus on new hash-based approaches to deal with large short/long string datasets

## 2. Background work

There are variety of string similarity measures as discussed here under

**Edit Distance:**

Given the edit distance, ed(s,t); between two strings, s and t, is the minimum number of operations (substitution, insertion, and deletion) required to transform one string to another string.

Examples:

1.Dog edit distance -1
Dof

2.Cat edit distance -2
Act

3.Cat 3 substitution operations; edit
Dog distance – 3 (min)
(or)

Deleting 3 characters of cat and inserting 3 characters of dog;

Edit distance – 6

Consider two strings s1 = a b c d e f and s2 = a z c e d. The following table represents the number of operations required to transform s1 to s2

ISSN PRINT 2319 1775 Online 2320 7876

Research paper

 UGC CARE Listed (Group -I) Journal Volume 8, Issue 3, 2019

**Table 1** operations required to transform S1 to S2

| | | A | b | c | d | e | f |
|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| A | 1 | 0 | 1 | 2 | 3 | 4 | 5 |
| Z | 2 | 1 | 1 | 2 | 3 | 4 | 5 |
| C | 3 | 2 | 2 | 1 | 2 | 3 | 4 |
| E | 4 | 3 | 3 | 2 | 2 | 2 | 3 |
| D | 5 | 4 | 4 | 3 | 2 | 3 | **3** |

**Algorithm:**

```
if(str[i]==str[j])
        T[i][j] = T[i-1[j-1]      //diagonal value
else
        T[i][j] = min(T[i-1[j],T[i-1][j-1],T[i][j-1])    //min of left, top, diagonal values
```

1)Consider two strings s = “effcionsy” and t =“efficiency”.

2)Assume the edit distance threshold t used for

ed(s,t); t=2, and q= 2 in this example.

3)The 2-gram set of string s is

Qs = {$e, ef, ff, fc, ci, io, on, ns, sy,y$}

the 2-gram set of string t is

Qt = {$e, ef, ff, fi, ic, ci, ie, en, nc, cy y$}.

4)Suppose the hash-labels used are of fixed size for L = 20,

and there is a hash function h ,which hashes a 2-gram

in the range of [1,20].

5)Table 1 shows the hash values of the q-grams

for Qs and Qt.

6) Note that there are hash collisions here. Both ‘ff’ and ‘sy’ are hashed into 14th bit position,

and both ‘fc’ and ‘io’ are hashed into 5th bit position in Fig.2

ISSN PRINT 2319 1775 Online 2320 7876

Research paper

 UGC CARE Listed (Group -I) Journal Volume 8, Issue 3, 2019

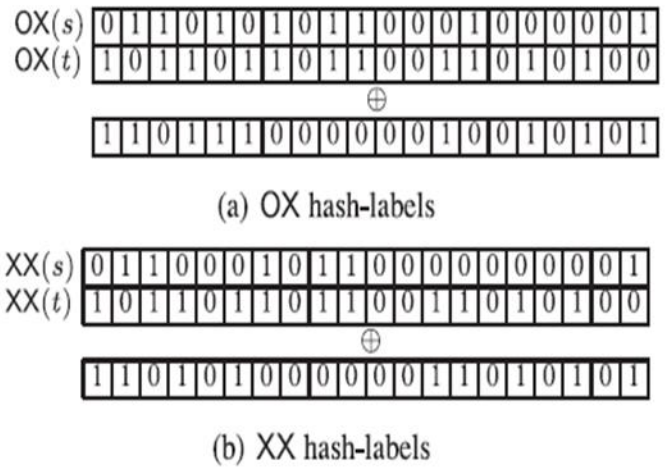


**Figure.1: Two different hash tables for the two strings s,t**

Fig. 1a shows the OX hash-labels, Hs and Ht, for s and t, and the resulting bit-sequence H by exclusive-or () of the two hash-labels. The number of different bits in H is 9. Hence, we can conclude that the two strings are not similar.

**Cosine Similarity**:

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any other angle in the interval [0,2π). It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1]. The name derives from the term "direction cosine": in this case, note that unit vectors are maximally "similar" if they're parallel and maximally "dissimilar" if they're orthogonal (perpendicular). This is analogous to the cosine, which is unity (maximum value) when the segments subtend a zero angle and zero (uncorrelated) when the segments are perpendicular.

Note that these bounds apply for any number of dimensions, and cosine similarity is most commonly used in high-dimensional positive spaces. For example, in information retrieval and text mining, each term is notionally assigned a different dimension and a document is characterised by a vector where the value of each dimension corresponds to the

ISSN PRINT 2319 1775 Online 2320 7876

Research paper  UGC CARE Listed (Group -I) Journal Volume 8, Issue 3, 2019

number of times that term appears in the document. Cosine similarity then gives a useful measure of how similar two documents are likely to be in terms of their subject matter

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

**Table 2** Document vector

| Document | Team | Coach | Hockey | Baseball | Scorer | Penal | Score | Win | Loss | Sea |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 0 | 3 | 0 | 2 | 0 | 0 | 2 | 0 | 0 |
| 2 | 3 | 0 | 2 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 3 | 0 | 7 | 0 | 2 | 1 | 0 | 0 | 3 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 1 | 2 | 2 | 0 | 3 | 0 |

**Hamming Distance:**

Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different.

i.e.., the minimum number of errors that could have transformed one string into the other.

Ex: The Hamming distance between:

"karolin" and "kathrin" is 3.

"karolin" and "kerstin" is 3.

1011101 and 1001001 is 2.

2173896 and 2233796 is 3.

SOUNDEX ALGORITHM:

- Soundex is a phonetic algorithm for indexing names by sound, as pronounced in English.
- The goal is for homophones to be encoded to the same representation so that they can be matched despite minor differences in spelling.
- The algorithm mainly encodes consonants; a vowel will not be encoded unless it is the first letter.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper  UGC CARE Listed (Group -I) Journal Volume 8, Issue 3, 2019

Algorithm:

1. Retain the first letter of the word
2. Change all occurrences of the following letters to '0'

'A','E','I','O','U','H','W','Y'

3. Change letters do digits as follows:

B,F,P,V -> 1

C,G,J,K,Q,S,X,Z -> 2

D,T -> 3

L -> 4

M,N -> 5

R -> 6

4. Remove all pairs of consecutive digits.
5. Remove all zeros from the resulting string.
6. Pad the resulting string with trailing zeros and return the first four positions, which will be of the form <uppercase letter> <digit> <digit> <digit>

Ex: Herman becomes H655

i.e.., Herman

H0rm0n (by step 2)

H06505 (by step 3)

H655 (by step 5)

Example 2:

Hermann -> H0rm0nn (by step 2)

H065055 (by step 3)

H06505 (by step 4)

H655 (by step 5)

Hence both Herman and Hermann becomes H655.

Therefore, the documents containing either of these words will be mapped to same H655 code

LEVENSNTEIN DISTANCE:

- Levensntein distance is a string metric for measuring the difference between two sequences

ISSN PRINT 2319 1775 Online 2320 7876

Research paper  UGC CARE Listed (Group -I) Journal Volume 8, Issue 3, 2019

- The distance between two words is the minimum number of single-character edits required to change one word into the other

## 3. Implementation

The above discussed string similarity metrics are implemented through R[8] studio and demonstrated the outputs generated. R is a programming language and free software environment for statistical computing and graphics that is supported by the R foundation for statistical computing. R[8] is a GNU package. R and its libraries implement a wide variety of statistical and graphical techniques including linear and nonlinear modelling, classical statistical tests, time series analysis, classification, clustering and others. The following R script demonstrate the various string similarity metrics

```
library(stringdist)
x<-"Good phone at this price "
y<-"It's always good to install updates as the updates fixes bugs and adds new features. "
stringsim(x,y,method="osa")
stringsim(x,y,method="lv")
stringsim(x,y,method="dl")
stringsim(x,y,method="hamming")
stringsim(x,y,method="lcs")
stringsim(x,y,method="qgram")
stringsim(x,y,method="cosine")
stringsim(x,y,method="jaccard")
stringsim(x,y,method="jw")
stringsim(x,y,method="soundex")
```

ISSN PRINT 2319 1775 Online 2320 7876

Research paper  UGC CARE Listed (Group -I) Journal Volume 8, Issue 3, 2019

```
> library(stringdist)
> x<-"Good phone at this price "
> y<-"It's always good to install updates as the updates fixes bugs and adds new features. "
> stringsim(x,y,method="osa")
[1] 0.1882353
> stringsim(x,y,method="lv")
[1] 0.1882353
> stringsim(x,y,method="dl")
[1] 0.1882353
> stringsim(x,y,method="hamming")
[1] 0
> stringsim(x,y,method="lcs")
[1] 0.3272727
> stringsim(x,y,method="qgram")
[1] 0.4
> stringsim(x,y,method="cosine")
[1] 0.7821525
> stringsim(x,y,method="jaccard")
[1] 0.48
> stringsim(x,y,method="jw")
[1] 0.5369561
> stringsim(x,y,method="soundex")
[1] 0
```

## 4. Conclusions

The main work in this paper gives an overview of various kinds of string similarity metrics and algorithms used by those metrics with some examples. All the metrics discussed above are implemented in machine learning software known as R programming. Each kind of metric yields different kind of values ranging from 0 to 1. 0 indicates dissimilar and 1

ISSN PRINT 2319 1775 Online 2320 7876

Research paper  UGC CARE Listed (Group -I) Journal Volume 8, Issue 3, 2019

indicates similar. In the implementation we have taken two reviews given by a customer from amazon website and measured similarity between two reviews. So, these string similarity metrics are very useful for documenting clustering of text mining.

**References**


1. Ekaterina Chernyak "Comparison of String Similarity Measures for Obscenity Filtering" Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing, pages 97–101, Valencia, Spain, 4 April 2017. c2017 Association for Computational Linguistics
2. M.K. Vijaymeena and K.Kavitha "A survey on similarity measures in Text Mining" Machine Learning and Application: An International Journal (MLAIJ) vol .3,No.1 March 2016
3. A. Arasu, V. Ganti, and R. Kaushik, "Efficient exact set-similarity joins," in Proc. Int. Conf. Very Large Data Bases, 2006, pp. 918–929.
4. S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani, "Robust and efficient fuzzy match for online data cleaning," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2003, pp. 313–324.
5. D. Deng, G. Li, and J. Feng, "A pivotal prefix based filtering algorithm for string similarity search," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2014, pp. 673–684.
6. Dr.Mark Gardener Beginning R The statistical Programming Language wrax
7. Yanchang Zhao RDM R and Data Mining: Examples and case studies
8. https://en.wikipedia.org/wiki/R_(programming_language)