Learn Languages First and Then Convert: Towards Effective Simplified to Traditional Chinese Conversion

Pranav A, S.F. Hui, I-Tsun Cheng, Ishaan Batra and Chiu Yik Hei

Dayta AI, Science Park, Hong Kong

Hong Kong University of Science and Technology, Hong Kong

Abstract

Simplified Chinese to Traditional Chinese conversion is a common preprocessing step in Chinese NLP. However, a simplified Chinese character could correspond to multiple traditional characters, and unfortunately, there is no accurate toolkit to disambiguate such mappings. We propose a sub-word segmentation model which relies on Simplified Chinese and Traditional Chinese language models and the character mapping table. Through these two language models, we effectively segment a sentence and use them to disambiguate between mappings. Our experiments show that we achieve the disambiguation accuracy of 98%.

1 Introduction

Traditional and Simplified are the two standardized character sets for written Chinese. Traditional Chinese is used throughout the history of ancient China and is currently still being used predominantly in Taiwan, Hong Kong, and Macau. Simplified Chinese was first introduced to mainland China in 1969 and has been officially used ever since due to its simplicity, which helps to encourage literacy. Traditional Chinese characters feature more complexity than Simplified Chinese characters. The traditional form of a character usually has more strokes and parts than its simplified counterpart. The differences between their forms call for the need of an accurate translator (Shi et al., 2011).

Converting from Traditional to Simplified Chinese is straightforward because there is a one-to-one or in some cases a many-to-one correspondence between the characters. A comprehensive mapping table like The *Table of General Standard Chinese Characters*[1] would be sufficient for this conversion.

[1] http://www.gov.cn/gzdt/att/att/site1/20130819/tygfhzb.pdf

However, conversion from Simplified to Traditional Chinese is a more difficult task as a Simplified character can be mapped to more than one Traditional character (see Table 1) depending on the context of the sentence. Around 12% of Simplified characters exist with one-to-many mappings to Traditional characters (Halpern and Kerman, 1999).

Simplified	台 (Stage, Table, Typhoon)			
Traditional	臺 (Stage)	台 (Stage, variant)	檯 (Table)	颱 (Typhoon)

Table 1: One to many mappings

A converter needs to consider the context to resolve this one-to-many character ambiguity. Currently, there is no such converter and toolkit which can disambiguate characters (section 4). On the other hand, in translation, sequence-to-sequence models are generally used (Sutskever et al., 2014). However, there are no parallel corpora available for converting the scripts. Also, seq2seq could reword the target sentence; this is undesirable since this is a character conversion problem, not a translation problem. Since conversion between the two scripts is typically a crucial step for pre-processing in Chinese NLP, there is a need for an accurate converter to perform this disambiguation.

We introduce an unsupervised translation approach which takes two features in consideration: context and correspondence. Firstly, we present our sub-word tokenization which jointly takes the context of sentence pair and mapping tables (section 2). After that, we use the language model for character disambiguation learned on the tokenized data (section 3). Our approach gives 99.6 % accuracy when compared to other character converters (section 5).

2 Segmentation for Conversion

A proper segmentation is the key for accurate conversion, especially in Chinese since it is not naturally tokenized. In this section, we present our conversion approach which is based on sub-word segmentation by considering language models and mapping table of source (Simplified Chinese) and target (Traditional Chinese) sentences.

Subwords are representations between words and characters (Mikolov et al., 2012). Subword segmentation has now become a widely used technique in machine translation (Kudo and Richardson, 2018). They handle OOV words and help in limiting the size of vocabulary (Wu and Zhao, 2018). Some of the notable methods for subword segmentation include BPE (Byte Pair Encoding) (Sennrich et al., 2015) and Unigram (Kudo, 2018).

BPE is a compression algorithm that combines frequent sequence of characters, which results in less frequent strings being segmented into subwords. Instead of relying on sub-word frequency, we made use of the mapping tables for subword segmentation, which could be thought of as "Dictionary-based BPE." Similar to BPE, the Dictionary-based BPE adapts a greedy longest-match-first approach in segmentation, where the individual tokens belong to the words in the mapping table.

Consider the sentence as an example "此貼文 在Facebook有15個 讚 好 。" (This post on Facebook has 15 likes). The words "此","貼文","在","有","個","讚好" are in the dictionary, so they will be tokenized as such. However, the words "facebook" and "15" are not in the mapping tables. Hence they will be tokenized into characters like "f|a|c|e|b|o|o|k" and "1|5". Therefore, the final tokenized sentence looks like: 此| 貼文| 在| f | a | c | e | b | o | o | k | 有| 1 | 5 | 個| 讚好.

The elements of the construction for this tokenizer are:

- **Mapping Table:** The mapping table is a dictionary that corresponds each Simplified Chinese token to a set of Traditional Chinese tokens. It is constructed using dictionaries from CC-CEDICT[2] and OpenCC[3].
- **Chunk based iteration:** This tokenizer traverses over the sentence and finds tokens starting from a chunk size of 8. If a chunk is found in the mapping table, it is formed as a token. Then it decreases the chunk size by 1 and captures the rest of the tokens. This process happens iteratively until the chunk size becomes 1 and all tokens are captured.

There are a few problems regarding sub-word tokenizers. Generally, they only take subword frequency as the feature. Dictionary-based BPE uses a mapping table instead of frequency. However, it could still lead to an undesirable segmentation due to the preference towards longer segments regardless of the context. Furthermore, sub-word tokenizers only take one end of the translation language into consideration for segmentation. We want to approach this research question: how to tokenize a sentence considering segments and the context of a source and target sentence pair?

Hence we propose a segmentation method which takes the source and target sentence pair jointly into consideration.

2.1 Theoretical Approach

A translator needs a source sentence $\mathbf{S}$ consisting of segments where $\mathbf{S} = s_0 s_1 \dots s_n$ and a target sentence $\mathbf{T}$ consisting of segments where $\mathbf{T} = t_0 t_1 \dots t_m$.

We want to find an optimally segmented sentence of $\mathbf{S}$ which is $\mathbf{S}^*$ and an optimally segmented sentence of $\mathbf{T}$ which is $\mathbf{T}^*$. Mathematically:

$$\mathbf{S}^*, \mathbf{T}^* = \underset{s_i \in \mathbf{S}, t_j \in \mathbf{T}}{\arg\max}\, P(\mathbf{S}, \mathbf{T}) \tag{1}$$

where $P(\mathbf{S}, \mathbf{T})$ is the joint probability of sentence pairs. Equation 1 can be rewritten as:

$$\mathbf{S}^* = \underset{s_i \in \mathbf{S}}{\arg\max} \prod_i P(s_i | \mathbf{S}, \mathbf{T}) \tag{2}$$

$$\mathbf{T}^* = \underset{t_j \in \mathbf{T}}{\arg\max} \prod_j P(t_j | \mathbf{S}, \mathbf{T}) \tag{3}$$

Looking further into equation 2:

$$P(s_i|\mathbf{S}, \mathbf{T}) \approx P(s_i|\mathbf{S}) P(s_i|\mathbf{T}) \tag{4}$$

$$\approx P(s_i|s_{i-1} \dots s_0) P(s_i|t_j t_{j-1} \dots t_0)$$

$$\approx P(s_i|s_{i-1} \dots s_0) P(s_i|t_j) P(t_j|t_{j-1} \dots t_0) \tag{5}$$

Intuition: In equation 4, we want to see how likely the segment s_i could belong to sentences $\mathbf{S}$ and $\mathbf{T}$. This could be approximated to the probability of $P(s_i|\mathbf{S})P(s_i|\mathbf{T})$. Here $P(s_i|\mathbf{S})$ is a language model score on sentence $\mathbf{S}$. The probability $P(s_i|t_j t_{j-1} \dots t_0)$ is approximated using

[2] https://cc-cedict.org/wiki/

[3] https://github.com/yichen0831/opencc-python/tree/master/opencc/dictionary

Markov assumption to $P(s_i|t_j)P(t_j|t_{j-1}\dots t_0)$. Here, $P(t_j|t_{j-1}\dots t_0)$ is language model score of sentence $\mathbf{T}$. Similarly, $P(t_j|\mathbf{S},\mathbf{T})$ can be calculated. $P(s_i|t_j)$ can be determined from the mapping table.

$$P(s_i|t_j) = \begin{cases} 1, & \text{if } s_i \rightarrow t_j \text{ in mapping table} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

If mapping t_j for s_i is found in the mapping table it is converted accordingly and is skipped otherwise. In a nutshell, our conversion uses the following elements:

1. language model score of source sentence of a candidate segment.
2. language model score of target sentence of a candidate segment.
3. mapping conversions from source segment to target segment.

2.2 Practical Approach

In this subsection, we describe the construction details of the elements mentioned in the previous subsection.

1. **Training Datasets for Segmentation:** We used SIGHAN-2005 Bakeoff dataset to train the segmentation-based language model (Emerson, 2005). For Simplified Chinese, we used PKU and MSR partitions, and for Traditional Chinese, we used Academia Sinica and CityU partitions.
2. **Language Model:** We used 5-gram Modified Kneser-Ney smoothing model for language modelling on this segmented dataset (James, 2000; Heafield, 2011).
3. **Segmentation:** We chose Viterbi for segmenting the given sentence (Luo and Roukos, 1996). The scoring function is obtained from equation 5 as mentioned earlier.
4. **Efficiency Heuristics:** For each segment in the training data, we applied Dictionary-based BPE to avoid any out of vocabulary words from mapping tables. To avoid OOVs as output segments, we imposed a penalty on OOV outputs, which is given by: $\alpha \times \frac{\texttt{len(segment)}}{\texttt{len(sentence)}}$. This is a length based penalty to avoid longer segments[4]. The value of α is generally near 15.0.

[4]Generally, we found the tokens were 1-4 characters long. Also, inclusion of penalty function also results in commonly occurring long segments, because we only impose penalty on OOV segments.

After this process, we obtain two corpora: segmented Traditional and Simplified Chinese datasets. Segmented Simplified Chinese dataset will be used for tokenizing the input and segmented Traditional Chinese dataset will be used for training language model for mapping disambiguation, which will be explained in next section.

3 Mappings Disambiguation

Once the sentences are tokenized, the mapping table is used to convert from Simplified Chinese to Traditional Chinese. For one-to-one mappings, we can convert them as it is. For one-to-many mappings, we will use language model to pick the best candidate.

Consider the following sentence in simplified Chinese: ”今天出论文了” (I published a paper today). The segmentation of the sentence is, 今天| 出| 论文| 了. The words 今天,出,论文 have one-to-one mapping corresponding to Traditional, which gives 今天,出,論文. However, the word 了 corresponds to two traditional tokens, which are 了 (auxiliary) and 瞭 (clear). Hence, we use language model to disambiguate between these two. The sentence 今天 出 論文 了 gives a log probability of -16.42 (based on modified Kneser-Ney smoothing language model) and the sentence 今天 出 論文 瞭 gives log probability of -19.58. Hence we pick the 今天 出 論文 了 as it has higher score and it’s the correct translation.

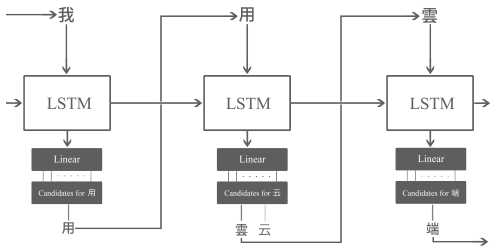

Figure 1: Consider this Simplified Chinese sentence 我用云端. The character 云 in Simplified Chinese corresponds to two characters in Traditional Chinese which are 云, 雲. The character with a higher score 雲 is predicted (shown in bold as the output of second cell) and passed on to the next cell to predict the next character.

Figure 1 explicates the process of disambiguation through neural language model. We use 2-layer LSTM (Long Short-term Memory (Sundermeyer et al., 2012)) with 512 cells which has been trained on the tokenized Traditional Chinese as mentioned in the previous subsection.

4 Baselines and Related Work for Converters

Translation models from Simplified Chinese to Traditional Chinese do exist; however, they are not perfect, and here we address their limitations that lower their accuracies in translation. Open-sourced translation models available are STCP, XMUCC, OpenCC, Hanziconv, and Mafan.

STCP (Xu et al., 2017) and XMUCC[5] use tokenizers Jieba and Moses respectively, which are Hidden Markov Models based tokenizers. These tokenizers are not reproducible because resulting tokens change for the same word in a different sentence. This leads to mismatch to the mapping table. Hence, tokens formed by them may be regarded as a new token, resulting in low probability. Also, STCP does character-level modelling which does not take word level in context or disambiguation. STCP claims that their materials are open-source, but we were not able to find any data and code that they used.

OpenCC[6], which uses character-level conversion, only takes the first candidate if there are multiple candidates available. Similarly for HanziConv[7] and Mafan[8], they use a simple character to character dictionary mapping, which is prone to translation errors in cases with multiple candidates.

5 Experiments and Results

To evaluate our converter, we need an evaluation corpus to compare the accuracy among different conversion systems. Due to the lack of high-quality parallel text corpus of Traditional and Simplified Chinese, we created corpus using book 1 of the novel *Twin of Brothers* written initially in Traditional Chinese and 150,696 articles scraped from multiple Hong Kong news sources.

To prepare our Simplified Chinese dataset of the corpus, we converted the corpus into Simplified Chinese using the OpenCC character conversion tool. It is assumed that this conversion performs with 100% accuracy since Traditional to Simplified mapping is one-to-one or many-to-one and hence every Traditional character can be mapped to a Simplified character with no errors.

[5]http://jf.xmu.edu.cn/

[6]https://github.com/yichen0831/opencc-python

[7]https://github.com/berniey/hanziconv

[8]https://github.com/hermanschaaf/mafan

Conversion System	Overall Accuracy	Micro-Average Accuracy
HanziConv	97.010%	80.102%
Mafan	99.126%	94.506%
OpenCC	99.257%	95.120%
STCP	98.533%	90.916%
This paper	99.625%	98.073%

Table 2: Conversion accuracies among different converters. We use overall accuracy and micro-average accuracy (disambiguation accuracy) as the evaluation metrics.

After creating Simplified Chinese dataset, we convert the dataset to Traditional Chinese using each conversion system and compare their outputs to the source in our corpus. We use overall accuracy and micro-average accuracy to evaluate the performances of the systems.

Overall accuracy is the ratio of correctly converted characters to number of characters, and micro-average accuracy is the accuracy for characters that have one-to-many mappings only.

6 Discussions

Our model attains 99.6 % overall accuracy and 98% disambiguation accuracy when compared to other converters (see table 2). We investigated why our model does not give perfect results, and we came up with two observations.

Firstly, the conversion errors may involve variant characters, which are sets of characters that have the same meaning and pronunciation but different scripts. It should be noted that they can be used interchangeably. Therefore mismatches, in this case, do not mean an incorrect conversion but rather an acceptable one. Fortunately, this is not a common phenomenon and does not affect the evaluation results substantially. Secondly, named entities could result in an incorrect conversion. It is difficult to disambiguate named entities such as names, locations, organizations as many are unique, so there are less training samples.

Simplified to Traditional scripts conversion is a common preprocessing step in Chinese NLP. We hope that our work would really help the Chinese NLP community in providing an open-source toolkit. For future work, we would like to extend our sub-word segmentation for other language pairs suited for neural machine translation.

References

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*.

Jack Halpern and Jouni Kerman. 1999. Pitfalls and complexities of chinese to chinese conversion. In *International Unicode Conference (14th) in Boston*.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*, pages 187–197. Association for Computational Linguistics.

Frankie James. 2000. Modified kneser-ney smoothing of n-gram models. *Research Institute for Advanced Computer Science, Tech. Rep. 00.07*.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Xiaoqiang Luo and Salim Roukos. 1996. An iterative algorithm to build chinese language models. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 139–143. Association for Computational Linguistics.

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. 2012. Subword language modeling with neural networks. *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)*, 8.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Xiaodong Shi, Yidong Chen, and Xiuping Huang. 2011. Key problems in conversion from simplified to traditional chinesecharacters. In *International Conference on Asian Language Processing*.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Yingting Wu and Hai Zhao. 2018. Finding better subword segmentation for neural machine translation. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 53–64. Springer.

Jiarui Xu, Xuezhe Ma, Chen-Tse Tsai, and Eduard Hovy. 2017. Stcp: Simplified-traditional chinese conversion and proofreading. *Proceedings of the IJCNLP 2017, System Demonstrations*, pages 61–64.