Joint Tokenization and Translation

Xinyan Xiao † **Yang Liu** † **Young-Sook Hwang** ‡ **Qun Liu** † **Shouxun Lin** †

†Key Lab. of Intelligent Info. Processing
Institute of Computing Technology
Chinese Academy of Sciences
{xiaoxinyan,yliu,liuqun,sxlin}@ict.ac.cn

‡HILab Convergence Technology Center
C&I Business
SKTelecom
yshwang@sktelecom.com

Abstract

As tokenization is usually ambiguous for many natural languages such as Chinese and Korean, tokenization errors might potentially introduce translation mistakes for translation systems that rely on 1-best tokenizations. While using lattices to offer more alternatives to translation systems have elegantly alleviated this problem, we take a further step to tokenize and translate jointly. Taking a sequence of atomic units that can be combined to form words in different ways as input, our joint decoder produces a tokenization on the source side and a translation on the target side simultaneously. By integrating tokenization and translation features in a discriminative framework, our joint decoder outperforms the baseline translation systems using 1-best tokenizations and lattices significantly on both Chinese-English and Korean-Chinese tasks. Interestingly, as a tokenizer, our joint decoder achieves significant improvements over monolingual Chinese tokenizers.

1 Introduction

Tokenization plays an important role in statistical machine translation (SMT) because tokenizing a source-language sentence is always the first step in SMT systems. Based on the type of input, Mi and Huang (2008) distinguish between two categories of SMT systems : *string-based* systems (Koehn et al., 2003; Chiang, 2007; Galley et al., 2006; Shen et al., 2008) that take a string as input and *tree-based* systems (Liu et al., 2006; Mi et al., 2008) that take a tree as input. Note that a tree-based system still needs to first tokenize the input sentence and then obtain a parse tree or forest of the sentence. As shown in Figure 1(a), we refer to this pipeline as **separate** tokenization and translation because they are divided into single steps.

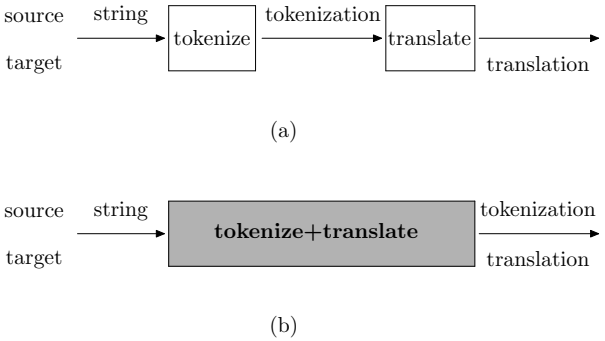

Figure 1: (a) Separate tokenization and translation and (b) joint tokenization and translation.

As tokenization for many languages is usually ambiguous, SMT systems that separate tokenization and translation suffer from a major drawback: tokenization errors potentially introduce translation mistakes. As some languages such as Chinese have no spaces in their writing systems, how to segment sentences into appropriate words has a direct impact on translation performance (Xu et al., 2005; Chang et al., 2008; Zhang et al., 2008). In addition, although agglutinative languages such as Korean incorporate spaces between "words", which consist of multiple morphemes, the granularity is too coarse and makes the training data

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1200–1208,
Beijing, August 2010

considerably sparse. Studies reveal that segmenting "words" into morphemes effectively improves translating morphologically rich languages (Oflazer, 2008). More importantly, a tokenization close to a gold standard does not necessarily leads to better translation quality (Chang et al., 2008; Zhang et al., 2008). Therefore, it is necessary to offer more tokenizations to SMT systems to alleviate the tokenization error propagation problem. Recently, many researchers have shown that replacing 1-best tokenizations with lattices improves translation performance significantly (Xu et al., 2005; Dyer et al., 2008; Dyer, 2009).

We take a next step towards the direction of offering more tokenizations to SMT systems by proposing **joint** tokenization and translation. As shown in Figure 1(b), our approach tokenizes and translates jointly to find a tokenization and a translation for a source-language string simultaneously. We integrate translation and tokenization models into a discriminative framework (Och and Ney, 2002), within which tokenization and translation models interact with each other. Experiments show that joint tokenization and translation outperforms its separate counterparts (1-best tokenizations and lattices) significantly on the NIST 2004 and 2005 Chinese-English test sets. Our joint decoder also reports positive results on Korean-Chinese translation. As a tokenizer, our joint decoder achieves significantly better tokenization accuracy than three monolingual Chinese tokenizers.

2 Separate Tokenization and Translation

Tokenization is to split a string of characters into meaningful elements, which are often referred to as words. Typically, machine translation separates tokenization from decoding as a preprocessing step. An input string is first preprocessed by a tokenizer, and then is translated based on the tokenized result. Take the SCFG-based model (Chiang, 2007) as an example. Given the character sequence of Figure 2(a), a tokenizer first splits it into the word sequence as shown in Figure 2(b), then the decoder translates the word sequence using the rules in Table 1.

This approach makes the translation process simple and efficient. However, it may not be

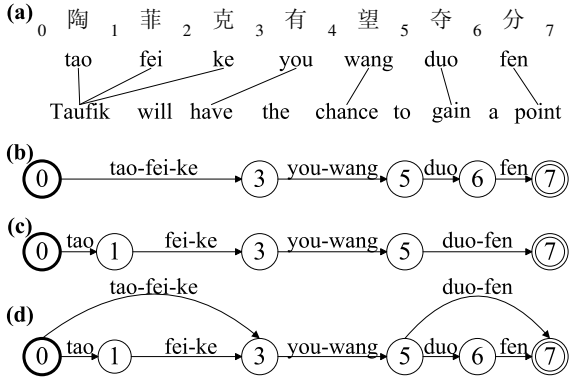

Figure 2: Chinese tokenization: (a) character sequence; (b) and (c) tokenization instances; (d) lattice created from (b) and (c). We insert "-" between characters in a word just for clarity.

r_1	*tao-fei-ke* $\rightarrow$*Taufik*
r_2	*duo fen* $\rightarrow$ *gain a point*
r_3	x_1 *you-wang* $x_2 \rightarrow x_1$ *will have the chance to* x_2

Table 1: An SCFG derivation given the tokenization of Figure 2(b).

optimal for machine translation. Firstly, optimal granularity is unclear for machine translation. We might face severe data sparseness problem by using large granularity, while losing much useful information with small one. Consider the example in Figure 2. It is reasonable to split *duo fen* into two words as *duo* and *fen*, since they have one-to-one alignments to the target side. Nevertheless, while *you* and *wang* also have one-to-one alignments, it is risky to segment them into two words. Because the decoder is prone to translate *wang* as a verb *look* without the context *you*. Secondly, there may be tokenization errors. In Figure2(c), *tao fei ke* is recognized as a Chinese person name with the second name *tao* and the first name *fei-ke*, but the whole string *tao fei ke* should be a name of the Indonesian badminton player.

Therefore, it is necessary to offer more tokenizations to SMT systems to alleviate the tokenization error propagation problem. Recently, many researchers have shown that replacing 1-best tokenizations with lattices improves translation performance significantly. In this approach, a lattice compactly encodes many tokenizations and is fixed before decoding.

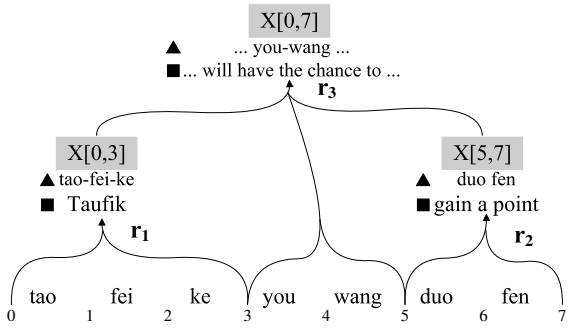

Figure 3: A derivation of the joint model for the tokenization in Figure 2(b) and the translation in Figure 2 by using the rules in Table 1. ▲ means tokenization while ■ represents translation.

3 Joint Tokenization and Translation

3.1 Model

We take a next step towards the direction of offering more tokenizations to SMT systems by proposing **joint** tokenization and translation. As shown in Figure 1(b), the decoder takes an untokenized string as input, and then tokenizes the source side string while building the corresponding translation of the target side. Since the traditional rules like those in Table 1 natively include tokenization information, we can directly apply them for simultaneous construction of tokenization and translation by the source side and target side of rules respectively. In Figure 3, our joint model takes the character sequence in Figure 2(a) as input, and synchronously conducts both translation and tokenization using the rules in Table 1.

As our model conducts tokenization during decoding, we can integrate tokenization models as features together with translation features under the discriminative framework. We expect tokenization and translation could collaborate with each other. Tokenization offers translation with good tokenized results, while translation helps tokenization to eliminate ambiguity. Formally, the probability of a derivation D is represented as

$$P(D) \propto \prod_i \phi_i(D)^{\lambda_i} \quad (1)$$

where ϕ_i are features defined on derivations including translation and tokenization, and λ_i are feature weights. We totally use 16 features:

- 8 traditional translation features (Chiang, 2007): 4 rule scores (direct and reverse translation scores; direct and reverse lexical translation scores); language model of the target side; 3 penalties for word count, extracted rule and glue rule.
- 8 tokenization features: maximum entropy model, language model and word count of the source side (Section 3.2). To handle the Out Of Vocabulary (OOV) problem (Section 3.3), we also introduce 5 OOV features: OOV character count and 4 OOV discount features.

Since our model is still a string-based model, the CKY algorithm and cube pruning are still applicable for our model to find the derivation with max score.

3.2 Adding Tokenization Features

Maximum Entropy model (ME). We first introduce ME model feature for tokenization by casting it as a labeling problem (Xue and Shen, 2003; Ng and Low, 2004). We label a character with the following 4 types:

- **b**: the **b**egin of a word
- **m**: the **m**iddle of a word
- **e**: the **e**nd of a word
- **s**: a **s**ingle-character word

Taking the tokenization *you-wang* of the string *you wang* for example, we first create a label sequence *b e* for the tokenization *you-wang* and then calculate the probability of tokenization by

$$\begin{aligned} &P(\textit{you-wang} \mid \textit{you wang}) \\ &= P(\mathrm{b\ e} \mid \textit{you wang}) \\ &= P(\mathrm{b} \mid \textit{you}, \textit{you wang}) \\ &\quad \times P(\mathrm{e} \mid \textit{wang}, \textit{you wang}) \end{aligned}$$

Given a tokenization w_1^L with L words for a character sequence c_1^n, we firstly create labels l_1^n for every characters and then calculate the probability by

$$P(w_1^L|c_1^n) = P(l_1^n|c_1^n) = \prod_{i=1}^{n} P(l_i|c_i, c_1^n) \quad (2)$$

Under the ME framework, the probability of assigning the character c with the label l is represented as:

$$P(l|c, c_1^n) = \frac{\exp[\sum_i \lambda_i h_i(l, c, c_1^n)]}{\sum_{l'} \exp[\sum_i \lambda_i h_i(l', c, c_1^n)]} \quad (3)$$

where h_i is feature function, λ_i is the feature weight of h_i. We use the feature templates the same as Jiang et al., (2008) to extract features for ME model. Since we directly construct tokenization when decoding, it is straight to calculate the ME model score of a tokenization according to formula (2) and (3).

Language Model (LM). We also use the n-gram language model to calculate the probability of a tokenization w_1^L:

$$P(w_1^L) = \prod_{i=1}^{L} P(w_i|w_{i-n+1}^{i-1}) \quad (4)$$

For instance, we compute the probability of the tokenization shown in Figure 2(b) under a 3-gram model by

$$\begin{aligned} & P(\textit{tao-fei-ke}) \\ \times & P(\textit{you-wang} \mid \textit{tao-fei-ke}) \\ \times & P(\textit{duo} \mid \textit{tao-fei-ke}, \textit{you-wang}) \\ \times & P(\textit{fen} \mid \textit{you-wang}, \textit{duo}) \end{aligned}$$

Word Count (WC). This feature counts the number of words in a tokenization. Language model is prone to assign higher probabilities to short sentences in a biased way. This feature can compensate this bias by encouraging long sentences. Furthermore, using this feature, we can optimize the granularity of tokenization for translation. If larger granularity is preferable for translation, then we can use this feature to punish the tokenization containing more words.

3.3 Considering All Tokenizations

Obviously, we can construct the potential tokenizations and translations by only using the extracted rules, in line with traditional translation decoding. However, it may limits the potential tokenization space. Consider a string *you wang*. If *you-wang* is not reachable by the extracted rules, the tokenization *you-wang* will never be considered under this way. However, the decoder may still create a derivation by splitting the string as small as possible with tokenization *you wang* and translating *you* with *a* and *wang* with *look*, which may hurt the translation performance. This case happens frequently for named entity especially. Overall, it is necessary to assure that the decoder can derive all potential tokenizations (Section 4.1.3).

To assure that, when a span is not tokenized into a single word by the extracted rules, we will add an operation, which is considering the entire span as an OOV. That is, we tokenize the entire span into a single word with a translation that is the copy of source side. We can define the set of all potential tokenizations $\tau(c_1^n)$ for the character sequence c_1^n in a recursive way by

$$\tau(c_1^n) = \bigcup_{i}^{n-1} \{\tau(c_1^i) \bigotimes \{w(c_{i+1}^n)\}\} \quad (5)$$

here $w(c_{i+1}^n)$ means a word contains characters c_{i+1}^n and $\bigotimes$ means the times of two sets. According to this recursive definition, it is easy to prove that all tokenizations is reachable by using the glue rule ($S \Rightarrow SX, SX$) and the added operation. Here, glue rule is used to concatenate the translation and tokenization of the two variables S and X, which acts the role of the operator $\bigotimes$ in equation (5).

Consequently, this introduces a large number of OOVs. In order to control the generation of OOVs, we introduce the following OOV features:

OOV Character Count (OCC). This feature counts the number of characters covered by OOV. We can control the number of OOV characters by this feature. It counts 3 when *tao-fei-ke* is an OOV, since *tao-fei-ke* has 3 characters.

OOV Discount (OD). The chances to be OOVs vary for words with different counts of characters. We can directly attack this problem by adding features OD_i that reward or punish OOV words which contains with i characters, or $OD_{i,j}$ for OOVs contains with i to j characters. 4 OD features are used in this paper: 1, 2, 3 and 4+. For example, OD_3 counts 1 when the word *tao-fei-ke* is an OOV.

Method	Train	#Rule	Test	TFs	MT04	MT05	Speed
Separate	ICT	151M	ICT	×	34.82	33.06	2.48
	SF	148M	SF	×	35.29	33.22	2.55
	ME	141M	ME	×	33.71	30.91	2.34
	All	219M	Lattice	×	35.79	33.95	3.83
				√	35.85	33.76	6.79
Joint	ICT	151M	Character	√	36.92	34.69	17.66
	SF	148M			37.02	34.56	17.37
	ME	141M			36.78	34.17	17.23
	All	219M			**37.25****	**34.88****	17.52

Table 2: Comparison of Separate and Joint methods in terms of BLEU and speed (second per sentence). Columns *Train* and *Test* represents the tokenization methods for training and testing respectively. Column *TFs* stands for whether the 8 tokenization features is used (√) or not (×). *ICT*, *SF* and *ME* are segmenter names for preprocessing. *All* means combined corpus processed by the three segmenters. Lattice represent the system implemented as Dyer et al., (2008). ** means significantly (Koehn, 2004) better than Lattice ($p < 0.01$).

4 Experiments

In this section, we try to answer the following questions:

1. Does the joint method outperform conventional methods that separate tokenization from decoding. (Section 4.1)

2. How about the tokenization performance of the joint decoder? (Section 4.2)

4.1 Translation Evaluation

We use the SCFG model (Chiang, 2007) for our experiments. We firstly work on the Chinese-English translation task. The bilingual training data contains 1.5M sentence pairs coming from LDC data.[1] The monolingual data for training English language model includes Xinhua portion of the GIGAWORD corpus, which contains 238M English words. We use the NIST evaluation sets of 2002 (MT02) as our development data set, and sets of 2004(MT04) and 2005(MT05) as test sets. We use the corpus derived from the People's Daily (Renmin Ribao) in Feb. to Jun. 1998 containing 6M words for training LM and ME tokenization models.

Translation Part. We used GIZA++ (Och and Ney, 2003) to perform word alignment in both directions, and grow-diag-final-and (Koehn et al., 2003) to generate symmetric word alignment. We extracted the SCFG rules as describing in Chiang (2007). The language model were trained by the SRILM toolkit (Stolcke, 2002).[2] Case insensitive NIST BLEU (Papineni et al., 2002) was used to measure translation performance.

Tokenization Part. We used the toolkit implemented by Zhang (2004) to train the ME model. Three Chinese word segmenters were used for comparing: ICTCLAS (*ICT*) developed by institute of Computing Technology Chinese Academy of Sciences (Zhang et al., 2003); *SF* developed at Stanford University (Huihsin et al., 2005) and *ME* which exploits the ME model described in section (3.2).

4.1.1 Joint Vs. Separate

We compared our joint tokenization and translation with the conventional separate methods. The input of separate tokenization and translation can either be a single segmentation or a lattice. The lattice combines the 1-best segmentations of segmenters. Same as Dyer et al., (2008), we also extracted rules from a combined bilingual corpus which contains three copies from different segmenters. We refer to this version of rules as *All*.

Table 2 shows the result.[3] Using *all* rule table, our joint method significantly outperforms the best single system *SF* by $+1.96$ and $+1.66$ points on MT04 and MT05 respectively, and also outperforms the lattice-based system by $+1.46$ and $+0.93$ points. However, the 8 tokenization features have small impact on the lattice system, probably because the tokenization space limited

[1] including LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06

[2] The calculation of LM probabilities for OOVs is done by the SRILM without special treatment by ourself.

[3] The weights are retrained for different test conditions, so do the experiments in other sections.

ME	LM	WC	OCC	OD	MT05
×	×	×	×	×	24.97
√	×	×	×	×	25.30
×	√	×	×	×	24.70
×	×	√	×	×	24.84
×	×	×	√	×	25.51
×	×	×	×	√	25.34
×	√	√	×	×	25.74
√	√	√	√	√	26.37

Table 3: Effect of tokenization features on Chinese-English translation task. "√" denotes using a tokenization feature while "×" denotes that it is inactive.

Method	BLEU	#Word	Grau	#OOV
ICT	33.06	30,602	1.65	644
SF	33.22	30,119	1.68	882
ME	30.91	29,717	1.70	1,614
Lattice	33.95	30,315	1.66	494
Joint$_{ICT}$	34.69	29,723	1.70	996
Joint$_{SF}$	34.56	29,839	1.69	972
Joint$_{ME}$	34.17	29,771	1.70	1,062
Joint$_{All}$	34.88	29,644	1.70	883

Table 4: Granularity (Grau, counts of character per word) and counts of OOV words of different methods on MT05. The subscript of joint means the type of rule table.

by lattice has been created from good tokenization. Not surprisingly, our decoding method is about 2.6 times slower than lattice method with tokenization features, since the joint decoder takes character sequences as input, which is about 1.7 times longer than the corresponding word sequences tokenized by segmenters. (Section 4.1.4).

The number of extracted rules with different segment methods are quite close, while the *All* version contains about 45% more rules than the single systems. With the same rule table, our joint method improves the performance over separate method up to $+3.03$ and $+3.26$ points (*ME*). Interestingly, comparing with the separate method, the tokenization of training data has smaller effect on joint method. The BLEU scores of MT04 and MT05 fluctuate about 0.5 and 0.7 points when applying the joint method, while the difference of separate method is up to 2 and 3 points respectively. It shows that the joint method is more robust to segmentation performance.

4.1.2 Effect of Tokenization Model

We also investigated the effect of tokenization features on translation. In order to reduce the time for tuning weights and decoding, we extracted rules from the FBIS part of the bilingual corpus, and trained a 4-gram English language model on the English side of FBIS.

Table 3 shows the result. Only using the 8 translation features, our system achieves a BLEU score of 24.97. By activating all tokenization features, the joint decoder obtains an absolute improvement by 1.4 BLEU points. When only adding one single tokenization feature, the *LM* and *WC* fail to show improvement, which may result from their bias to short or long tokenizations. However, these two features have complementary advantages and collaborate well when using them together (line 8). The OCC and OD features also contribute improvements which reflects the fact that handling the generation of OOV is important for the joint model.

4.1.3 Considering All Tokenizations?

In order to explain the necessary of considering all potential tokenizations, we compare the performances of whether to tokenize a span as a single word or not as illustrated in section 3.3. When only tokenizing by the extracted rules, we obtain 34.37 BLEU on MT05, which is about 0.5 points lower than considering all tokenizations shown in Table 2. This indicates that spuriously limitation of the tokenization space may degenerate translation performance.

4.1.4 Results Analysis

To better understand why the joint method can improve the translation quality, this section shows some details of the results on the MT05 data set.

Table 4 shows the granularity and OOV word counts of different configurations. The lattice method reduces the OOV words quite a lot which is 23% and 70% comparing with ICT and ME. In contrast, the joint method gain an absolute improvement even thought the OOV count do not decrease. It seems the lattice method prefers to translate more characters (since smaller granularity and less OOVs), while our method is inclined to maintain integrity of words (since larger granularity and more OOVs). This also explains the difficulty of deciding optimal tokenization for translation before decoding.

There are some named entities or idioms that

Method	Type	F_1	Time
Monolingual	ICT	97.47	0.010
	SF	97.48	0.007
	ME	95.53	0.008
Joint	ICT	97.68	9.382
	SF	97.68	10.454
	ME	97.60	10.451
	All	97.70	9.248

Table 5: Comparison of segmentation performance in terms of F_1 score and speed (second per sentence). *Type* column means the segmenter for monolingual method, while represents the rule tables used by joint method.

are split into smaller granularity by the segmenters. For example:"史 东" which is an English name "Stone" or "豆-蔻 年-华" which means "teenage". Although the separate method is possible to translate them using smaller granularity, the translation results are in fact wrong. In contrast, the joint method tokenizes them as entire OOV words, however, it may result a better translation for the whole sentence.

We also count the overlap of the segments used by the $Joint_{All}$ system towards the single segmentation systems. The tokenization result of $Joint_{All}$ contains $29,644$ words, and shares $28,159$, $27,772$ and $27,407$ words with ICT, SF and ME respectively. And 46 unique words appear only in the joint method, where most of them are named entity.

4.2 Chinese Word Segmentation Evaluation

We also test the tokenization performance of our model on Chinese word segmentation task. We randomly selected 3k sentences from the corpus of People's Daily in Jan. 1998. 1k sentences were used for tuning weights, while the other 2k sentences were for testing. We use MERT (Och, 2003) to tune the weights by minimizing the error measured by F_1 score.

As shown in Table 5, with all features activated, our joint decoder achieves an F_1 score of 97.70 which reduces the tokenization error comparing with the best single segmenter *ICT* by 8.7%. Similar to the translation performance evaluation, our joint decoder outperforms the best segmenter with any version of rule tables.

Feature	F_1
TFs	97.37
TFs + RS	97.65
TFs + LM	97.67
TFs + RS + LM	97.62
All	97.70

Table 6: Effect of the target side information on Chinese word segmentation. *TFs* stands for the 8 tokenization features. *All* represents all the 16 features.

4.2.1 Effect of Target Side Information

We compared the effect of the 4 Rule Scores (RS), target side Language Model (LM) on tokenization. Table 6 shows the effect on Chinese word segmentation. When only use tokenization features, our joint decoder achieves an F_1 score of 97.37. Only integrating language model or rule scores, the joint decoder achieves an absolute improvement of 0.3 point in F_1 score, which reduces the error rate by 11.4%. However, when combining them together, the F_1 score deduces slightly, which may result from the weight tuning. Using all feature, the performance comes to 97.70. Overall, our experiment shows that the target side information can improve the source side tokenization under a supervised way, and outperform state-of-the-art systems.

4.2.2 Best Tokenization = Best Translation?

Previous works (Zhang et al., 2008; Chang et al., 2008) have shown that preprocessing the input string for decoder by better segmenters do not always improve the translation quality, we reverify this by testing whether the joint decoder produces good tokenization and good translation at the same time. To answer the question, we used the feature weights optimized by maximizing BLEU for tokenization and used the weights optimized by maximizing F_1 for translation. We test BLEU on MT05 and F_1 score on the test data used in segmentation evaluation experiments. By tuning weights regarding to BLEU (the configuration for Joint_{All} in table 2), our decoder achieves a BLEU score of 34.88 and an F_1 score of 92.49. Similarly, maximizing F_1 (the configuration for the last line in table 6) leads to a much lower BLEU of 27.43, although the F_1 is up to 97.70. This suggests that better tokenization may not always lead to better translations and vice versa

Rule	#Rule	Method	Test	Time
Morph	46M	Separate	21.61	4.12
Refined	55M		21.21	4.63
All	74M	Joint	21.93*	5.10

Table 7: Comparison of Separate and Joint method in terms of BLEU score and decoding speed (second per sentence) on Korean-Chinese translation task.

even by the joint decoding. This also indicates the hard of artificially defining the best tokenization for translation.

4.3 Korean-Chinese Translation

We also test our model on a quite different task: Korean-Chinese. Korean is an agglutinative language, which comes from different language family comparing with Chinese.

We used a newswire corpus containing 256k sentence pairs as training data. The development and test data set contain 1K sentence each with one single reference. We used the target side of training set for language model training. The Korean part of these data were tokenized into morpheme sequence as atomic unit for our experiments.

We compared three methods. First is directly use morpheme sequence (Morph). The second one is refined data (Refined), where we use selective morphological segmentation (Oflazer, 2008) for combining morpheme together on the training data. Since the selective method needs alignment information which is unavailable in the decoding, the test data is still of morpheme sequence. These two methods still used traditional decoding method. The third one extracting rules from combined (All) data of methods 1 and 2, and using joint decoder to exploit the different granularity of rules.

Table 7 shows the result. Since there is no gold standard data for tokenization, we do not use ME and LM tokenization features here. However, our joint method can still significantly ($p < 0.05$) improve the performance by about +0.3 points. This also reflects the importance of optimizing granularity for morphological complex languages.

5 Related Work

Methods have been proposed to optimize tokenization for word alignment. For example, word alignment can be simplified by packing (Ma et al., 2007) several consecutive words together. Word alignment and tokenization can also be optimized by maximizing the likelihood of bilingual corpus (Chung and Gildea, 2009; Xu et al., 2008). In fact, these work are orthogonal to our joint method, since they focus on training step while we are concerned of decoding. We believe we can further the performance by combining these two kinds of work.

Our work also has connections to multilingual tokenization (Snyder and Barzilay, 2008). While they have verified that tokenization can be improved by multilingual learning, our work shows that we can also improve tokenization by collaborating with translation task in a supervised way.

More recently, Liu and Liu (2010) also shows the effect of joint method. They integrate parsing and translation into a single step and improve the performance of translation significantly.

6 Conclusion

We have presented a novel method for joint tokenization and translation which directly combines the tokenization model into the decoding phase. Allowing tokenization and translation to collaborate with each other, tokenization can be optimized for translation, while translation also makes contribution to tokenization performance under a supervised way. We believe that our approach can be applied to other string-based model such as phrase-based model (Koehn et al., 2003), string-to-tree model (Galley et al., 2006) and string-to-dependency model (Shen et al., 2008).

Acknowledgement

The authors were supported by SK Telecom C&I Business, and National Natural Science Foundation of China, Contracts 60736014 and 60903138. We thank the anonymous reviewers for their insightful comments. We are also grateful to Wenbin Jiang, Zhiyang Wang and Zongcheng Ji for their helpful feedback.

References

Chang, Pi-Chuan, Michel Galley, and Christopher D. Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *the Third Workshop on SMT*.

Chiang, David. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Chung, Tagyoung and Daniel Gildea. 2009. Unsupervised tokenization for machine translation. In *Proc. EMNLP 2009*.

Dyer, Christopher, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proc. ACL 2008*.

Dyer, Chris. 2009. Using a maximum entropy model to build segmentation lattices for mt. In *Proc. NAACL 2009*.

Galley, Michel, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. ACL 2006*.

Huihsin, Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter. In *Fourth SIGHAN Workshop*.

Jiang, Wenbin, Liang Huang, Qun Liu, and Yajuan Lü. 2008. A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *Proc. ACL 2008*.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. HLT-NAACL 2003*.

Koehn, Philipp. 2004. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP 2004*.

Liu, Yang and Qun Liu. 2010. Joint parsing and translation. In *Proc. Coling 2010*.

Liu, Yang, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. ACL 2006*.

Ma, Yanjun, Nicolas Stroppa, and Andy Way. 2007. Bootstrapping word alignment via word packing. In *Proc. ACL 2007*.

Mi, Haitao, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proc. of ACL 2008*.

Ng, Hwee Tou and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In *Proc. EMNLP 2004*.

Och, Franz J. and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. ACL 2002*.

Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Och, Franz Josef. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL 2003*.

Oflazer, Kemal. 2008. Statistical machine translation into a morphologically complex language. In *Proc. CICL 2008*.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. ACL 2002*.

Shen, Libin, Xu Jinxi, and Weischedel Ralph. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. ACL 2008*.

Snyder, Benjamin and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *Proc. ACL 2008*.

Stolcke, Andreas. 2002. Srilm – an extensible language modeling toolkit.

Xu, Jia, Evgeny Matusov, Richard Zens, and Hermann Ney. 2005. Integrated chinese word segmentation in statistical machine translation. In *Proc. IWSLT2005*.

Xu, Jia, Jianfeng Gao, Kristina Toutanova, and Hermann Ney. 2008. Bayesian semi-supervised chinese word segmentation for statistical machine translation. In *Proc. Coling 2008*.

Xue, Nianwen and Libin Shen. 2003. Chinese word segmentation as LMR tagging. In *SIGHAN Workshop*.

Zhang, Hua-Ping, Hong-Kui Yu, De-Yi Xiong, and Qun Liu. 2003. Hhmm-based chinese lexical analyzer ictclas. In *the Second SIGHAN Workshop*.

Zhang, Ruiqiang, Keiji Yasuda, and Eiichiro Sumita. 2008. Improved statistical machine translation by multiple Chinese word segmentation. In *the Third Workshop on SMT*.

Zhang, Le. 2004. Maximum entropy modeling toolkit for python and c++.