A Hybrid Sentence Splitting Method by Comma Insertion for Machine Translation with CRF

Yang Shuli, Feng Chong, and Huang Heyan

Beijing Institute of Technology
Beijing Engineering Research Center of High Volume
Language Information Processing and Cloud Computing Application
{ysl2007,fengchong,hhy63}@bit.edu.cn

Abstract. When writing formal articles many English writers often use long sentences with few punctuation marks. Since long sentences bring difficulty to machine translation systems, many researchers try to split them using punctuation marks before translation. But dealing with sentences with few punctuation marks is still intractable. In this paper we use a log linear model to insert commas into proper positions to split long sentence, trying to shorten the length of sub-sentence and benefit to machine translation. Experiment results show that our method can reasonably segment long sentences, and improve the quality of machine translation.

1 Introduction

In writing English, especially in formal articles, authors often use long sentences with very complicated syntactic structure to enhance the coherence and fluency. But as the length of a sentence increases, the quality of machine translation fails dramatically. For example:

Source: Reconstruction and repair have been put on hold in some instances due to workers' fears that the spirits of the dead who passed away a year ago will bring them bad luck if they continue.

Reference: 某些重建和修复工作被暂时搁置，因为工人害怕如果继续下去，一年前的亡灵会给他们带来厄运。

Translation: 重建和修理已经被在一些实例里推迟进行，由于工人的恐惧即一年前消失的死者的情绪将给他们带来坏运如果他们继续。

As the example shows, the translation of the source sentence is quite messy because of the long length and complicated structure. The rule-based systems have difficulties in covering all possible language phenomena; the statistical systems often generate syntactically strange outputs.

Thus, lots of methods on sentence splitting have been explored. But we notice that most works on sentence segmentation try to use the punctuation marks that already exist in the sentences, so if the sentence to be split has few punctuation

marks, these methods won't get improved results at all. In this paper we use a log-linear model together with a rule-based method to automatically insert commas into a sentence, in order to recover the omitted commas, reduce the length of each sub-sentence and improve the machine translation result. We tested our method on both rule-based and statistical machine translation systems. The results show that our method is quite helpful for improving the quality of translation.

2 Related Work

Lots of works have been done trying to split or simplify the sentences before translation or conducting other NLP tasks. Some works are rule-based. [2] proposed a multi-strategy method to analyze long sentences. The algorithm is mainly based on pattern match combined with rule analysis. [12] proposed a top-down analysis method to split long sentences. The author used regular expressions to match several sentence structures, and analyzed and split the matched sentences by rules. Despite the efficiency of these rule-based methods, the main problem of them is that the handcrafted rules cannot cover enough language phenomena, and cannot process sentences that contain more complicated components such as multiple nested clauses.

Some other works try to mine the information in the punctuation marks or conjunctions. [13] tried to use punctuation marks and conjunctions such as "and" and "or" to find split points in Chinese long sentences, and dealt with the two situations individually. [11] believed that in Chinese some commas could be "replaced" by periods because the contents on both sides of the comma did not have strong connection, so these commas could be treated as splitting position. They used a max-entropy classifier to find these commas. [10] also believed that some commas could be used to split Chinese long sentences and the authors used an algorithm based on syntactic parsing to find these commas. But these methods can only tackle with the commas already exist in the sentence, and cannot deal with more complicated syntactic structures.

We also notice that sentence splitting for English-Chinese machine translation is less researched in recent years.

In this paper we try to split sentences by finding proper positions and adding commas in them. A hybrid method is proposed considering that the rule-based algorithm can deal with simple situations, and the machine learning method can be better at covering more language phenomena. In this way we effectively shorten the length of sub-sentences and improve the quality of machine translation while keeping the original meaning of sentences.

3 Analysis on English Long Sentence

3.1 Standard of Long Sentence

In this paper we mainly deal with English long sentences with few punctuation marks. In fact there is no clear definition of long sentence. To decide the scope

of the length of "long sentence", we extracted four groups of bilingual texts by different length(word count) from the news part of UM-Corpus[8], which is an English-Chinese sentence-aligned corpus. The four groups of sentences cover different sentence length and each group contains 2,000 sentences. We evaluated the BLEU score to analyze the influence of sentence length. The detailed information and results are listed in Table 1.

Table 1. Machine translation results on different groups by sentence length.

Group	Sentence length	BLEU
1	0 - 10	17.68
2	10 - 20	17.12
3	20 - 30	15.85
4	>30	15.11

From the table it can be clearly seen that when the sentence length is over 20 words, the BLEU score obviously drops. Also, some experts on linguistics also claims that 20 words per sentence is a desirable average length of a sentence[5]. So This paper defines "long sentences" as the sentences contain more than 20 words, and all the long sentence data sets used in this paper are extracted and built by this standard.

3.2 Difference in Processing Long Sentence in English and Chinese

In Chinese long sentences are usually made up of several short and comma-delimited sub-sentences, so we can easily dig information from these commas for splitting. In English, many writers often use long sentences especially in formal materials, and even worse they omit commas in some situations where there could have been one. In the example below, "Source" is the source sentence and the bracketed commas are the suggested splitting place where can we insert commas. "Trans-1" and "Trans-2" are respectively the machine translations before and after splitting.

Source: Colleges and universities are a community (,) and everyone within that community needs to be treated with dignity (,) and that means paying them a wage they can live on for their work.

reference: 高校是一个社区，社区内的每个人都需要有尊严的待遇，这意味着给他们发放的工资必须能让他们维持基本生活。

Trans-1: 大专院校一社区和每个人在那社区内需要用尊严处理并且那的意思是付款给他们能适合他们工作继续活着的一工资他们。

Trans-2: 大专院校是一个社区，并且在那个社区内的每个人需要被用尊严处理，并且那表明付款给他们他们能为他们的工作生活的一份工资。

As we can see, the source sentence has only one sub-sentence, while the corresponding Chinese translation is made up of three sub-sentences separated by commas. In fact the source sentence is concatenated by three simple sentence, but three simple sentences adding together will bring trouble to MT system. If we can find out the splitting points and divide the sentence into three, the source sentence will be closer to the target reference and achieve better translation quality.

This phenomenon of English brings two problems. First, it confuses the syntactic parser and leads to wrong parsing results, which is of great importance to RBMT systems. Second, it deviates the expressing pattern of the sentence from the target language. Previous study has found that when syntactic parsing fails to work, MT systems, especially RBMT systems, tend to conduct word-by-word translation[6]. At this time the similarity between the source and target sentence will greatly affect the translation results[7]. So we consider if we can add commas at positions where there could have been one, we can make the English sentence closer to the target language, and shorten the length of each sub-sentence to make parsing easier.

4 The Hybrid Model of Comma Insertion

4.1 Strategy of Choosing Splitting Point

The procedure of our model is shown in Fig 1. The two methods works in parallel. A rule-based algorithm is designed to deal with several specific kinds of phrases or clauses while CRF is used to cover more complicated situations.

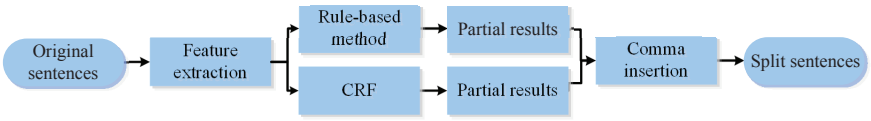

Fig. 1. The overview of our hybrid model.

4.2 Pattern Match Based on Dependency Structure

After analyzing many long sentences we conclude some situations that are easy to recognized by rules-based algorithm. We list them in Table 2. As is shown, sentences that contain adverbial clauses or prepositional phrases are very common, and sometimes these constituents can be very long. We find these sentences are easy to find out with dependency structure.

These constituents can be easily recognized by dependency structure. First we try to find the head word and then locate the boundary. The head words

Table 2. Splitting places easily recognized by rules.

Type	Example
Prepositional phrases	According to the most up to date marriage data from the Office for National Statistics (,) the number of marriages in the winter months has increased by around seven percent in two years.
Adverbial clauses	When Bradley Wiggins goes for gold tomorrow afternoon in the men 's cycling time trial (,) the Tour de France champion could be forgiven for checking the crowd nervously for the face of the prime minister , who is starting to get a reputation as a bit of a jinx .

of prepositional phrases can be easily found by dependency label "prep"; the leading word of an adverbial clauses is always labeled "advmob" and the word depended by the leading word should be labeled "advcl". After the head word is located, an algorithm is designed to find the boundary. The dependency structure of these components have a common characteristic which is the core of our algorithm: all words except the head word depend on words that are in the component. Fig 2 shows an simple example, the whole prepositional phrase can be easily recognized.

Algorithm 1: our rule-based algorithm

```
program findSplitPoint:
if advcl:
  leftList = [headWord, headWord.dep]
if prep:
  leftList = [headWord]
rightList = []
for word in sentence:
  if word.dep in leftList:
    leftList.add(word)
    do
      for word in rightList:
        if word.dep in leftList:
          leftList.add(word)
    while (rightList changed)
  else:
    rightList.add(word)
splitWord = maxIndex(leftWord)
```

Algorithm 1 is the python-like pseudo code of our rule-based splitting algorithm. In the pseudo code `headWord` represents the leading word of phrase or clause; `word.dep` represents the dependent of the governor, i.e., the current word. The algorithm maintains two lists to find the splitting point. the `leftList`

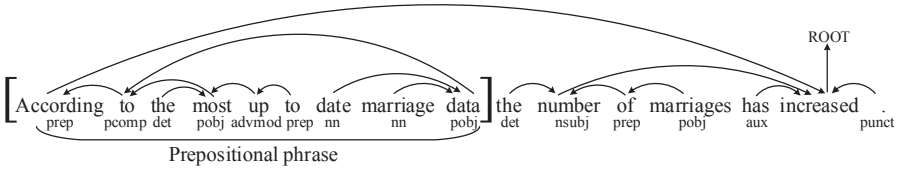

Fig. 2. The dependency structure of a prepositional phrase. All words except the head word depend on words that are in the prepositional phrase.

contains words within the phrase or clause, and `rightList` contains words in other places. The splitting points are before and after the phrase or clause. Note that we only segment prepositional phrases longer than five words.

4.3 Comma Insertion Using CRF

Table 3 lists other complex situations which simple handcrafted rules cannot deal with, so we use CRF to solve them.

Table 3. Other complex situations for splitting.

Type	Example
Adjective and object clauses	The son of wartime emperor Hirohito said (,) he had been in a good health in a year (,) in which he marked 20 years on the Chrysanthemum Throne and 50 years of marriage to Empress Michiko.
Coordinative components	Colleges and universities are a community (,) and everyone within that community needs to be treated with dignity (,) and that means paying them a wage they can live on for their work.

Since what we need to do is to find the proper place to insert a comma, we cast this task into a sequence labeling problem by tagging the word before insertion place with a label of "COM" and other words "NUL". Sequence labelling tasks can be solved by conditional random field[4], which tries to find the global best solution and avoids the labeling bias problem.

Given a word sequence $w_1, w_2, ..., w_L$, the CRF tries to find a best label sequence $y_1, y_2, ..., y_L$ to make probability $p(y_1 y_2 ... y_L | w_1 w_2 ... w_L)$ maximal. According to the theory of CRF, it can be formulated and finally represented as:

$$Y^* = \underset{y_1 y_2 \cdots y_L}{\operatorname{argmax}}\, p(y_1 y_2 ... y_L | w_1 w_2 ... w_L)$$

Using CRF needs careful consideration about feature selection. Since many factors might take part in finding the split points in a long sentence, we decide to utilize the following features.

– Words in sentence. We use this feature to recognize some fixed collocations.
– Dependency structure. This is the main feature we use. Dependency structure not only shows the constituent of each word in a sentence, but also conveys the dependent relationship between words.
– POS tag of words. POS tag can recognize the different usage of a word in different part-of-speech.

Besides the syntactic labels and POS tags of individual words we also need context information to find the proper positions for splitting. So we add a lot of context features on the basis of the above single features. Table 3 shows the actual feature template we use. The feature template seems very complicated. In fact we tested different number of context features and the template we present here is actually the best.

Table 4. Feature template we use.

ID	Feature	Comments
1 - 5	$w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$	Words in sentence
6 - 9	$w_{i-2}\&w_{i-1}, w_{i-1}\&w_i,$ $w_i\&w_{i+1}, w_{i+1}\&w_{i+2}$	Context of words
10 - 14	$p_{i-2}, p_{i-1}, p_i, p_{i+1}, p_{i+2}$	POS tags
15 - 18	$p_{i-2}\&p_{i-1}, p_{i-1}\&p_i,$ $p_i\&p_{i+1}, p_{i+1}\&p_{i+2}$	Context of POS tag
19 - 21	$p_{i-2}\&p_{i-1}\&p_i,$ $p_{i-1}\&p_i\&p_{i+1},$ $p_i\&p_{i+1}\&p_{i+2}$	Context of POS tag
22 - 30	$d_{i-4}, ..., d_i, ..., d_{i+4}$	Dependency tag
31 - 38	$d_{i-4}\&d_{i-3}, d_{i-3}\&d_{i-2},$ ..., $d_{i+2}\&d_{i+3}, d_{i+3}\&d_{i+4}$	Context of dependency tag
39 - 43	$d_{i-3}\&d_{i-2}\&d_{i-1},$ ..., $d_{i+1}\&d_{i+2}\&d_{i+3}$	Context of dependency tag

5 Experiments

5.1 Data Sets

Training Corpus The WMT workshop released large quantity of free corpora for researchers to train MT systems. We used "News Crawl (articles from 2007)" part of "monolingual language model training data" as our training data. This is a large and high-quality monolingual English news corpora for training language models, which can be downloaded freely on the WMT2013's web site[1].

[1] `http://www.statmt.org/wmt13/translation-task.html`

We extracted about 450,000 long sentences and did preprocessing on them, such as removing the news source header and deleting special marks. Then we used the syntactic parser[1] and POS tagger[9] from Stanford University to conduct typed dependency parsing and POS tagging.

Test Corpus To evaluate the effectiveness and rationality of comma insertion method we extracted 2,000 sentences which have at least two commas from the WMT corpora as Testset 1. Note that we didn't consider the length and these sentences are not included in the training data.

To evaluate the influence of our method on machine translation, a sentence-aligned bilingual corpora provided by HuaJian company was used. We extracted 2,027 long sentences from the corpora as Testset 2 to do our experiments, which has no more than one comma and are longer than 20 words.

Table 5 shows the detailed description of our data sets. The column of "Long sentence" stands for whether the data set is a corpus of long sentence.

Table 5. Detailed description of data sets.

Data set	Size	Long sentence	Comma limitation
Training set	About 450,000	Yes	At least 1
Testset 1	2000	Not limited	At least 2
Testset 2	2027	Yes	At most 1

5.2 Experiments on Effectiveness of Comma Insertion

Experiment Setup First we tested the effectiveness and rationality of our method. For effectiveness we deleted all the commas in Testset 1, hoping that the method could recover the original commas as many as possible. For rationality we mean that the places which the algorithm find to insert commas are all reasonable, i. e., not to split the sentences at wrong places.

We first conducted feature extraction and then deleted all the commas in Testset 1. Both methods of CRF and CRF+rule were used to find where the original commas were. Precision, recall and the number of recovered commas were calculated and size of training corpus were considered. We gradually enlarged the size of the training corpus. The results are shown in Table 6 and Fig 3 respectively. More concretely we listed several examples in Table 7.

Results From Table 6 we see that the ability of recovering commas of our method is quite promising. the precision of CRF only reaches 77.2% and the recall reaches 85.7%. By adding rule-based method we gain the recall by 0.5% at the cost of 0.1% precision, and can also add more commas.

We notice that the recall is much larger than the precision, which accords with our expectation. we expected that the positions found by the algorithm

Table 6. Effectiveness experiment results on Testset 1.

	Precision	Recall	Count	Count of insert
CRF only	77.2	85.7	3850	154
CRF + rule	77.0	86.2	3910	214

should cover most of the original ones which we deleted from the sentences, thus the recall would be relatively high while the precision wouldn't. The results proved our asumption.

Besides, this experiment is designed to test the ability to find the correct comma positions of our method. When building the data set we did'nt consider the sentence length, which resulted that many short sentences presented in the data set. There was not much room for new positions for splitting. As a result, the number of newly added commas is quite small.

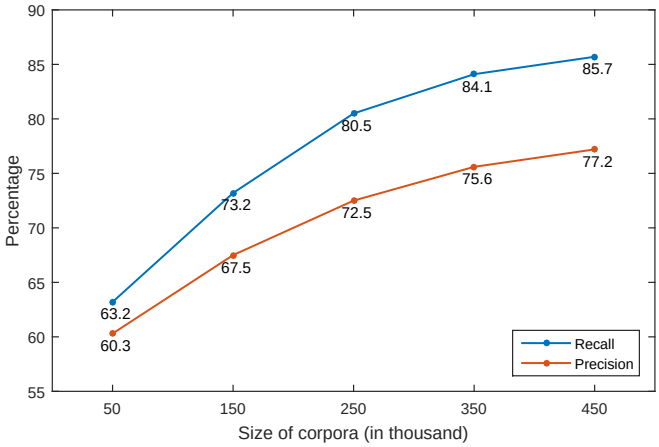

Fig. 3. Influence of training corpora size.

From Fig 3 we can see that the precision and recall all increases as the size of corpus enlarges. Because the monolingual long sentences are very easy to obtain, we deliberately chose larger corpus to train the CRF model. Of course larger corpus means larger memory to consume and longer time to run CRF, so we have to trade off between performance and resource.

5.3 Experiments on Translation Results

Experiment Setup Both RBMT and SMT systems were used to test whether our method was helpful for machine translation. We utilized HJTrans and Moses[3]

Table 7. Examples of Testset 1. The bracketed commas are recovered from the original sentence, and the underlined commas are inserted by the algorithm.

Before splitting	After splitting
The team had not won since their last trip to suburban Washington , starting the season 0-2 and having lost 9 of 11 games , dating to November .	The team had not won since their last trip to suburban Washington (,) starting the season 0-2 , and having lost 9 of 11 games (,) dating to November .
By law , Cubans can not sell their homes and because the state controls almost all property moves must be approved .	By law (,) Cubans can not sell their homes , and because the state controls almost all property , moves must be approved .
The fires have torched 1,790 homes but more than a dozen had been surrounded and nine others were 40 to 97 percent contained .	The fires have torched 1,790 homes , but more than a dozen had been surrounded , and nine others were 40 to 97 percent contained .

to conduct this part of experiment. HJTrans is a mature rule-based machine translation system with high quality and good robustness; Moses is a widely used statistical machine translation system. Both are very suitable for our experiment. The Moses system were trained on a bilingual corpus contains about 200,000 sentences.

We also set up a baseline system which is described in [10] to segment Testset 2, and used HJTrans to translate the segmented sentences. Since we used a web interface of HJTrans and couldn't generate n-best translation lists, we didn't follow the original paper's "sub translation combining" procedure. We just translated each segment, and concatenated the translated segments in the order of source sentence.

For our own method, we conducted feature extraction on Testset 2, used CRF to decode the features, and added commas after words that were labeled as "COM", as we designed in the previous chapter. Then we translated the segmented sentences by HJTrans, and used test script `mteval13a.pl` released by NIST to score the translation results.

Results Table 8 shows the results of translation experiments. The table shows the number of split sentences, number of total commas, and BLEU and NIST metric scores of Moses and HJTrans respectively. The baseline method doesn't add new commas into sentences so the number of commas is the same as the original corpus.

From the table we see that the CRF-only method segmented more than 1,400 sentences, and combining the two methods we could split more sentences in the test set. The baseline system did not achieve good performance, we think that it is because the sentences we chose to build Testset 2 are all too long and have too few punctuation marks. Also, the baseline method is mainly designed to process

Table 8. Results of translation experiments.

		#of split	#of commas	BLEU	NIST
Moses	Before split	0	1384	10.61	3.6431
	Baseline	577	1384	10.65	3.6642
	Rule	1198	2971	10.62	3.6873
	CRF	1481	3463	10.72	3.6971
	CRF+Rule	1575	3621	**10.81**	**3.7023**
HJTrans	Before split	0	1384	18.08	6.1152
	Baseline	577	1384	18.20	6.1384
	Rule	1198	2971	18.29	6.1462
	CRF	1481	3463	18.52	6.2820
	CRF+Rule	1575	3621	**18.60**	**6.3915**

Chinese long sentences. As we analyzed in the previous chapters, Chinese and English language have two very different patterns of using long sentences, these differences also deteriorate the performance of the baseline method.

The translation quality is also becoming better as the number of segmentations increases. After segmentation we obtain a gain of about 2.5% of the BLEU score on the rule-based system, and a gain of about 1% on the statistical Moses system. Although to maintain the original meaning of sentences we only split the sentences and didn't conduct further reordering or rewriting, such little changes to sentences still gets encouraging results.

6 Conclusion and Future Work

In this paper we try to add commas into sentences at proper positions. In this way we split long sentences improve the translation result. We use a rule-based algorithm to deal with some simple situations, and process complex situations by CRF. The experiments show that the precision of adding commas is quite high, and our method can reasonably insert commas into sentences and improves the translation quality.

Future work may focus on feature extraction and better template choosing, or developing better algorithms. Because in a sentence the potential split places not only depend on words or syntactic structure, but also and maybe more on semantic information, so we think add more semantic features may help improve the performance.

7 Acknowledgements

The work of this paper was supported by the National Basic Research Program of China (973 Program, Grant No. 2013CB329303) and National Natural Science Foundation of China (Grant No. 61201351, 61132009).

References

1. De Marneffe, M.C., MacCartney, B., Manning, C.D.: Generating typed dependency parses from phrase structure parses. In: Proceedings of LREC. vol. 6, pp. 449–454 (2006)
2. Huang, H., Chen, Z.: Tlie hybrid strategy processing approach of complex long sentence. Journal of Chinese Information Processing 16(3), 1–7 (2002)
3. Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al.: Moses: Open source toolkit for statistical machine translation. In: Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions. pp. 177–180. Association for Computational Linguistics (2007)
4. Lafferty, J., McCallum, A., Pereira, F.C.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data (2001)
5. Mudrak, B.: When two parts of a sentence should go their separate ways. http://expertedge.aje.com/2013/04/16/editing-tip-of-the-week-when-two-parts-of-a-sentence-should-go-their-separate-ways/ (April 2013)
6. Somers, H.: Round-trip translation: What is it good for. In: Proceedings of the Australasian Language Technology Workshop. pp. 127–133 (2005)
7. Sun, Y., O'Brien, S., O'Hagan, M., Hollowood, F.: A novel statistical pre-processing model for rule-based machine translation system. Proceedings of EAMT, 8pp (2010)
8. Tian, L., Wong, D.F., Chao, L.S., Quaresma, P., Oliveira, F., Yi, L.: Um-corpus: A large english-chinese parallel corpus for statistical machine translation. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14). European Language Resources Association (ELRA) (2014)
9. Toutanova, K., Klein, D., Manning, C.D., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1. pp. 173–180. Association for Computational Linguistics (2003)
10. Xiong, H., Xu, W., Mi, H., Liu, Y., Liu, Q.: Sub-sentence division for tree-based machine translation. In: Proceedings of the ACL-IJCNLP 2009 Conference Short Papers. pp. 137–140. Association for Computational Linguistics (2009)
11. Xue, N., Yang, Y.: Chinese sentence segmentation as comma classification. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2. pp. 631–635. Association for Computational Linguistics (2011)
12. Yin, B., Zuo, J., Ye, N.: Long sentence partitioning using top-down analysis for machine translation. In: Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on. vol. 3, pp. 1425–1429. IEEE (2012)
13. Yin, D., Ren, F., Jiang, P., Kuroiwa, S.: Chinese complex long sentences processing method for chinese-japanese machine translation. In: Natural Language Processing and Knowledge Engineering, 2007. NLP-KE 2007. International Conference on. pp. 170–175. IEEE (2007)