CHINESE SENTENCE TOKENIZATION USING VITERBI DECODER

LI Haizhou, BAI Shuanhu and LIN Zhiwei
Kent Ridge Digital Labs, Heng Mui Keng Terrace, Singapore 116913
{hzli;bai;lzw}@krdl.org.sg

ABSTRACT

In this paper, an approach to Chinese sentence tokenization is proposed whereby word segmentation and text normalization could be conducted at the same time within the framework of Viterbi decoding. In the process, not only lexical words but also the new word classes could be identified. The approach demonstrated is very practical in sentence tokenization for *n*-gram statistical language modeling.

1. INTRODUCTION

It is known that Chinese is an ideographic language and there is no word delimiter between words in written Chinese sentences. Chinese sentence tokenization is to segment a sentence into word or word-like tokens given lexicon and word productive rules. It is an important task in natural language processing, for instance, in building an *n*-gram statistical language model for speech recognition applications.

Sentence tokenization has been widely studied in recently years and there are quite a number of publications on it [1-3]. Word boundary disambiguation and unknown word detection have been the major foci. To deal with new word detection, it is common to segment the text into lexical words in the first pass, then merge tokens into a plausible new word in the second pass according word formation rules. As a result, there is a risk when error segmentation happens in the first pass, which can not be recovered in the second pass. In this paper, we propose a unified solution to lexical word and productive word identification, based on the Viterbi decoding algorithm and finite-state-grammar parsing.

Of the algorithms or methods for sentence tokenization, most of them fall into two categories, that is, lexical knowledge based and linguistic knowledge based methods. Lexical knowledge based word segmenter only make use of the lexicon knowledge to conduct segmentation judgement. It is efficient and straightforward in practice. Resorting to more powerful decoding approaches, linguistic knowledge based segmenters combine lexical information with langauge statistical information to provide higher accuracy. The proposed approach in this paper belongs to the second category.

In section 2, we will describe the decoding mechanism; In section 3, we discuss how word hypotheses are produced. After showing an experiment in section 4, we summarize the discussions.

2. VITERBI DECODING

The task here is to decompose a Chinese sentence into words. It is known that the definition of a Chinese word itself is a complicated issue in Chinese linguistics, and worth a separate paper. In this context, words are referred to as both lexical entries as well as phrases derived via various productive processes, such as proper names, numeric sequences, expressions for dates, and so forth. There is an argument that one could obtain a correct tokenization only when a

sentence is perfectly parsed. It is true in the sense that the semantic class affiliation of a word and its contextual constraint constitutes useful information in the tokenization. We know that an *n*-gram langauge model describes the regularities of a langauge whereby a better tokenization could be expected.

Speaking in information theory, it is considered that a sentence with boundary marks go through a noisy channel. After the noisy channel, all the marks are gone. The job of the decoder is to model the original process that marked the boundaries before they went through the noisy channel.

More formally, we are going to find a word sequence $W = \{w_i\}$ in a sentence $S : p(W \mid S)$. Under the Bayes' rule, we have

$$p(W \mid S) = \frac{p(W, S)}{p(S)} \quad (1)$$

Since the *a priori* probability of the sentence is constant for any segmentation patterns, we can maximize Eq.(1) by maximizing the numerator alone. Given a word *2*-gram langauge model, the likelihood could be factored into the following:

$$p(W, S) = \prod_i p(w_i \mid w_{i-1}) p(w_{i-1} \mid s_i) p(s_i) \quad (2)$$

where $p(w_i \mid w_{i-1})$ is the word bigram, $p(w_i \mid s_i)$ serves as the word unigram and $p(s_i)$ the *a priori* probability of a character being a breaking point. Obviously, $p(s_i)$ is a normalizatin or penalty factor to balance word merging or spliting whenever there exists an ambiguous segmentation.

The parameters in Eq.(2) reflect the tokenization criteria in the process. By replacing them with other heuristics, one can easily derive various tokenization methods within the Viterbi framework. When $p(w_{i-1} \mid s_i)$ takes the token length and $p(w_i \mid w_{i-1})$ is kept constant, the Viterbi decoder leads to a Maximum Match solution. A Least Word method could be derived in a similar way.

It is presumed that all single characters are also words by themselves, therefore, the word lattice is built in a character-synchronized manner. The number of lattice entries are pruned down during the forward pass to get rid of unreasonable partial paths. The pruning reduces the search space and consequently speeds up the process. Fig.1 shows the tokenization mechanism.

3. WORD HYPOTHESIS

3.1 Lexical words

Given a lexicon, one can easily construct a word lattice from a character string where all the possible word segmentation results are retained. Each word is associated with a unigram, the word frequency. Furthermore, when a statistical *n*-gram is available, each *n*-word string is associated with a word or a word class[1] *n*-gram[5] probability.

At each potential breaking point, the lattice generator firstly looks backward in the sentence to search for all the potential lexical words. This search is carried out for each character boundary in the sentence from left to right, named forward pass. When a word hypothesis is found, its identity and the boundary are kept in the lattice. To be efficient, a lexicon is usually encoded in a tree structure. This approach is essentially the one proposed in Sproat et al.[3] where a lexical word are encoded as a WFST with costs at the transition arcs.

3.2 Finite-state-grammar words

[1] A class of words consists of a group of words which share common part-of-speech attributes

A finite-state-grammar is used as a means to augment the static lexicon by accomodating some classes of regular expressions. A FSG defines a language for a type of regular expression, which could be a word-like token in Chinese, for example, 五月六日 as a date, 百分之六十 as a percentage, 董建华 as a people's name, and so on.

A FSG attempts to match against a sequence of tokens (words) in much the same way as a lexical word in the forward pass. To propose a word hypothesis, two issues have to be addressed:

1) To find the best parse for a FSG if there is more than one parse for a given token;
2) To provide the back-off scores for new words which are instances of the language defined by a given FSG.

Multiple parses means that ambiguation exists, therefore, a good answer to the first question is to train up a within-class character *n*-gram model which helps score the parses. Good-Turing estimate [6] is one of the solutions to the second point. For instance, the aggregate probability or the unigram, of previsously unseen instances of a construction is estimated as n/N, where n is the number of instances observed only once and N the total number of observed tokens in the training corpus. The same back-off scheme is also applicable to a higher order *n*-gram model.

Once a token is sucessfully parsed by a FSG, it is tagged with an entity identity and recorded into the lattice as a word hypothesis. It is noted that a hypothesis means a word or word class identity together with its boundary. At the end of the forward pass, we have exhausted all the word hypotheses in the input sentence. A backward pass will allow us to apply word or word class *n*-gram to the lattice and decode the best path which gives the lowest cost.

4. EXPERIMENTS

Text normalization is to convert an original text into a word entity sequence. A word entity is referred to as its lexical index when it is a lexical word. In the cases of new words and symbols including punctuation marks, text normalization is intended to encode the spotted entities with their predefined class indices. With a lexicon augmented with FSG words, one is able to conduct text segmentation and normalization at the same time.

Performance for Chinese segmentation and unknown word detection systems is generally reported in terms of the dual measures of precision and recall (P&R). Precision is defined to be the number of correct hits divided by the total number of items selected; and recall rate is defined to be the number of correct hits divided by the number of items that should have been selected. As the definition of what constitutes a correct segmentation is not clear and human judges differ a lot when given the same segmentation task, it is difficult to compare system performance. Here we report an experiment involving Chinese named entity extraction where, in addition to a *50,000* word lexicon, *9* classes of regular expressions are defined as FSGs, such as people's name, place name and date, etc. The within-class character *n*-gram model is trained up with a tagged corpus of *5* million words. The P & R rate for entities of the *9* classes are reported at *86%*, the overall word tokenization accuracy achieves *99%* over a test corpus of *200* thousand words[5].

5. CONCLUSIONS

Using the Viterbi decoder, one is able to re-estimate the word *n*-gram language model as well. We start with an initial estimate of the *n*-gram model. The tokenization and re-estimation of the language model are repeated until a converge is obtained.

In this paper, the Viterbi decoder is proposed as a tokenization framework which allows us to explore different tokenization criteria. An iterative procedure of tokenization and re-estimation is a practical way to refine a *n*-gram langauge model.

REFERENCES

[1] J. Guo, "Critical Tokenization and its properties", Computational Linguistics, Vol. 23, No. 4. P569-596, 1997

[2] H. Li and B. Yuan, "Chinese word segmentation", Language, Information and Computation (PACLIC12), Singapore, 1998

[3] R. Sproat, C Shih, W Gale and N. Chang, "A stochastic finite-state word-segmentation algorithm for Chinese", Computational Linguistics 4 (4):377-404

[4] D. M. Bikel, S. Miller, R. Schwartz and R. Weischedel, "Nymble: a High performance learning name finder", MUC-6 proceedings, 1997

[5] T. Keenan, "Accessing the state of the art in multilingual named entity extraction", Message Entity Tracking Project Report.

[6] K. Church and W. Gale, "A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams", Computer Speech and Language, 5(1):19-54

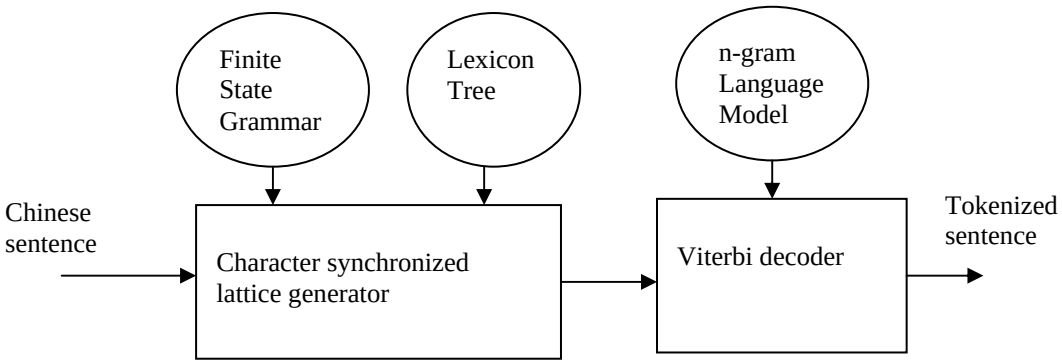

Fig.1 Character synchronized sentence tokenization