COMP90049 Knowledge Technologies Report

JUNWEN ZHANG
791773

1. Introduction

This task is about using spelling correction to solve the problem of back–transliteration in the Persian language. Persian names are treated like misspelled Latin names and using approximate matching method to predict the best match of Latin name for Persian name. In this task, I solved this problem based on four approximate matching methods and compared them with their precision and recall as evaluation metric. Furthermore, I make some changes to Global Edit Distance method through using replacement matrix instead of r parameter.

The data set which I use to evaluate the systems is the given train.txt file and names.txt file. The number of cases of misspelled Latin names which are actual Persian names are the first column of train text and intended words for each case are the second column of train text. The dictionary of all the correct Latin names are in names text. I use these two files to calculate the precision and recall of each method.

$$\text{precision} = \frac{\text{the number of the correct cases}}{\text{the number of the predicted words}}$$

$$\text{recall} = \frac{\text{the number of the correct cases}}{\text{the number of the interested words (Persian names)}}$$

2. Global Edit Distance

2.1. Basic Global Edit Distance

The Global Edit Distance is trying to transform one string into another. The less operations it needs in the transform, the closer these two strings are. The parameters I set in basic Global Edit Distance for match, insertion, deletion and replacement are 0,1,1,1 and 0,1,2,1. The parameter set 0,1,1,1 is Levenshtein Distance. The lower the score is, the closer two strings are. Therefore the best matches are the ones have the lowest score. As I noticed that when transforming Persian names to Latin ones, it is more likely to insert or replace character instead of deleting one. For example, the Persian name ‘ABRAPT’ and Latin name ‘abrupt’, they just replace A with u. So I set more value to the deletion parameter to increase the score which has deletion operation in the transforming path. And the precision and recall are shown in the below table1:

methods	recall (%)	precision (%)
Global Edit Distance (0,1,1,1)	42.99	4.46
Global Edit Distance (0,1,2,1)	45.27052	5.92

Table 1: basic Global Edit Distance

According to the result, the parameter effects the recall and precision. When the parameter set as 0,1,2,1 due to the regulation between Persian names and Latin names, the precision and recall become higher.

2.2. **Modifying the r parameter**

As we can see from the train file, not all the Latin names are the lower case of its Persian names. Some characters are substituting with another character. For example, Persian name ‘ABST’ and its Latin name ‘obst’, character A is replacing with o. While throughout the train file, A is never replacing with t. In this case, replacing A with o should has the different score with replacing A with z and the r parameter of the former case should be closer to the matching parameter. In order to decide the r parameter, I need all the ‘correct’ replacements in the train file. So firstly, I created a matrix called replacement of size 26*26(the set of total characters) to store the frequency of replacement from one character to another, for instance, replacement[i][j] has the frequency of i replacing with j and a matrix called replacementParameter to store the parameter of each replacement.

I calculate the frequency by matching all the Persian names and its Latin name in the train file with Global Edit Distance set parameter with 0,1,1,1. Instead of getting the score between two names, I mostly focus on the best way of transforming one into another. If there is a replacement (i with j) in the best way, then the value of replacement[i][j] plus one. After this operation, I get the replacement matrix that contains all replacement frequency which looks like the below table2:

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
A	0.0	0.0	1.0	1.0	110.0	0.0	0.0	5.0	71.0	0.0	0.0	2.0	0.0	0.0	268.0	0.0	0.0	0.0	0.0	0.0	244.0	0.0	0.0	0.0	0.0	1.0
B	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
C	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
D	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0
E	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
F	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
G	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
H	29.0	0.0	0.0	0.0	226.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
I	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
J	0.0	0.0	0.0	0.0	2.0	0.0	31.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
K	3.0	0.0	326.0	0.0	8.0	0.0	1.0	1.0	3.0	0.0	0.0	1.0	0.0	0.0	2.0	0.0	46.0	0.0	0.0	0.0	6.0	0.0	1.0	14.0	0.0	0.0
L	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	2.0	0.0	0.0	0.0	0.0	0.0
M	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
N	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
O	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Q	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
R	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
S	1.0	0.0	66.0	0.0	22.0	0.0	1.0	1.0	4.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	2.0	0.0	1.0	0.0	0.0	1.0	19.0	0.0	2.0
T	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	2.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
U	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
V	8.0	0.0	1.0	0.0	25.0	2.0	1.0	2.0	6.0	0.0	0.0	4.0	0.0	0.0	1640.0	1.0	2.0	1.0	2.0	4.0	577.0	0.0	237.0	2.0	1.0	0.0
W	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
X	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Y	44.0	1.0	18.0	1.0	341.0	0.0	3.0	2.0	2181.0	6.0	0.0	9.0	0.0	8.0	5.0	0.0	1.0	0.0	9.0	7.0	22.0	0.0	4.0	0.0	0.0	2.0
Z	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	5.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	2.0	282.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0

Table 2: replacement frequency martix

Certainly, this operation does not return the actual frequency of replacement, so I only count the value above 70, and change their values in the replacementParameter matrix equal to the matching parameter. And the values of those replacement times which equal to 0 the same as the deletion parameter. Others set as 2. And the replacementParameter matrix looks like the table below:

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
A	3.0	3.0	2.0	2.0	0.0	3.0	3.0	2.0	0.0	3.0	3.0	2.0	3.0	3.0	0.0	3.0	3.0	3.0	3.0	3.0	0.0	3.0	3.0	3.0	3.0	2.0
B	2.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	2.0	3.0	3.0	3.0
C	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	2.0	3.0	3.0	3.0	3.0	3.0	3.0
D	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	2.0	3.0	2.0	3.0	3.0	3.0	2.0	3.0	3.0	2.0	3.0	2.0	3.0	3.0	3.0	3.0	3.0	3.0
E	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
F	3.0	3.0	3.0	3.0	3.0	3.0	3.0	2.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
G	3.0	3.0	3.0	1.0	3.0	3.0	3.0	3.0	3.0	2.0	2.0	3.0	3.0	3.0	3.0	3.0	2.0	3.0	2.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
H	2.0	3.0	3.0	3.0	0.0	3.0	3.0	3.0	3.0	3.0	3.0	2.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
I	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
J	3.0	3.0	3.0	3.0	2.0	3.0	2.0	3.0	2.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
K	2.0	3.0	0.0	3.0	2.0	3.0	2.0	2.0	2.0	3.0	3.0	2.0	3.0	3.0	2.0	3.0	2.0	3.0	3.0	3.0	2.0	3.0	2.0	2.0	3.0	3.0
L	3.0	3.0	3.0	3.0	2.0	3.0	3.0	3.0	2.0	3.0	3.0	3.0	3.0	3.0	2.0	3.0	3.0	3.0	3.0	2.0	2.0	3.0	3.0	3.0	3.0	3.0
M	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	2.0	3.0	3.0	3.0	2.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
N	3.0	3.0	2.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	2.0	3.0	3.0	3.0	3.0	3.0	2.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
O	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
P	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
Q	3.0	3.0	3.0	3.0	3.0	3.0	2.0	3.0	3.0	3.0	2.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
R	3.0	3.0	2.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	2.0
S	2.0	3.0	2.0	3.0	2.0	3.0	2.0	2.0	2.0	3.0	3.0	3.0	3.0	2.0	2.0	3.0	3.0	2.0	3.0	2.0	3.0	3.0	2.0	2.0	3.0	2.0
T	3.0	3.0	3.0	2.0	3.0	2.0	3.0	3.0	2.0	3.0	3.0	2.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
U	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
V	2.0	3.0	2.0	3.0	2.0	2.0	2.0	2.0	2.0	3.0	3.0	2.0	3.0	3.0	0.0	2.0	2.0	2.0	2.0	2.0	0.0	3.0	0.0	2.0	2.0	3.0
W	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
X	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
Y	2.0	2.0	2.0	2.0	0.0	3.0	2.0	2.0	0.0	2.0	3.0	2.0	3.0	2.0	2.0	3.0	2.0	3.0	2.0	2.0	2.0	3.0	2.0	3.0	3.0	2.0
Z	3.0	3.0	3.0	3.0	2.0	3.0	3.0	3.0	3.0	2.0	2.0	3.0	3.0	3.0	3.0	3.0	2.0	2.0	0.0	3.0	3.0	3.0	3.0	2.0	3.0	3.0

table 3: replacement parameter matrix

Again, calculating the precision and recall through Global Edit Distance with replacementParameter matrix instead of replacement parameter. Each replacement will have different parameter which depends on the replacementParameter matrix. And results are shown in the below table4:

method	recall (%)	precision (%)
Global Edit Distance With replacement matrix(0,1,3,matrix)	66.29	28.97

Table 4: Global Edit Distance with r matrix parameter

As we can see from the table4 that the recall and precision become higher obviously, so this improved method is more suitable.

3. Other approximate matching methods

3.1 Setting parameter

The parameter I set in Local Edit Distance is 1,-1,-1,-1. The largest value in the Local Edit Distance matrix is the length of the longest substring between two strings. So the higher the value is, the closer two strings are.

The substring length of N-grams is 2. As some length of the Persian name is very short, for instance, some Persian name just has 4 characters. I set 2-grams so that the system can apply to every Persian names.

3.2 Precision and Recall

method	recall (%)	precision (%)
Local Edit Distance	33.40	0.48
N-grams (N=2)	16.49	7.67
Soundex	53.55	3.21

Table 5: other approximate matching methods

3.3 Evaluation

3.3.1 Edit distance

The recall in Local Edit Distance is lower than Global Edit Distance. From the train text, we can see that the length of Pertain names and Latin names are nearly the same. The difference between these two names are more likely to change some characters. So the intended Latin name for Pertain name may not have the longest substring among the dictionary. For instance, Latin name ‘abraham’ has the longest substring with Persian name ABR, while the intended Latin name is ‘aabar’. Local Edit Distance is more suitable for matching two strings with very different lengths and the most common thing they have is substring, while Global Edit Distance is trying to match the whole string through some operations. Apart from that, the precision in Local Edit Distance is also lower. Due to many names in dictionary may have the same substring with the Persian name, it is harder to differ the names by the length of substrings. So many names will be scored as the predicted ones and the tie will be large which leads to the lower precision.

3.3.2 N-grams

N-grams has a lower recall but has a higher precision. In 2-grams, the more substrings of length

2 that two names have in common, the closer they are. In this task, the Persian name and its intended Latin name may not have many substrings of length n in common. Because some of the characters in Persian name are replaced by other characters in Latin name. Lower distance does not mean that two names are closer so that N-grams is less suitable for this system than Global Edit Distance. And the constrain of N-gram score is very strict so the number of words that have the same N-grams score is small. So the precision will be higher than basic Global Edit Distance.

3.3.3 **Soundex**

Soundex is a phonetic algorithm and homophones will be encoded to the same representation. Although orthography is not a good predictor of phonetics and homophones may have very different meanings. Persian name and Latin name may sound quite the same and have the same meanings because they are just names. According to the result, Soundex may be suitable for the task but improved Global Edit Distance works better than Soundex.

4. **Conclusions**

In the task like back– transliteration that the length of two strings are similar, we could use Global Edit Distance. And if the strings we are matching like names that sounds similar and has the same meaning, we could use Soundex. Local Edit Distance and N-gram may not be suitable for this task. Furthermore, if the frequency of replacement in the given matches are different, which means that not all the replacements are equivalent, then we could use a replacement matrix to take the place of the replacement parameter.

References

Sarvnaz Karimi, Andrew Turpin, and Falk Scholer (2006) English to Persian Translitera- tion. In Proceedings of the 13th Symposium on String Processing and Information Retrieval (SPIRE'06), Glasgow, UK, pp. 255–266.

Sarvnaz Karimi, Andrew Turpin, and Falk Scholer (2007) Corpus Effects on the Evaluation of Automated Transliteration Systems. In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07), Prague, Czech Republic, pp. 640– 647.

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.