Transliteration Project

REPORT_lhe_0817

Table of contents

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

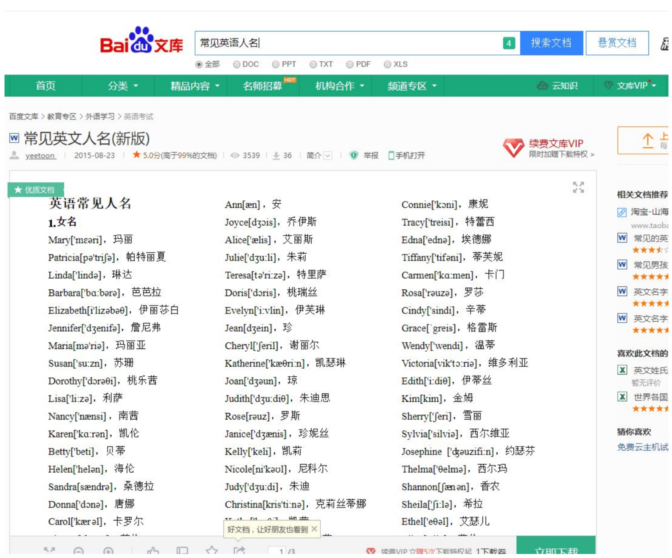
Date: 19/06/2018

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 ^ $).

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 435 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.
    b. 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 ; right_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. <u>(Whether this interpretation is right or wrong needs to be confirmed with my supervisors) Answer: do not remove.</u>
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.

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/

Original IPA table:

https://en.wikipedia.org/wiki/Template:Transcription_into_Chinese

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. <u>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.</u>

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 [əˈθiːnə], 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 (Old)

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_gMqxVu3CCPF3RlYT1lg/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 (en2chi)

a. Divide the 57998 word pairs in a ratio of 8:1:1 (**Training: 46358, development: 5794, testing: 5796**). 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), **overall 27 characters**;the Chinese words in the training dataset contain 434 characters， an additional 14 characters appear in dev/tst dataset, so **in total, # of Chinese characters is 448**.

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 Creation of the IPA table)

4. Preprocess (NET)

a. The original data is divided in a ratio of 64:16:20 as the training, development and test dataset. Each word is in a form of :

English word (no spacing) ; word in other language (with spacing) ; a number

Each part is separated by a tab character.

E.g.

mannerheim م ا ن ر ه ا ي م 2

b. I first removed the tailing number by using `re.sub(r'\t[0-9]+', '', line)`

c. Then find all word pairs that are complete (i.e. no missing part on the src or tgt side) by using `re.findall(r'(.+)\t(.+)', word_pair)`

d. For both the English side and the other-language side, skip the word if it does not match: `re.match(r'^[\'\w]+$', word`

This means a valid word must contain at least one valid character or one inverted comma (name like O'Neil).

e. After preprocessing, the numbers of characters in each dataset are the following:

   i. Arabic: 52 ; English: 136

   ii. Hebrew: 30 ; English: 125

   iii. Japanese: 86 ; English: 156

   iv. Russian: 50 ; English: 164

f. Compared to the original data, the number of characters decrease noticeably whereas the number of examples only decrease by 0.1%.

5. 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.

iv. Left (Right) alt unigram, i.e. a one-shot vector indicating the second character to the left(right) of the current character.

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)

iii. [Left_alt_unigram; left_unigram]

iv. [right_unigram; right_alt_unigram]

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)

iii. [Left_alt_unigram; left_unigram; left_IPA; right_IPA]

iv. [right_unigram; right_alt_unigram; left_IPA; right_IPA]

v. etc.

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|1 m|2 y|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.

6. Training

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

a. Experiment 1 and 2

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. 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

i. 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)]

d. Experiment 5 (Testing Left_two_unigrams)

i. 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_alt_unigram; left_unigram; IPA (for Chinese only)]

v. A small experiment on +- and ++ of Chinese data

e. Experiment 6 (Testing Right_two_unigrams)

i. 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: [right_unigram; right_alt_unigram; IPA (for Chinese only)]

v. A small experiment on +- and ++ of Chinese data

f. Additional experiment (NET transliteration)

i. 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)]

v. Generating comparable results using the data from the NET-Transliteration.

g. Experiment on features

i. Do features ablation. Details in: Additional Experiment (exp-features)

7. 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: 0.29686	WER: 0.30066	WER: 0.29496	WER: 0.29755	WER: 0.31550	WER: 0.31688	WER: 0.32016	WER: 0.32568
	CER: 0.12105	CER: 0.12272	CER: 0.12105	CER: 0.12239	CER: 0.13026	CER: 0.13031	CER: 0.13231	CER: 0.13279
+-	WER: 0.30186	WER: 0.29703	WER: 0.27874	WER: 0.29289	WER: 0.28616	WER: 0.29565	WER: 0.29254	WER: 0.28564
	CER: 0.12466	CER: 0.12353	CER: 0.11243	CER: 0.11959	CER: 0.11528	CER: 0.12175	CER: 0.11954	CER: 0.11598
-+	WER: 0.29134	WER: 0.30790	WER: 0.28875	WER: 0.29893	WER: 0.31636	WER: 0.31498	WER: 0.32258	WER: 0.32361
	CER: 0.11819	CER: 0.12681	CER: 0.11641	CER: 0.12191	CER: 0.13150	CER: 0.13107	CER: 0.13295	CER: 0.13284
-- (Baseline)	WER: 0.29841							
	CER: 0.12239							

Corresponding Best checkpoints

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							

Test results

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

ar2en

Dev results

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

Corresponding Best checkpoints

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							

Test results

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

en2jp

Dev results

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

Corresponding Best checkpoints

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**							

Test results

en2jp								
Ann/Cls	2	4	5	7	9	10	12	15
++	WER: 0.54607	WER: 0.54045	WER: 0.55730	WER: 0.58034	WER: 0.55900	WER: 0.63427	WER: 0.62303	WER: 0.65056
	CER: 0.23015	CER: 0.22445	CER: 0.23058	CER: 0.24344	CER: 0.23489	CER: 0.28004	CER: 0.27987	CER: 0.29135
+-	WER: 0.54326	WER: 0.53315	WER: 0.53539	WER: 0.53989	WER: 0.53427	WER: 0.53933	WER: 0.54551	WER: 0.53820
	CER: 0.23092	CER: 0.22600	CER: 0.22505	CER: 0.22281	CER: 0.21970	CER: 0.22453	CER: 0.22704	CER: 0.22790
-+	WER: 0.55112	WER: 0.54775	WER: 0.56236	WER: 0.60618	WER: 0.56236	WER: 0.65787	WER: 0.57584	WER: 0.58933
	CER: 0.23265	CER: 0.22920	CER: 0.23679	CER: 0.27141	CER: 0.23955	CER: 0.29325	CER: 0.24119	CER: 0.24456
-- (Baseline)	WER: 0.54157							
	CER: 0.22807							

en2he

Dev results

en2he								
Ann/Cls	2	4	5	7	9	10	12	15
++	WER: 0.92	WER: 0.92	WER: 0.9	WER: 0.94	WER: 0.96	WER: 0.9	WER: 0.9	WER: 0.9
	CER: 0.49237	CER: 0.53817	CER: 0.54580	CER: 0.5916	CER: 0.59542	CER: 0.54962	CER: 0.56107	CER: 0.58779
+-	WER: 0.84	WER: 0.82	WER: 0.84	WER: 0.88	WER: 0.82	WER: 0.82	WER: 0.9	WER: 0.84
	CER: 0.41221	CER: 0.41603	CER: 0.40840	CER: 0.42748	CER: 0.41603	CER: 0.44275	CER: 0.41603	CER: 0.41603
-+	WER: 0.9	WER: 0.9	WER: 0.9	WER: 0.94	WER: 0.92	WER: 0.92	WER: 0.94	WER: 0.9
	CER: 0.51908	CER: 0.53435	CER: 0.55725	CER: 0.55725	CER: 0.55344	CER: 0.57634	CER: 0.59924	CER: 0.59160
-- (Baseline)	WER: 0.82							
	CER: 0.43130							

Corresponding Best checkpoints

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							

Test results

en2he								
Ann/Cls	**2**	**4**	**5**	**7**	**9**	**10**	**12**	**15**
++	**WER: 0.95667**	**WER: 0.94333**	**WER: 0.96**	**WER: 0.96**	**WER: 0.95667**	**WER: 0.96**	**WER: 0.95667**	**WER: 0.96**
	CER: 0.58364	**CER: 0.55904**	**CER: 0.57442**	**CER: 0.59594**	**CER: 0.59471**	**CER: 0.62239**	**CER: 0.61685**	**CER: 0.64268**
+-	**WER: 0.90667**	**WER: 0.90667**	**WER: 0.91333**	**WER: 0.90667**	**WER: 0.9**	**WER: 0.91**	**WER: 0.89333**	**WER: 0.89333**
	CER: 0.47294	**CER: 0.50923**	**CER: 0.44957**	**CER: 0.45879**	**CER: 0.45264**	**CER: 0.48954**	**CER: 0.45756**	**CER: 0.45633**
-+	**WER: 0.94333**	**WER: 0.94667**	**WER: 0.96**	**WER: 0.94667**	**WER: 0.96**	**WER: 0.95667**	**WER: 0.96**	**WER: 0.95667**
	CER: 0.54797	**CER: 0.56581**	**CER: 0.59041**	**CER: 0.55966**	**CER: 0.61255**	**CER: 0.62054**	**CER: 0.62116**	**CER: 0.62854**
-- (Baseline)	**WER: 0.9**							
	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: 0.29928	WER: 0.30497	WER: 0.30117	WER: 0.29858	WER: 0.30739	WER: 0.31757	WER: 0.30877	WER: 0.31136
	CER: 0.12056	CER: 0.12498	CER: 0.1225	CER: 0.12056	CER: 0.12579	CER: 0.13241	CER: 0.12552	CER: 0.12827
+-	WER: 0.30497	WER: 0.29254	WER: 0.28737	WER: 0.28616	WER: 0.29306	WER: 0.29341	WER: 0.29496	WER: 0.28909
	CER: 0.12557	CER: 0.11841	CER: 0.11798	CER: 0.11663	CER: 0.11841	CER: 0.11728	CER: 0.11868	CER: 0.11652
-+	WER: 0.29375	WER: 0.30066	WER: 0.29479	WER: 0.29928	WER: 0.31912	WER: 0.32085	WER: 0.32033	WER: 0.31412
	CER: 0.11932	CER: 0.12514	CER: 0.12019	CER: 0.12223	CER: 0.13198	CER: 0.13134	CER: 0.13058	CER: 0.12816
-- (Baseline)	WER: 0.29841							
	CER: 0.12239							

Corresponding Best checkpoints

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							

Test results

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

ar2en

Dev results

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

Corresponding Best checkpoints

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							

Test results

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

en2jp

Dev results

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

Corresponding Best checkpoints

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							

Test results

en2jp								
Ann/Cls	**2**	**4**	**5**	**7**	**9**	**10**	**12**	**15**
++	**WER: 0.54101**	**WER: 0.54831**	**WER: 0.55506**	**WER: 0.60899**	**WER: 0.55**	**WER: 0.58258**	**WER: 0.58202**	**WER: 0.58483**
	CER: 0.22583	**CER: 0.22868**	**CER: 0.22799**	**CER: 0.27011**	**CER: 0.23187**	**CER: 0.24283**	**CER: 0.24655**	**CER: 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: 0.23472	**CER: 0.22704**	**CER: 0.22479**	**CER: 0.22427**	**CER: 0.22289**	**CER: 0.22686**	**CER: 0.22289**	**CER: 0.22756**
-+	**WER: 0.54213**	**WER: 0.55281**	**WER: 0.54326**	**WER: 0.55618**	**WER: 0.55955**	**WER: 0.57135**	**WER: 0.61910**	**WER: 0.58764**
	CER: 0.22488	**CER: 0.22911**	**CER: 0.22635**	**CER: 0.23429**	**CER: 0.23446**	**CER: 0.24732**	**CER: 0.27374**	**CER: 0.24171**
-- (Baseline)	**WER: 0.54157**							
	CER: 0.22807							

en2he

Dev results

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

Corresponding Best checkpoints

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							

Test results

en2he								
Ann/Cls	2	4	5	7	9	10	12	15
++	WER: 0.95	WER: 0.94667	WER: 0.94333	WER: 0.96333	WER: 0.95333	WER: 0.95333	WER: 0.96	WER: 0.95667
	CER: 0.54244	CER: 0.54920	CER: 0.54797	CER: 0.60578	CER: 0.59902	CER: 0.60332	CER: 0.63100	CER: 0.65806
+-	WER: 0.91333	WER: 0.89333	WER: 0.89333	WER: 0.89333	WER: 0.90333	WER: 0.89	WER: 0.9	WER: 0.89667
	CER: 0.47663	CER: 0.45572	CER: 0.47970	CER: 0.46863	CER: 0.46494	CER: 0.45572	CER: 0.44834	CER: 0.46617
-+	WER: 0.94333	WER: 0.94667	WER: 0.93333	WER: 0.95667	WER: 0.95333	WER: 0.96	WER: 0.95667	WER: 0.96333
	CER: 0.52522	CER: 0.56765	CER: 0.57011	CER: 0.59287	CER: 0.60578	CER: 0.59287	CER: 0.61747	CER: 0.65191
-- (Baseline)	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: 0.27925	WER: 0.27977	WER: 0.28150	WER: 0.28478	WER: 0.28788	WER: 0.28754	WER: 0.29548	WER: 0.29772
	CER: 0.11227	CER: 0.11361	CER: 0.11421	CER: 0.11442	CER: 0.11636	CER: 0.11604	CER: 0.12035	CER: 0.12094
+-	WER: 0.27856	WER: 0.27477	WER: 0.27925	WER: 0.27408	WER: 0.27736	WER: 0.27425	WER: 0.27684	WER: 0.27459
	CER: 0.11318	CER: 0.11114	CER: 0.11324	CER: 0.11049	CER: 0.11140	CER: 0.11006	CER: 0.11135	CER: 0.11070
-+	WER: 0.28064	WER: 0.28322	WER: 0.27943	WER: 0.28167	WER: 0.28926	WER: 0.29341	WER: 0.29928	WER: 0.30394
	CER: 0.11345	CER: 0.11534	CER: 0.11345	CER: 0.11588	CER: 0.11631	CER: 0.11852	CER: 0.12148	CER: 0.12261
-- (Baseline)	WER: 0.28322							
	CER: 0.11383							

Corresponding Best checkpoints

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: 0.28295**	**WER: 0.27657**	**WER: 0.28209**	**WER: 0.28244**	**WER: 0.29624**	**WER: 0.28520**	**WER: 0.29055**	**WER: 0.29279**
	CER: 0.11316	**CER: 0.11273**	**CER: 0.11252**	**CER: 0.11449**	**CER: 0.11715**	**CER: 0.11465**	**CER: 0.11838**	**CER: 0.11827**
+-	**WER: 0.28019**	**WER: 0.27484**	**WER: 0.27502**	**WER: 0.27450**	**WER: 0.27709**	**WER: 0.27191**	**WER: 0.27933**	**WER: 0.27570**
	CER: 0.11326	**CER: 0.11156**	**CER: 0.11150**	**CER: 0.10969**	**CER: 0.11129**	**CER: 0.10846**	**CER: 0.11310**	**CER: 0.11262**
-+	**WER: 0.28019**	**WER: 0.27536**	**WER: 0.28399**	**WER: 0.29106**	**WER: 0.29693**	**WER: 0.28623**	**WER: 0.29310**	**WER: 0.29658**
	CER: 0.11252	**CER: 0.11118**	**CER: 0.11475**	**CER: 0.11598**	**CER: 0.11907**	**CER: 0.11475**	**CER: 0.11897**	**CER: 0.11966**
-- (Baseline)	**WER: 0.27278**							
	CER: 0.11054							

ar2en

Dev results

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

Corresponding Best checkpoints

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: 0.75346**	**WER: 0.76730**	**WER: 0.76164**	**WER: 0.77044**	**WER: 0.78113**	**WER: 0.77925**	**WER: 0.77296**	**WER: 0.78365**
	CER: 0.22685	**CER: 0.22933**	**CER: 0.22722**	**CER: 0.23088**	**CER: 0.23574**	**CER: 0.23674**	**CER: 0.23610**	**CER: 0.23473**
+-	**WER: 0.75975**	**WER: 0.76038**	**WER: 0.77233**	**WER: 0.76478**	**WER: 0.75849**	**WER: 0.75849**	**WER: 0.75597**	**WER: 0.76164**
	CER: 0.22813	**CER: 0.22850**	**CER: 0.23134**	**CER: 0.22832**	**CER: 0.22740**	**CER: 0.22511**	**CER: 0.22484**	**CER: 0.22859**
-+	**WER: 0.77044**	**WER: 0.76792**	**WER: 0.77736**	**WER: 0.78050**	**WER: 0.77799**	**WER: 0.77358**	**WER: 0.77799**	**WER: 0.78113**
	CER: 0.23143	**CER: 0.23226**	**CER: 0.23528**	**CER: 0.23281**	**CER: 0.23528**	**CER: 0.23473**	**CER: 0.23638**	**CER: 0.23564**
-- (Baseline)	**WER: 0.75849**							
	CER: 0.22484							

en2jp

Dev results

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

Corresponding Best checkpoints

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

en2he

Dev results

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

Corresponding Best checkpoints

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: 0.88**	**WER: 0.91667**	**WER: 0.91**	**WER: 0.93**	**WER: 0.93333**	**WER: 0.92667**	**WER: 0.94667**	**WER: 0.95333**
	CER: 0.42374	**CER: 0.48401**	**CER: 0.47294**	**CER: 0.50185**	**CER: 0.52706**	**CER: 0.51415**	**CER: 0.54305**	**CER: 0.56150**
+-	**WER: 0.88667**	**WER: 0.87**	**WER: 0.86333**	**WER: 0.87333**	**WER: 0.85**	**WER: 0.87333**	**WER: 0.86**	**WER: 0.87333**
	CER: 0.41451	**CER: 0.41636**	**CER: 0.40590**	**CER: 0.42866**	**CER: 0.42128**	**CER: 0.41082**	**CER: 0.42189**	**CER: 0.41205**
-+	**WER: 0.87667**	**WER: 0.92333**	**WER: 0.93333**	**WER: 0.93**	**WER: 0.94333**	**WER: 0.93667**	**WER: 0.95**	**WER: 0.94667**
	CER: 0.41513	**CER: 0.47478**	**CER: 0.47232**	**CER: 0.49569**	**CER: 0.52276**	**CER: 0.54736**	**CER: 0.54982**	**CER: 0.55843**
-- (Baseline)	**WER: 0.85**							
	CER: 0.40836							

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

Setting 4 (exp5)

{ biLSTM encoder,

LSTM decoder,

2-layers,

500 hidden units,

K-means ,

Clusters: 2 4 5 7 9 10 12 15,

[left_alt_unigram; left_unigram; IPA (for Chinese only)] }

Note: A small experiment testing the left-two-uni-grams feature vector.

en2ch

Dev results

en2ch								
Ann/Cls	**2**	**4**	**5**	**7**	**9**	**10**	**12**	**15**
++	**WER: 0.27805**	**WER: 0.28340**	**WER: 0.28219**	**WER: 0.28616**	**WER: 0.28115**	**WER: 0.28616**	**WER: 0.29237**	**WER: 0.29099**
	CER: 0.11216	**CER: 0.11442**	**CER: 0.11431**	**CER: 0.11442**	**CER: 0.11415**	**CER: 0.11663**	**CER: 0.11830**	**CER: 0.11895**
+-	**WER: 0.27408**	**WER: 0.27356**	**WER: 0.27546**	**WER: 0.27408**	**WER: 0.27287**	**WER: 0.27270**	**WER: 0.27736**	**WER: 0.27252**
	CER: 0.11060	**CER: 0.11006**	**CER: 0.11184**	**CER: 0.10957**	**CER: 0.11140**	**CER: 0.10882**	**CER: 0.11140**	**CER: 0.10930**
-- (Baseline)	**WER: 0.28322**							
	CER: 0.11383							

Corresponding Best checkpoints

en2ch								
Ann/Cls	**2**	**4**	**5**	**7**	**9**	**10**	**12**	**15**
++	**14100**	**14700**	**14600**	**14200**	**15000**	**14600**	**14600**	**14100**
+-	**14200**	**14600**	**14200**	**14600**	**14200**	**14200**	**14900**	**14500**
--	**In exp4**							

Test results

en2ch								
Ann/Cls	**2**	**4**	**5**	**7**	**9**	**10**	**12**	**15**
++	**WER: 0.27847**	**WER: 0.28744**	**WER: 0.27743**	**WER: 0.28675**	**WER: 0.28088**	**WER: 0.28537**	**WER: 0.29124**	**WER: 0.29469**
	CER: 0.11241	**CER: 0.11491**	**CER: 0.11209**	**CER: 0.11481**	**CER: 0.11310**	**CER: 0.11603**	**CER: 0.11822**	**CER: 0.11966**
+-	**WER: 0.27277**	**WER: 0.27657**	**WER: 0.27588**	**WER: 0.27312**	**WER: 0.27726**	**WER: 0.27088**	**WER: 0.27139**	**WER: 0.27295**
	CER: 0.11022	**CER: 0.11108**	**CER: 0.11129**	**CER: 0.11044**	**CER: 0.11124**	**CER: 0.10953**	**CER: 0.10852**	**CER: 0.10964**
-- (Baseline)	**WER: 0.27278**							
	CER: 0.11054							

Setting 5 (exp6)

{ biLSTM encoder,

LSTM decoder,

2-layers,

500 hidden units,

K-means ,

Clusters: 2 4 5 7 9 10 12 15,

[right_unigram; right_alt_unigram; IPA (for Chinese only)] **}**

Note: A small experiment testing the right-two-uni-grams feature vector.

en2ch

Dev results

en2ch								
Ann/Cls	**2**	**4**	**5**	**7**	**9**	**10**	**12**	**15**
++	**WER: 0.28788**	**WER: 0.29151**	**WER: 0.29134**	**WER: 0.29375**	**WER: 0.29220**	**WER: 0.30273**	**WER: 0.29876**	**WER: 0.29997**
	CER: 0.11685	**CER: 0.11668**	**CER: 0.11760**	**CER: 0.11970**	**CER: 0.11970**	**CER: 0.12169**	**CER: 0.12089**	**CER: 0.12261**
+-	**WER: 0.28012**	**WER: 0.27908**	**WER: 0.28236**	**WER: 0.27787**	**WER: 0.27477**	**WER: 0.27649**	**WER: 0.28167**	**WER: 0.27425**
	CER: 0.11361	**CER: 0.11275**	**CER: 0.11474**	**CER: 0.11286**	**CER: 0.11043**	**CER: 0.11049**	**CER: 0.11453**	**CER: 0.11135**
-- (Baseline)	**WER: 0.28322**							
	CER: 0.11383							

Corresponding Best checkpoints

en2ch								
Ann/Cls	**2**	**4**	**5**	**7**	**9**	**10**	**12**	**15**
++	**14100**	**14600**	**14600**	**15000**	**14700**	**15000**	**14800**	**14800**
+-	**14700**	**14400**	**14400**	**14800**	**14400**	**14100**	**14800**	**14100**
--	**In exp4**							

Test results

en2ch								
Ann/Cls	**2**	**4**	**5**	**7**	**9**	**10**	**12**	**15**
++	**WER: 0.28571**	**WER: 0.28865**	**WER: 0.28934**	**WER: 0.29244**	**WER: 0.29003**	**WER: 0.29227**	**WER: 0.29986**	**WER: 0.30297**
	CER: 0.11342	**CER: 0.11545**	**CER: 0.11753**	**CER: 0.11715**	**CER: 0.11737**	**CER: 0.11747**	**CER: 0.12121**	**CER: 0.12201**
+-	**WER: 0.27536**	**WER: 0.27588**	**WER: 0.27295**	**WER: 0.27346**	**WER: 0.27364**	**WER: 0.27415**	**WER: 0.27139**	**WER: 0.26984**
	CER: 0.11289	**CER: 0.11049**	**CER: 0.11140**	**CER: 0.11033**	**CER: 0.11060**	**CER: 0.11044**	**CER: 0.10793**	**CER: 0.10974**
-- (Baseline)	**WER: 0.27278**							
	CER: 0.11054							

Additional Experiment (NET Transliteration)

Language & Direction:

- _2en
  - ar2en, he2en, jp2en, ru2en
- en2_
  - en2ar, en2he, en2jp, en2ru

Note1: The Japanese here is actually Katakana, a special part of Japanese.

Note2: Compared to the original data, a small portion of words that totally make no sense are removed.

Setting: (Using the same as exp4)

{ biLSTM encoder,

LSTM decoder,

2-layers,

500 hidden units,

K-means ,

Clusters: 2 5 7 10 12 15

[left_unigram ; right_unigram] }

ru2en

Dev results

ru2en								
Ann/Cls	2	4	5	7	9	10	12	15
++	WER: 0.49933	NA	WER: 0.50059	WER: 0.50711	NA	WER: 0.50714	WER: 0.50779	WER: 0.51263
	CER: 0.14339	NA	CER: 0.14379	CER: 0.14468	NA	CER: 0.14443	CER: 0.14468	CER: 0.14579
+-	WER: 0.49804	NA	WER: 0.49964	WER: 0.50040	NA	WER: 0.49708	WER: 0.50044	WER: 0.50105
	CER: 0.14345	NA	CER: 0.14387	CER: 0.14392	NA	CER: 0.14321	CER: 0.14371	CER: 0.14407
-+	WER: 0.50070	NA	WER: 0.50444	WER: 0.50654	NA	WER: 0.50935	WER: 0.50894	WER: 0.51427
	CER: 0.14355	NA	CER: 0.14451	CER: 0.14489	NA	CER: 0.14609	CER: 0.14499	CER: 0.14604
-- (Baseline)	WER: 0.49872							
	CER: 0.14378							

Corresponding Best checkpoints

ru2en								
Ann/Cls	2	4	5	7	9	10	12	15
++	29100	NA	29100	29100	NA	29100	29200	29100
+-	39900	NA	39900	39900	NA	39900	39900	29100
-+	39200	NA	39900	39200	NA	39900	39900	39200
--	39100							

Note: This time all the last ten checkpoints give the same WER/CER, so I pick checkpoint according to the best training accuracy (Please see models/logs).

Test results

ru2en								
Ann/Cls	**2**	**4**	**5**	**7**	**9**	**10**	**12**	**15**
++	**WER: 0.50507**	**N/A**	**WER: 0.50657**	**WER: 0.50861**	**N/A**	**WER: 0.51379**	**WER: 0.51327**	**WER: 0.51641**
	CER: 0.14566		**CER: 0.14588**	**CER: 0.14579**		**CER: 0.14764**	**CER: 0.14745**	**CER: 0.14808**
+-	**WER: 0.50294**		**WER: 0.50010**	**WER: 0.50264**		**WER: 0.49870**	**WER: 0.50010**	**WER: 0.50169**
	CER: 0.14559		**CER: 0.14528**	**CER: 0.14605**		**CER: 0.14485**	**CER: 0.14567**	**CER: 0.14557**
-+	**WER: 0.50449**		**WER: 0.50849**	**WER: 0.51166**		**WER: 0.51354**	**WER: 0.51696**	**WER: 0.51918**
	CER: 0.14586		**CER: 0.14672**	**CER: 0.14729**		**CER: 0.14789**	**CER: 0.14811**	**CER: 0.14902**
-- (Baseline)	**WER: 0.50175**							
	CER: 0.14528							

ar2en

Dev results

ar2en								
Ann/Cls	2	4	5	7	9	10	12	15
++	WER: 0.81759	NA	WER: 0.81161	WER: 0.81422	NA	WER: 0.81084	WER: 0.80612	WER: 0.81354
	CER: 0.26007	NA	CER: 0.25713	CER: 0.25786	NA	CER: 0.25869	CER: 0.25733	CER: 0.25850
+-	WER: 0.81354	NA	WER: 0.80997	WER: 0.80949	NA	WER: 0.80968	WER: 0.81104	WER: 0.81335
	CER: 0.25862	NA	CER: 0.25868	CER: 0.25760	NA	CER: 0.25815	CER: 0.25753	CER: 0.25757
-+	WER: 0.81239	NA	WER: 0.81229	WER: 0.81190	NA	WER: 0.81682	WER: 0.81007	WER: 0.81702
	CER: 0.25843	NA	CER: 0.25773	CER: 0.25836	NA	CER: 0.25934	CER: 0.25673	CER: 0.25891
-- (Baseline)	WER: 0.80949							
	CER: 0.25802							

Corresponding Best checkpoints

ar2en								
Ann/Cls	2	4	5	7	9	10	12	15
++	24900	NA	24900	24900	NA	24400	24400	24900
+-	24200	NA	24400	24400	NA	24900	24400	24900
-+	24900	NA	24900	24800	NA	24500	24400	24900
--	24300							

Note: Sometimes all the last ten checkpoints give the same WER/CER, so in that case I pick checkpoint according to the best training accuracy (Please see models/logs).

Test results

ar2en								
op[Ann /Cls	2	4	5	7	9	10	12	15
++	WER: 0.81038	N/A	WER: 0.80814	WER: 0.80814	N/A	WER: 0.80906	WER: 0.80953	WER: 0.81076
	CER: 0.26143		CER: 0.25995	CER: 0.26132		CER: 0.25965	CER: 0.26000	CER: 0.26013
+-	WER: 0.81038		WER: 0.80791	WER: 0.80891		WER: 0.80451	WER: 0.80292	WER: 0.80868
	CER: 0.26162		CER: 0.26091	CER: 0.26060		CER: 0.26000	CER: 0.25966	CER: 0.25957
-+	WER: 0.81115		WER: 0.81524	WER: 0.81208		WER: 0.81177	WER: 0.80292	WER: 0.81292
	CER: 0.26146		CER: 0.26160	CER: 0.26235		CER: 0.26149	CER: 0.26001	CER: 0.26108
-- (Baseline)	WER: 0.81061							
	CER: 0.26068							

he2en

Dev results

he2en								
Ann/Cls	2	4	5	7	9	10	12	15
++	WER: 0.83974	NA	WER: 0.84351	WER: 0.84602	NA	WER: 0.84388	WER: 0.84351	WER: 0.84665
	CER: 0.27517		CER: 0.27658	CER: 0.27704		CER: 0.27697	CER: 0.27356	CER: 0.27686
+-	WER: 0.84250		WER: 0.84225	WER: 0.83899		WER: 0.83911	WER: 0.83811	WER: 0.84250
	CER: 0.27465		CER: 0.27580	CER: 0.27343		CER: 0.27571	CER: 0.27363	CER: 0.27574
-+	WER: 0.84074		WER: 0.84363	WER: 0.84162		WER: 0.84677	WER: 0.84187	WER: 0.84351
	CER: 0.27678		CER: 0.27600	CER: 0.27573		CER: 0.27571	CER: 0.27495	CER: 0.27469
-- (Baseline)	WER: 0.83748							
	CER: 0.27334							

Corresponding Best checkpoints

he2en								
Ann/Cls	2	4	5	7	9	10	12	15
++	24900	NA	24900	24900	NA	24900	24400	24900
+-	24400		24200	24400		24400	24400	24400
-+	24300		24900	24900		24900	24300	24900
--	24900							

Note: This time all the last ten checkpoints give the same WER/CER, so I pick checkpoint according to the best training accuracy (Please see models/logs).

Test results

he2en								
Ann/Cls	2	4	5	7	9	10	12	15
++	WER: 0.84149	NA	WER: 0.84139	WER: 0.84270	NA	WER: 0.84561	WER: 0.83737	WER: 0.84903
	CER: 0.27553		CER: 0.27396	CER: 0.27538		CER: 0.27598	CER: 0.27391	CER: 0.27819
+-	WER: 0.84159		WER: 0.83828	WER: 0.84260		WER: 0.84220	WER: 0.83436	WER: 0.83938
	CER: 0.27352		CER: 0.27520	CER: 0.27509		CER: 0.27527	CER: 0.27228	CER: 0.27445
-+	WER: 0.84149		WER: 0.83878	WER: 0.84159		WER: 0.84581	WER: 0.83938	WER: 0.84400
	CER: 0.27358		CER: 0.27481	CER: 0.27693		CER: 0.27786	CER: 0.27666	CER: 0.27571
-- (Baseline)	WER: 0.83626							
	CER: 0.27088							

jp2en

Dev results

jp2en								
Ann/Cls	**2**	**4**	**5**	**7**	**9**	**10**	**12**	**15**
++	WER: 0.72464	NA	WER: 0.73390	WER: 0.73256	NA	WER: 0.73688	WER: 0.73428	WER: 0.73821
	CER: 0.21840		CER: 0.21989	CER: 0.21943		CER: 0.22088	CER: 0.22165	CER: 0.22148
+-	WER: 0.72749		WER: 0.73066	WER: 0.72851		WER: 0.72996	WER: 0.72622	WER: 0.72838
	CER: 0.21839		CER: 0.21868	CER: 0.21809		CER: 0.21995	CER: 0.21848	CER: 0.21844
-+	WER: 0.72870		WER: 0.73022	WER: 0.73757		WER: 0.73713	WER: 0.73345	WER: 0.73967
	CER: 0.21846		CER: 0.21908	CER: 0.22071		CER: 0.22116	CER: 0.22256	CER: 0.22211
-- (Baseline)	WER: 0.72819							
	CER: 0.21857							

Corresponding Best checkpoints

jp2en								
Ann/Cls	**2**	**4**	**5**	**7**	**9**	**10**	**12**	**15**
++	29600	NA	29600	29600	NA	29600	29600	29600
+-	29600		29600	29600		29600	29600	29600
-+	29600		29600	29600		29600	29600	29600
--	29600							

Note: This time all the last ten checkpoints give the same WER/CER, so I pick checkpoint according to the best training accuracy (Please see models/logs).

Test results

jp2en								
Ann/Cls	**2**	**4**	**5**	**7**	**9**	**10**	**12**	**15**
++	**WER: 0.72423**	**NA**	**WER: 0.72378**	**WER: 0.72784**	**NA**	**WER: 0.72966**	**WER: 0.72900**	**WER: 0.72591**
	CER: 0.21697		**CER: 0.21762**	**CER: 0.21863**		**CER: 0.21963**	**CER: 0.22035**	**CER: 0.21946**
+-	**WER: 0.72119**		**WER: 0.72083**	**WER: 0.71875**		**WER: 0.71972**	**WER: 0.71946**	**WER: 0.72002**
	CER: 0.21654		**CER: 0.21642**	**CER: 0.21641**		**CER: 0.21638**	**CER: 0.21717**	**CER: 0.21679**
-+	**WER: 0.72180**		**WER: 0.72347**	**WER: 0.72789**		**WER: 0.72865**	**WER: 0.73174**	**WER: 0.73022**
	CER: 0.21617		**CER: 0.21751**	**CER: 0.21880**		**CER: 0.21946**	**CER: 0.22097**	**CER: 0.22144**
-- (Baseline)	**WER: 0.72170**							
	CER:0.21694							

Note 16/08/2018:

For the direction: en -> other languages, the training will be finished without checking the results.

Additional Experiment (exp-features)

This is an experiment testing several combinations of features including:

Features

{

--- Finished ---

LR: [left_unigram ; right_unigram] (done in exp4)

--- Partially finished ---

LL: [left_alt_unigram ; left_unigram] (done in exp5 except for -+)

RR: [right_unigram ; right_alt_unigram] (done in exp6 except for -+)

--- New ---

L: [left_unigram]

R: [right_uingram]

LL R: [left_alt_unigram ; left_unigram ; right_unigram]

RR L: [rigt_unigram ; right_alt_unigram ; left_unigram]

LR bLR: [left_unigram ; right_unigram ; left_right_bigram]

LL RR LR: Just combination of above LL RR and LR

}

IPA features

For Chinese data an additional [left_IPA ; right_IPA] is augmented to every feature vector.

Parameters

{ biLSTM encoder, LSTM decoder, 2 layers, 500 hidden units, k-means, clusters: **10**}

Table (Dev results)

Features	-+	+-	++
LR	**WER: 0.29341** **CER: 0.11852**	**WER: 0.27425** **CER: 0.11006**	**WER: 0.28754** **CER: 0.11604**
LR bLR	**WER: 0.29151** **CER: 0.11835**	**WER: 0.27908** **CER: 0.11162**	**WER: 0.28875** **CER: 0.11728**
LL	**WER: 0.29047** **CER: 0.11738**	**WER: 0.27270** **CER: 0.10882**	**WER: 0.28616** **CER: 0.11663**
RR	**WER: 0.29755** **CER: 0.12159**	**WER: 0.27649** **CER: 0.11049**	**WER: 0.30273** **CER: 0.12169**
L	**WER: 0.28426** **CER: 0.11620**	**WER: 0.28098** **CER: 0.11318**	**WER: 0.28737** **CER: 0.11652**
R	**WER: 0.28892** **CER: 0.11825**	**WER: 0.27477** **CER: 0.11017**	**WER: 0.28530** **CER: 0.11782**
LL R	**WER: 0.29254** **CER: 0.11808**	**WER: 0.27459** **CER: 0.11049**	**WER: 0.29669** **CER: 0.11997**
RR L	**WER: 0.30031** **CER: 0.12223**	**WER: 0.27528** **CER: 0.11092**	**WER: 0.30273** **CER: 0.12374**
LL RR LR	**WER: 0.29341** **CER: 0.11873**	**WER: 0.27787** **CER: 0.11297**	**WER: 0.29427** **CER: 0.11927**
Baseline (From exp4)	**WER: 0.28322** **CER: 0.11383**		

Analysis (en2ch)

Based on setting 1 experiment:

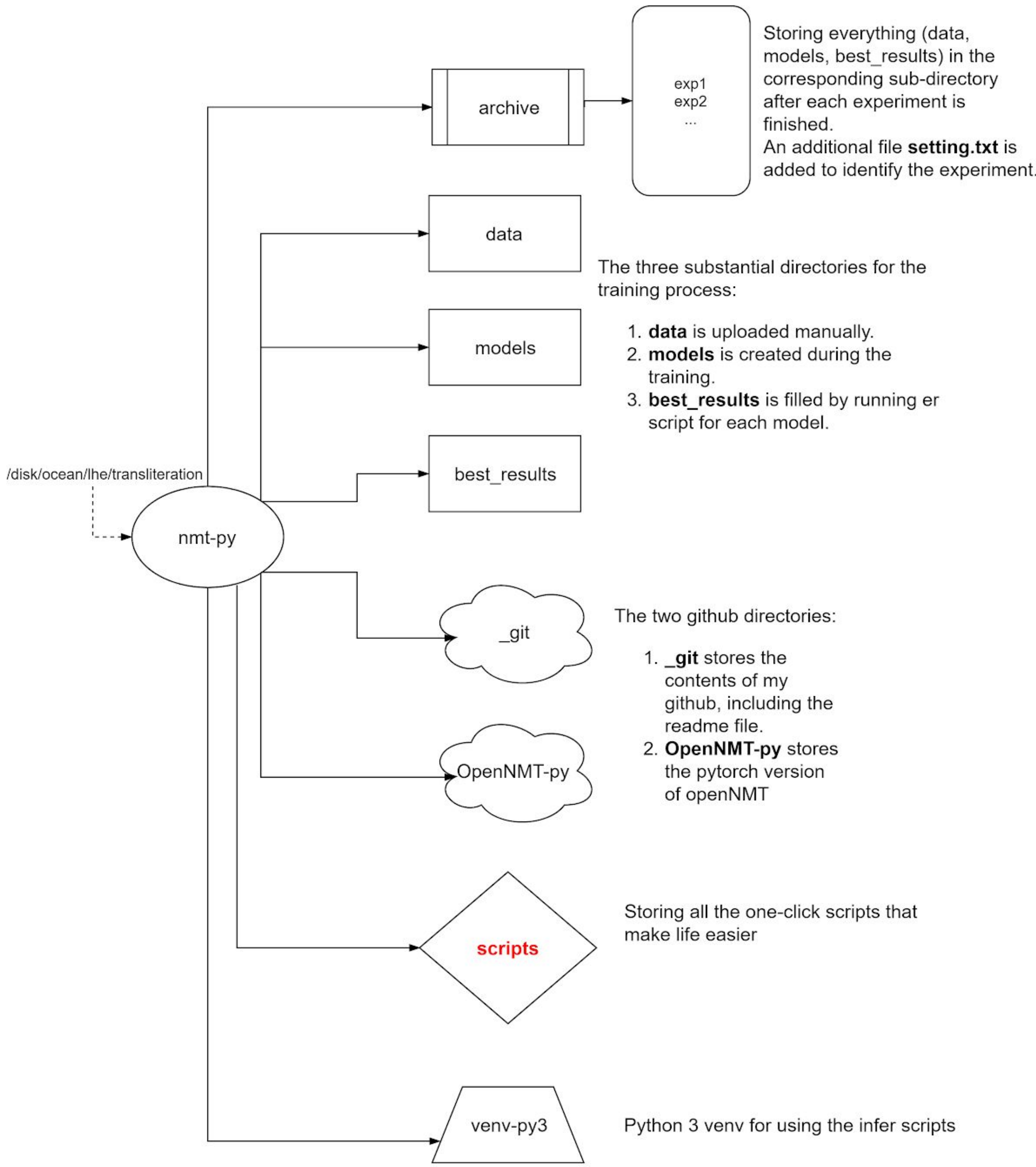
- Comparing baseline result and the best result (5cls +-):
  - The sequence “ch” could be transliterated into many ways in Chinese, such as ‘奇’ (sounds like ‘chee’), ‘赫’ (sounds like ‘her’), ‘克’ (sounds like ‘ker’) etc. Based on my observations, the machine does not work very well when it tries to predict ‘ch’ in different cases, and this is a general problem that happens in both results.
  - 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 ‘^’ and the end symbol ‘$’ should have a significant power because of the way we construct the feature vectors. (This may explain why generally the 2 clusters does not give better results—the ‘^’ and ‘$’ 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



Details of each independent sub-directories are written in GitHub readme.