COMP90049 Report

Shixun Liu 766799

1. Introduction

The project is implemented in JAVA. BI-SIM[1] is utilized as the mainly approximate matching strategy. Considering the big size of dataset, Levenshtein distance is used to setup a tree structure of location dictionary which optimize the retrieval speed effectively.

Each tweet is processed to an array of word that tokenized by space symbol after deleting any analphabetic symbols. Each location context is seen as a string completely disregarding the number of words contained. In addition, all strings are processed to lower-case.

2. Matching Strategy

BI-SIM can be seen as a modification based on n-gram distance method taught in class but eliminates its flaws and combines the property of edit distance. It is more sensitive to context and able to detect the similar string in some special cases where n-gram distance or global edit distance (GED) cannot.

2.1.Effectiveness Comparison with N-gram and GED

With the email feedback from Prof. Grzegorz Kondrak, the writer of "N-Gram Similarity and Distance", I was provided with an online string comparison tool (http://www.cs.toronto.edu/~aditya/strcmp2/), which helped prove the output of my Java-based implementation of BI-SIM, below comparison and latter context are based on this accuracy.

From human perception of words, the Op_1 seems the most similar to query in a way, however, N-gram distance (N=2) or GED may give out the ambiguous output.

	Op_1	Op_2	
Query: sevens	**seven**	**vensev**	**Choice**
N-gram Distance	1	0	Op_2
BI-SIM	83.33%	58.33%	Op_1
Query: sevens	**savina**	**kenepe**	**Choice**
N-gram Distance	10	10	Both
BI-SIM	58.33%	33.33%	Op_1

Table 1: BI-SIM and N-gram Comparison

Table 1 indicates two significant drawbacks of N-gram distance that it does associate the n-grams even if they occur in radically different word positions and do not consider the similarity within n-grams, however, BI-SIM eliminates these two drawbacks by making the n-grams absolute positions in the strings and n-gram itself similarity both contribute to the overall score.

	Op_1	Op_2	Op_3	
Query: abcde	**abcfg**	**afcge**	**fgcde**	**Choice**
GED	1	1	1	All
BI-SIM	70.00%	60.00%	50.00%	Op_1

Table 2: BI-SIM and GED Comparison

Table 2 shows that though GED can indicates score of transformation between two strings, it does not consider the influence of relative transformation position within string while neighbouring identity matches are a stronger indication of similarity. In view of BI-SIM, it is more sensitive to initial symbols at string boundaries in human perception of words.

2.2.Effectiveness Testing

Due to BI-SIM seems like a modified N-gram method, so I compare it with GED instead.

In the 20th tweet, there is a word “mcdonough” indicates the location. The retrieval results are shown below:

Method	**Threshold**	**Matching words**	**Result**
BI-SIM	≥ 90%	mcdonough	100%
BI-SIM	≥ 80%	mcdonough mcdonogh	100% 83.3%
BI-SIM	≥ 70%	mcdonough mcdonogh ***mcdougal***	100% 83.3% 72.2%
GED	≥ 7	mcdonough mcdonogh	9 7
GED	≥ 4	mcdonough mcdonogh east mcdonough mcdonough peak mcdonough lake mcdonough park	9 7 4 4 4 4

Table 3: Testing

It can be seen that the BI-SIM is more sensitive and has a high measurement precision for variation between words; it is able to point out the word “mcdougal” which seems similar to “mcdonough” as well in human perception. In reverse, the GED is insensitive to context, as it cannot distinguish the words with result 4, namely, its measurement precision is much less than BI-SIM. Table 4 illustrates the similarity of these words provided by BI-SIM against GED.

Method	Matching words	Result
BI-SIM	east mcdonough	60.7%
	mcdonough peak	64.3%
	mcdonough lake	64.3%
	mcdonough park	64.3%

Table 4: Similarity provided by BI-SIM

In addition, it tells that GED threshold value is also undeterminable: if the threshold is too high, the more similar words in human perception will be missed; if threshold is too low, more ambiguous words with same distance will be detected.

So if GED is implemented in this project, it is better to set the threshold dynamically based on the query string length but not a static value to all queries.

2.3.Conclusion

Compared with GED and N-gram, BI-SIM is better in precision and recall. I utilized it as the main approach through this project. However, all these three methods can only detect the similarity through the “appearance” of two strings, but not semantic analysis in context. This obstacle makes it cannot indicate whether the tweet contains a location name or not, but only indicates the tweet contains words which are similar to location words in appearance.

3. Retrieval speed optimization

In order to retrieval the 1.3M location names for each tweet quickly, the location names are categorised by the Levenshtein distance instead of matching each name one by one.

3.1.Tree data structure

A location name is set as the tree root randomly, it is “water well” in this project. Then all the other names are categorised into different groups based on their Levenshtein distance with “water well”. The finalized structure is shown in Figure 1.

During retrieval, the Levenshtein distance between query and “water well” is *d*, the threshold of Levenshtein distance is set as *k*, then it only need to retrieval the groups whose Levenshtein distance with “water well” is in the range of *d-k* to *d+k*.

This processing method can be seen as a pre-processing that filter out the strings beyond the distance threshold, then BI-SIM is used to match the strings precisely in further.

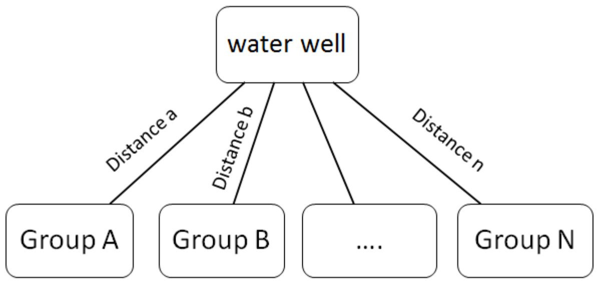

Figure 1: Tree structure of location dictionary

3.2.Testing

Compared with brute force, the retrieval speed is improved by 66.7% in average for each tweet with no influence on retrieval result.

	Brute Force	Tree	Improvement
Time / Tweet	12 min	4 min	66.7%

Table 5: Retrieval speed optimization

4. Conclusions and Challenges

From this project, it shows that the BI-SIM approximate matching method is more effective than N-gram distance and GED in precision and recall. But all of them can only focus on the “appearance” of the strings, not the meaning in context.

This project is trying to detect the misspelled location names in tweets. Using N-gram, GED or BI-SIM is only the first step to identify the words whose “appearance” is similar to location name even if it does not mean location in context at all. One key challenge is how to know the meaning of the word in context in order to get the potential words, even for some abbreviatory words in context, like “Los Angeles” is written as “L.A.” in tweet.

I think there is a way can help reduce the range of potential words by calculating all the words statistical frequency within big amount of tweets, the word with so high frequency, like “the”, “it” etc. should be ignored even if their “appearance” is similar to some location names.

Actually, there has been some tools offered, like StanfordNER which can help identify the name, organization, location, etc. from the context in multiple languages.

References

[1] Grzegorz Kondrak "N-Gram Similarity and Distance" (2005) In: M. Consens and G. Navarro (Eds.): SPIRE 2005, LNCS 3772, pp. 115–126, 2005. Springer-Verlag Berlin Heidelberg 2005