University of Melbourne

Project 1 Report
Misspelled Location Names

COMP90049 Knowledge Technologies, Semester 2 2016

Mengfei Hu – 719434

2016-8

mergfeih@student.unimelb.edu.au

1. Introduction

When people using social network tools like Facebook or Twitter, there are possibilities to misspell words when posting (Zinglaa,, et al., 2015). Location name in one of the common spelling mistakes that people would make. If trying to analyse twitters based on the location names, it is important to identify misspelling location names mentioned in twitters before move to a further step. Therefore, this project is aim at finding misspelling location names among text file of people's twitters.

There are two main data set have been used in this project:

1) "mergfeih_tweets_small.txt"
   a real data sub-sample from Twitter without any filter, which provided by University of Melbourne for teaching purpose only. Each line with a record of one tweet, with following format:
   user id (tab) tweet id (tab) tweet text (tab) time stamp (newline)
   There are 3798 lines in the file in total.
2) "US-loc-names.txt"
   Simplified version dictionary of US location names, already sorted without duplication. It contains one location name per line, and 1294704 lines in total.

In the project, two methods of approximate matching have been implemented by Python programs to match location names in the above two files.

2. Overview of approximate matching methods

There are different methods being implemented in different programming languages to solve approximate matching problem, and also popular algorithms exist. As Python is one of the most useful language for text analysis problem for its convenient I/O of files especially for string operations (Bird, et al., 2009), also mature pre-implemented libraries dealing with approximate searching such as *Fuzzywuzzy, NGram, NLTK (Natural Language Toolkit)* etc.

Two pre-implemented methods: The *Fuzzywuzzy (Levenshtein Distance matching)*, *NGram (N-gram distance matching)* and one self-developed method based on Global Edit Distance have been used to solve the problem.
System name with corresponding method used shown as follow table:

System name	**Method Used**
FuzzyMatch	Levenshtein Distance
NgramMatch	N-gram Distance
GED	Global Edit Distance

Table 1 - System and Method

In all methods, there are two main stages:

1) pre-processing data set
2) apply package

For matching location names problem, multi-words location names have been noticed in the file. However, it would make sense to check spelling of each words(token) and unnecessary and inefficient to match the full location name. So no matter which method is chosen, location name should be split into tokens and remove duplications so that get ready to be matched.

As for tweets, based on application features of different methods, they tweeters in different ways after matching regular expression of location pattern:

- Levenshtein Distance matching

Token Set Ratio function from fuzzywuzzy is used: it calculates the similarity between a token and a set based on Levenshtein distance in a range of 0 to 100. Higher score means more close to find the token in the set.

Tweets have been stored in a set together. It will be treated as "set" parameter of the function so that to compare with location tokens.

- N-gram distance matching

"Ngram" in python is a set class using N-gram string similarity to match (ref), search function is adopted from it: it will compare the token to be matched with each item in the set and return the similar item from the set with their similarity range between 0 to 1. 1 means the exact match.

In order to compare each location tokens with each tweets, all tweets should be in a one-dimensional list.

- Global Edit Distance

In theory, it is suitable to match between tokens in similar length because in this method, it will scan through each entry in dictionary and looking for the "best" match based their global edit distance(GED). GED function has been built based on Needleman–Wunsch algorithm, it calculates a match ratio:

match ratio = GED / length of token wait to be matched

(namely location name in the project).

Tweets and locations should be compared token by token. Therefore, tweets are split into single tokens as N-gram method do.

3. Effectiveness of the approximate matching methods

Because of several problems identified in the project (will be illustrate in later section) including data processing still not effective enough, the estimate execution time for all three systems will over 10 hours, a random selection of tweets has been applied to reduce time of checking effectiveness of each method so that repeat execution could be conduction to avoid coincidence.

3.1 Effectiveness of "FuzzyMatch"

No. of Execution	Number of Tweets Checked	Misspelling Found	Misspelling Actual	Precision
1	100	39	0	0%
2	100	49	0	0%

Table 2-Effectiveness of "FuzzyMatch"

Unfortunately, the use of token set ratio function has shown not effective at all for finding misspelling location names in tweets. Based on observation, it tends to return location tokens with punctuations especially "- ", and those with high score of match usually have another more common meaning in context.

For example, location token "Once-upon-a-time" has been found with 81 points. Although there is a location named "Once-Upon-A-Time Elementary School" in location name dictionary, it doesn't have this meaning in tweet context. Instead, it contains the natural language phrase "Once upon a time" in tweet. They are useless results for this problem.

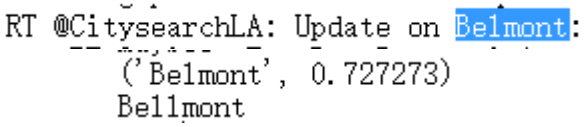
Therefore, this method has been proved to take relatively high weight on punctuations into account for calculating similarity. Without removing punctuation in location names first would lead to quite poor result even define approximate match in high similarity, given 80 is used in the project.

3.2 Effectiveness of "NGramMatch"

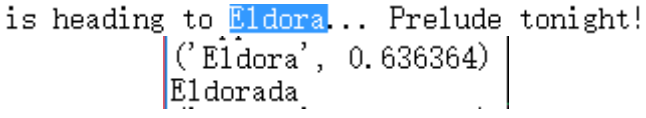
No. of Execution	Number of Tweeters Checked	Misspelling Found	Misspelling Actual	Precision
1	100	116	1	0.86%
2	100	117	2	1.7%

Table 3-Effectiveness of "NGramMatch"

The threshold is set to 0.6 based on consideration that the one-letter misspelling in a minimum length of three-words token (Soemarmo, 1983). Effectiveness of NGramMatch is still poor, but at least it proved to be able to find two kinds of misspelling location names: omit double writing letter and leave out tale. Words contain double writing letters as "Forrest" may be misspelt as "Forest"; "Bellmont" may be written as "Belmont" are the misspellings found by this method:



Another one is "Eldorada" has been misspelt as "Eldora" in the tweet:



3.3 Effectiveness of "GED"

No. of Execution	Number of Tweeters Checked	Misspelling Found	Misspelling Actual	Precision
1	10	12	1	9%
2	10	31	1	3.2%

Table 4-Effectiveness of "GED"

Due to execution time nearly 60 seconds per tweet, the number of tweets being checked has been reduced accordingly. The operation score for match has been set to 1 and others are -1, in this case, full match would get match ratio as 1. So approximate match for this method has been defined as match ratio bigger than 0.8 but no equals to 1. Luckily, it found misspellings among the small number of tweets for writing "Tennessee" as "Tennesse" and "Phillis" as "Phillies":

```
lets go Phillies
0.8571428571428571
'Phillies'
Phillis
```

It is possible that it is able to find more misspelling location names if given enough time to check more tweets. But time consuming is a key issue for this method.

3.4 Comparison

Among these three methods, we compare execution time, threshold, effectiveness as below:

	EXECUTION TIME	THRESHOLD	EFFECTIVENESS
FUZZYMATCH	Fast	60 out of 100	Ineffective
NGRAMMATCH	Medium	0.6 out of 1	Poorly effective
GED	Slow	0.8 out of 1	Potentially effective

Table 5-Comparison of Methods

4. Conclusion

There are several lessons learnt in this project:

- Pre-processing data is key to any of approximate matching method.

Each method may need certain input data format, the importance of getting them ready for method to apply could be seen from the "FuzzyMatch" method. When the input string is not ideal for the method, it would obviously lead to poor result no matter how good the method is.

Also, regular expression is a critical step for pre-pressing data. There is a jump in performance for all three methods after using regular expression to match tweeter content so that to remove digital number and punctuation.

- Define "misspelling" by parameter and threshold

For any knowledge technology problem, the context is critical. For this particular misspelling location name issue, the context is related to how accurate the result need to be and who is asking the question. Therefore, there are dynamic and flexible parameter in all methods to define what is "misspelling" which are parameters and thresholds.

In conclusion, there may be no perfect answer for the question. Each method has their advantage and disadvantages, also require different types of input. It is important to manage input data as well as define the vague in question by context.

References

Zinglaa,, M. A., Chirazb, L., Slimani, Y. & Berrut, C., 2015. Statistical and Semantic Approaches for Tweet Contextualization. *Procedia Computer Science,* Volume 60, pp. 498-507.

Bird, S., Klein, E. & Loper, E., 2009. *Natural Language Processing with Python.* 1st ed. Sebastopol: O'Reilly Media, Inc..

Soemarmo, M., 1983. Programming for Misspelling Extended Input. *CALICO Journal,* Volume 1, pp. 31-39.