COMP90049 Project One: Lexical Normalization of Tweets

1 Introduction

Microblogging services, especially twitter, are evolving to become a more and more important source for various kind of information. Twitter data has been widely used in sentiment analysis (Agarwal et al., 2011), sentiment detection (Barbosa and Feng, 2010) and even predicting movements in stock markets (Pagolu et al., 2016).

However, compared to other kinds of text corpus often used in NLP tasks, twitter data is more noisy, containing many abbreviations, acronyms and typos, which makes tasks like sentiment analysis much more difficult. In this case, the twitter text will be normalized to its canonical form. The main purpose of this report is to implement different approximate string matching methods, including Levenshtein distance (Levenshtein, 1966), two- and three-gram with Jaccard distance (Levandowsky and Winter, 1971), metaphone algorithm (Philips, 1990), match rating approach (Moore et al., 1977) and refined soundex (Ahmed, 2015) for lexical normalization of the provided twitter data (Baldwin et al., 2015). Furthermore, external data[1] is adopted to improve the quality of the provided dictionary.

2 Related Work

The noisy channel model (Shannon, 1948) has been the primary approach for lexical normalization traditionally (Han et al., 2013). Also, machine translation methods are adopted to tackle text normalization problems. (Aw et al., 2006) adopted a phrased-based model translating SMS language to canonical English language as a preprocessing step. Unsupervised models are also applied to text normalization problems. (Yang and Eisenstein, 2013) proposed a log-linear model for unsupervised statistical text normalization.

3 Dataset

The dataset used in this report is a part of a twitter dataset curated by Han and Baldwin (Baldwin et al., 2015). The misspelling words file (misspell.txt) contains the original twitter text, and the file "correct.txt" contains the parallel texts for "misspell.txt" with words (supposed to be) in their canonical forms.

We preprocess the data and label each words with "OOV" (correct spelling not in dictionary) or "IV" (correct form in dictionary) according to whether one word in the original dictionary ("dict.txt", containing 370099 words), as well as the expanded dictionary (containing 500808 tokens), or not, and store the preprocessed data into "data_entries.json" and "new_data_entries.json" under the "Code_Output" folder.

Table 1: OOV/IV Status of the Data (Orignal Dictionary)

	Number of Words	**Ratio**
OOV	1779	0.172
IV	8543	0.828

Table 2: OOV/IV Status of the Data (Expanded Dictionary)

	Number of Words	**Ratio**
OOV	1490	0.144
IV	8832	0.856

From the preprocessing outcome, we can see that the expansion of dictionary failed to make

[1] Obtained from https://github.com/first20hours/google-10000-english and https://github.com/dwyl/english-words.

a significant drop on the OOV ratios of the correction word list. By taking a look into the file "correct.txt", we can see that a lot of words that are supposed to be correct are in fact not corrected by the ones who produced the data. In other words, the data is very noisy-labelled. Also, there are considerable number of words are "slangs" which do not exist in any kind of formal dictionaries.

From the misspelling-correction dataset, we can see among all the forms of non-standard English words, most of them are slangs, special terms such as names, ill-formed words with substitution of similar-pronounced characters and contractions as well as abbreviations.

4 Methodology

4.1 String Distance Based Methods

4.1.1 Levenshtein Distance

Levenshtein distance, which is a special case for global edit distance, measures the minimum number of edit operations to transform a string to another. Edit distance includes insertion, replacement and deletion. In this project, the edit distance is calculated using the "edit_distance" function in the python package NLTK[2] (Bird et al., 2009). For each token in the misspell list, this project will return three kind of results (corresponding to the three approaches involving Levenshtein distance):

- the token in the dictionary which has the minimum Levenshtein distance to the input token from the misspell list (the query);
- the tokens in the dictionary which have Levenshtein distance no more than 3 to the query;
- the tokens in the dictionary which have Levenshtein distance no more than 2 to the query.

For both original and expanded dictionary.

4.1.2 N-Gram with Jaccard Distance

N-Gram measures the similarity of two strings based on the number of similar substrings of length N (Kondrak, 2005). Instead of the metrics mentioned in the lecture, in this project we adopt the Jaccard similarity to measure the similarity between two N Gram substring sets (unlike bag of words, the n gram method usually do not count duplicates).

$$D(X,Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|} \tag{1}$$

Where X and Y are both sets, and $|X|$ means the number of elements in set X. In this project, for a single query, the script will return four kinds of tokens from the dictionary (both the original one and the expanded one):

- 2-gram with minimum Jaccard similarity;
- 3-gram with minimum Jaccard similarity;
- 2-gram with Jaccard similarity no more than 0.2;
- 3-gram with Jaccard similarity no more than 0.2;

4.2 Phonetic Representation Based Method

In this report, three phonetic representation method are adopt for approximate string searching. For each query and each kind of phonetic algorithm, the program will return the tokens that have the same phonetic representation as the query, both the original dictionary and the expanded one. The implementation of the following phonetics algorithms is modified from a python package "PyPhonetics[3]". Since phonetic algorithms does not work properly when encountered with digits, we adopted python package "num2words[4]" to transform digits to words before generating phonetic representations for the tokens as the following rules:

- if the first character of the query token is 0, convert that 0 to o;
- if a 0 in the token does not has other digit on its left side (eg. "er0" or "45iuy0d", not "gh10k"), convert that 0 to o;
- convert the rest numbers (consecutive digits treated as a single number) using the num2word package.

4.2.1 Refined Soundex

Compared to original soundex, refined soundex provides a more detailed grouping for the alphabetic characters:

[2] https://www.nltk.org/

[3] Obtained from https://github.com/Lilykos/pyphonetics

[4] Obtained from https://pypi.org/project/num2words/

Table 3: Original Soundex (Ahmed, 2015)

Characters	Group Number
b,f,p,v	1
c, g, j, k, q, s, x, z	2
d, t	3
l	4
m,n	5
r	6

Table 4: Refined Soundex (Ahmed, 2015)

Characters	Group Number
b, p	1
f, v	2
c, k, s	3
g, j	4
q, x, z	5
d, t	6
l	7
m, n	8
r	9

4.2.2 Metaphone

Metaphone was invented for indexing words by their English pronunciation, which improves on the Soundex algorithm by taking the variation as well as inconsistency information of English words into account to generate an encoding with more accuracy than nomral Soundex algorithms. This helps a lot when matching words and names with similar pronunciation. Same to Soundex, similar pronounced words should have the same phonetic representation. Original Metaphone codes use the 16 consonant symbols "0BFHJKLMNPRSTWXY". The '0' represents "th" (Kuhn, 1995). Detailed transformation rules can be accessed on http://aspell.net/metaphone/metaphone-kuhn.txt.

4.2.3 Match Rating Approach

The match rating approach (MRA) was initially developed for indexing and comparing homophonous names (Moore et al., 1977). MRA itself contains a simple set of encoding rules with a more detailed set of comparison rules. Main mechanism for calculating the similarity between two strings is to calculate the number of unmatched characters by comparing the strings from left to right and then right to left and removing identical characters. This value is subtracted from 6 and then compared to a minimum threshold.

5 Evaluation and Analysis

5.1 Evaluation Metrics

In this project, we evaluate the matching results with precision, recall and accuracy from the area of information retrieval. The defination of these three metrics for this project are as follows:

$$Precision = \frac{CTQ}{TQT} \tag{2}$$

Where CTQ represents the number of correct tokens (that is the same as those in the correct.txt) returned by all the queries and TQT represents the total number of tokens returned for all the queries (If query 1, 2, 3 returns 10, 20, 30 tokens respectively, and 1 correct for query 1 while 0 correct for both of quires 2 and 3, the CTQ will be 1 and the TQT will be 60).

$$Recall = \frac{CiQ}{AC} \tag{3}$$

Where CiQ represents the number of queries by which returned tokens contain the correct one (meaning the correct form of the word is in all the returned tokens for a single query) and also the input of the query should also in the correct form (hence the "positive class" of the dataset, which are the entries with same misspell word and correct word), and AC represents for the number of words in the dataset that does not need any correction. Recall here basically reflects whether the program wrongly "corrects" a correct word.

$$Accuracy = \frac{RQ}{AQ} \tag{4}$$

Where RQ stands for the number of queries whose returned tokens contains the right answer, and AQ stands for the number of total queries, hence the total number of tokens in misspell.txt or correct.txt.

In this project, we apply the evaluation metrics mentioned above on different part of the

dataset, hence PED, PMIV, AMIV, RED and RIV. See the following explainations.

- PED: Precision on Entire Dataset.
- PMIV: Precision for on the Misspellings, IV (in vocabulary)
- AMIV: Accuracy on the Misspellings, IV
- RED: Recall on Entire Dataset
- RIV: Recall for all the IVs

5.2 Analysis

The evaluation results on our matching outcome by adopting the metrics mentioned in the previous subsection are presented in Table 5 (on the fifth page) for the original dictionary, and Table 6 (also on the fifth page) for the expanded dictionary.

From the two tables, we can see that expanding dictionary will make an increase of RED, but not very significant.

However, an expanded dictionary affect other metrics differently, especially the precisions. We can observe a drop of PMIV on all methods, especilly those will return more than one tokens for a single match query, which is very reasonable because an expanded dictionary will increase the number of tokens that fall within the selection criteria.

AMIV, which reflects the ability of the algorithm to correct a really wrong word to its canonical form which is in the dictionary (We do not consider those words with canonical form falls outside the dictionary, which is impossible to correct with current methods), reacts differently to the expanded dictionary for different methods. Methods involving n-gram of the words and minimum Levenshtein distance suffers a decline of AMIV as the number of words in the dictionary increases, while the AMIVs of other methods rise. For the minimum Levenshtein distance method, such a situation can occur due to some of the misspells actually exist in the expanded dictionary. This also applied for the methods involving n-grams and Jaccard distance. In fact, even two words with small edit distance can differ a lot under Jaccard distance after the n-gram process. From this we can also learn that even n-gram works well for sentences (token level), it performs not so good on words (character level).

If choose AMIV as our main evaluation metric, we can see that both Edit Distance No More Than 2 and Edit Distance No More Than 1 did pretty well on both the original and the expanded dictionary. However, Edit Distance No More Than 2 outperforms other methods by a great margin on both dictionaries.

6 Conclusions

In this report, we explored the performance of several approximate string matching based methods on the lexical normalization of twitter data. Although good precision on the in vocabulary, misspelled part of the dataset was achieved by the method Edit Distance No More Than 2, this method still has a serious problem of low precision, which means it returned too much words for a single query. All the methods adopted in this project have great difficulties to notice the long-range dependency in the misspellings, such as "cu" for "see you", "$n8$" for "night" and "aka" for "also known as", which is very common in today's social media corpus. Some of these contractions are in fact abbreviations, and some of them are substitutions with similar pronounced characters. To mechanically set up a pack of rules with edit distance and phonetic algorithms are becoming more and more unlikely to tackle such problems.

However, with enough parallel texts, problems mentioned above may be solved with methods inspired from the attention models (Vaswani et al., 2017) used in machine translation. Also, techniques from translating without parallel data(Conneau et al., 2017) can be applied to such lexical normalization tasks, since the corrected text is hard to obtain.

Table 5: Results on the Original Dictionary

Method	**PED(%)**	**PMIV (%)**	**AMIV (%)**	**RED(%)**	**RIV(%)**
Minimum Levenshtein Distance	76.18	7.984	7.984	83.23	100.0
Edit Distance No More Than 2	0.1604	0.1594	76.73	83.23	100.0
Edit Distance No More Than 1	2.686	1.173	33.96	83.23	100.0
3-Gram with Jaccard Best Match	76.58	13.67	13.67	83.23	100.0
2-Gram with Jaccard Best Match	76.49	12.45	12.45	83.22	99.98
2-Gram Jaccard Distance No More Than 0.2	77.36	5.811	3.924	83.23	100.0
3-Gram Jaccard Distance No More Than 0.2	92.21	0.5115	0.2706	83.23	100.0
Metaphone	1.847	0.6670	27.60	83.23	100.0
Refined Soundex	2.652	0.9792	20.97	83.23	100.0
MRA	3.410	1.033	24.22	83.23	100.0

Table 6: Results on the Expanded Dictionary

Method	**PED(%)**	**PMIV (%)**	**AMIV (%)**	**RED(%)**	**RIV(%)**
Minimum Levenshtein Distance	77.95	7.962	7.962	85.08	100.0
Edit Distance No More Than 2	0.1039	0.09828	78.22	85.08	100.0
Edit Distance No More Than 1	2.256	1.054	40.28	85.08	100.0
3-Gram with Jaccard Best Match	78.36	12.99	12.99	85.08	100.0
2-Gram with Jaccard Best Match	78.24	12.41	12.41	85.00	99.89
2-Gram Jaccard Distance No More Than 0.2	77.16	4.545	3.395	85.08	100.0
3-Gram Jaccard Distance No More Than 0.2	92.03	0.3809	0.2341	85.08	100.0
Metaphone	0.9984	0.4213	35.59	85.08	100.0
Refined Soundex	1.468	0.7390	29.15	85.08	100.0
MRA	1.864	0.7320	32.55	85.08	100.0

References

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2011. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media*, LSM '11, pages 30–38, Stroudsburg, PA, USA. Association for Computational Linguistics.

B. Ahmed. 2015. Lexical normalisation of twitter data. In *2015 Science and Information Conference (SAI)*, pages 326–328, July.

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for sms text normalization. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 33–40, Stroudsburg, PA, USA. Association for Computational Linguistics.

Timothy Baldwin, 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 Workshop on Noisy User-generated Text*, pages 126–135, Beijing, China.

Luciano Barbosa and Junlan Feng. 2010. Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 36–44, Stroudsburg, PA, USA. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *CoRR*, abs/1710.04087.

Bo Han, Paul Cook, and Timothy Baldwin. 2013. Lexical normalization for social media text. *ACM Trans. Intell. Syst. Technol.*, 4(1):5:1–5:27, February.

Grzegorz Kondrak. 2005. N-gram similarity and distance. In *Proceedings of the 12th International Conference on String Processing and Information Retrieval*, SPIRE'05, pages 115–126, Berlin, Heidelberg. Springer-Verlag.

Michael Kuhn. 1995. Phonetic search codes, Nov.

Michael Levandowsky and David Winter. 1971. Distance between sets. *Nature*, 234(5323):34–35.

VI Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

G.B. Moore, Institute for Computer Sciences, Technology, and Inc Operating Systems. 1977. *Accessing Individual Records from Personal Data Files Using Non-unique Identifiers*. Accessing Individual Records from Personal Data Files Using Non-unique Identifiers. U.S. Department of Commerce, National Bureau of Standards.

V. S. Pagolu, K. N. Reddy, G. Panda, and B. Majhi. 2016. Sentiment analysis of twitter data for predicting stock market movements. In *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, pages 1345–1350, Oct.

Lawrence Philips. 1990. Hanging on the metaphone. *Computer Language Magazine*, 7(12):39–44, December.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 7.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Yi Yang and Jacob Eisenstein. 2013. A log-linear model for unsupervised text normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 61–72, Seattle, Washington, USA, October. Association for Computational Linguistics.