Transliteration Project

REPORT_lhe_0811

Table of contents

<u>Table of contents</u>
Introduction
<u>Papers</u>
<u>Courses</u>
<u>Data Log</u>
<u>Date: 19/06/2018</u>
Date: 20/06/2018
Date: 23/06/2018
Date: 24/06/2018 (Important)
<u>Creation of the IPA table</u>
Date: 02/07/2018
Date: 03/07/2018
Date: 17/07/2018
Relevant URLs
<u>Problems Discussion</u>
<u>Experiments</u>
Pipeline (Appendix I)
<u>Data source</u>
Explanation of data source
<u>Data extraction</u>
<u>Preprocess</u>
Feature_vector and Clustering
<u>Types of features</u>
<u>Dimension of features</u>
The feature vector types for English characters (no smoothing)
The feature vector types for Chinese characters (before smoothing
Smoothing process
Storing feature vectors

```
K-means clustering
   Training
      Experiment 1 and 2
      Experiment 3
      Experiment 4
   Post-training
Results (Appendix II)
   Setting 1 (exp1 and exp2)
      en2ch
          Dev results
          Corresponding Best checkpoints
          Test results
      ar2en
          Dev results
          Corresponding Best checkpoints
          Test results
      en2jp
          Dev results
          Corresponding Best checkpoints
          Test results
      en2he
          Dev results
          Corresponding Best checkpoints
          Test results
   Setting 2 (exp3)
      en2ch
          Dev results
          Corresponding Best checkpoints
          Test results
      ar2en
          Dev results
          Corresponding Best checkpoints
          Test results
      en2jp
          Dev results
          Corresponding Best checkpoints
          Test results
      en2he
```

```
Corresponding Best checkpoints
          Test results
   Setting 3 (exp4)
       en2ch
          Dev results
          Corresponding Best checkpoints
          Test results
       <u>ar2en</u>
          Dev results
          Corresponding Best checkpoints
          Test results
       <u>en2jp</u>
          Dev results
          Corresponding Best checkpoints
          Test results
       en2he
          Dev results
          Corresponding Best checkpoints
          Test results
Analysis (en2ch)
```

File Structure (Appendix IV)

Introduction

The project can be divided into four parts, i.e. reading papers, learning deeplearning.ai courses, collecting data, and building/training some models of the English-to-Chinese

transliteration.

Papers

Date: 19/06/2018

I mainly focused on the sequence-to-sequence paper from Google, but I could not understand some concepts such as the bidirectional encoder. Also, this paper is rather short and hides many details. What I am currently interested is its discussion and description of the transliteration dataset (which will be discussed below), as it reveals some problems that I am also concerned while gathering my first dataset. Since I was trying to run the Google's model and adjusting my data, I have not finished the reading of the paper concerning the Monolingual Corpora, I will study it after the meeting on Wednesday.

Courses

Date: 19/06/2018

The Coursera website provides me with great materials to learn Machine Learning. I currently finished the first simple project of the binary classification using logistic regression, it is implemented on Jupyter/iPython Notebook. Since two month is a short period of time, I will accelerate the learning process, hopefully will know LSTM/GNN before building the final model.

Date: 10/07/2018

Heading to the hyperparameters section.

Data Log

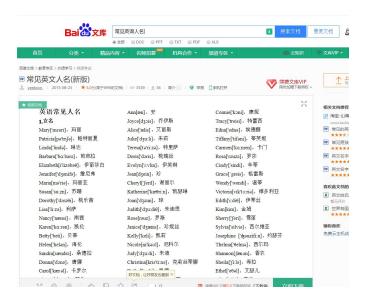
Date: 19/06/2018

Description:

The original en2chi dataset I have created has about 7000 word pairs, including people's names from different origins (particularly there is a small list of the ancient Greek Gods such as Zeus, and a small list of NBA players), places' names from various countries (cities in America/Britain/New Zealand take up the majority), and a list of brands and borrowed words. After some refinement of the data, the real number of word pairs for training (start date: 19/06/2018) is 6828.

Data Sources:

In practice, it is easier to use a Chinese search engine to search for some lists of English-Chinese word pairs. The engine I used for collecting the majority of the data is Baidu, where there is an embedded library called '百度文库' (Baidu's Library). An example is shown in the picture below:



In this example, I typed '常见英文人名' (common English names of people) in the textbox beside the logo of Baidu's Library, it actually gave a list of choices. Then I selected the documents whose words are reasonably formatted by the author, and wrote some code (in

Python) to extract the relevant information. There are some words coming from other websites, but all searched using Baidu.

Date: 20/06/2018

Description: Collecting a set of names of all Western celebrities' in Chinese.

Date: 23/06/2018

Description:

After finding two giant dictionaries of worldwide names of people and places, with Chinese transliteration aside, I was very excited because this data source is excellent (For one, the **transliteration style is unified** because of the single author; Also, the number of data is guaranteed to be large enough). I wrote some code in Python to extract the relevant information from the dictionary. After the meeting on 20th June, my supervisors and I agreed on **using people's names first** because they are more regular than places' names. I extracted all the people's names from the categories (the author has labelled each word with its related country(ies)) containing the key character '英' or '美', which are abbreviations of **the UK and the US** in Chinese respectively. Overall, there are **57948 distinct word pairs**, I divided them in a ratio of 8:1:1, the 80% part is used for training purposes, the rest are used for development and testing respectively.

Date: 24/06/2018 (Important)

Description:

The next step of data processing is to use some clustering algorithm (e.g. K-means) on the dataset I built. One of my supervisors, Professor Shay Cohen kindly taught me the idea of how to process the data, especially the construction of feature vectors. We decided to consider the left unigram and right unigram as features of each character of a English/Chinese word, and add 'IPA feature' for Chinese characters only. So a feature vector would look like:

[left_unigram ; right_unigram ; left_IPA; right_IPA; character_itself; original_word; class_to_be_determined]

For English character, the dimension would be (26+2)+(26+2)+1+1+1 (2 for $^{^{^{^{5}}}}$).

For Chinese character, the dim would be $(435+17+2)+(435+17+2)+(Table size)^2+1+1+1$.

Currently in the **training dataset** there are <u>435</u> distinct Chinese characters (and 17 characters in the original IPA table but not shown in the training dataset), it could be more in the whole dataset. So the dimension shown above is for **inductive setting**: cluster training dataset only. The **transductive learning** will be discussed later.

Creation of the IPA table

Today I spent the whole day to construct a 'reasonable' IPA table of which the columns are consonants and the rows are vowels, each cell is filled with zero or more Chinese characters or a mark 'Bigram'. (The 'Bigram' here is used for indicating the cells where some special pairs of Chinese bigram represent a composite sound, since we only consider sound for single character here, these special pairs are removed and marked as 'Bigram') In total, out of 435 characters, there are 150+ character missing in the original table (discussion about these characters will be presented in the Problems Discussion section of this report). I added all of them into the original IPA table from Wikipedia, the rules of insertions I used are as follows:

- 1. The top priority is not the original sound of the Chinese character, but its factual usage in English transliteration.
- 2. Bear 1. in mind, I did:
 - a. Search each of the 150+ Chinese characters in the training dataset first and look at all the English sequences of characters it corresponds to.
 - Search the pronunciation of the English sequences using Google, if there is an exact IPA presentation, insert the character into the table where
 objectively appropriate, otherwise analyse the sound of the original English

- word the sequence is attached to, and insert the Chinese character where **subjectively** appropriate.
- c. If the sound information of some particular English words is ambiguous and the Chinese transliteration looks weird and unmatched, I will insert the Chinese character according to its original sound.

The method I used for constructing the IPA table may not be the best, but even my linguistician friend, who knows both English and Chinese very well, cannot resolve the potential risks here. The final IPA table is uploaded to GitHub, compared to the original one, I added two more columns for the composite sound 'tr' and 'di' in order to deal with some special characters.

Date: 02/07/2018

Below are the recent achievements:

- 1. Create a csv file storing all the feature vectors generated from the English names dataset, each in a format of [left_unigram; right_unigram].
- 2. Create a csv file storing all the feature vectors generated from the Chinese names dataset, each in a format of [left_unigram; right_unigram; left_IPA].
- 3. It turns out that the training dataset does not include all the Chinese characters used in the whole dataset, But since I am using an inductive setting, the unknown information from the development and test dataset is not added to the unigram features. Only the IPA table is updated accordingly. (Whether this interpretation is right or wrong needs to be confirmed with my supervisors) Answer: do not remove.
- 4. The smoothing process is applied, so that the features with a frequency no more than three are deleted before the clustering process. Hopefully, this will minimize the negative effect brought by rare characters and special usage.
- 5. Create a virtual environment to prepare for the final training process.
- 6. All the data tagged by 'eva' are actually belong to the development dataset.
- 7. The annotated datasets using 2, 5, 10, and 15 clusters are ready for training. The clustering algorithm used is K-means.

9

Date: 03/07/2018

Description:

After figuring out how to use the openNMT model to train the data (I gave up on Nematus because of the lack of instructions), I tried to use it to train the annotated dataset with 5 clusters first, the training process is ongoing now. However, the tensorflow version of openNMT does not give enough and clear instructions for me to choose the right model type, so I chose the NMT_medium instinctively (the toy examples use the NMT_small), this part along with the specific configurations should be discussed on Wednesday's meeting.

Date: 17/07/2018

Description:

After some efforts and struggles, everything is prepared for running multiple settings of experiments.

----DATA LOG END----

Relevant URLs

Baidu: www.baidu.com

Baidu's Library: https://wenku.baidu.com

Transliterated Names of celebrities: http://www.manmankan.com/dy2013/mingxing/oumei/

Full dictionaries of names (people/places):

http://resource.ttplus.cn/publish/app/data/2016/04/04/6754/share1.html

The GitHub address of my project:

https://github.com/Lawhy/En2Chi-Transliteration

OpenNMT: http://opennmt.net/

Problems Discussion

Date: 19/06/2018

As mentioned in the Seq2Seq paper, there are many exceptional words whose transliteration probably cannot be learnt at all. Although currently I can erased these exceptions by checking the data line by line, such noisiness cannot be resolved manually when the dataset becomes larger and larger. In my observation, the exceptions occur a lot in the names of places, brand or simply by convention. The reasons behind may result from historical factors or literary manipulation of words. For example, Paris is transliterated to 巴黎, but it actually sounds more like 帕里斯. Nevertheless, 巴黎 has a sense of more 'beauty" than 帕里斯 in Chinese. My idea is that should we divide the dataset into names of peoples and names of others, because exceptions are much less frequent in people's names.

The second issue is about the general noisiness among the places' names. There are dozens of names containing certain parts irrelevant to transliteration, such as 'River', 'Port', 'Mount', 'Sea', 'Island', 'North/East/West/South', etc., which correspond to '河', '港', '山', '海', '岛, '北/东/西/南' in Chinese. Even though I can erase the irrelevant parts by seeking for some patterns, it will result in a 'collateral damage' in the sense that some words containing these seemingly irrelevant parts as sound components will lose them accidentally. For instance, the English name 'Westbrook' contains 'West', but in this case 'West' does not represent a direction. Instead, 'West' will be transliterated to '韦斯特'. Again, for a small dataset, I can resolve all the conflicts by checking it line by line. But it is impossible for a giant dataset.

The final concern is that Chinese characters are pictograms. There is no way to infer the sound by simply looking at the character itself. Also, Chinese is a many-to-many language in the sense that a single Chinese character may have several sounds and a particular sound may be possessed by several characters, whereas the sound of a single English word is almost fixed (although there exists different accents). Consequently, a model simply dealing with the textual data may treat alternative and other valid transcriptions as errors. (Mihaela Rosca et al., 2016)

Date: 25/06/2018

About the IPA table:

The original table was taken from Wikipedia, which uses a classical set of Chinese characters to represent the IPA sound made up by a consonant component and a vowel component. Most of the cells contain only one Chinese character, some contain a substitute for the alternative gender, a few of them contain a bigram because for some IPA combination, there is no way to use one character to represent that sound.

I believed that the original table is trying to give a one-to-one correspondence for the IPA sounds and the Chinese characters, and it actually covers a large portion of characters used in transliteration. In order to make the IPA feature more concrete, I added the missing 150+ Chinese characters into the IPA table in a way described in the Data Collection & Manipulation Log part. Among these 150+ characters, some actually appear multiple times in the dataset and their IPA positions are relatively easier to locate. However, the are some characters used in a special/weird way, the cases are:

- 1. Characters chosen because of convention, e.g. Athena (雅典娜), the pronunciation of Athena is [ə¹θi:nə], whereas 雅典娜 sounds like 'yudiana'. In this case, I did not put the character 雅 into the cell of its closest sound ([jʌ]), instead, I put it into the cell of [ə]. The same principle is also applied on '典", the second character, whereas the third one '娜' can be located easily because it sounds very similar to [nə]. This is a concrete example of what I mean by 'Factual usage prior to the original sound' in the Data Log section.
- 2. Characters used only one or two times, e.g. '典' in '雅典娜' (Athena), '耀' in 'Jauchem'. I did not delete them because they are minority.
- 3. Characters used in some **marginal names** which are used not frequently in English (so that I cannot find a proper IPA from Wikipedia). In this case, I first tried to listen to several sounds of that name from some website offering pronunciation.

 Moreover, I also took account of the original sound of the Chinese character.

After finishing the table, I kindly asked a linguistician friend to check its validity. He pointed out that the IPA table actually includes some pronunciation not commonly used in English, I believed that the author of the original table also tried to include some exceptional names.

Experiments

Date: 19/06/2018

ERROR 2470000 0.374696 1078 2877

In the first round, I used 7011 word pairs to train the seq2seq model and obtained an error rate of 37.4696% for the training. However, the model behaves rather badly on the test data.

I am now re-training the model using a refined dataset (By refined I mean that I erase some exceptions manually) of the size 6800+. Hopefully it will be finished by tomorrow's meeting.

Date: 21/06/2018

ERROR 2760000 0.333927 938 2809

In the second round, I removed some fatal errors from the dataset, and the error rate decreases to 33.3927% at the 2,760,000th trail. However, I just found an excellent data source which includes about 820 thousand word pairs (although not all of them are English-to-Chinese). The baseline experiment for the current dataset ends at this point.

Date: 08/07/2018

Using openNMT to train --, -+, +-, ++. First round on tensorflow. Encounter low-level bugs.

(Corresponding log/results not shown here)

Date: 10/07/2018

Using openNMT to train --, -+, +-, ++. Second round on pyTorch with GPU support.

Here is the experiment setup:

https://github.com/Lawhy/En2Chi-Transliteration/blob/master/README.md

Date: 20/07/2018

Finish and record a full **en2ch** experiment with setting 1.

Details in here:

https://docs.google.com/document/d/1vIZGSELcZVQyAs_9dorSxc0_gMqxVu3CCPF3RIYT1lg/edit?usp=sharing

Date: 22/07/2018 (Important)

Add more feature vector choices, prepare for next setting of experiments by changing the features. Available features are:

```
{
"L": [L_unigram],

"R": [R_unigram],

"LR": [L_unigram ; R_unigram],

"bLR": [LR_bigram],

"LRbLR": [L_unigram ; R_unigram ; LR_bigram]

# IPA only for Chinese data
}
```

Details are in the GitHub file: feature_vector.py.

A problem: Do not know how to use spectral clustering algorithm in this particular project.

A reminder: exp1 results were not totally collected. Need to discuss new ideas of processing Chinese data with Shay on Monday.

Date: 24/07/2018-12/08/2018

Run several other experiments using different settings, clarify the results, do some analysis and execute the designed human evaluation.

Pipeline (Appendix I)

1. Data source

The worldwide names of people, a giant dictionary written by the Chinese official media, Xinhua news. The language used is Mandarin.

Explanation of data source

I only choose **people's names** because they are more regular (without too many exceptional transliterations), thus less noisy. The experiment results can prove that.

2. Data extraction

57998 word pairs (English-to-Chinese) are extracted from the dictionary. The dictionary contains names in all languages in the world, I select those originated from the US and the UK. Some words have two kinds of transliteration for male and female respectively, these words are removed from the dataset.

3. Preprocess

- a. Divide the 57998 word pairs in a ratio of 8:1:1 (Training, development, testing dataset). And further divide the pairs into parallel files, so in total 6 data files are generated.
- b. Apply inductive learning (Learn everything from the training dataset).
- c. All the English words are transformed to their lower case.
- d. The English words contain 26 characters plus inverted comma (e.g. O'Neil), the Chinese words in the training dataset contain 434 characters.
- e. An IPA table of 408 non-empty cells is created for Chinese data. Each cell represents a sound combined by a consonant and a vowel. The table was built from the incomplete table created by the dictionary's author. I added another 150 characters into the table by observing how they are used in the training dataset. (Details in <u>Creation of the IPA table</u>)

4. Feature vector and Clustering

The main technique used in this project is to construct feature vectors for every character of each word, then apply clustering algorithm to the feature vectors, finally assign the cluster number to each character.

a. Types of features

- i. Left (Right) unigram, i.e. a one-shot vector indicating which character is in the left (right) of the current character.
- ii. Left-right bigram, i.e. a one-shot vector indicating which two characters (order matters) are adjacent to the current character.
- iii. Left (Right) IPA, i.e. a sparse vector indicating the positions of the left (right) character in the IPA table. Note: The IPA table for Chinese only.

b. Dimension of features

A start symbol (^) and an end symbol (\$) are added to the word during the creation of feature vectors, but '\$' is removed from the left context, '^' is removed from the right context, '^\$' is removed from the left-right context. So for English, the dimension of left (right) unigram is 26+1+1=28, the dimension of left-right bigram is 28*28-1=783. For Chinese, the dimension of the unigram is 434+1=435, the dimension of the left-right bigram is 435*435-1=189225.

c. The feature vector types for English characters (no smoothing)

- i. [left_unigram; right_unigram] (number of features: 28*2 = 56)
- ii. [left_unigram; right_unigram; left_right_bigram] (number of features:28*2 + 783 = 839)

d. The feature vector types for Chinese characters (before smoothing)

- i. [left_unigram; right_unigram; left_IPA; right_IPA] (435*2+408*2=1686)
- ii. [left_unigram; right_unigram; left_right_bigram; left_IPA; right_IPA](435*2+408*2+189225=190911)

e. Smoothing process

A smoothing process is applied on Chinese data and all the features with a total frequency <= 3 are removed.

f. Storing feature vectors

The feature vectors are stored in csr_matrix so as to save space and memory.

g. K-means clustering

Apply k-means clustering to the sparse matrices of training data, then predict the labels of the development and test data according to the training data.

- i. The labelled source word looks like: $a \mid 1 \text{ m} \mid 2 \text{ y} \mid 3$.
- ii. The labelled target word looks like: 拉-1 帕-1 迪-1 斯-0.
- iii. Use u'|' for source side only because OpenNMT only supports source side features.

5. Training

The Seq2seq model with attention used in the experiments is OpenNMT: http://opennmt.net/

a. Experiment 1 and 2

- Model: a 2-layer LSTM with 500 hidden units on both the encoder/decoder. (The default openNMT model)
- ii. Clusters: 2 4 5 7 9 10 12 15

iii. Algorithm: K-means

iv. Feature vector types: [left_unigram; right_unigram; IPA (for Chinese only)]

b. Experiment 3

 i. Model: a 2-layer LSTM with 500 hidden units on both the encoder/decoder. (The default openNMT model)

ii. Clusters: 2 4 5 7 9 10 12 15

iii. Algorithm: K-means

iv. [left_unigram; right_unigram; left_right_bigram; IPA (for Chinese only)]

c. Experiment 4

 Model: 2-layer, biLSTM encoder and LSTM decoder with 500 hidden units.

ii. Clusters: 2 4 5 7 9 10 12 15

iii. Algorithm: K-means

iv. Feature vector types: [left_unigram; right_unigram; IPA (for Chinese only)]

6. Post-training

Evaluate the last ten checkpoints on the development dataset, pick the best checkpoint and use it to predict the test dataset.

Results (Appendix II)

```
Setting 1 (exp1 and exp2)
{ LSTM encoder/decoder
2-layers,
500 hidden units,
K-means,
Clusters: 2 5 10 15 (In exp1) 4 7 9 12 (In exp2)
[left_unigram; right_unigram; IPA (for Chinese only)]}
```

en2ch

Dev results

				en2ch					
Ann/Cls	2	4	5	7	9	10	12	15	
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.29686	0.30066	0.29496	0.29755	0.31550	0.31688	0.32016	0.32568	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.12105	0.12272	0.12105	0.12239	0.13026	0.13031	0.13231	0.13279	
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.30186	0.29703	0.27874	0.29289	0.28616	0.29565	0.29254	0.28564	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.12466	0.12353	0.11243	0.11959	0.11528	0.12175	0.11954	0.11598	
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.29134	0.30790	0.28875	0.29893	0.31636	0.31498	0.32258	0.32361	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.11819	0.12681	0.11641	0.12191	0.13150	0.13107	0.13295	0.13284	
 (Docali	WER: 0.29841								
(Baseli ne)				CER: 0	.12239				

	en2ch											
Ann/Cls	2	4	5	7	9	10	12	15				
++	13500	14600	12600	14100	14600	12700	14700	13500				
+-	12300	14200	12000	14100	14300	12000	14700	12100				
-+	12300	14900	12600	14500	14500	12500	15000	12300				
		12000										

				en2ch						
Ann/Cls	2	4	5	7	9	10	12	15		
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.28658	0.29848	0.29779	0.29934	0.31384	0.31522	0.32022	0.32885		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.11502	0.12233	0.12062	0.12126	0.12867	0.12979	0.13150	0.13374		
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.29917	0.29210	0.28675	0.29762	0.28865	0.29089	0.29055	0.28727		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.12281	0.11897	0.11614	0.12014	0.11763	0.11902	0.11694	0.11529		
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.29365	0.30435	0.29486	0.30210	0.31677	0.31970	0.32419	0.31522		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.11870	0.12483	0.11982	0.12227	0.12942	0.13086	0.13230	0.12985		
 (Deceli		WER: 0.29279 CER: 0.12030								
(Baseli ne)										

ar2en

Dev results

				ar2en					
Ann/Cls	2	4	5	7	9	10	12	15	
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.78057	0.78267	0.78407	0.78686	0.78477	0.79455	0.78477	0.80014	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.23724	0.23300	0.23652	0.24097	0.23517	0.23641	0.23786	0.26260	
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.77498	0.78197	0.77498	0.78197	0.77638	0.75961	0.77708	0.77009	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.23455	0.23372	0.23145	0.23186	0.23341	0.22720	0.23217	0.22637	
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.78337	0.77987	0.77778	0.78407	0.79595	0.79874	0.78756	0.78407	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.23797	0.23579	0.23621	0.24045	0.24118	0.24294	0.24066	0.24024	
 (Deceli	WER: 0.77638								
(Baseli ne)	CER: 0.23735								

	ar2en										
Ann/Cls	2	4	5	7	9	10	12	15			
++	10700	14400	10300	14300	14500	10400	14100	10200			
+-	10600	14900	10800	14900	14900	10300	14800	10400			
-+	10500	14300	10300	14100	15000	10800	14300	10300			
		14800									

				ar2en						
Ann/Cls	2	4	5	7	9	10	12	15		
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.77107	0.77233	0.78239	0.78050	0.78553	0.77987	0.77987	0.80189		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.23519	0.23445	0.23857	0.23683	0.24004	0.24004	0.24196	0.27090		
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.77987	0.76730	0.77484	0.76289	0.76891	0.76667	0.76415	0.76792		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.23509	0.23317	0.23766	0.23134	0.23079	0.22987	0.23345	0.23134		
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.78113	0.76541	0.77984	0.78239	0.78365	0.78931	0.79245	0.79119		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.24022	0.23583	0.23784	0.24187	0.24196	0.24480	0.24956	0.24654		
 (Deceli		WER: 0.77044								
(Baseli ne)		CER: 0.23720								

en2jp

<u>Dev results</u>

				en2jp				
Ann/Cls	2	4	5	7	9	10	12	15
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:
	0.54637	0.54637	0.56436	0.56773	0.55481	0.65037	0.64025	0.66667
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:
	0.23072	0.23731	0.23971	0.24664	0.23705	0.29398	0.28688	0.30622
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:
	0.54188	0.54019	0.54525	0.52333	0.53907	0.50815	0.51771	0.53345
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:
	0.23525	0.23363	0.23380	0.22900	0.23388	0.22284	0.22601	0.22798
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:
	0.54919	0.55031	0.57167	0.62057	0.57729	0.63969	0.58066	0.58179
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:
	0.23260	0.23269	0.24279	0.27986	0.24724	0.29886	0.25383	0.25674
 (Danal'	WER: 0.54300							
(Baseli ne)				CER: 0	.23380			

	en2jp											
Ann/Cls	2	4	5	7	9	10	12	15				
++	14700	14400	14800	14100	14400	14500	14500	14100				
+-	14300	14200	14600	14700	14800	14700	14600	14600				
-+	14900	15000	14700	14400	14800	14900	14700	15000				
		14900										

				en2jp					
Ann/Cls	2	4	5	7	9	10	12	15	
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.54607	0.54045	0.55730	0.58034	0.55900	0.63427	0.62303	0.65056	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.23015	0.22445	0.23058	0.24344	0.23489	0.28004	0.27987	0.29135	
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.54326	0.53315	0.53539	0.53989	0.53427	0.53933	0.54551	0.53820	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.23092	0.22600	0.22505	0.22281	0.21970	0.22453	0.22704	0.22790	
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.55112	0.54775	0.56236	0.60618	0.56236	0.65787	0.57584	0.58933	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.23265	0.22920	0.23679	0.27141	0.23955	0.29325	0.24119	0.24456	
 (Danali	WER: 0.54157								
(Baseli ne)				CER: 0	.22807				

en2he

Dev results

				en2he					
Ann/Cls	2	4	5	7	9	10	12	15	
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.92	0.92	0.9	0.94	0.96	0.9	0.9	0.9	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.49237	0.53817	0.54580	0.5916	0.59542	0.54962	0.56107	0.58779	
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.84	0.82	0.84	0.88	0.82	0.82	0.9	0.84	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.41221	0.41603	0.40840	0.42748	0.41603	0.44275	0.41603	0.41603	
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.9	0.9	0.9	0.94	0.92	0.92	0.94	0.9	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.51908	0.53435	0.55725	0.55725	0.55344	0.57634	0.59924	0.59160	
 (Danali	WER: 0.82								
(Baseli ne)				CER: 0	.43130				

	en2he											
Ann/Cls	2	4	5	7	9	10	12	15				
++	400	14100	800	14100	14100	700	14100	900				
+-	700	14100	1200	14100	14100	1000	14100	900				
-+	500	14100	400	14100	14100	900	14100	500				
		700										

				en2he						
Ann/Cls	2	4	5	7	9	10	12	15		
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.95667	0.94333	0.96	0.96	0.95667	0.96	0.95667	0.96		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.58364	0.55904	0.57442	0.59594	0.59471	0.62239	0.61685	0.64268		
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.90667	0.90667	0.91333	0.90667	0.9	0.91	0.89333	0.89333		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.47294	0.50923	0.44957	0.45879	0.45264	0.48954	0.45756	0.45633		
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.94333	0.94667	0.96	0.94667	0.96	0.95667	0.96	0.95667		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.54797	0.56581	0.59041	0.55966	0.61255	0.62054	0.62116	0.62854		
 (Dood!				WER	: 0.9	•		•		
(Baseli ne)		CER: 0.47786								

Setting 2 (exp3)

```
{ LSTM encoder/decoder,
2-layers,
500 hidden units,
K-means,
Clusters: 2 4 5 7 9 10 12 15 (all in exp3),
[left_unigram; right_unigram; left_right_bigram; IPA (for Chinese only)]}
```

Note: Compared to Setting 1, Setting 2 only changes the feature vector type.

en2ch

Dev results

				en2ch					
Ann/Cls	2	4	5	7	9	10	12	15	
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.29928	0.30497	0.30117	0.29858	0.30739	0.31757	0.30877	0.31136	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.12056	0.12498	0.1225	0.12056	0.12579	0.13241	0.12552	0.12827	
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.30497	0.29254	0.28737	0.28616	0.29306	0.29341	0.29496	0.28909	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.12557	0.11841	0.11798	0.11663	0.11841	0.11728	0.11868	0.11652	
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.29375	0.30066	0.29479	0.29928	0.31912	0.32085	0.32033	0.31412	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.11932	0.12514	0.12019	0.12223	0.13198	0.13134	0.13058	0.12816	
 (Docali				WER: 0	.29841				
(Baseli ne)	CER: 0.12239								

	en2ch										
Ann/Cls	2	4	5	7	9	10	12	15			
++	14200	14100	14900	15000	14200	14800	14200	15000			
+-	14800	14800	14500	14400	14700	14400	14200	14500			
-+	14200	15000	14100	14600	14800	14100	14500	14600			
		In exp_1									

				en2ch					
Ann/Cls	2	4	5	7	9	10	12	15	
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.29831	0.30452	0.29486	0.29607	0.31401	0.31246	0.29831	0.30814	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.11875	0.12483	0.11822	0.11945	0.12809	0.12921	0.12051	0.12585	
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.29934	0.29520	0.29262	0.29417	0.29486	0.28761	0.29572	0.28830	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.12307	0.11758	0.11817	0.11838	0.11907	0.11731	0.11833	0.11646	
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.28968	0.30090	0.29952	0.29796	0.31228	0.31694	0.31073	0.30469	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.11763	0.12281	0.12078	0.11950	0.12809	0.13081	0.12649	0.12441	
 (Docali	WER: 0.29279								
(Baseli ne)	CER: 0.12030								

ar2en

Dev results

				ar2en						
Ann/Cls	2	4	5	7	9	10	12	15		
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.78896	0.77918	0.79525	0.78267	0.77918	0.79525	0.78616	0.77568		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.23372	0.23807	0.23962	0.23797	0.23735	0.24180	0.23714	0.24325		
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.78197	0.77848	0.77149	0.77638	0.76450	0.76240	0.76869	0.77289		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.23869	0.23331	0.22731	0.23186	0.22596	0.23000	0.23672	0.22865		
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.80503	0.77778	0.78686	0.79525	0.77638	0.77987	0.78826	0.79385		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.25474	0.23103	0.23745	0.23662	0.23755	0.24449	0.24045	0.24242		
 (Deceli	WER: 0.77638									
(Baseli ne)				CER: 0	.23735					

	ar2en										
Ann/Cls	2	4	5	7	9	10	12	15			
++	15000	14100	14200	14100	14800	14100	15000	14600			
+-	14900	14700	14800	14100	14100	14300	14600	15000			
-+	14800	14400	14800	14200	14700	14900	14600	14700			
		In exp_1									

				ar2en						
Ann/Cls	2	4	5	7	9	10	12	15		
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.76855	0.78931	0.77673	0.78428	0.77107	0.78616	0.78491	0.78239		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.23116	0.23720	0.23455	0.23748	0.23500	0.23986	0.23711	0.24270		
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.77044	0.77799	0.77421	0.76289	0.76604	0.76604	0.77107	0.77107		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.23803	0.23729	0.23354	0.23400	0.23299	0.23345	0.23198	0.22768		
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.78553	0.78616	0.77233	0.78994	0.77673	0.77736	0.78113	0.77421		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.24911	0.24260	0.23867	0.24215	0.24462	0.23986	0.24242	0.24288		
 (Daneli	- WER: 0.77044									
(Baseli ne)		CER: 0.23720								

en2jp

<u>Dev results</u>

				en2jp					
Ann/Cls	2	4	5	7	9	10	12	15	
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.54356	0.55031	0.53963	0.60821	0.56043	0.56998	0.58010	0.58460	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.22935	0.23876	0.23406	0.27635	0.23953	0.24030	0.25751	0.24844	
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.53288	0.53907	0.53513	0.53513	0.51714	0.52389	0.53738	0.52501	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.23354	0.22952	0.23097	0.23183	0.22378	0.22960	0.22729	0.22858	
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.53007	0.55705	0.54413	0.55818	0.57111	0.58123	0.62619	0.58460	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.22883	0.24150	0.23380	0.24450	0.24638	0.24972	0.28319	0.25195	
	WER: 0.54300								
(Baseli ne)	CER: 0.23380								

	en2jp										
Ann/Cls	2	4	5	7	9	10	12	15			
++	14700	14100	15000	14900	14200	14100	14100	14100			
+-	15000	14500	14400	14800	14800	14700	15000	14200			
-+	14700	14700	14200	14100	14200	14400	14100	14100			
		In exp_1									

				en2jp					
Ann/Cls	2	4	5	7	9	10	12	15	
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.54101	0.54831	0.55506	0.60899	0.55	0.58258	0.58202	0.58483	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.22583	0.22868	0.22799	0.27011	0.23187	0.24283	0.24655	0.24577	
+-	WER: 0.55506	WER: 0.53539	WER: 0.54101	WER: 0.54157	WER: 0.53708	WER: 0.54045	WER: 0.53034	WER: 0.54157	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.23472	0.22704	0.22479	0.22427	0.22289	0.22686	0.22289	0.22756	
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.54213	0.55281	0.54326	0.55618	0.55955	0.57135	0.61910	0.58764	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.22488	0.22911	0.22635	0.23429	0.23446	0.24732	0.27374	0.24171	
 (Danali	WER: 0.54157								
(Baseli ne)				CER: 0	.22807				

en2he

Dev results

				en2he						
Ann/Cls	2	4	5	7	9	10	12	15		
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.9	0.88	0.88	0.92	0.9	0.92	0.9	0.96		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.49237	0.49237	0.49237	0.5458	0.51908	0.53817	0.5458	0.61832		
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.86	0.86	0.92	0.86	0.86	0.88	0.86	0.9		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.42366	0.40458	0.44275	0.42748	0.40076	0.41603	0.41985	0.41221		
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.88	0.92	0.94	0.9	0.92	0.92	0.92	0.94		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.47328	0.50382	0.5	0.57252	0.54198	0.57252	0.59924	0.58015		
 (Baseli ne)	WER: 0.82									
	CER: 0.43130									

		Спескропп								
en2he										
Ann/Cls	2	4	5	7	9	10	12	15		
++	5000	4200	4800	4300	4400	5000	5000	4200		
+-	4300	4200	4700	4200	4200	5000	4700	4900		
-+	4800	4900	4900	4100	4600	4500	4100	4500		
		In exp_1								

en2he								
Ann/Cls	2	4	5	7	9	10	12	15
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:
	0.95	0.94667	0.94333	0.96333	0.95333	0.95333	0.96	0.95667
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:
	0.54244	0.54920	0.54797	0.60578	0.59902	0.60332	0.63100	0.65806
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:
	0.91333	0.89333	0.89333	0.89333	0.90333	0.89	0.9	0.89667
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:
	0.47663	0.45572	0.47970	0.46863	0.46494	0.45572	0.44834	0.46617
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:
	0.94333	0.94667	0.93333	0.95667	0.95333	0.96	0.95667	0.96333
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:
	0.52522	0.56765	0.57011	0.59287	0.60578	0.59287	0.61747	0.65191
 (Baseli ne)	WER: 0.9							
	CER: 0.47786							

Setting 3 (exp4)

```
{ biLSTM encoder,
  LSTM decoder,
  2-layers,
  500 hidden units,
  K-means ,
  Clusters: 2 4 5 7 9 10 12 15 (all in exp4, including new baseline),
  [left_unigram ; right_unigram ; IPA (for Chinese only)]}
```

Note: Compared to Setting 1, Setting 3 only changes the encoder type. This results in a new baseline using biLSTM structure.

en2ch

Dev results

				en2ch						
Ann/Cls	2	4	5	7	9	10	12	15		
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.27925	0.27977	0.28150	0.28478	0.28788	0.28754	0.29548	0.29772		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.11227	0.11361	0.11421	0.11442	0.11636	0.11604	0.12035	0.12094		
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.27856	0.27477	0.27925	0.27408	0.27736	0.27425	0.27684	0.27459		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.11318	0.11114	0.11324	0.11049	0.11140	0.11006	0.11135	0.11070		
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.28064	0.28322	0.27943	0.28167	0.28926	0.29341	0.29928	0.30394		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.11345	0.11534	0.11345	0.11588	0.11631	0.11852	0.12148	0.12261		
 (Deceli	WER: 0.28322									
(Baseli ne)	CER: 0.11383									

	en2ch										
Ann/Cls	2	4	5	7	9	10	12	15			
++	14900	14600	15000	14300	15000	14600	14900	14600			
+-	14300	14600	14900	15000	14300	14500	14400	14500			
-+	14100	14600	14100	14600	14800	14700	14100	15000			
		14400									

Test results

				en2ch						
Ann/Cls	2	4	5	7	9	10	12	15		
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.28295	0.27657	0.28209	0.28244	0.29624	0.28520	0.29055	0.29279		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.11316	0.11273	0.11252	0.11449	0.11715	0.11465	0.11838	0.11827		
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.28019	0.27484	0.27502	0.27450	0.27709	0.27191	0.27933	0.27570		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.11326	0.11156	0.11150	0.10969	0.11129	0.10846	0.11310	0.11262		
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:		
	0.28019	0.27536	0.28399	0.29106	0.29693	0.28623	0.29310	0.29658		
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:		
	0.11252	0.11118	0.11475	0.11598	0.11907	0.11475	0.11897	0.11966		
	WER: 0.27278									
(Baseli ne)	CER: 0.11054									

ar2en

Dev results

				ar2en					
Ann/Cls	2	4	5	7	9	10	12	15	
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.76799	0.77149	0.76869	0.77568	0.77638	0.78337	0.78337	0.77638	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.22648	0.22979	0.22700	0.23258	0.23051	0.23610	0.23445	0.23238	
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.76171	0.75611	0.76171	0.76730	0.75472	0.76520	0.75961	0.77778	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.22658	0.22710	0.22192	0.22979	0.22461	0.22658	0.22772	0.22720	
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.75961	0.77149	0.76450	0.77708	0.76799	0.77708	0.77428	0.77428	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.22948	0.22958	0.22855	0.22969	0.23310	0.23372	0.23662	0.23517	
 (Baseli ne)	WER: 0.77079								
	CER: 0.22772								

ar2en										
Ann/Cls	2	4	5	7	9	10	12	15		
++	14300	14100	14800	15000	14700	14800	14700	14200		
+-	15000	14500	14300	14100	15000	15000	14800	14100		
-+	14900	14700	14200	14300	15000	14900	15000	14700		
	14100									

Test results

	ar2en											
Ann/Cls	2	4	5	7	9	10	12	15				
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:				
	0.75346	0.76730	0.76164	0.77044	0.78113	0.77925	0.77296	0.78365				
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:				
	0.22685	0.22933	0.22722	0.23088	0.23574	0.23674	0.23610	0.23473				
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:				
	0.75975	0.76038	0.77233	0.76478	0.75849	0.75849	0.75597	0.76164				
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:				
	0.22813	0.22850	0.23134	0.22832	0.22740	0.22511	0.22484	0.22859				
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:				
	0.77044	0.76792	0.77736	0.78050	0.77799	0.77358	0.77799	0.78113				
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:				
	0.23143	0.23226	0.23528	0.23281	0.23528	0.23473	0.23638	0.23564				
 (Danali	WER: 0.75849											
(Baseli ne)	CER: 0.22484											

en2jp

Dev results

				en2jp					
Ann/Cls	2	4	5	7	9	10	12	15	
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.45363	0.47049	0.47836	0.47499	0.47836	0.47611	0.49185	0.50197	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.20067	0.20161	0.20880	0.20794	0.20709	0.21625	0.21796	0.21454	
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.46374	0.45475	0.45700	0.45981	0.45868	0.46599	0.46431	0.45419	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.20358	0.20247	0.19861	0.19973	0.19553	0.20743	0.20170	0.19990	
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.46037	0.47218	0.47274	0.47723	0.48679	0.48904	0.49803	0.50365	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.19878	0.20555	0.20692	0.20700	0.20743	0.20735	0.21180	0.21976	
 (Baseli ne)	WER: 0.47049								
	CER: 0.20058								

en2jp											
Ann/Cls	2	2 4 5 7 9 10 12 15									
++	14800	14500	14500	14200	14200	14100	14700	14200			
+-	14400	14600	14800	14700	14200	14200	14400	14300			
-+	15000	14900	14200	14400	14200	14700	14900	14900			
		14900									

Test results

				en2jp					
Ann/Cls	2	4	5	7	9	10	12	15	
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.45112	0.45843	0.47360	0.48483	0.48371	0.49213	0.48034	0.50337	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.19121	0.19441	0.19631	0.20131	0.20045	0.20839	0.20166	0.20865	
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.45169	0.45618	0.45449	0.44551	0.46236	0.46236	0.45056	0.45281	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.19803	0.19561	0.19026	0.19052	0.19786	0.19121	0.19044	0.19208	
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.46348	0.46517	0.46685	0.47191	0.47640	0.48764	0.49888	0.49831	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.19734	0.19631	0.19415	0.19933	0.20028	0.20563	0.20830	0.20917	
 (Baseli ne)	WER: 0.45787								
	CER: 0.19302								

en2he

Dev results

				en2he					
Ann/Cls	2	4	5	7	9	10	12	15	
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.88	0.82	0.82	0.88	0.88	0.92	0.88	0.9	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.4084	0.40076	0.4542	0.49237	0.48473	0.52672	0.51527	0.5229	
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.82	0.8	0.88	0.88	0.84	0.8	0.84	0.82	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.37405	0.36260	0.39695	0.38168	0.40076	0.37405	0.37786	0.36641	
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.82	0.88	0.84	0.9	0.9	0.92	0.88	0.9	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.3626	0.47710	0.42366	0.46947	0.53053	0.52672	0.52672	0.51145	
	WER: 0.9								
(Baseli ne)	CER: 0.4084								

	en2he										
Ann/Cls	2	4	5	7	9	10	12	15			
++	4900	4700	4500	4300	4600	4800	4100	4100			
+-	4100	4500	4100	4400	4800	4100	4600	4900			
-+	4800	4200	4300	4100	4900	4100	4600	4800			
		4100									

Test results

				en2he					
Ann/Cls	2	4	5	7	9	10	12	15	
++	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.88	0.91667	0.91	0.93	0.93333	0.92667	0.94667	0.95333	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.42374	0.48401	0.47294	0.50185	0.52706	0.51415	0.54305	0.56150	
+-	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.88667	0.87	0.86333	0.87333	0.85	0.87333	0.86	0.87333	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.41451	0.41636	0.40590	0.42866	0.42128	0.41082	0.42189	0.41205	
-+	WER:	WER:	WER:	WER:	WER:	WER:	WER:	WER:	
	0.87667	0.92333	0.93333	0.93	0.94333	0.93667	0.95	0.94667	
	CER:	CER:	CER:	CER:	CER:	CER:	CER:	CER:	
	0.41513	0.47478	0.47232	0.49569	0.52276	0.54736	0.54982	0.55843	
 (Baseli ne)	WER: 0.85								
	CER: 0.40836								

Conclusion: By changing encoder to bilstm, the results seem to be insensitive to hyperparameters tuning.

Analysis (en2ch)

Based on setting 1 experiment:

- Comparing baseline result and the best result (5cls +-):
 - o The sequence "ch" could be transliterated into many ways in Chinese, such as '奇' (sounds like 'chee'), '赫' (sounds like 'her'), '克' (sounds like 'ker') etc. Based on my observations, the machine does not work very well when it tries to predict 'ch' in different cases, and this is a general problem that happens in both results.
 - o The best result is able to capture 'th' as a common English sequence of characters and the transliteration of such sequence is more accurate in various cases, whereas the baseline result **occasionally** breaks 'th' down into 't' and 'h' and only transliterates 't' into '特'(sounds like ter) when 'th' appears in the initial positions or the end positions. This might prove that my hypothesis of what clustering does (see below) is on the right track.

What clustering does:

I think what the clustering basically does it to assign each character to it 'popular' neighbour. For example, in the 5cls English file, most characters that are assigned to cluster 1 happen to have an adjacent character 'e' in their right. It indicates that 'e' occurs very frequently in the names, and character such as 'h' does not have that influence. Certainly, the start symbol 'A' and the end symbol '\$' should have a significant power because of the way we construct the feature vectors. (This may explain why generally the 2 clusters does not give better results—the 'A' and '\$' already need 2 clusters in the first place). So in my own understanding, the clustering actually divide characters into groups that **centre on the most frequent characters or their combinations** (if the left-right-bigram is used as a feature). However, this is not a 'pure' improvement in the sense that it does not preserve all the right instances in the baseline model, then make

progress based on that. There are 400 words that the best result got right whereas the baseline got wrong. And 288 words that the baseline got right but the best result got wrong. So if we combine the baseline and the best results, actually we can get a WER=22.9% (There are 5794 words in the development set, 1615 wrong predictions in the best result, if the 288 words are replaced using the baseline result, we can get only 1327 wrong predictions and thus a WER=22.9%).

File Structure (Appendix IV)

Here specifies the arrangement of files in /disk/ocean/lhe/transliteration

Main directory using openNMT-py: /disk/ocean/lhe/transliteration/nmt-py

