Learning Phrase Break Detection in Thai Text-to-Speech

Virongrong Tesprasit[1], Paisarn Charoenpornsawat[1] and Virach Sornlertlamvanich[2]

[1]Information Research and Development Division
National Electronics and Computer Technology Center, Thailand
{virong, paisarn}@nectec.or.th

[2]Thai Computational Linguistics Laboratory
CRL Asia Research Center, Thailand
virach@crl-asia.org

Abstract

One of the crucial problems in developing high quality Thai text-to-speech synthesis is to detect phrase break from Thai texts. Unlike English, Thai has no word boundary delimiter and no punctuation mark at the end of a sentence. It makes the problem more serious. Because when we detect phrase break incorrectly, it is not only producing unnatural speech but also creating the wrong meaning. In this paper, we apply machine learning algorithms namely C4.5 and RIPPER in detecting phrase break. These algorithms can learn useful features for locating a phrase break position. The features which are investigated in our experiments are collocations in different window sizes and the number of syllables before and after a word in question to a phrase break position. We compare the results from C4.5 and RIPPER with a based-line method (Part-of-Speech sequence model). The experiment shows that C4.5 and RIPPER appear to outperform the based-line method and RIPPER performs better accuracy results than C4.5.

1. Introduction

The most important goal of speech synthesis is to synthesize natural speech sound. The crucial problems of high quality speech synthesis not only relates to how natural speech is produced but also concerns how style of synthetic speech is created. Producing appropriate phrase break in input text is the one feature that makes the TTS system reach the naturalness. We cannot accept if the TTS system produces the natural speech with inappropriate pauses. It leads to wrong meaning. Usually people pause their speech after a phrase. Thus to insert appropriate pauses, we have to identify phrase boundaries.

To identify phrase break, several approaches have been proposed such as rule-based, stochastic [2, 4, 11] and learning [1, 6] approaches. Rule-based approaches are easy to implement but these methods consume much linguistic knowledge including syntactic and semantic knowledge. It is difficult to improve the accuracy. Stochastic approaches employ statistics from a training set in processing and do not require linguistic knowledge. However, these techniques cannot capture useful information in long distance. In recent years, several machine-learning algorithms have been proposed in solving various tasks of Natural Language Processing (NLP) such as word sense disambiguation, spelling correction, word segmentation and part-of-speech tagging. Like stochastic approaches, these techniques require training data and do not need linguistic knowledge. Nevertheless, several machine learning algorithms can capture local, long distance context and other information to solve a problem.

In this paper, we proposed to apply machine learning techniques namely C4.5 and RIPPER in identifying phrase breaks. Both algorithms can learn the useful features from training examples. Features which are investigated in our experiments are collocations in different window sizes and the number of syllables before and after a juncture to a phrase break position.

2. Applying Machine Learning Techniques

In this section, we briefly reflect on C4.5 and RIPPER algorithms, and features that we will use in solving our problem - detecting phrase breaks.

2.1. C4.5

C4.5, decision tree, is a traditional classifying technique that proposed by Quinlan [8]. C4.5 have been successfully applied in many NLP problems such as word extraction [10] and sentence boundary disambiguation [7].

C4.5 proceeds by evaluation content of series of attributes and iteratively building a tree from the attribute values with the leaves of the decision tree being the valued of the goal attribute. At each step of learning procedure, the evolving tree is branched on the attribute that partitions the data items with the highest information gain. Branches will be added until all items in the training set are classified. To reduce the effect of overfitting, C4.5 prunes the entire decision tree constructed. It recursively examines each subtree to determine whether replacing it with a leaf or branch would reduce expected error rate. This pruning makes the decision tree better in dealing with the data different from training data.

2.2. RIPPER

RIPPER is the one of the famous machine learning techniques applying in NLP problems proposed by Cohen [3]. On his experiment [12] shows that RIPPER is more efficient than C4.5 on noisy data and it scales nearly linearly with the number of examples in a dataset. Therefore, we decide to choose RIPPER in evaluating and comparing results with C4.5.

RIPPER is a propositional rule learning algorithm that constructs a ruleset which accurately classifies the training

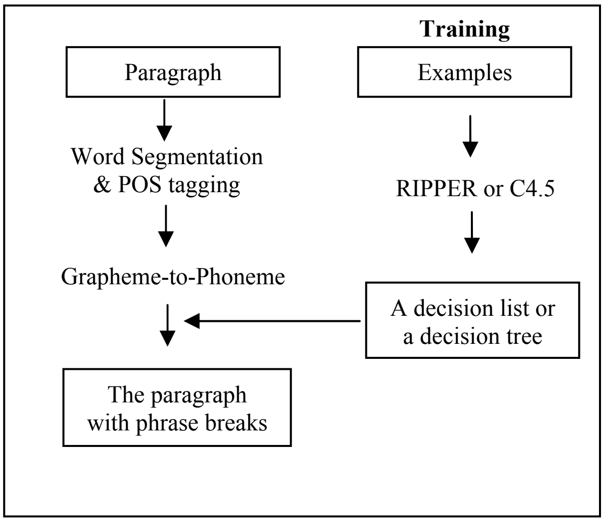
Training
Paragraph
Examples
Word Segmentation & POS tagging
RIPPER or C4.5
Grapheme-to-Phoneme
A decision list or a decision tree
The paragraph with phrase breaks

Figure 1: Architecture of phrase break detection

data. A rule in the constructed ruleset is represented in the form of a conjunction of conditions:

if T_1 and T_2 and ... T_n then class C_x.

T_1 and T_2 and ... T_n is called the body of the rule. C_x is a target class to be learned; it can be a positive or negative class. A condition T_i tests for a particular value of an attribute, and it takes one of four forms: $A_n = v$, $A_c \geq \theta$, $A_c \leq \theta$ and $v \in A_s$, where A_n is a nominal attribute and v is a legal value for A_n; or A_c is a continuous variable and θ is some value for A_c that occurs in the training data; or A_s is a set-value attribute and v is a value that is an element of A_s. In fact, a condition can include negation. A set-valued attribute is an attribute whose value is a set of strings. The primitive tests on a set-valued attribute A_s are of the form "$v \in A_s$".

When constructing a rule, RIPPER finds the test that maximizes information gain for a set of examples S efficiently, making only a single pass over S for each attribute. All symbols v, that appear as elements of attribute A for some training examples, are considered by RIPPER. One of the most powerful ability of RIPPER is to learn the set-valued attributes, and this ability is one of the main reasons for choosing RIPPER for our task. Using the set-valued attributes, examples can be class concept. We provide training data set composed of positive and negative examples of the concept. Given the training set, RIPPER first partitions the data into two sets, a "growing set" and "pruning set", and the repeatedly constructs on rule at a time that classifies data in the growing set, until all positive examples are covered. A rule is initialized with an empty body until no negative example is covered. After a rule is constructed, the rule is immediately pruned. To prune a rule, RIPPER deletes any final sequence of conditions from the rule according to some heuristic. The goal of pruning a rule, which may overfit the growing set when the set contains noisy data, is to improve error rates on unseen data in the pruning set. The algorithm can also handle multiple classes with slight modification.

2.3. Features

To train the algorithms for identifying phrase breaks, two main features are investigated. The first feature is *a collocation feature* which is a pattern of up to n contiguous words and part-of-speech tags around the target juncture. The second feature is *a syllable feature* which is the number of syllables from a target juncture to the previous and the next phrase break.

3. An Overview of the System

Our algorithm for identifying phrase breaks (in Figure 1) consists of three steps as follows:

3.1. Word Segmentation & Part-Of-Speech Tagging

For each input paragraph, probabilistic trigram model [5] is applied to separate the paragraph into a sequence of words and assign their parts of speech (POS), then the best segmented sequence is selected. The probabilistic trigram model can be described formally as following:

Let $C = c_1c_2\ldots c_m$ be an input string, $W_i = w_1w_2..w_n$ be a possible word segmentation, and $T_i = t_1t_2..t_n$ be a sequence of parts of speech. Find W_i which is the highest probability of sequence of words. We can compute $P(W_i)$ in the following fashion:

$$\begin{aligned} P(W_i) &= \sum_T P(W_i, T_i) \\ &= \sum_T \prod_i P(t_i \mid t_{i-1}, t_{i-2}) * P(w_i \mid t_i) \end{aligned} \qquad (1)$$

where $P(t_i|t_{i-1}\ t_{i-2})$ and $P(w_i|t_i)$ are computed from the corpus.

3.2. Grapheme-to-Phoneme

In this step, every word from step 3.1 is converted from grapheme to phoneme, and then counts the number of syllables in a word. This step can be ignored, in case of excluding the features of the number of syllables.

3.3. Predicting Phrase Breaks

After finishing step 3.2, the input paragraph is separated into words. Each word is assigned with the number of syllables and an appropriate POS tag by using probabilistic trigram model. Then we form features surrounding every juncture that is a point between 2 words. Finally, C4.5 or RIPPER will predict a juncture is whether a phrase break or not.

4. Experiments and Results

4.1. Data

To test the performance of our approach, 1,169 sentences from ORCHID corpus[9] are selected. The data contains 12,538 words and 1,960 phrase breaks. Every sentence is manually separated into words, and assigned their parts-of-speech tags, pronunciation and phrase breaks by linguists. The data is divided into two parts; the first part, containing 10,059 words and 1,596 phrase breaks, is utilized for training.

The rest, consisting of 2,479 words and 364 phrase breaks, is used for testing.

4.2. Training Methodology

To train the algorithms, we construct both positive and negative examples.

- Positive examples: we construct positive examples by forming collocation and syllable features of junctures which are phrase breaks.
- Negative examples: we create negative examples similar to positive examples but we change a target juncture to a juncture which is not a phrase break.

4.3. Performance Criteria

The performance criteria that used in this work are derived from [11]. We measure the performance of our approach in terms of the percentage of breaks-correct, junctures-correct and false-break. These measures are explained as follows:

Breaks-correct = (CB / RB) x 100%
Junctures-correct = (CS / RS) x 100%
False-break = (FB / RS) x 100%

where

CB is the number of correct classified phrase breaks from the test set.
FB is the number of false classified phrase breaks from the test set.
CS is the number of correct classified phrase breaks and non-phrase breaks from test set.
RB is the number of phrase breaks from the reference.
RS is the number of phrase breaks and non-phrase breaks from the reference.

The difference between the breaks-correct and junctures-correct is that junctures-correct includes the non-phrase breaks in calculation. The junctures-correct score consider on accuracies of both phrase breaks and non-phrase breaks, while the breaks-correct score only accounts on the phrase breaks. In indicating the accuracy of identifying phrase break, both breaks-correct and junctures-correct score are investigated. In our test set, the ratio of the number of non-phrase breaks by the number of phrase breaks is about 5:1. The measures are used to evaluate on the phrase-break accuracy only. Therefore, if an algorithm classifies all junctures as phrase breaks then the break-correct is 100%, and the junctures-correct is about 17%. False-break is the assessment of the insertions, this score indicate how often an algorithm returns the unreliable phrase breaks. In conclusion, the good algorithm must yield high junctures-correct and break-correct scores, and low false-break scores.

4.4. Results

We design the experiments to investigate the impacts of the two major factors; 1) window sizes and 2) syllable features. Then we evaluate our approach with different window sizes of collocations from 1 to 10. To evaluate an impact of syllable features, we test our approach by comparing the accuracy results that consider only collocation features and the accuracy results that employ both collocation and syllable features.

In Table 1 to 3, Size stands for a window size. Coll. and Coll + Syl. represent the algorithm that employs only collocation features, or collocation features and syllable features, respectively.

In table 1 to 3, most of results show that using both features, collocation and syllable features, give higher accuracy results than considering only collocation features. The various window sizes of collocations have an effect on the accuracy results. However, the accuracy results are not quite different. If the longer window size is applied, it cannot guarantee that we will get the better accuracy results. The accuracy results of RIPPER outperform the accuracy result of C4.5 in Table 1 and 3.

Table 4 shows the results of detecting phrase breaks by using POS sequence model which is described by Taylor and Black [11].

5. Discussion

From the experimental results, C4.5 and RIPPER give higher breaks-correct scores and junctures-correct scores than the scores of POS sequence model, whereas false-break scores of C4.5 and RIPPER are lower than the scores of POS sequence model. It means that C4.5 and RIPPER outperform POS sequence model. When C4.5 and RIPPER employed syllable features, the accuracy results of such algorithms increased. It is because syllable features have an effect on a decision whether a juncture is a phrase break or not.

The accuracy results from various sizes of collocations are not quite different. It is because only words and/or POS tags before and after a target juncture have an effect on identifying phrase breaks.

The experimental results show that C4.5 and RIPPER give better accuracy than POS sequence model, because POS sequence cannot capture words and syllable features to solve the problem of phrase break identification. C4.5 and RIPPER show a good performance in extracting both useful collocation and syllable features. Moreover, the accuracy results of RIPPER are better than the accuracy results of C4.5. It means that RIPPER is the best algorithm in solving our problem.

Table 1: Breaks-correct score

Size	C4.5 (%)		RIPPER (%)	
	Coll.	Coll.+Syl.	Coll.	Coll.+Syl.
1	71.70	71.98	**80.21**	**84.60**
2	71.15	71.70	78.48	83.24
3	72.25	73.63	79.12	82.42
4	72.53	73.90	76.92	78.85
5	72.53	74.18	74.18	75.55
6	72.53	73.90	76.65	79.95
7	72.25	73.90	72.80	80.77
8	71.70	73.35	75.82	83.24
9	71.70	73.63	74.45	82.97
10	71.70	73.08	71.15	81.59

Table 2: Junctures-correct score

Size	C4.5 (%)		RIPPER (%)	
	Coll.	Coll.+Syl.	Coll.	Coll.+Syl.
1	93.42	93.63	90.56	91.57
2	93.47	93.55	90.92	93.14
3	93.55	93.67	91.89	93.59
4	93.26	93.30	92.46	93.26
5	93.22	93.38	**93.67**	93.51
6	93.10	93.30	93.83	93.06
7	**95.44**	93.26	92.82	93.34
8	93.22	93.42	92.98	**94.15**
9	93.22	**93.79**	93.30	93.26
10	93.10	93.38	92.05	92.66

Table 3: False-break score

Size	C4.5 (%)		RIPPER (%)	
	Coll.	Coll.+Syl.	Coll.	Coll.+Syl.
1	4.15	4.11	**2.44**	2.99
2	4.24	4.15	3.59	**2.46**
3	4.07	3.87	3.07	2.58
4	**4.03**	**3.83**	3.39	3.11
5	**4.03**	3.79	3.79	3.59
6	**4.03**	**3.83**	3.43	2.94
7	4.07	**3.83**	3.99	2.82
8	4.15	3.91	3.55	**2.46**
9	4.15	3.87	3.75	2.50
10	4.15	3.95	4.24	2.70

Table 4: the accuracy of POS sequence model

Breaks-correct	Junctures-correct	False-break
69.75%	89.04%	7.97%

6. Conclusion

This paper applies machine learning techniques, C4.5 and RIPPER, to the task of predicting phrase breaks. The experimental results show that C4.5 and RIPPER have ability to capture useful features in solving the problem. Moreover, C4.5 and RIPPER give the better accuracy results than previous POS sequence model. The accuracy results of C4.5 and RIPPER are not quite different. Nevertheless, RIPPER appears to outperform C4.5. Considering both collocation and syllable features give the better results than considering only collocation features. The windows size of collocation has only little impact on the accuracy results.

In the future experiment, we will apply other machine learning techniques and investigate different types of features.

7. Acknowledgement

We would like to thank Mr. Teerapong Modhiran, an intern student from Department of Computer Engineering, King Mongkut's Institute of Technology Ladkrabang, for his help in experiment with C4.5 and RIPPER.

8. References

[1] Byeongchang Kim and Geunbae Lee (2000). “Decision-tree based error correction for statistical phrase break prediction in Korean”, *in Proc. of Coling.* 2000.

[2] Byeongchang Kim and Geunbae Lee. “Statistical/Rule-based Hybrid Phrase Break Detection”, *in Proc. of ICSP.* 1999.

[3] Cohen, William W. Fast effective rule induction, *In Proceedings of the Twelfth International Conference on Machine Learning*, Lake Taho, California, Morgan Kauffman. 1995.

[4] Eric Sanders and Paul A. Taylor (1995). “Using Statistical Models to Predict Phrase Boundaries for Speech Synthesis”, *in Proc. of Eurospeech.* 1995.

[5] Meknavin, Surapant, Charoenpornsawat, Paisarn. and Kijsirikul, Boonserm. 1997. Feature-based Thai Word Segmentation. *Proceeding of NLPRS.* 1997.

[6] Navas, I. Hernaez, and N. Ezeiza. “Assigning Phrase Breaks using CART's in Basque TTS”, *in Proc. of the 1st Int. Conf. on Speech Prosody, Aix-en-Provence.*2002.

[7] Palmer, David D., Hearst, Mati A. “Adaptive Multilingual Sentence Boundary Disambiguation”. *Computational Linguistics Volumn 23 No. 2.* 1997.

[8] Quinlan, Ross. C4.5: Programs for Machine Learning Morgan Kauffman. 1993.

[9] Sornlertlamvanich, Virach, Charoenporn, Thatsanee. and Isahara, Hitoshi. ORCHID: Thai Part-Of-Speech Tagged Corpus. *Technical Report Orchid Corpus*. National Electronics and Computer Technology Center. 1997.

[10] Sornlertlamvanich, Virach, Potipiti, Tanapong and Charoenporn, Thatsanee. *Automatic Corpus-based Thai Word Extraction with the C4.5 Learning Algorithm*. Proceedings of the 18th International Conference on Computational Linguistics, Saarbrucken, Germany, pp 802-807, 2000.

[11] Taylor, Paul. and Black, Alan. Assigning Phrase Breaks from part-of-speech Sequences. Computer Speech and Language 12. 1998.

[12] William Cohen & Yoram Singer, *Context-sensitive learning methods for text categorization*, in *ACM Transactions on Information Systems}*, v17 pp 171-173 1999.