Goal

Building a machine translation system that translates Modern Chinese into Classical Chinese.

Data

I use Sima Qian's *Records of the Grand Historian*, also known by its Chinese name *Shiji* (史記), except for the *Table* (表) volumes. The original text contains more than five hundred thousand Chinese characters.

The original text I use comes from several sources. I mainly use the text from Gutenberg Project, which comes as a nice single plain text file[1]. However, about 1,000 characters are corrupted, which are found with the following regular expression:

```
[^\u2E80-\u9FFF，。；：“”‘’《》！？、·「」『』…\r\n]
```

I recovered each corrupted character by cross-referencing Wikisource[2] and the Chinese Text Project[3]. The original text is then converted from traditional Chinese characters to simplified Chinese characters using ConvertZ[4].

The translation I use mainly comes from read126.cn[5]. (Obviously this is not originally where the translation was published, but I cannot find the original source.) There are also corrupted characters in translations, which are harder to deal with. Here's what I did:

1. Try to find the same translation from other sources by Googling the nearby uncorrupted sentence;
2. Try to find other translations by Googling the corresponding original text in classical Chinese;
3. If the corrupted character is a name of some person / location / plant /animal, try to recover by consulting the corresponding original sentence in classical Chinese;
4. Occasionally when all the above fail, try to fill in a translation (as a native speaker).

I also tried to correct the wrong punctuation by looking for consecutive ，。；：！？、• characters. For each case, I checked whether it indicated a missing sentence or was just a typo.

The data is divided into three parts: `train` (4192 paragraphs, 1.29MB), `dev` (524 paragraphs, 259KB) and `test` (524 paragraphs, 200KB) for model training, tuning and evaluation respectively.

Evaluation

I use the per-paragraph average character-level BLEU score as evaluation metric.

---

[1] http://www.gutenberg.org/files/24226/24226-0.txt

[2] http://zh.wikisource.org/wiki/史記

[3] http://ctext.org/shiji

[4] http://dl.pconline.com.cn/html_2/1/76/id=495&pn=0.html

[5] http://www.read126.cn/194c6894-51d5-4df3-a4bc-fa1282139f82!c0856342-2132-4498-921c-d81450904044!66ee2899-4ffa-41e9-ae09-126ca0281c65.html

I choose character-level BLEU rather than word-level BLEU for two reasons. Firstly, according to *Li et al. 2011*[6], character-level metrics correlate better with human assessment for evaluating Chinese translation. Secondly, the method and even the necessity of classical Chinese word segmentation is still an open question – I will come back to this later.

I manually performed paragraph alignment instead of using automatic sentence alignment algorithms, because the performance of such algorithms needs to be evaluated, and consequently the validity of our evaluation metric would become questionable – this is especially true in our case, and again I will come back to this later. The reason why I use BLEU score averaged over paragraphs rather than sentences is that I found the workload unacceptable having tried to manually align sentences of a volume.

In the calculation of BLEU, punctuation marks are treated indiscriminately, and are not included in n-grams.

1-Hour Baseline

Without a classical-modern Chinese dictionary, the best we can do in an hour is to replace/remove some most frequent words. I replaced 的 with 之, 说 with 曰, 到 with 至, 秦/楚/齐国 with 秦/楚/齐. I also removed the following words and characters: 了, 他们, 所以, 军队, 在, 他, 就, 是, 这. I also tried with other frequent words and characters but didn't work out well.

The performance of this simple baseline is as follows:

		dev	test
1-BLEU	overall	0.5080	0.5166
	per-paragraph	0.4221	0.4516
2-BLEU	overall	0.3331	0.3380
	per-paragraph	0.2701	0.3026
3-BLEU	overall	0.2081	0.2091
	per-paragraph	0.1742	0.2102
4-BLEU	overall	0.1324	0.1352
	per-paragraph	0.1072	0.1422

Method

I use Moses[7] as decoder of the system. The model training process mostly follows the "Baseline System" tutorial from Moses' website[8]. IRSTLM is used to build a trigram language model with improved Kneser-Ney smoothing. MERT is used for tuning.

However, there're three issues specific to our case, namely:

1. Sentence alignment of bi-text with ambiguous punctuation and weak correlation of sentence length

[6] Li, M., Zong, C., & Ng, H. T. (2011). Automatic evaluation of Chinese translation output: word-level or character-level?

[7] http://www.statmt.org/moses/index.php?n=Main.HomePage

[8] http://www.statmt.org/moses/?n=moses.baseline

2. Word segmentation (both modern and classical text)
3. During word alignment, prefer word pairs that share same characters

The problems are addressed as described in the following three sections.

Sentence Alignment

Although I avoided sentence alignment in evaluation, it is still unavoidable for training the model. I use hunalign[9] to perform sentence alignment.

There are two major challenges:

1. Translation from classical Chinese to modern Chinese is often not literal. Sometimes the classical text is paraphrased; sometime additional contextual information is filled in. As a result, the length of classical sentences and corresponding modern sentences often doesn't correlate very well.
2. The original text doesn't have punctuation marks. There are different versions of punctuation by modern scholars that sometimes don't agree with each other, and a translation may adopt any of them, or even have its own version.

This put us in a dilemma. If we split the text into sentences only by periods, a one-to-one alignment wouldn't exist for many sentences, as a result many sentences need to be concatenated into overly-long ones, which choke the follow-up word alignment algorithm. If we treat commas and periods indiscriminately, the correlation of sentence length becomes much weaker, which severely harms the performance of sentence alignment algorithm.

I use a two-pass approach to solve the problem. In both passes, to utilize the fact that corresponding sentences very often share same characters, all characters are treated as single tokens. In the first pass, the bi-text is split by periods only, and the sentence alignment algorithm groups them into chunk pairs with quite high quality. In the second path, each chunk is separated by both commas and periods, and (sub-)sentence alignment is performed again within each chunk. If there are still overly-long sentences, they are simply abandoned.

By doing this, fewer sentences are abandoned, and the resulting sentences are shorter, which is friendlier to the word alignment algorithm.

Word Segmentation

The challenge of word segmentation is two-fold. For modern text, we can use an off-the-shelf word segmenter (e.g. Stanford Word Segmenter[10]), but it does poorly when dealing with names of people, locations, ancient official positions, sacrificial vessels, legendary animals and plants, etc. For classical text, neither an off-the-self word segmenter nor any word-segmented text that can be used to train a segmenter is available.

I believe this problem can be solved by learning a vocabulary from the bi-text. The vocabulary can be used as part of the input data of a CRF-based modern Chinese segmenter like Stanford Word Segmenter to improve its performance. For classical Chinese, unlike modern Chinese, most words consist of only one character, thus ambiguity is not a big issue and simple forward maximum matching with the vocabulary would be reasonably good.

[9] http://mokk.bme.hu/resources/hunalign/

[10] http://nlp.stanford.edu/software/segmenter.shtml

I implemented a fairly simple unsupervised algorithm to extract vocabulary. In short, for each 2-gram and 3-gram that appears in both classical and modern texts, we compute its frequency, boundary entropy and mutual information, and if they reach a certain threshold, we put the n-gram into the vocabulary[11]. This gives us a vocabulary with a few thousand words. I then manually remove some of them and add some new ones.

Word Alignment

Before performing word alignment, sentences that have more than 25 words are abandoned. In addition, if a modern sentence is more than 3 times longer than its classical counterpart, it is also abandoned since this often suggests aggressive paraphrasing that doesn't generalized very well.

I implemented a modified IBM Model2 to perform word alignment: for a pair of words that share x common characters, an extra $0.9x$ pseudo-count is collected in the E-step.

Result

The performance of our final model is as follows:

		dev	test
1-BLEU	overall	0.7373	0.7206
	per-paragraph	0.5853	0.5636
2-BLEU	overall	0.5556	0.5213
	per-paragraph	0.4292	0.4051
3-BLEU	overall	0.3941	0.3505
	per-paragraph	0.3228	0.2997
4-BLEU	overall	0.2892	0.2459
	per-paragraph	0.2427	0.2182

As a comparison, if we use GIZA++ (with default parameters) instead of our modified IBM Model 2, the result is as follows:

		dev	test
1-BLEU	overall	0.7373	0.7163
	per-paragraph	0.5851	0.5585
2-BLEU	overall	0.5556	0.5187
	per-paragraph	0.4287	0.4006
3-BLEU	overall	0.3943	0.3498
	per-paragraph	0.3211	0.2958
4-BLEU	overall	0.2890	0.2458
	per-paragraph	0.2417	0.2129

Our model clearly out-performs the 1-hour baseline. Despite of being a simpler model, it also slightly out-performs GIZA++.

[11] I got the idea from here: http://www.matrix67.com/blog/archives/5044 , but similar techniques are used in several more formal papers, too.