COMP90049 Project 1: Lexical Normalisation of Twitter Data

Anonymous

1. Introduction

The problem of Lexical Normalisation is finding a canonical form for each token with a document. This report describes a couple attempts to Lexical Normalisation on twitter data using approximate string-matching methods. Twitter is a rich source of lexical variants and since the tokens used for this paper are sourced from tweets, we can obtain unbiased results.

The most well-known methods for approximate matching are edit distance and n-grams. It is also well-known that Phonetic methods could prove to be very useful in some cases. This paper specifically attempts to understand how Edit Distance compares to Edit Distance with a Phonetic method.

Related studies have been conducted on this topic. The main study that I will be following and referencing throughout this report is the study done by Zobel and Dart in 1996.

2. Dataset

This project contains three datasets: 'misspell.txt', 'correct.txt' and 'dict.txt'. The goal of this project is to correct the spellings/match the words in 'misspell.txt' to the words in 'correct.txt' using the large list of words (dictionary) in 'dict.txt'.

Datasets	Description
misspell.txt	This dataset contains a list of 10322 words which, for the purpose of this project, have been identified as misspelled.
correct.txt	This dataset contains a list of 10322 "correct" words corresponding to each misspelled word in the aforementioned dataset.
dict.txt	This dataset contains 370099 words and acts as a dictionary using which the canonical forms are predicted from the tokens.

Table 1: The description of the datasets.

2.1. Properties of Datasets

Even though 'dict.txt' acts as a dictionary, it is not complete, hence it would not be suitable for a much larger list of misspelled words.

The list of misspelled words contains a large subset of words that actually have the correct spelling. This could in many ways skew the final results obtained.

Few of the words in 'correct.txt', do not exist in the dictionary, making it impossible to match the corresponding misspelled words using the dictionary. Hence, the precision could be largely skewed however the recall remains intact since only the possible correct results are considered.

3. Methodologies

The main method of approximate string matching that I have chosen for this project is Global Edit Distance specifically the Levenshtein Distance method. I have implemented the Levenshtein method in two ways, one using the spelling of the words and the other using Phonetic methods. For phonetic methods, I have used Soundex.

Both methods have been implemented using C and the results containing the matches for each of the methods are stored in a JSON file. This JSON file is used for evaluation using Python.

3.1. Global Edit Distance

Levenshtein edit distance attempts to find the lowest number of operations required to transform a token into it's canonical form using single character operations which include insertion, substitution and deletion where each operation has a cost of 1 (Hasan et al, 2015).

3.2. Phonetic Method

The Phonetic method chosen for the purpose of this paper is Soundex. Soundex uses codes based on the sound of each letter transforming tokens into a code with 4 character which is then used to find similar sounding strings.

According to Zobel and Dart (1996), Soundex has an issue where it transformes strings that sound similar, to different codes and codes that sound different, to similar codes, this could skew the results obtained.

For this project, I have combined Levenshtein distance with Soundex where, if no exact matches for Soundex are found from the dictionary, then the closest matches are returned.

4. Evaluation

The evaluation metrics used through this report are Precision and Recall.

4.1. Metrics

4.1.1. Precision

Precision is the proportion of the retrieved matches that are correct.

$$Precision = \frac{number\ of\ correct\ results\ returned}{total\ number\ of\ results\ returned}$$

4.1.2. Recall

Recall is the proportion of correct matches that are retrieved.

$$Recall = \frac{number\ of\ correct\ results\ returned}{number\ of\ possible\ correct\ results}$$

4.2. Results

	Levenshtein Distance	**Levenshtein Distance+Soundex**
Correct results returned	7987	8068
Total matches returned	40717	1442763
Average matches/token	3.9	139.8

Table 2: Statistics of matches returned.

	Levenshtein Distance	**Levenshtein Distance+Soundex**
Precision	19.6%	0.6%
Recall	93.5%	94.4%

Table 3: Precision and Recall.

Token (Canonical Form)	**Levenshtein Distance**	**Levenshtein Distance+Soundex**
'tmrw' (tomorrow)	amra mew mr tare …	tamara … tomorrow …
'rt' (rt)	rt	raad rad radio rate rt rte …(132)

Table 4: Examples of tokens and matches.

4.3. Analysis

4.3.1. Global Edit Distance

This method manages to return the canonical form from the misspelled words 7987 times out of the total 10322 words. This is a high percentage however it is important to consider the fact that most of the tokens are already in their canonical form.

The total number of matches returned is 40717 which is relatively small, this means that the average matches per token is low, specifically 3.9. Therefore, it can also be deduced that most tokens receive just one match making this method highly effective. This results in a high precision.

One of the key issues observed from the data was the fact that tweets contain a lot of short forms of words which are formed by how the word sounds hence finding the edit distance relative to the spelling becomes redundant. This was evident in the misspelled token 'u' where the canonical form was 'you' however 'you' was not even in the list of matches as it is not similar to the token 'u' in terms of spelling.

4.3.2. Phonetic Method

The key characteristic of using Edit Distance + Soundex is the total number of matches it produced. This method had an average matches returned of 139.8. This is mainly due to phonetic matching being very abstract. It is due to many different sounding words ending up with the same Soundex code, which is a key issue faced when using Soundex. This results in a large number of matches, reducing the precision but increasing the chance of the canonical forms of the tokens being present in the relevant matches.

4.3.3. Comparison

The key observable difference between basic edit distance and edit distance with Soundex is the difference in precision. This stems from the difference in the total number of matches, specifically 40717 in the basic edit distance method versus 1442763 in the edit distance

method with Soundex. This is largely due the characteristics of phonetic matching which have been mentioned in 4.3.2. This is evident in the example given in the second row of Table 4. The token ‘rt’ has the same word in it’s canonical form. Edit distance returns one match which is the correct canonical form. The Phonetic method returns 138 matches of which ‘rt’ is one but such a low precision makes this method redundant.

It is important to note that 1778 of the total canonical forms do not exist in the dictionary which means only 83% of the total dataset can produce correct matches (possible correct matches). Therefore, it is understandable as to how the recall manages to eclipse 90% for both methods. This skews the effectiveness measurement of precision however this makes recall a more important measurement.

Note that Edit Distance + Soundex produces a slightly higher Recall, which is due it’s ability to match more tokens to their canonical form as the misspelled tokens are sourced from tweets which contain multiple short forms that are easily captured by Soundex. An example of this is visible in the example given in the first row of Table 4. The token is ‘tmrw’ which is a common short hand for ‘tomorrow’ (the canonical form for this token). Edit distance displays a large number of matches but none of them match ‘tomorrow’. The Phonetic method, given it’s properties, returns a list of matches which include the canonical form. This is the key reason that the Phonetic method has a higher recall than the basic Edit Distance.

5. Future Improvements

Using Soundex did not prove to be as useful as expected in Lexical Normalisation. As proven by Zobel and Dart (1996), using Editex could prove much more effective than Soundex.

Since we use Twitter data, it might be useful to use additional dictionaries which include slang and other common short-hands. This could vastly improve the efficiency and quality of the results received.

Running the code took a long time since it was run brute-force, a key future improvement would be to streamline this process increasing the efficiency.

6. Conclusions

Due to the many challenges and noise within the dataset, it is difficult to compare both methods for Lexical Normalisation described in this project. However, it can be deduced that the Phonetic method has a much lower precision than the basic Levenshtein distance. Therefore, it is evident that Soundex is not a great method for Lexical Normalisation compared to Levenshtein distance.

In my opinion, if the improvements described in 5. are implemented, it could improve on the effectiveness achieved by the methods described in this report.

7. References

Baldwin, Timothy, Marie Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu (2015) Shared Tasks of the 2015 Workshop on Noisy User-generated Text: Twitter Lexical Normalization and Named Entity Recognition. In *Proceedings of the ACL 2015 Workshop on Noisy User-generated Text*, Beijing, China, pp. 126–135.

Zobel, Justin and Philip Dart. (1996). Phonetic String Matching: Lessons from Information Retrieval. In *Proceedings of the Eighteenth International ACM SIGIR Conference on Research and Development in Information Retrieval*. Zurich, Switzerland. pp. 166–173.

Syeda Shabnam Hasan, Fareal Ahmed, and Rosina Surovi Khan. Approximate string matching algorithms: A brief survey and comparison. *International Journal of Computer Applications*, 120 (8), 2015.