Effectiveness of Approximate String Search with N-Word and Edit Distance Matching Strategy

Jian-Heng Tang

1 Introduction

Social network community has become quite popular in recent years. Many people share information via community so that hundreds of thousands of messages are created everyday. These messages are too large to allow us to find desirable information from posts by adopting a simple single method. This is because the content of information we are searching for is so largely different that we have to take different analytical methods into considerations. The information, however, still have common attribute that we can classify them into category for evaluating effectiveness. Hence, this report will focus on the effectiveness of evaluating a mixture of approximate match methods to identify the appropriate information from Tweeter texts.

2 Query Evaluation and Match Method

The aim of this report is to find the best approximate matches to a set of queries including US locations and a set of target strings from Tweet texts. The data set of query is from one query file, file size is 1K. The data set of tweet is from one target file, file size is 3K.

2.1 Query Type

The query in the file is composed of different part of tokens, for each token can be expressed as explicit or common location, abbreviation or whole length, and noun or preposition. For example, "Lakehill Shopping Centre" are composed of explicit location "Lakehill" and common expression "shopping centre". L.A. is the abbreviation of Los Angels. "Village of Evergreen Park" are composed of 3 parts of noun and one preposition "of".

The type of query can be divided into two categories: string model and token model. The string model is one location as a whole string of a query. The token model is one location can be separated into a set of one to three consecutive words.

The string model and token model have their advantages and disadvantages. First, the string model has less number of comparison, which means that the execution time is faster than the token model. The drawbacks, however, is that the comparison could match common words, such as "High School" or match the preposition, such as "of", making the search result become meaningless.

Second, the token model seems not to have the drawbacks that the string model has. The queries with only one to three token are easier to find the location name in tweet texts, leaving the preposition or common expression behind. This makes comparison result become more accurate. Conversely, the disadvantage of the token model is that it will take lots of searching time to finish the token comparison. For instance, if I divide one string into three tokens, the number of comparisons of each tweet text becomes triple. This means that searching by tokens is an exhaustive search, increasing the time and space complexity.

2.2 Approximate Match Method

The approximate matching strategy is local edit distance (LED) for string model, and global edit distance (GED) for token model respectively. But there is a problem: how do I know which token in a query best match the location name? My method is that, for string model, I simply execute query and tweet comparison by LED. For token model, I parse query texts and tweet texts into separate one to three consecutive words as a sub-query string and sub-tweet string and then run GED algorithm. I call the consecutive word as N-word. For instance, if there are three consecutive words, it is called 3-word.

3 Result and Effectiveness

3.1 String Model

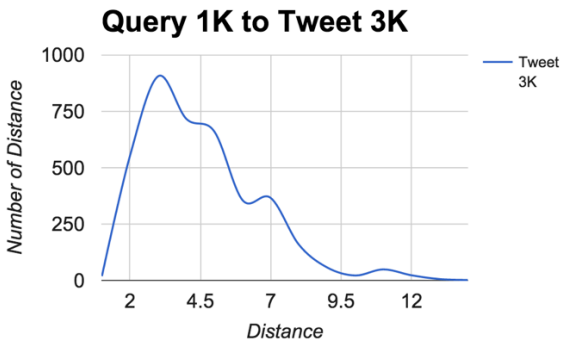

Figure 1. Distribution of LED between Query 1K and Tweet 3K

The graph in Figure 1, the largest distance of LED is 14 and there are only 2 matches. This means that a large proportion of matches are meaningless. Even that two matches the result is quite irrelevant to location name.

3.2 Token Model

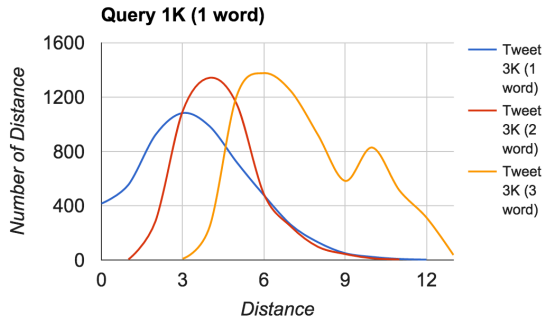

Figure 2. Distribution of Distance between Query 1K 1-word and Tweet 3K

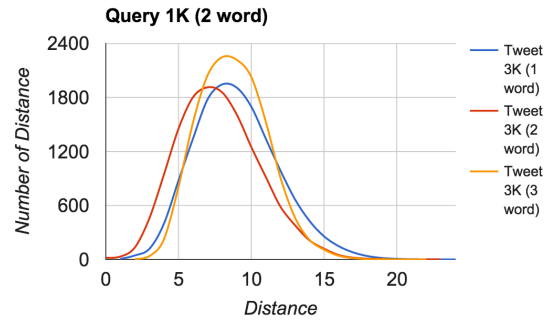

Figure 3. Distribution of Distance between Query 1K 2-word and Tweet 3K

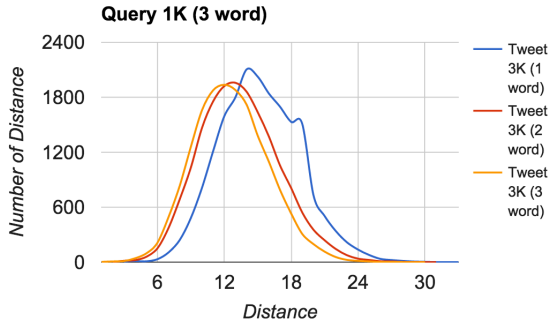

Figure 3. Distribution of Distance between Query 1K 3-word and Tweet 3K

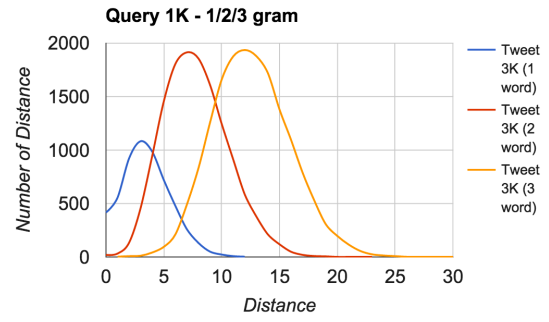

Figure 5. Distribution of Distance between Query 1K 1/2/3-word and Tweet 3K 1/2/3-word

The graph in Figure 2, Tweet 3K 1-word and Tweet 3K 2-word are normal distribution. The graph is far from normal distribution as the number of word is getting larger. I think this is because when matching Tweet 3-word by query 1-word, there are more mismatch words when tweet is changing from 1-word to 3-word. Likewise, in Figure 4, there are more mismatch words when tweet is changing from Tweet 3-word to Tweet 1-word. Thus, the best match in Figure 2 and Figure 4 should be query 1-word to tweet 1-word and query 3-word to tweet 3-word.

In Figure 3, the distribution and distance is quite similar because the query is 2-word, but tweet 2-word has less distance compared to other tweet N-word. Thus, best match in Figure 2 should be query 2-word to tweet 2-word.

	1-1*	2-2*	3-3*
Smallest Distance	0	0	1
Total Number	416	20	3
Relevant	9.6%	25%	33%
Irrelevant	90.4%	75%	67%

Table 1: The percentage of recognizable location name. *1-1 means that Query 1-word to Tweet 1-word.

The number in Table 1, although the Tweet 1-word from query 1-word has 416 data that distance is zero, only about 9.6% of matches can be recognised as location name, such as “Boston”. The rest of data about 90.4% is ambiguous, too common or meaningless, such as “Luther”, “School” or “of”.

The relevant percentage from 3-3 can reach to 33%. However, the recognizable locations of 2-2 can be found in 3-3, no matter 3-3 is relevant or irrelevant. This means that 2-2 has more number of matches than 3-3. Moreover, in Figure 5, the distribution of 2-word and 3-word is quite similar, but the overall distance of 2-word is much less than 3-word. Therefore, I think that 2-2 is more effective than the others because there are more location name found in the smallest distance.

3.3 Corpus of Models

Now that 2-2 is the best match, I started analysing the best case and worst case of 2-2 in Table 2 and Table 3.

	Best Case of Token Model
Corpus (Query)	**San Francisco** Hot Springs
Corpus (Tweet)	5671000230 Chicago vs **San Francisco** Late Betting Line Moves httpbitlyWDHVS
Distance	0 (Smallest number from GED)

Table 2: Query 2-word to Tweet 2-word: Best Case

	Worst Case of Token Model
Corpus (Query)	Cheyney University of **Pennsylvania Chickasawhatchee** Baptist Cemetery
Corpus (Tweet)	1471339544 The Examiner is keeping an eye on the sky **and the** Giants home opener httpbitlyoyyIK
Distance	23 (Largest number from GED)

Table 3: Query 2-word to Tweet 2-word: Worst Case

The worst case is that 2 consecutive token is one from one location name and the second token is from the next. And the match is quite different.

	String Model
Corpus (Query)	Lake**hill Shopping Center**
Corpus (Tweet)	5894814858 RT darrencurrin Windsor **Hill**s **shopping center** in Oklahoma City reopens its movie theater under new ownership httptinyurlcomyklps
Distance	14 (Largest number from LED)

Table 4: String model and features: Best Case

In Table 4, LED method examines the string locally, so only “hill”, “shopping”, and “centre” match up with each other.

Therefore, we can see the result from two 3 tables, the token model can exactly find the location name in tweet texts. Thus, the token model is more effective than string model.

4 Scalability

In Figure 6, Given that the size of tweet data is 3K, total execution time increase as the query data size and N-word increase. This is because as the total number of queries increase, the total execution time goes up as well.

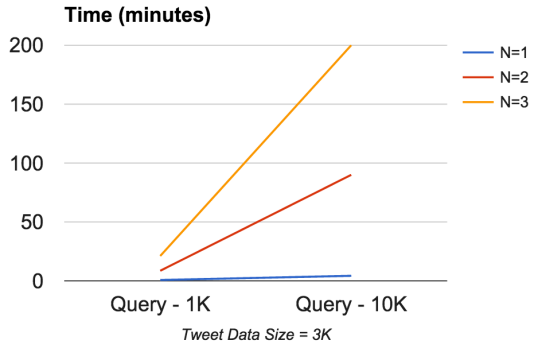

Figure 6. Performance of Queries

In Figure 7, Comparing to string model, overall performance of token model is much slower. In token model, since many strings are divided into many tokens as a query, the total number of queries increases, the total execution time will increase.

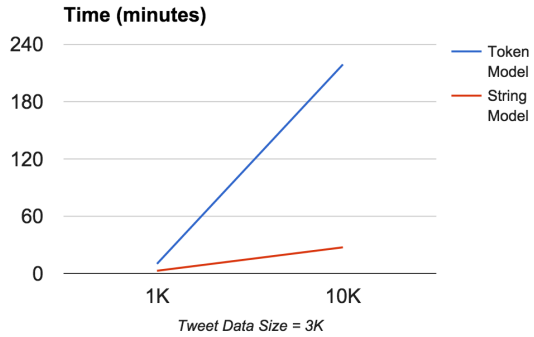

Figure 7. Performance of Two Models

5 Conclusions

According to the result of model comparison, the most effective way to find the location name in tweet texts is when each string in query texts is divided into many 2 consecutive words, and the approximately match method is global edit distance. The scalability of this method, however, does not perform very well. Therefore, in the future study, there may be a trade-off between accuracy and scalability.

References

Kondrak, G.: N-Gram Similarity and Distance. *Proceeding SPIRE'05 Proceedings of the 12th international conference on String Processing and Information Retrieval*. Pages 115-126. (2005)

Miller, E., Shen, D., Liu, J., and Nicholas, C.: Performance and Scalability of a Large-Scale N-Gram Based Information Retrieval System. *Journal of Digital Information,* Vol. 1(5). (2000)