how to implement jaccard coefficient in java?

Asked 5 years ago Modified 5 years ago Viewed 4k times

1

Assume I have 2 strings like this.

Query1: "Ideas of March"

Query2: "Ceaser died in March"

Function(j) = (Query1 intersection Query2)/ (Query1 union Query2)

I am looking at accuracy with respect to number of tokens(words), irrespective their position.

Query1 intersection Query2 = 1 {March}

Query1 union Query2 = 6 {Ideas, of, March,Ceaser, died, in}

In this context Function(j) should return 1/6.

Is there anyway I can find the intersection count and union count of two sentences? For an example, in here,

```
public double calculateSimilarity(String  oneContent,  String otherContent)
{
    double numerator   = intersection(oneContent,otherContet);
    double denominator = union(oneContent,otherContet);

    return denominator.size() > 0 ?
    (double)numerator.size()/(double)denominator.size() : 0;
}
```

Is these any available function in Java to get intersection count and union count without using any external libraries like Google Guava?

java string function string-comparison

Share Improve this question Follow

edited Apr 14, 2017 at 13:41

asked Apr 14, 2017 at 13:26

Yash
235 1 7 18

Note that the Jaccard Index / Tanimoto coefficient has no single definition. Rather, it is a method of relating an intersection and a union to an index of accuracy, and therefore its interpretation depends on the particular intersection and union measures used. You need to be specific what you're after, e.g. accuracy in terms of number of characters in common when treated independent of position, or number of characters in common given optimal alignment (which is then subject to the particular alignment algorithm used). – Tasos Papastylianou Apr 14, 2017 at 13:35

1 @TasosPapastylianou I am looking at accuracy with respect to number of tokens(words), irrespective their position. – Yash Apr 14, 2017 at 13:43

In that case, tokenise and sort alphabetically (ignoring case or not, up to you) the concatenation of the two texts (i.e. keeping only unique tokens), then make two arrays keeping counts for the number of times a word appears for each individual text. Your interesection is the sum of the minimum between the two arrays over all tokens, and your union is the sum of the corresponding maximum.
– Tasos Papastylianou Apr 14, 2017 at 13:54

2 Answers

Sorted by: Highest score (default)

1

You can use Apache commons text, which has no other external dependency. (https://commons.apache.org/proper/commons-text/)

Text similarity algorithm documentation: https://commons.apache.org/sandbox/commons-text/apidocs/org/apache/commons/text/similarity/package-summary.html

And you can find the Jaccard Coefficient implementation here: https://github.com/apache/commons-text/blob/master/src/main/java/org/apache/commons/text/similarity/JaccardDistance.java

Share Improve this answer Follow

answered Apr 14, 2017 at 13:33
jchampemont
2,525 15 19

-1

As you are only interested in the size of the union/intersection, you can calculate the size of these two sets without actually creating the union and intersection set (`union(a, b).size()` is `a.size() + b.size() - intersection(a, b).size()` -> only intersection size is required).

```
public static void main(String[] args) {
    final String a = "Ideas of March";
    final String b = "Ceaser died in March";
    final java.util.regex.Pattern p
        = java.util.regex.Pattern.compile("\\s+");
    final double similarity = similarity(
            p.splitAsStream(a).collect(java.util.stream.Collectors.toSet()),
            p.splitAsStream(b).collect(java.util.stream.Collectors.toSet()));
    assert similarity == 1d / 6;
    System.out.println(similarity); // 0.1666...
}

public static double similarity(Set<?> left, Set<?> right) {
    final int sa = left.size();
    final int sb = right.size();
    if ((sa - 1 | sb - 1) < 0)
        return (sa | sb) == 0 ? emptyJaccardSimilarityCoefficient : 0;
    if ((sa + 1 & sb + 1) < 0)
        return parallelSimilarity(left, right);
    final Set<?> smaller = sa <= sb ? left : right;
    final Set<?> larger  = sa <= sb ? right : left;
    int intersection = 0;
    for (final Object element : smaller) try {
        if (larger.contains(element))
            intersection++;
    } catch (final ClassCastException | NullPointerException e) {}
```

```
        final long sum = (sa + 1 > 0 ? sa : left.stream().count())
                       + (sb + 1 > 0 ? sb : right.stream().count());
        return 1d / (sum - intersection) * intersection;
    }
```

Share Improve this answer Follow

answered Apr 14, 2017 at 15:26

Nevay

754 5 9