A Text Analyzer for Korean Text-to-Speech Systems

Sangho Lee, Yung-Hwan Oh

Department of Computer Science
Korea Advanced Institute of Science and Technology,
373-1 Kusong-dong Yusong-gu, Taejon 305-701, Korea
shlee@bulsai.kaist.ac.kr

ABSTRACT

In developing a text-to-speech system, it is well known that the accuracy of information extracted from a text is crucial to produce high quality synthesized speech. In this paper, by transferring probabilistic natural language processing techniques into TTS system field, we develop a more robust text analyzer with high accuracy for Korean TTS systems. The proposed system is composed of five modules: a preprocessor, a morphological analyzer, a part-of-speech tagger, a grapheme-to-phoneme module, and a parser. Among these modules, the part-of-speech tagger and the parser are designed under probabilistic framework, and trained automatically. Given a text, our system produces the structures of word phrases, word pronunciations, and governor-dependent relationships that represents the structure of the sentence. Experimental results showed that the tagger got 90.33% correctness for finding the structure of word phrases in the word level, and the parser, 80.87% for finding governor-dependent relationships of sentences respectively.

1. INTRODUCTION

A text-to-speech (TTS) system is a program that inputs a stream of text and outputs a speech signal. The goal is to develop a program which reads a text as if it were a skilled speaker who had understood the text[1]. To achieve this goal, the program generally extracts as much informations as possible from a text, and synthesizes speech signals by using this information. This paper primarily discusses the first phase of the TTS system, text analysis, which is important to produce high quality synthesized speech.

Until now, most existing Korean text analyzers have been developed based on simple strategies. For example, they interpret only one morphological analysis by utilizing the longest-matching method without a tagging procedure, and gets word pronunciations by using only letter contexts. This is prone to produce word pronunciations and syntactic structures incorrectly[2, 3].

By transferring probabilistic natural language processing techniques into text-to-speech system field[4], we are trying to develop a system that extracts more accurate information from a text. The proposed system consists of five modules: a preprocessor, a morphological analyzer, a part-of-speech tagger, a grapheme-to-phoneme module, and a parser. Among these modules, the tagger and the parser is designed under probabilistic framework, which is currently widely used in natural language processing field.

Given a text, the preprocessor converts non-Korean letters into Korean letters by simulating nondeterministic finite automata, and makes a sequence of sentences for the following modules. The morphological analyzer and the tagger produce a sequence of morphemes with their parts-of-speech, which will be assigned to each phoneme at grapheme-to-phoneme phase. The grapheme-to-phoneme module generates phoneme strings by using three peak-triggering morphophonemic rules and 27 coda-triggering rules, because all the morphophonemic rules of Korean are triggered by peaks and codas. The last component of our system, the parser, produces governor-dependent relationships. Since Korean has governor post-positioning property, the parser was easily implemented by using restricted Chomsky normal-form rules.

This paper is organized as follows: Five components of the system will be explained in section 2, and the system will be evaluated in section 3 by experimenting the tagger and the parser. In the last section, some conclusions will be drawn.

2. THE COMPONENTS OF THE SYSTEM

Given a text, our system calls the preprocessor, the morphological analyzer, the tagger, the grapheme-to-phoneme module, the parser in order, and outputs pronunciations and the structures of all the word phrases, and a dependency tree.

2.1. Preprocessor

The function of a preprocessor for TTS systems is to divide a text into sentences and convert symbols into words. To perform preprocessing, our preprocessor uses nondeterministic finite automata, of which the number of states is 48, and

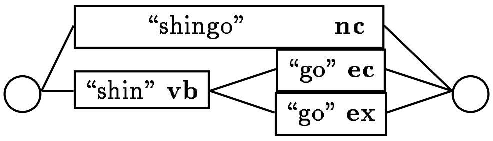

Figure 1: The lattice representation of the morphological analysis of "shingo"

12 of them are final states. All Korean and English letters including symbols, and digits, can be symbols for the automata, and while the automata scans a text, if the current state is a final state, the automata calls the corresponding function which converts the input stream into Korean letters.

The main functions of final states are interpreting times, digits in Korean and Chinese style, telephone numbers, years, etc. Among these, since digit-reading style is dependent on what words follow, the preprocessor looks for the following symbols ahead.

2.2. Morphological Analyzer

Since almost all Korean word phrases[1] can be interpreted differently in a sentence, we need to get all the possible interpretations at the morphological analysis phase.

To represent all the interpretations in a compact way, we analyze each word phrase into word (morpheme) lattice. For example, let us assume that the ith word phrase w_i is "shingo," then the morphological analyzer produces the word lattice as shown in Figure 1[2]. While the upper path of the lattice means *report*, the rest paths mean *to put on*, and each path of them is different from the other grammatically.

To make a word lattice, the morphological analyzer uses the algorithm suggested by Sangho Lee *et. al.*[5]. The algorithm analyzes an input phrase by scanning from the right side of the phrase. Since the major mechanism of Korean word combination is affixation (e.g., nominal stem + postpositions, verbal stem + verb endings, ... etc.), postpositions and various endings appear at the right side of a phrase. Furthermore, since the postpositions and endings can be classified as a closed word class (i.e., there are only limited number of words in a closed word class.), the system can find postpositions or endings even when unknown nouns or verbs appear in an input word phrase. By utilizing these characteristics of Korean, the morphological analyzer can always produce a word lattice by guessing unknown words based on the partial word lattice.

2.3. Part-of-Speech Tagger

Given a sequence of word lattices, the part-of-speech tagger selects the most probable morphological analyses which are described by a sequence of morpheme sequences and a sequence of part-of-speech sequences. This is quite different from English tagging systems which produce only a sequence of tags for an input sentence[6]. Since Korean is a highly inflected agglutinative language like Finnish and Japanese, we can't process a sentence further with only the tags of word phrases. This implies that Korean tagger should extract more information, i.e., morphological analysis that represents the structure of a word phrase.

The proposed tagger is implemented based on hidden Markov model[6], and modified for handling unknown words. Before getting tagging formula, let's define some notations which are used in the formula. w_i is ith word phrase in a sentence and it's morphological analysis is described by using a pair (m_i,t_i), such that m_i, t_i are a morpheme sequence, a tag sequence respectively. Assuming that N_i is the length of the morpheme sequence of ith word phrase, m_i and t_i can be expanded into $m_{i^1},\ldots,m_{i^{N_i}}$, $t_{i^1},\ldots,t_{i^{N_i}}$. In terms of English, if w_i is "tries (try/verb + s/suffix)," m_i is $m_{i^1}m_{i^2}$ = (try,s), t_i is $t_{i^1}t_{i^2}$ = (verb,suffix), and N_i is 2, the length of the sequence. The tagging formula can then be formally defined as finding the sequence of morphological analyses which is the result of the following function:

$$\begin{aligned}\phi(w_{1..n}) &\overset{\text{def}}{=} \arg\max_{m_{1..n},t_{1..n}} P(m_{1..n},t_{1..n}|w_{1..n}) &(1)\\ &\cong \arg\max_{m_{1..n},t_{1..n}} \prod_{i=1}^{n} P(m_i|t_i)P(t_i|t_{i-1}) &(2)\end{aligned}$$

Equation 2 is derived by applying Bayes' rule and Markov assumption. Now we can break m_i and t_i into $m_{i^1}m_{i^2}\ldots m_{i^{N_i}}$ and $t_{i^1}t_{i^2}\ldots t_{i^{N_i}}$ respectively, and rewrite $P(t_i|t_{i-1})$ as the following:

$$P(t_i|t_{i-1}) \cong P(t_{i^1}|t_{{i-1}^{N_{i-1}}})\prod_{j=2}^{N_i} P(t_{i^j}|t_{i^{j-1}}) \qquad (3)$$

Equation 3 is derived by assuming that the tag sequence of a current word phrase is dependent on the tag of the last morpheme of the preceding word phrase and by applying the chain rule and Markov assumption.

Before moving to $P(m_i|t_i)$, we need to define a random variable k_{i^1}, which means that the first morpheme of the word is an entry of the dictionary or is not and entry, i.e., if k_{i^1} is 1, then the morpheme m_{i^1} is registered as an entry, if not (k_{i^1} = 0), considered as an unknown word. Using this variable k_{i^1} for handling unknown words, we can rewrite $P(m_i|t_i)$ as the following:

$$\begin{aligned}P(m_i|t_i) &= P(m_i,k_{i^1}|t_i) &(4)\\ &\cong P(k_{i^1}|t_i)[k_{i^1}P(m_i|t_i,k_{i^1}=1)+ \\ &\qquad (1-k_{i^1})P(m_i|t_i,k_{i^1}=0)] &(5)\end{aligned}$$

[1] Word phrase in this paper means *eojeol* in Korean, which is like *bunsetsu* in Japanese.

[2] The part-of-speech of a morpheme is represented by two English letters. In this figure, **nc**, **vb**, **ec**, **ex** represent common noun, verbal stem, conjunctive ending, auxiliary conjunctive ending respectively.

Table 1: The pronunciations of "shingo" corresponding to it's parts-of-speech

syllable	shin						go					
letter	sh		i		n		g		o			
phoneme	sh		i		n		*kk*		o			
part-of-speech	**verbal stem**						**ending**					
syllable	shin			go			ha			rŏ		
letter	sh	i	n	g	o		h	a		r	ŏ	
phoneme	sh	i	n	*g*	o		h	a		r	ŏ	
part-of-speech	**noun**						**suffix**			**ending**		

$$\cong \quad P(k_{i^1}|t_{i^1})[k_{i^1}\prod_{j=1}^{N_i}P(m_{ij}|t_{ij}) + (1-k_{i^1})P(t_{i^1}|t_{i^2}m_{i^2})\prod_{j=2}^{N_i}P(m_{ij}|t_{ij})] \quad (6)$$

Equation 4 follows because we can always know if m_{i^1} is known or not. In Equation 5 we rewrite the different cases $k_{i^1}=0$ and $k_{i^1}=1$ into one form by making the other term always zero. By using the chain rule and Markov assumption, we can get Equation 6, of which term $P(t_{i^1}|t_{i^2}m_{i^2})$ is made by replacing m_{i^1} with the tag of m_{i^1}, since we can't get the information about the unknown word m_{i^1}. Based on these formulas, the tagging module selects the most probable sequence of morphological analyses among the multiple results by using Viterbi algorithm[7].

2.4. Grapheme-to-Phoneme Module

Compared with English, while the writing system of Korean is not also phonetically transparent, most pronunciations can usually be obtained from orthographic representation by applying morphophonemic rules deterministically. However, since there are many Korean words, such as the nouns which come from Chinese, that can not be covered by rules. Therefore we need to look up these words in the dictionary.

The grapheme-to-phoneme module of our system is composed of three phases: handling irregularly-pronounced words, labeling tags to each letters, and applying morphophonemic rules. Given an output of the tagger, this module searches the pronunciation cache, which is filled when the morphological analysis is performed. After obtaining pronunciations of irregularly-pronounced words, the module performs assigning tag to each letter, and with scanning all the peaks and codas from left to right, calls the routine corresponding to each peak and coda. Korean morphophonemic rules are triggered by only three peaks ("yŏ, ye, ŭi"), all the 27 codas with their tags, and the tags of following letters. We simply coded these 30 rules into 30 functions, of which input parameters are letters and their parts-of-speech.

To recognize the importance of the tag of the letter, let's take two sentences as examples, "nanŭn shinŭl shingo gatta. (I put on shoes, and I went.)," and "nanŭn shingoharŏ gatta. (I went to report.)." In the two sentences, while two "shingo"s are written in the same spelling, because the "shin" of the first "shingo" is a verbal stem and "go" is an ending, "go" must be pronounced "kko" as shown in Figure 1.

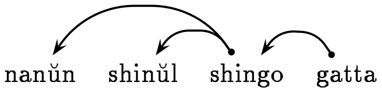

Figure 2: The dependency tree of "nanŭn shinŭl shingo gatta."

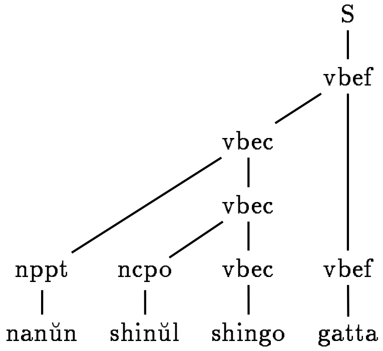

Figure 3: The phrase-structure tree of "nanŭn shinŭl shingo gatta."

2.5. Parser

In addition to the modules which we have described so far, we implemented the probabilistic parser, which generates the most probable dependency tree. Dependency tree is defined as a set of governor-dependent relationships. Since it can be represented as a phrase-structure tree, the parser can parse a sentence using best-parse parsing method[8]. For example, dependency tree and phrase-structure tree version of the sentence "nanŭn shinŭl shingo gatta." are shown in Figure 2 and 3 respectively.

Due to the characteristics of Korean, the governor post-positioning property, parsing can be performed using only the restricted Chomsky normal-form rules like the following:

1) $\mathbf{S} \rightarrow \mathbf{A}$
2) $\mathbf{A} \rightarrow \mathbf{B}\ \mathbf{A}$
3) $\mathbf{A} \rightarrow \mathbf{a}$

where **S** is a start symbol, **A, B** are the parts-of-speech of word phrases (the sequence of parts-of-speech of morphemes), and **a** is a word phrase.

In this research, since we only need to find the parse tree and not it's probability, we assign 1.0 for rule probabilities to all the rules which are type 1 or 3. This is done by utilizing the

Table 2: The performance of the tagger

	word phrase level	word level
total word phrase	85.11 %	90.33 %
known word phrase	88.03 %	93.46 %
unknown word phrase	62.10 %	65.25 %

fact that the last word phrase in a Korean sentence always becomes the head word phrase of the sentence (rule 1). Since only the best parts-of-speech of word phrases are obtained at the tagging phase, the probability that a word phrase is generated is considered to be constant at the parsing phase (rule 3). Hence the probability of the tree in Figure 3 is obtained by multiplying P(vbef → vbec vbef), P(vbec → nppt vbec), and P(vbec → ncpo vbec).

3. EXPERIMENTAL RESULTS

In this section, we evaluate our system in terms of the accuracies of the tagger and the parser.

3.1. Part-of-Speech Tagger

The part-of-speech tagger was trained with a corpus of 49,506 word phrases (116,031 words) and tested with a corpus of 4,729 word phrases. Among the corpus of 49,506 word phrases, 45,851 word phrases were used to get conditional probabilities of $P(t_{ij}|t_{ij-1})$, $P(t_{i^1}|t_{{i-1}^{N_{i-1}}})$, $P(m_{ij}|t_{ij})$, $P(t_{i^1}|t_{i^2}m_{i^2})$, and to build the dictionary. To obtain the conditional probability of $P(k_{i^1}|t_{i^1})$, the corpus of 3,652 word phrases were used.

The performance of the tagger is shown in Table 2. According to the table, the correctness ratio of the tagger is a little bit lower than that of the ordinary English tagger. This is due to the error of the morphological analyzer. In Figure 1, it can be possible that while "shingo" is used with the meaning of the upper path in a given sentence, "shingo" is not registered in the dictionary. In this case, the morphological analyzer constructs the lattice which contains only the lower paths of Figure 1, and the tagger chooses the path among the wrong interpretations, which degrades the performance of the tagger. Hence in the experiment with no unknown words, we got 96.46% correctness at the word phrase level, 98.01% at the word level. Therefore, we expect that we can get much better performance if we have a dictionary which contains a large vocabulary.

3.2. Parser

To evaluate the parser, we trained the parser with a hand-bracketed corpus of 498 sentences, and tested the parser with a corpus of 100 sentences. In the first experiment, we evaluated only the parser by making outputs of the tagger perfect. The parser yielded 80.87% correctness for finding governor-dependent relationships. In addition to this experiment, we also tested the parser with outputs of the tagger, and got 78.68% correctness.

4. CONCLUSIONS

In this paper, we have presented a text analyzer for Korean text-to-speech systems, which is composed of five modules: a preprocessor, a morphological analyzer, a part-of-speech tagger, a grapheme-to-phoneme module, and a parser. Among these modules, the tagger and the parser are implemented based on probabilistic formalism. The performance of our system was evaluated by testing the tagger and the parser. The former yielded 90.33% correctness in the word level, and the latter, 80.87% correctness for finding governor-dependent relationships of a sentence.

5. REFERENCES

1. Liberman, Mark Y. and Church, Kenneth W., "Text Analysis and Word Pronunciation in Text-to-Speech Synthesis," *Advances in Speech Signal Processing*, Marcel Dekker Inc., 1992.
2. Hong, SungHoon, Jeon, BumKi, Hong, JoonMo, Lee, JooHun, Rheem, JaeYeol, Lim, HeungSoon and Ann, Souguil, "A Study on the Language Processing in Korean Text-to-Speech System," *Proc. of the 10th Workshop on Speech Communication and Signal Processing Workshop*, pp. 99-103, 1993(in Korean).
3. Kang, Yong-Bum and Kim Jin-young, "A Preprocessing for the Unlimited Korean Text to Speech System," *Proc. of the 11th Workshop on Speech Communication and Signal Processing Workshop*, pp. 334-337, 1994(in Korean).
4. Charniak, Eugene, *Statistical Language Learning*, The MIT Press, 1993.
5. Lee, Sangho, Kim, Jae-Hoon, Cho, Jeong Mi and Seo, Jungyun, "Korean Morphological Analysis Sharing Partial Analyses," *Proc. of the 11th Workshop on Speech Communication and Signal Processing Workshop*, pp. 75-79, 1994(in Korean).
6. Church, Kenneth W., "A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text," *Proc. of Applied Natural Language Processing*, Austin, Texas, 1988.
7. Rabiner, L. R., "A Tutorial on Hidden Markov Models and Selected Application in Speech Recognition," *Proc. of the IEEE*, Vol. 77, no. 2, pp. 257-286, Feb. 1989.
8. Allen, J., *Natural Language Understanding*, The Benjamin/Cummings Publishing Company, Inc., 1995.