Edit Distance for Approximate String Search on Geolocation of Tweets

Xichang Zhao

Abstract

This paper introduces a new approach for solving the problem of approximate string search, specifically on the searching of locations from a large data of tweets. The software uses the global edit distance algorithm as the core method for tackling the task. Additionally, mechanisms are designed exclusively for solving the specific problem of finding locations in tweets. The accuracy of result for this approach appeared to be optimistic, though the efficiency can still be large improved.

Introduction

Given a location file containing location names and their corresponding information, and a tweet file containing one tweet per line with its user id tweet id and timestamp, the task asks to search for the tweets that have words match with the locations listed in the location file. The algorithm using for this problem is the global edit distance.

Method

In order to correctly implement this algorithm, each tweet in the tweet file must be split into words (by empty space). This is easily processed in Python by using the built-in string method str.split().

In actual implementation, the software extracts the locations (queries) and tweets successively from the files, obtaining two lists. The location list contains a list of location objects, with each object containing a string for location name, and an integer indicating the number of words in that location string. This count is useful at later stage to determine how many words of tweet to test each time when calculating the edit distance. The tweet list contains a list of tweet objects, with each object containing a list of words which are split from a complete tweet, and a user id whom the tweet is created from. After extracting, a function is called to implement the algorithm. An outer loop iterates every query object in the list to test the edit distance with each word of each tweet in the tweet list. The definition of “word” here is defined by the count of words for each query. If a query has two words, then two words from each tweet is tested at a time with it. This mechanism remedies the defect of treating the whole tweet string as words, thus if a word at a time is tested with a multiple-words query, the edit distance will be huge and result will be inaccurate.

RESULT

The sample results below are produced under the default threshold setting of 1. This threshold is chosen due to several reasons. First is that the threshold cannot be 0 since this loses the meaning of "approximate string searching" and turns into exact string search. It is also not optimal to let the threshold be 2 since this allows too much room for differences. In reality, misspelling for human unlikely causes two or more edits to remedy, therefore, threshold of 1 should be the most appropriate choice.

	Goose Hill	Mountain Creek Baptist Church	Seventyseven th Street Church of Christ	Call
@SuaveSerg lol I think **Mountain Creek Baptist Churca** I'll survive! I'm calling u when its time to go to memphis!!!		1		
Q&A: Director **Goose Hill** CLAIRE DENIS (Chocolat) on her new film, White Material http://bit.ly/2viPhg	1			
@PrinceSammie happy thanksgiving bro **Seventyseventh Street Church of Christ**!			1	
my photography teacher brought in books for everyone to look at in the class **Seventyseventh Street Church of Christ** and she brought me in a book on food photography. love it!			1	
the gift of art is "not" what **Seventyseventh Street Charch of Chbist** they expect--original art starting at 50.00 at www.deanrussoart.etsy.com (via @deanrussoartist)				
@ReDoubleD **ALL** DAY!! THATS REAL!				1

The table above shows the result from the sample location file "b.txt" and tweet file "a.txt". Rows are tweets and columns are queries. The first row shows a match with the second column, this is correct even though the underlined word is one character different from the query. The second row shows an exact match between query and a word in the tweet. Third and fourth row are for the purpose of testing multiple-words queries, the software returns matches correctly when the queries are 5 words long. The underlined word in fifth row has two characters difference against the query, the software therefore did not return matches. The last row is an example of the sort of returns that are not wanted to see. The "all" here is not a misspelling of the location "call", yet the software still returned match under the threshold of 1. This is unavoidable under current algorithms, one solution might be to have a list of words that are frequently appeared in English texts but can't possibly be misspelled to (ex. "the", "of"), when matches are found for these words, the software would not return. This could be the direction of improvement for this software in future. In general, the result produced by the software can be seen as accurate.

Scalability

The method for this software is descripted in previous sections, an outer loop iterates x (number of queries) times, an inner loop iterates y times (number of tweets). In each inner loop, there is another loop iterates z times (number of words in a tweet), each word (t long) does edit distance with the query (s long). Therefore the total running time should be O(xyzts) (since edit distance takes $O(n^2)$ time). Simplifying this, if assuming z is the length of the whole tweet string, more further, if m is the length of whole queries combined, and n is z*y (length of a tweet times quantity), the time complexity of this software would be O(mn), polynomial growth.

The software takes about 3 minutes to complete running when the sample texts are "US_small.txt" and "trainingset_tweet_small.txt". These two files are the 0.1% sub-samples of the original files, thus the execution time for running original files would be 1000000 times longer, which is 3*1000000/60=5000h. This is way beyond acceptable. It proves that the algorithm chosen for tackling this task is inefficient.

Analysis

Choosing global edit distance approach has the advantage of being easy to implement on the programming side, and allowing mismatches to happen by choosing appropriate threshold. The results produced are also more accurate comparing to N-gram.

On the other hand, running the software takes enormous amount of time comparing to other approaches (local edit distance, tries), and it will grow in polynomial when input gets larger. This could be improved by shifting to more running-time-efficient algorithms.

Challenges that have been encountered were mainly the inconsistency of format for input files and dealing with multiple-words query. The first was solved by adding special cases to eliminate them and the second was solved by adding in indicators to determine how many words to test at a time.

Conclusion

The global edit-distance approach produces a relatively accurate result, the yardstick can be easily adjusted by changing appropriate threshold. Though it is less inefficient running-time wise, it is still highly recommended when input scale is relatively small.

References

Allison, L. (1999). *Dynamic Programming Algorithm (DPA) for Edit-Distance*. Retrieved from Monash University Website: http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Dynamic/Edit/

Levenshtein distance. (2014). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Levenshtein_distance