Phrase-Based Approach for Adaptive Tokenization

Jianqiang Ma
Department of Linguistics
University of Tübingen
Wilhelmstr. 19, Tübingen, 72074, Germany
jma@sfs.uni-tuebingen.de

Dale Gerdemann
Department of Linguistics
University of Tübingen
Wilhelmstr. 19, Tübingen, 72074, Germany
dg@sfs.uni-tuebingen.de

Abstract

Fast re-training of word segmentation models is required for adapting to new resources or domains in NLP of many Asian languages without word delimiters. The traditional tokenization model is efficient but inaccurate. This paper proposes a phrase-based model that factors sentence tokenization into phrase tokenizations, the dependencies of which are also taken into account. The model has a good OOV recognition ability, which improves the overall performance significantly. The training is a linear time phrase extraction and MLE procedure, while the decoding is via dynamic programming based algorithms.

1 Introduction

In many Asian languages, including Chinese, a sentence is written as a character sequence without word delimiters, thus word segmentation remains a key research topic in language processing for these languages. Although many reports from evaluation tasks present quite positive results, a fundamental problem for real word applications is that most systems heavily depend on the data they were trained on. In order to utilize increasingly available language resources such as user contributed annotations and web lexicon and/or to dynamically construct models for new domains, we have to either frequently re-build models or rely on techniques such as incremental learning and transfer learning, which are unsolved problems themselves.

In the case of frequent model re-building, the most efficient approach is the *tokenization model* (using the terminology in Huang et al., 2007), in which the re-training is just the update of the dictionary and the segmentation is a *greedy* string matching procedure using the dictionary and some disambiguation heuristics, e.g. Liang (1986) and Wang et al. (1991). An extension of this approach is the *dynamic programming* search of the most probable word combination on the *word lattice*, such as Ma (1996) and Sproat et al. (1996), which utilize information such as word frequency statistics in a corpus to build the model and are less efficient but more accurate.

However, all the methods mentioned above are mostly based on the knowledge of in-vocabulary words and usually suffer from poor performance, as the out-of-vocabulary words (OOV) rather than segmentation ambiguities turn out to the dominant error source for word segmentation on real corpora (Huang and Zhao, 2007). This fact has led to a shift of the research focus to modeling the roles of individual characters in the word formation process to tackle the OOV problem. Xue (2003) proposes a *character classification model*, which classifies characters according to their positions in a word using the maximum entropy classifier (Berger et al., 1996). Peng et al. (2004) has further extended this model to its sequential form, i.e. sequence labeling, by adopting linear-chain conditional random fields (CRFs, Lafferty et al., 2001). As it is capable of capturing the morphological behaviors of characters, the character classification model has significantly better performance in OOV recognition and overall segmentation accuracy, and has been the state-of-art since its introduction, suggested by the leading performances of systems based on it in recent international Chinese word

Proceedings of the Twelfth Meeting of the Special Interest Group on Computational Morphology and Phonology (SIGMORPHON2012), pages 17–25,
Montréal, Canada, June 7, 2012.

segmentation bakeoffs (Emerson, 2005; Levow, 2006; Zhao and Liu, 2010).

The tokenization model has advantages in **simplicity** and **efficiency**, as the basic operation in segmentation is *string matching* with *linear time complexity* to the sentence length and it only needs a dictionary thus requires *no* training as in the character classification model, which can easily have millions of features and require hundreds of iterations in the training phase. On the other hand, it has inferior performance, caused by its poor OOV induction ability.

This work proposes a framework called **phrase-based tokenization** as a generalization of the tokenization model to cope with its deficiencies in OOV recognition, while preserving its advantages of simplicity and efficiency, which are important for adaptive word segmentation. The segmentation hypothesis unit is extended from a *word* to a *phrase*, which is a character string of arbitrary length, i.e. combinations of partial and/or complete words. And the statistics of different tokenizations of the same phrase are collected and used for parameters estimation, which leads to a *linear time* model construction procedure. This extension makes hypothesis units capable of capturing richer context and describing morphological behavior of characters, which improves OOV recognition. Moreover, *overlapping* hypothesis units can be combined once certain consistency conditions are satisfied, which avoids the unrealistic assumption of independence among the tokenizations of neighboring phrases.

Phrase-based tokenization decomposes the sentence tokenization into phrase tokenizations. We use a graph called *phrase tokenization lattice* to represent all the hypotheses of phrase tokenization in a given sentence. Under such a formulation, tokenizing a sentence is transformed to the shortest path search problem on the graph, which can be efficiently solved by dynamic programming techniques similar to the Viterbi (1967) algorithm.

2 Phrase-Based Model

The hypothesis unit of the tokenization model is the *word*, i.e. it selects the best word sequence from all the words that can be matched by substrings of the sentence (usually in a greedy manner). Once a word is chosen, the corresponding boundaries are determined. This implies that as the characters in a word are always considered as a whole, the morphological behavior of an individual character, e.g. the distribution of its positions in words, is ignored thus makes it impossible to model the word formation process and recognize OOV.

Critical tokenization (Guo, 1997) suggests a method of discovering all and only unambiguous token boundaries (critical points) and generating longest substrings with all inner positions ambiguous (critical fragments) under the assumption of *complete dictionary*. Then an *example-based* method using the context can be adopted to disambiguate the tokenization of critical fragments (Hu et al, 2004). However, the complete dictionary assumption is *not* realistic in practice, as the word formation is so dynamic and productive that there is no dictionary that is even close to the complete lexicon. Given the presence of OOV, a word, including a monosyllabic word, in the original dictionary may be a substring, i.e. a *partial word*, of an OOV. In this case, the critical points found by the dictionary are *not* guaranteed to be unambiguous.

As the complete dictionary does not exist as a static object, a possible solution is to make a dynamic dictionary, which induces words on the fly. But this will not be discussed in this paper. Instead, we attempt to generalize the tokenization model to work *without* the complete dictionary. Different from making distinctions of critical fragments and "non-critical" fragments in critical tokenization, we suggest using *phrases* to represent potentially ambiguous fragments of sentences in a unified way. We define a phrase as a substring of a sentence, the boundaries of which, depending on the tokenization, may or may not necessarily match word boundaries. The fact that partial words, including single characters, may appear on both ends of a phrase makes it possible to describe "morphemes in the context" for OOV induction. A consequence of introducing phrase in tokenization is that a manually segmented corpus is needed in order to collect phrases.

2.1 Tokenization

Tokenization is the process of separating words or word-like units from sentences or character strings. We can consider sentence tokenization as a mapping from each position in the sentence to a

binary value, which indicates the presence (denoted as #) or the absence of word boundary (denoted as $) at that position. A specific tokenization realization of a sentence can be represented by a list of binary values, which can be generated by the concatenations of its sub-lists. In other words, a tokenization of a given sentence can be represented as the *concatenation* of the tokenizations of its component phrases.

If we assume that the tokenization of a phrase is *independent* of other phrases in the same sentence, the sentence tokenization problem is decomposed to smaller phrase tokenization problems, which are unrelated to each other. The independency assumption is not necessarily true but in general is a good approximation. We take this assumption by default, unless there exists evidence that suggests otherwise. In that case, we introduce a method called *backward dependency match* to fix the problem, which will be discussed in Section 3.3.

2.2 Phrase Tokenization Lattice

Informally a **phrase tokenization lattice**, or *lattice* in short, is a set of hypothesized tokenization of phrases in the given sentence, which is a compact representation of all the possible tokenization for that sentence. Using the notations in Mohri (2002), we formally define a **lattice** as a *weighted directed graph* $<V,E>$ with a mapping $W:E\mapsto A$, where V is the set of *nodes*, E is the set of *edges*, and the mapping W assigns each edge a *weight* w from the semiring $<A,\oplus,\otimes,\bar{0},\bar{1}>$[1].

For a given sentence $s[0...m]$, each node $v\in V$, denotes a *sentence position* (the position between a pair of adjacent characters in a untokenized sentence). Each edge $e\in E$ from node v_a to node v_b, denotes a tokenization of the phrase between the positions defined by v_a and v_b. And for each edge e, a weight w is determined by the mapping W, denotes the *phrase tokenization probability*, the probability of the phrase defined by the two nodes of the edge being tokenized as the tokenization defined by that edge. A *path* π in the lattice is a sequence of consecutive edges, i.e. $\pi=e_1e_2...e_k$, where e_i and e_{i+1} are connected with a node. The *weight* for the path π can be defined as:

$$w(\pi)=\bigotimes_{i=1}^{k}w(e_i) \quad (1)$$

which is the product of the weights of its component edges. A path from the source node to the sink node, represents a *tokenization* of the sentence *being factored as the concatenation of tokenizations* of phrases represented by those edges of on that path.

For example, with some edges being pruned, the lattice for the sentence 有人质疑他 ‘Someone questions him’ is shown in Figure 1.

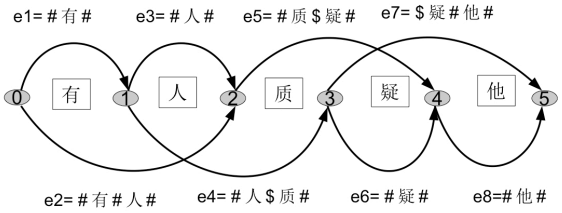

Figure 1. A pruned phrase tokenization lattice. Edges are tokenizations of phrases, e.g. e_5 represents tokenizing 质疑 ‘question’ into a word and e_7 represents tokenizing 疑他 ‘doubt him’ into a partial word 疑 ‘doubt’ followed by a word 他 ‘him’.

2.3 Tokenization as the Best Path Search

After the introduction of the lattice, we formally describe the tokenization (disambiguation) problem as the best path searching on the lattice:

$$\hat{T}=\arg\max_{T\in D}w(T) \quad (2)$$

where D is the set of all paths from the source node to the sink node, and $\hat{T}$ is the path with the highest weight, which represents the best tokenization of the sentence. Intuitively, we consider the *product* of phrase tokenization probabilities as the probability of the sentence tokenization that is generated from the concatenation of these phrase tokenizations.

Note that every edge in the lattice is from a node represents an earlier sentence position to a node that represents a later one. In other words, the lattice is *acyclic* and has a clear topological order.

[1] A semiring defines an algebra system with certain rules to compute path probabilities and the max probabilities from a node to another. See Mohri (2002) for details.

In this case, the best path can be found using the Viterbi (1967) algorithm efficiently[2].

3 Training and Inference Algorithms

3.1 Model Training

In order to use the lattice to tokenize unseen sentences, we first have to build a model that can generate the edges and their associated weight, i.e. the tokenization of all the possible phrases and their corresponding phrase tokenization probability. We do it by collecting all the phrases that have occurred in a training corpus and use *maximum likelihood estimation* (MLE) to estimate the phrase tokenization probabilities. The estimation of the probability that a particular phrase $A = a_1 a_2 ... a_n$ being tokenized as the tokenization $T = t_1 t_2 .. t_m$ is given in equation (3), where $C(\bullet)$ represents the empirical count, and the set of all T' stands for all possible tokenizations of A. To avoid extreme cases in which there is no path at all, techniques such as *smoothing* can be applied.

$$P(T \mid A) = \frac{C(T,A)}{\sum_{T'} C(T',A)} = \frac{C(T,A)}{C(A)} \quad (3)$$

The result of the MLE estimation is stored in a data structure called phase tokenization table, from which one can retrieval all the possible tokenizations with their corresponding probabilities for the every phrase that has occurred in the training corpus. With this model, we can construct the lattice, i.e. determine the set of edges E and the mapping function W (defining nodes is trivial) for a given sentence in a simple string matching and table retrieval manner: when a substring of sentence is matched to a stored phrase, an edge is built from the its starting and ending node to represent a tokenization of that phrase, with the weight of the edge equals to the MLE estimation of the stored phrase-tokenization pair.

3.2 Simple Dynamic Programming

Once the model is built, we can tokenize a given sentence by the inference on the lattice which represents that sentence. The proposed simple dynamic programming algorithm (**Algorithm 1**, as shown in Figure 2) can be considered as the phrase tokenization lattice version of the *evalUtterance* algorithm in Venkataraman (2001). The best tokenization of the partial sentence up to a certain position is yielded by the best combination of one previous best tokenization and one of the phrase tokenizations under consideration at the current step.

The upper bound of the time complexity of Algorithm 1 is $O(kn^2)$, where n is the sentence length and k is the maximum number of the possible tokenization for a phrase. But in practice, it is neither necessary nor possible (due to data sparseness) to consider phrases of arbitrary length, so we set a constraint of maximum phrase length of about 10, which makes the time complexity de-facto linear.

Algorithm 1: Simple Dynamic Programming

Prerequisite: Phrase Tokenization Table (PT)
Input: Sentence S[0...N]
Initialization:
BestScore= *N*-dimension zero vector
BestTokenization= *N*-dimension null-string vector
Algorithms:

```
for i=1 to N do :            // i: current position in the sentence
    for j=i-1 to 0 do :      // j: starting position of the last phrase
        phrase=S[j:i]
        if phrase in PT :
            tokenization=GetTopTokenization(PT, phrase)
            tokenization_prob=GetProbability(PT, tokenization)
            score=BestScore[j] * tokenization_prob

            if score>BestScore[i] :
                BestTokenization[i]=tokenization
                BestScore[i]=score
                back_pointer[i]=j
        else:
            break    // if the phrase not in PT, exist the inner loop

BestPath ← Path traced back from back_pointer[N]
SentenceTokenization← Concatenation of the BestTokenizaiton
entries of the edges on the BestPath (which are phrase tokenizations)
```

Output: SentenceTokenization

Figure 2. The pseudo code of Algorithm 1.

The key difference to a standard word-lattice based dynamic programming lies in the phrase lattice representation that the algorithm runs on. Instead of representing a word candidate as in Venkataraman (2001), each edge now represents a

[2] More rigid mathematical descriptions of this family of problems and generic algorithms based on semirings are discussed in Mohri (2002) and Huang (2008).

tokenization of a phrase defined by two nodes of the edge, which can include full and partial words. The combination of phrase tokenizations may yield new words that are not in the dictionary, i.e. our method can recognize OOVs.

Let us consider a slightly modified version of the lattice in Figure 1. Suppose edge e_5 =#质$疑# does not exist , i.e. the word 质疑 'question' is not in the dictionary, and there is new edge e_5'= #质$ that links node 2 and node 3 and represents a partial word. Two of possible tokenizations of the sentence are path $p_1 = e_1 e_4 e_6 e_8$ and path $p_2 = e_2 e_5' e_7$. Note that p_2 recognizes the word 质疑 'question' by combining two partial words, even though the word itself has not seen before. Of course, this OOV is finally recognized only if a path that can yield it is the best path found by the decoding algorithm.

Once the best path is found, the procedure of mapping it back to segmented words is as follows. The phrase tokenizations represented by the edges of the best path are concatenated, before substituting meta symbols # and $ into white space and empty string, respectively. For example, if $p_2 = e_2 e_5' e_7$ is the best path, the concatenation of the phrase tokenizations of the three edges on the path will be #有#人##质$$疑#他#, and removal of $ and substitution of # into the white space will further transform it into 有 人 质疑 他 'Somebody questions him', which is the final result of the algorithm.

3.3 Compatibility and Backward Dependency Match

As mentioned in Section 2, the independency assumption of phrase tokenization is not always true. Considering the example in Figure 1, e_4 and and e_7 are not really *compatible*, as e_4 represents a word while e_7 represents a partial word that expects the suffix of its preceding phrase to form a word with its prefix. To solve this problem, we require that the last (meta) symbol of the preceding tokenization must equal to the first (meta) symbol of the following tokenization in order to concatenate the two. This, however, has the consequence that there may be no valid tokenization at all for some positions. As a result, we have to maintain the top *k* hypotheses and use the *k-best* path search algorithms instead of 1-best (Mohri, 2002). We adopt the naïve k-best path search, but it is possible to use more advanced techniques (Huang and Chiang, 2005).

The compatibility problem is just the most salient example of the general problem of variable independency assumptions, which is the "unigram model" of phrase tokenization. A natural extension is a higher order Markov model. But that is inflexible, as it assumes a fixed variable dependency structure (the current variable is always dependent on previous *n* variables). So we propose a method called *backward dependency match*, in which we start from the independency assumption, then try to explore the longest sequence of adjacent dependencies that we can reach via string match for a given phrase and its precedent.

To simplify the discussion, we use sequence labeling, or conditional probability notation of the tokenization. A tokenization of the given character sequence (sentence) is represented as a specific label sequence of same length. The label can be those in the standard 4-tag set of word segmentation (Huang and Zhao, 2007) or the #/$ labels indicating the presence or absence of a word boundary after a specific character.

The possible tokenizations of character sequence $a_1 a_2 a_3$ are represented as the probability distribution $P(t_1 t_2 t_3 \mid a_1 a_2 a_3)$, where $t_1 t_2 t_3$ are labels of $a_1 a_2 a_3$. If a tokenization hypothesis of $a_1 a_2 a_3$ decomposes its tokenization into the concatenation of the tokenization of $a_1 a_2$ and the tokenization of a_3, this factorization can be expressed as $P(t_1 t_2 \mid a_1 a_2) \times P(t_3 \mid a_3)$, as shown in Figure 3a. For a specific *assignment* $<\overline{a_1}\,\overline{a_2 a_3}; \overline{t_1}\,\overline{t_2 t_3}>$, if we find that $<\overline{a_2 a_3}>$ can be tokenized as $<\overline{t_2}\,\overline{t_3}>$, it suggests that t_3 may be *dependent on* a_2 and t_2 as well, so we update the second part of the factorization (at least for this assignment) to: $P(t_3 \mid a_3; a_2 t_2)$, which can be estimated as:

$$P(\overline{t_3} \mid \overline{a_3}; \overline{a_2}\,\overline{t_2}) = \frac{C(\overline{a_2 a_3 t_2 t_3})}{\sum_{t_3} C(\overline{a_2 a_3 t_2 t_3})} \quad (4)$$

In this case, the factorization of the tokenization $P(t_1t_2t_3 \mid a_1a_2a_3)$ is $P(t_1t_2 \mid a_1a_2) \times P(t_3 \mid a_3;a_2t_2)$, as shown in Figure 3b.

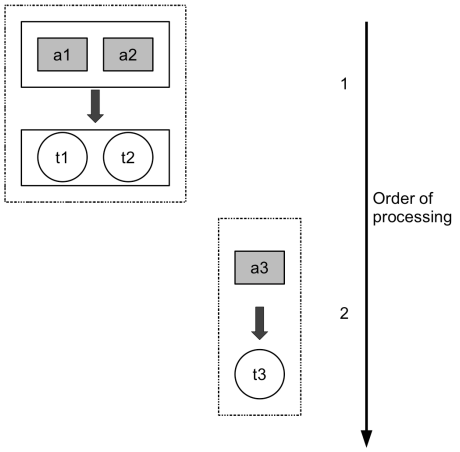

Figure 3a. The factorization of $P(t_1t_2t_3 \mid a_1a_2a_3)$ into $P(t_1t_2 \mid a_1a_2) \times P(t_3 \mid a_3)$.

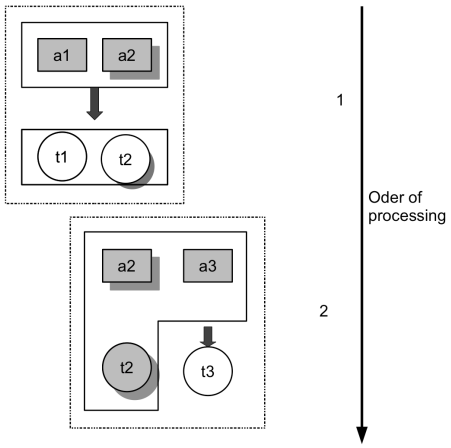

Figure 3b. The factorization of $P(t_1t_2t_3 \mid a_1a_2a_3)$ into $P(t_1t_2 \mid a_1a_2) \times P(t_3 \mid a_3;a_2t_2)$. Note that in the 2nd factor, in addition to a_3, a_2 and t_2 are also observed variables and all of them are treated as a unit (shown by the L-shape). The shadowed parts (a_2 and t_2) represent the matched items.

Algorithm 2 is based on the k-best search algorithm, which calls the backward dependency match after a successful compatibility check, and match as far as possible to get the largest probability of each tokenization hypothesis. In extreme cases, where no tokenization hypothesis survives the compatibility check, the algorithm backs off to Algorithm 1.

4 Experiments

We use the training and testing sets from the second international Chinese word segmentation bakeoff (Emerson, 2005), which are freely available and most widely used in evaluations. There are two corpora in simplified Chinese provided by Peking University (PKU) and Microsoft Research (MSR) and two corpora in traditional Chinese provided by Academic Sinica (AS) and the City University of Hong Kong (CityU). The experiments are conducted in a closed-test manner, in which no extra recourse other than the training corpora is used. We use the same criteria and the official script for evaluation from the bakeoff, which measure the overall segmentation performance in terms of *F-scores*, and the OOV recognition capacity in terms of *Roov*.

Precision is defined as the number of correctly segmented words divided by the total number of words in the segmentation result, where the correctness of the segmented words is determined by matching the segmentation with the gold standard test set. Recall is defined as the number of correctly segmented words divided by the total number of words in the gold standard test set. The evenly-weighted F-score is calculated by:

$$F = 2 \times p \times r / (p + r) \qquad (5)$$

Roov is the recall of all the OOV words. And *Riv* is the recall of words that have occurred in the training corpus. The evaluation in this experiment is done automatically using the script provided with the second bakeoffs data.

We have implemented both Algorithm 1 and Algorithm 2 in Python with some simplifications, e.g. only processing phrase up to the length of 10 characters, ignoring several important details such as pruning. The performances are compared with the baseline algorithm maximum matching (MM), described in Wang et al. (1991), and the best bakeoff results. The *F-score*, *Roov and Riv* are summarized in Table 1, Table 2, and Table 3, respectively.

All the algorithms have quite similar recall for the in-vocabulary words (*Riv*), but their *Roov* vary

greatly, which leads to the differences in F-score. In general both Algorithm 1 and Algorithm 2 improves OOV Recall significantly, compared with the baseline algorithm, maximum matching, which has barely any OOV recognition capacity. This confirms the effectiveness of the proposed phrase-based model in modeling morphological behaviors of characters. Moreover, Algorithm 2 works consistently better than Algorithm 1, which suggests the usefulness of its strategy of dealing with dependencies among phrase tokenizations.

Besides, the proposed method has the *linear* training and testing (when setting a maximum phrase length) time complexity, while the training complexity of CRF is the proportional to the feature numbers, which are often over millions. Even with current prototype, our method takes only minutes to build the model, in contrast with several hours that CRF segmenter needs to train the model for the same corpus on the same machine.

Admittedly, our model still underperforms the best systems in the bakeoff. This may be resulted from that 1) our system is still a prototype that ignores many minor issues and lack optimization and 2) as a generative model, our model may suffer more from the data sparseness problem, compared with discriminative models, such as CRF.

As mentioned earlier, the OOV recognition is the dominant factor that influences the overall accuracy. Different from the mechanism of tokenization combination in our approach, state-of-art systems such as those based on MaxEnt or CRF, achieve OOV recognition basically in the same way as in-dictionary word recognition. The segmentation is modeled as assigning labels to characters. And the probability of the label assignment for a character token is mostly determined by its features, which are usually local contexts in the form of character co-occurrences.

There are many other OOV recognition methods proposed in literature before the rise of machine learning in the field. For example, the Sproat et al. (1996) system can successfully recognize OOVs of strong patterns, such as Chinese personal names, transliterations, using finite-state techniques. Another typical example is Ma and Chen (2003), which proposed context free grammar like rules together with a recursive bottom-up merge algorithm that merges possible morphemes after an initial segmentation using maximum matching. It would be fairer to compare the OOV recognition performance of our approach with these methods, rather than maximum matching. But most earlier works are not evaluated on standard bake-off corpora and the implementations are not openly available, so it is difficult to make direct comparisons.

F-score	As	CityU	MSR	PKU
Best Bakeoff	0.952	0.943	0.964	0.950
Algorithm 2	0.919	0.911	0.946	0.912
Algorithm 1	0.897	0.888	0.922	0.890
MM	0.882	0.833	0.933	0.869

Table 1. The F-score over the bakeoff-2 data.

Roov	AS	CityU	MSR	PKU
Best Bakeoff	0.696	0.698	0.717	0.636
Algorithm 2	0.440	0.489	0.429	0.434
Algorithm 1	0.329	0.367	0.411	0.416
MM	0.004	0.000	0.000	0.059

Table 2. The Roov over the bakeoff-2 data.

Riv	AS	CityU	MSR	PKU
Best Bakeoff	0.963	0.961	0.968	0.972
Algorithm 2	0.961	0.961	0.970	0.951
Algorithm 1	0.955	0.940	0.950	0.940
MM	0.950	0.952	0.981	0.956

Table 3. The Riv over the bakeoff-2 data.

5 Conclusion

In this paper, we have presented the phrase-based tokenization for adaptive word segmentation. The proposed model is efficient in both training and decoding, which is desirable for fast model re-construction. It generalizes the traditional

tokenization model by considering the phrase instead of the word as the segmentation hypothesis unit, which is capable of describing "morphemes in the context" and improves the OOV recognition performance significantly. Our approach decomposes sentence tokenization into phrase tokenizations. The final tokenization of the sentence is determined by finding the best combination of the tokenizations of phrases that cover the whole sentence. The tokenization hypotheses of a sentence are represented by a weighed directed acyclic graph called phrase tokenization lattice. Using this formalism, the sentence tokenization problem becomes a shortest path search problem on the graph.

In our model, one only needs to estimate the phrase tokenization probabilities in order to segment new sentences. The training is thus a linear time phrase extraction and maximum likelihood estimation procedure. We adopted a Viterbi-style dynamic programming algorithm to segment unseen sentences using the lattice. We also proposed a method called backward dependency match to model the dependencies of adjacent phrases to overcome the limitations of the assumption that tokenizations of neighboring phrases is independent. The experiment showed the effectiveness of the proposed phrase-based model in recognizing out-of-vocabulary words and its superior overall performance compared with the traditional tokenization model. It has both the efficiency of the tokenization model and the high performance of the character classification model.

One possible extension of the proposed model is to apply re-ranking techniques (Collins and Koo, 2005) to the k-best list generated by Algorithm 2. A second improvement would be to combine our model with other models in a log linear way as in Jiang et al. (2008). Since phrase-based tokenization is a model that can be accompanied by different training algorithms, it is also interesting to see whether discriminative training can lead to better performance.

Acknowledgments

The research leading to these results has received funding from the European Commission's 7th Framework Program under grant agreement n° 238405 (CLARA).

References

Adam Berger, Stephen Della Pietra, and Vincent Della Pietra. 1992. A Maximum Entropy Approach to Natural Language Processing. 1996. *Computational Linguistics*, 22(1): 39-71

Michael Collins and Terry Koo. 2005. Discriminative Reranking for Natural Language Parsing. *Computational Linguistics*, 31(1):25-69.

Thomas Emerson. 2005. The second international Chinese word segmentation bakeoff. In *Proceedings of Forth SIGHAN Workshop on Chinese Language Processing*. Jeju Island, Korea.

Jin Guo. 1997. Critical tokenization and its properties. *Computational Linguistics*, 23(4): 569-596

Qinan Hu, Haihua Pan, and Chunyu Kit. 2004. An example-based study on Chinese word segmentation using critical fragments. In *Proceedings of IJCNLP-2004*. Hainan Island, China

Changning Huang and Hai Zhao. 2007. Chinese Word Segmentation: a Decade Review. *Journal of Chinese Information Processing*, 21(3): 8-20

Chu-Ren Huang, Petr Simon, Shu-Kai Hsieh, and Laurent Prévot. Rethinking Chinese word segmentation: tokenization, character classification, or wordbreak identification. In *Proceedings of ACL-2007*. Prague, Czech

Liang Huang. 2008. Advanced dynamic programming in semiring and hypergraph frameworks. *In Proceedings of COLING 2008*. Manchester, UK.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*. Vancouver, Canada

Wenbin Jiang, Liang Huang, Qun Liu, Yajuan Lu. 2008. A Cascaded Linear Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging. In *Proceedings of ACL 2008: HLT*. Columbus, USA

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML 2001*. Williamstown, MA, USA

Gina-Anne Levow. 2006. The third international Chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*. Sydney, Australia

Nanyuan Liang. 1986. On computer automatic word segmentation of written Chinese. *Journal of Chinese Information Processing*, 1(1).

Wei-Yun Ma and Keh-Jiann Chen. 2003. A bottom-up merging algorithm for Chinese unknown word extraction. In *Proceedings of the second SIGHAN workshop on Chinese language processing*. Sapporo, Japan

Yan Ma. 1996. The study and realization of an evaluation-based automatic segmentation system. In Changning Huang and Ying Xia, editors, *Essays in Language Information Processing*. Tsinghua University Press, Beijing, China.

Mehryar Mohri. 2002. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of COLING*. Stroudsburg, PA, USA.

Richard Sproat, Chilin Shih, William Gale, and Nancy Chang. 1996. A stochastic finite-state word-segmentation algorithm for Chinese. *Computational Linguistics*, 22(3):377-404.

Anand Venkataraman. 2001. A Statistical Model for Word Discovery in Transcribed Speech. *Computational Linguistics*, 27(3): 351-372

Andrew Viterbi (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13 (2): 260–269.

Xiaolong Wang, Kaizhu Wang, and Xiaohua Bai. 1991. Separating syllables and characters into words in natural language understanding. *Journal of Chinese Information Processing,* 5(3):48-58.

Nianwen Xue. 2003. Chinese Word Segmentation as Characater Tagging. *Computational Linguistics and Chinese Language Processing*, 8(1): 29-48

Hongmei Zhao and Qun Liu. 2010. The CIPS-SIGHAN CLP 2010 Chinese Word Segmentation Bakeoff. In *Proceedings of the First CPS-SIGHAN Joint Conference on Chinese Language Processing*. Beijing, China.