Identifying Temporal Expression and its Syntactic Role Using FST and Lexical Data from Corpus

Juntae Yoon
jtyoon@daumcorp.com
Daum Communications Corp.
Kangnam-gu, Samsung-dong, 154-8
Seoul 135-090, Korea

Yoonkwan Kim Mansuk Song
{general,mssong}@december.yonsei.ac.kr
Dept. of Computer Science, Engineering College
Yonsei Univ.
Seoul 120-749, Korea

Abstract

Accurate analysis of the temporal expression is crucial for Korean text processing applications such as information extraction and chunking for efficient syntactic analysis. It is a complicated problem since temporal expressions often have the ambiguity of syntactic roles. This paper discusses two problems: (1) representing and identifying the temporal expression (2) distinguishing the syntactic function of the temporal expression in case it has a dual syntactic role. In this paper, temporal expressions and the context for disambiguation which is called *local context* are represented using lexical data extracted from corpus and the finite state transducer. By experiments, it turns out that the method is effective for temporal expression analysis. In particular, our approach shows the corpus-based work could make a promising result for the problem in a restricted domain in that we can effectievely deal with a large size of lexical data.

1 Introduction

Accurate analysis of the temporal expression is crucial for text processing applications such as information extraction and for chunking for efficient syntactic analysis. In information extraction, a user might want to get a piece of information about an event. Typically, the event is related with date or time, which is represented by temporal expression.

Chunking is helpful for efficient syntactic analysis by removing irrelevant intermediate constituents generated through parsing. It involves the task to divide sentences into non-overlapping segments. As a result of chunking, parsing would be a problem of analysis inside chunks and between chunks (Yoon, et al., 1999). Chunking prevents the parser from producing intermediate structures irrelevant to a final output, which makes the parser efficient without losing accuracy. Thus, it turns out that chunking is an essential stage for the application system like MT that should pursue both efficiency and precision.

Korean, an agglutinative language, has well-developed functional words such as postposition and ending by which the grammatical function of a phrase is decisively determined. Besides, because it is a head final language and so the head always follows its complement, the chunking is relatively easy. However, we are also faced with an ambiguity problem in chunking, which is often due to the temporal expression. This is because many temporal nouns are used as the modifier of noun and verb in a sentence. Let us consider the following examples:

[**Example**]

1a *jinan*(last) *yeoreum*(summer) *uri-neun* (we/NOM) *hamgge*(together) *san-e*(to mountain) *gassda*(went)
→ We went to the mountain together last summer.

1b *jinan*(last) *yeoreum*(summer) *banghag-e*(in vacation) *uri-neun*(we/NOM) *hamgge*(together) *san-e*(to mountain) *gassda*(went)
→ We went to the mountain together in the last summer vacation.

2a *10 weol*(October) *9 il*(9th) *jeonyeog*(evening) *7 si*(7 o'clock) *daetongryeong-yi* (president/GEN) *damhwa-ga*(talk/NOM) *issda*(be)
→ The president will give a talk at 7:00pm in Oct. 7th.

2b *10 weol*(October) *9 il*(9th) *jeonyeog*(evening) *7 si*(7 o'clock) *bihaenggipyo-reul*(flight ticket/ACC) *yeyaghal su issseubnigga*(can reserve)
→ Can I reserve the flight ticket for 7:00pm in Oct. 7?

In the examples, each temporal expression plays a syntactically different role used as noun phrase or adverbial phrase (The underlined is a phrase) although they comprise the same phrasal forms. The temporal expressions in 1a and 2a of the example serve as the temporal adverb to modify predicates. On the other hand, the temporal expressions in 1b and 2b are used as the modifier of other nouns. That is, as a temporal noun either contributes to construction of a noun compound or modifies a predicate, it causes a structural ambiguity.

One solution might be that the POS tagger assigns a different tag to each temporal noun e.g. NN and ADV. However, since dependencies of temporal nouns are lexically decided, it does not seem that their syntactic tags could be accurately predicted with a relatively small size of POS tagged corpus. Also, the simple rule based approach cannot make satisfactory results without lexical information. As such, identification of temporal expression is a complicated problem in Korean text analysis.

This paper discusses identification of temporal expressions and their syntactic roles. In this paper, we would deal with two problems: (1) representing and identifying the temporal expression (2) distinguishing its syntactic function in case it has a dual syntactic role. Actually, the two problems are closely related since the identification and disambiguation process would be done under the representation scheme of temporal expression. The process bases on lexical data extracted from corpus and the finite state transducer (FST). According to our observation of texts, we could see that a few words following a temporal noun have great effect on the syntactic function of the temporal noun. Therefore, we note that the structural ambiguity could be resolved in local contexts, and so obtain lexical information for the local contexts from corpus. The lexical data which contain contexts for disambiguation are represented with temporal word transition over the FST.

Briefly describing our methodology, we first extract concordance data of each temporal word using a concordance program. The co-occurrences represent relations between temporal words and also explain how temporal nouns and common nouns are combined to generate a compound noun. It would be the likelihood of word combination, which helps disambiguate the syntactic role if a temporal word have a syntactic duality. In particular, we classify temporal nouns into 26 classes in accordance with their meaning and function. Thus, the word co-occurrences become those among temporal classes or temporal classes and other nouns, which results in reducing the parameter space. Second, temporal expressions containing the co-occurrences of temporal classes and other nouns are represented with the FST to identify temporal expressions and assign their syntactic tags in a sentence. It has been shown that the FST presents a very efficient way for representing phrases with locality. The input of the FST is the result from morphological analysis and POS tagging (here, the temporal noun is tagged only as noun). Its output is the syntactic tag for each word in the sentence and temporal words are attached tags such as noun and adverb. Figure 1 shows the overall system from the morphological analyzer to the chunker.

Therefore, the process attaches syntactic labels to the previous examples so that chunking would be safely executed from the results as follows:

[**Example**]

1a′ [*jinan*(last) *yeoreum*(summer)]$_{TA}$ *uri-neun* (we/NOM) *hamgge*(together) *san-e*(to mountain) *gassda*(went)

→ We went to the mountain together last summer.

1b′ [*jinan*(last) *yeoreum*(summer)]$_{TN}$ *banghag-e*(in vacation) *uri-neun*(we/NOM) *hamgge* (together) *san-e*(to mountain) *gassda*(went)

→ We went to the mountain together in the last summer vacation.

2a′ [*10 weol*(October) *9 il*(9th) *jeonyeog*(evening) *7 si*(7 o'clock)]$_{TA}$ *daetongryeong-yi* (president/GEN) *damhwa-ga*(talk/NOM) *issda*(be)

→ The president will give a talk at 7:00pm in Oct. 7th.

2b′ [*10 weol*(October) *9 il*(9th) *jeonyeog*(evening) *7 si*(7 o'clock)]$_{TN}$ *bihaenggipyo-reul*(flight ticket/ACC) *yeyaghal su issseubnigga*(can reserve)

→ Can I reserve the flight ticket for 7:00pm in Oct. 7?

2 Related Works

Abney (1991) has proposed text chunking as a preliminary step to parsing on the basis of psychological evidence. In his work, the chunk was defined as a partitioned segment which corresponds in some way to prosodic patterns. In addition, complex attachment decisions as occurring in NP or VP analysis are postponed without being decided in chunking. Ramshaw and Marcus (1995) introduced a baseNP which is a non-recursive NP. They used transformation-based learning to identify non-recursive baseNPs in a sentence. Also, V-type chunk was introduced in their system, and so they tried to partition sentences into non-overlapping N-type and V-type chunks. Yoon, et al. (1999) have defined chunking in various ways for efficient analysis of Korean texts and shown that the method is very effective for practical application.

Besides, there have been many works based on the finite state machine. The finite state machine is often used for systems such as speech processing, pattern matching, POS tagging and so forth because of its efficiency of speed and space and its convenience of representation. As for parsing, it is not suitable for full parsing based on the grammar that has recurrent property, but for partial parsing requiring simple state transition. Roche and Schabes (1995) have transformed the Brill's rule based tagger to the optimized deterministic FST and improved the speed and space of the tagger. A notable one related to this work is about *local grammar* presented in Gross (1993), which is suitable for representing

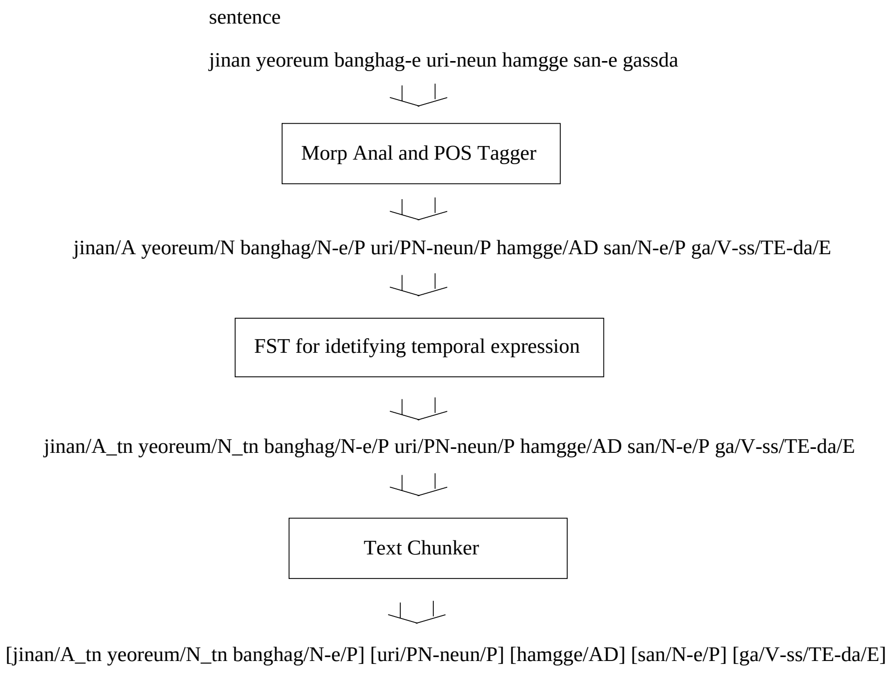

sentence
jinan yeoreum banghag-e uri-neun hamgge san-e gassda
Morp Anal and POS Tagger
jinan/A yeoreum/N banghag/N-e/P uri/PN-neun/P hamgge/AD san/N-e/P ga/V-ss/TE-da/E
FST for idetifying temporal expression
jinan/A_tn yeoreum/N_tn banghag/N-e/P uri/PN-neun/P hamgge/AD san/N-e/P ga/V-ss/TE-da/E
Text Chunker
[jinan/A_tn yeoreum/N_tn banghag/N-e/P] [uri/PN-neun/P] [hamgge/AD] [san/N-e/P] [ga/V-ss/TE-da/E]

Figure 1: System overview from the morphological analyzer from the chunker

rigid phrases, collocations and idioms unlike global grammar for describing sentences of a language in a formal level. The temporal expression was represented with local grammar in his work, where it was claimed that the formalism of finite automata could be easily used to represent them.

3 Acquiring Co-occurrence of Temporal Expression

3.1 Categorizing Temporal Nouns

Since many words have in common a similar meaning and function, they can be categorized by their features. So do temporal nouns. That is, we say that 'Sunday' and 'Monday' have the same features and so would take the similar behavior patterns such as co-occurring with the similar words in a sentence or phrase. Hence, in the first place we categorize temporal nouns according to their meaning and function. We first select 259 temporal nouns and divide them into 26 classes as shown in Table 1. Among them, some temporal words have syntactic duality and others play one syntactic role. Thus, the disambiguation process would be applied only to the words with dual syntactic functions.

3.2 Acquisition of Temporal Expressions from Corpus

Temporal words would be combined with each other in order to be made reference to time, which is called *temporal expression*. Since a temporal expression is typically composed of one or a few temporal words, it seems to be possible to describe a grammar of

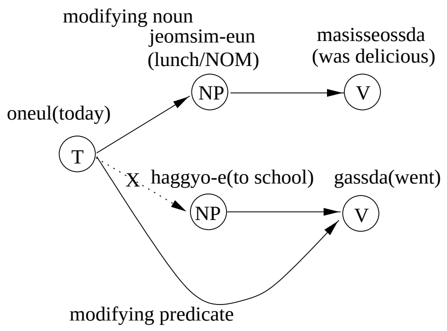


Figure 2: Syntactic functional ambiguity of temporal expression

the temporal expression with a simple model like finite automata. In the practical system, however, we are confronted with a complicated problem in treating temporal expressions since many temporal words have a functional ambiguity used as both a nominal and predicate modifier. For instance, a temporal noun *oneul*(today) could play a different role in the similar situation as shown in Figure 2. In the first and the second path, the words to follow *oneul* are all noun, but the roles (dependency relations) of *oneul* are different.

Accurate classification of their syntactic functions is crucial for the application system since great difference would be made according to accuracy of the dependency result. Practically, we therefore should take into consideration the structural ambiguity resolution as well as their representation itself in identi-

word category			class #	temporal words
temporal prefixes	modifier		1	*ol*(this), *jinan*(last), ...
	number		2	number ...
temporal nouns	temporal unit		3–10	*segi*(centry), *nyeon*(year), ...
	era, age		11	*gosaengdae*(Paleozoic), ...
	years		12	*geumnyeon*(this year), *saehae*(new year), ...
	months		13	*naedal*(next month), *jeongwol*(January), ...
	weeks		14	*geumju*(this week), *naeju*(next week), ...
	days	day of week	15	*ilyo'il*(Sunday), *wolyo'il*, ...
		day1	16–17	*haru*(one day), *chuseog*(Thanksgiving day), ...
		day2	18	*oneul*(today), *nae'il*(tomorrow), ...
	time and duration	time1	19	*saebyeog*(dawn), *achim*(morning), ...
		time2	20–21	*yeonmal*(year-end), ...
		season	22	*bom*(spring), *yeoreum*(summer), ...
		specific duration	23	*hwanjeolgi*(time of season changing), ...
		edge	24-25	*chogi*(early time), *jungban*(mid), ...
temporal suffixes	temporal suffixes		26	*dongan*(during), *naenae*(through), ...

Table 1: Categorization of temporal words

fying temporal expressions. The point that we note here is that we could predict the syntactic function of temporal words by looking ahead one or two words. Namely, looking at a few words that follows a temporal word we can figure out which word the temporal expression modifies, and call the following words *local context*.

Unfortunately, it is not easy to define the local context for determining the syntactic function of each temporal word because they are lexically related. That is, it is wholly different from each word whether a temporal noun would modify other noun to form a compound noun or modify a predicate as an adverbial phrase. Our approach is to use corpus to acquire information about the local context. Since we could obtain from corpus as many examples as needed, rules for compound word generation can be constructed from the examples. In this paper, we use co-occurrence relations of temporal nouns extracted from large corpus to represent and construct rules for identification of temporal expressions.

As mentioned before, we would pay attention to two points here: (1) In what order a temporal expression would be represented with temporal words, i.e. description of the temporal expression network. (2) how the local context would be described to resolve the ambiguity of the syntactic function of temporal expressions. For this purpose, we first extract example sentences containing each of 259 temporal words from corpus using the KAIST concordance program[1] (KAIST, 1998). The number of temporal words is small and so we could manually manipulate lexical data extracted from corpus. Figure 3 shows example sentences about *yeoreum*(summer) extracted by the concordance program.

[1]KAIST corpus consists of about 50 million *eojeols*. *Eojeol* is a spacing unit composed of a content word and functional words.

Second, we select only the phrases related with temporal words from the examples (Table 4). As shown in Table 4, *yeoreum* is associated with varying words. Temporal words like temporal prefixes can come before it and common nouns can follow it. In this stage we describe contexts of each temporal word and the output (syntactic tag of the temporal word) under the given context. In particular, each temporal word is assigned a temporal class. Besides, other nouns serve as local contexts for disambiguation of syntactic function of temporal words.

From the examples, we can see that if *bam*(night), *byeoljang*(villa), *banghag*(vacation) and so on follows it, *yeoreum* serves as a component of a compound noun with the following word. On the other hand, the word *naenae* which means *all the time* is a temporal noun and forms a temporal adverbial phrase with other preceding temporal noun. Moreover, *yeoreum*(summer) might represent time-related expression with preceding temporal prefixes.

4 Identifying Temporal Expressions and Chunking

4.1 Representing Temporal Expression Using FST

The co-occurrence data extracted by the way described in the previous section can be represented with a finite state machine (Figure 5). For syntactic function disambiguation and chunking, the automata should produce an output, which leads to a finite state transducer. In fact, individual description for each data could be integrated into one large FST and represented as the right-hand side in Figure 5. A finite state transducer is defined with a six-

left context	word	right context
썼고 나중에는 기름에 묻힌 솜을 썼다.	[여름	밤]에 물고기들이 물것로 나올 때
그리고는 소파 위에서 그만 잠이 들었다.	[여름	밤]만큼이나 짧았지만
정권을 빼앗기고 말았다. 나는 처음에는	[여름	방학] 후에 복학을 하기로 했었고
새로운 친구들을 빨리 사귀게 되었다.	[여름	방학] 때에는 부산에 사는
있는 성악 레슨을 받아 보도록 했다.	[여름	방학] 전날 열린 학급 잔치에서
변화되어야 한다고 완곡하게 설득했다.	[여름	방학]이 다것오면서 성근이는 여러
것 같았다. 신학과 3학년 1학기를 마치고	[여름	방학]이 시작된 후 최종 결심을 했다.
터였다) 봄에 시작된 상아의 생활문제는	[여름	방학]을 넘기고 것을로 접어들면서
거부하고 있는 형편이라고 한다. 마침	[여름	방학] 중이라서 시간도 있었고
이 집은 본래 저씨 부인의 친정 숙부의	[여름	별장]이더니, 그것 별세하매 이 집
들여다보자. 보충 수업,	[여름	주말]은 언제나 괴롭다. 매미는
뇌염 예방 주사를 맞히러 온 사람,	[여름	감기]에 잔기침을 해대거나 콧물을
되었다. 것게 일은 한나절이면 되지만	[여름	내내] 피서 한 번 못 것고 것게 일에
있는지 대학 교수들에게 묻고 싶다. [지난	여름]	나는 10여 년의 유학에서 돌아와
정말 다행스럽게 생각한다. 그 [해	여름]	6.25 전쟁이 터졌다. 학교는 피난민
간첩이라니 묘한 인연이구만!" 그 [해	여름]	마지막 늦더위것 기승을 부리는
생각되지 않는군요. [이번	여름]	전투는 누것 이긴거죠?" 이긴

Figure 3: Example concordance data of *yeoreum*(summer)

before	temporal noun *yeoreum*(summer)	after	output	freq
	여름/t_{22}	밤(*bam*,night)	TN	2
	여름/t_{22}	방학(*banghag*,vacation)	TN	7
	여름/t_{22}	별장(*byeoljang*,villa)	TN	1
	여름/t_{22}	주말(*jumal*,weekends)	TN	1
	여름/t_{22}	감기(*gamgi*,flu)	TN	1
	여름/t_{22}	내내/t_{26}(*naenae*,all the time)	TA	1
지난/t_1(*jinan*, last)	여름/t_{22}	나는(*naneun*,I/TOP)	TA	1
해/t_{10}(*hae*,year)	여름/t_{22}	6.25, 마지막(*majimag*,the last)	TA	2
이번/t_1(*ibeon*,this)	여름/t_{22}	전투는(*jeontuneun*,battle/TOP)	TA	1

Figure 4: Temporal expression phrases selected from examples

tuple $(\Sigma_1, \Sigma_2, Q, i, F, E)$ where: Σ_1 is a finite input alphabet; Σ_2 is a finite output alphabet; Q is a finite set of states or vertices; $i \in Q$ is the initial state; $F \subseteq Q$ is the set of final states; $E \subseteq Q \times \Sigma_1^* \rightarrow \Sigma_2^* \times Q$ is the set of transitions or edges.

Although the syntactic function of a temporal expression would be nondeterministically selected from the context, temporal expressions and the lexical data of local context can be represented in a deterministic way due to their finite length. For the deterministic FST, we define the partial functions $\otimes$ and $*$ where $q \otimes a = q'$ iff $d(q, a) = \{q'\}$ and $q * a = w'$ iff $\exists q' \in Q$ such that $q \otimes a = q'$ and $\delta(q, a, q') = \{w'\}$ (Roche and Schabes, 1995). Then, a subsequential FST is a eight-tuple $(\Sigma_1, \Sigma_2, Q, i, F, \otimes, *, \rho)$ where: Σ_1, Σ_2, Q, i and F are the same as the FST; $\otimes$ is the deterministic state transition function that maps $Q \times \Sigma_1$ on Q; $*$ is the deterministic emission function that maps $Q \times \Sigma_1$ on Σ_2^*; $\rho : F \rightarrow \Sigma_2^*$ is the final output function.

$T = (\Sigma_1, \Sigma_2, Q, i, F, \otimes, *, \rho\})$
$\Sigma_1 = \{t_1, t_22, t_26, w_i, w_j\}$
$\Sigma_2 = \{TN, TA, NT\}$
$Q = \{0, 1, 2, 3\}$
$i = 0, F = \{3\}$
$0 \otimes t_1 = 1, \quad 0 * t_1 = TN,$
$1 \otimes t_{22} = 2, \quad 1 * t_{22} = \epsilon,$
$2 \otimes t_{w_i} = 3, \quad 0 * t_{w_i} = TN\_NT,$
$2 \otimes t_{26} = 3, \quad 0 * t_{26} = TA,$
$2 \otimes t_{w_j} = 3, \quad 0 * t_{w_j} = TA\_NT,$
$\rho(3) = \epsilon$

Figure 6: Deterministic FST resulted from Figure 5

Our temporal co-occurrence data can be represented with a deterministic finite state transducer

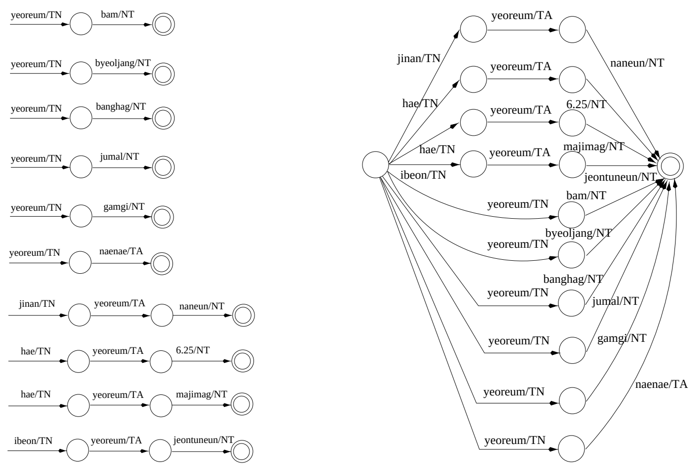
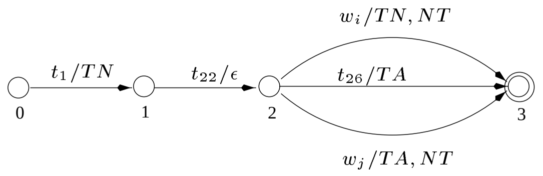


Figure 5: Finite state machine constructed with the data in Figure 4



$W = \{bam, jumal, banghag, \ldots\}$
$w_i \in W$
$w_j \notin W$

Figure 7: A deterministic finite state transducer to process temporal expression

in a similar way. The subsequential FST for our system is defined as in Figure 6 and Figure 7 illustrates the transducer in Figure 6. In the figure, t_i is a class to which the temporal word belongs in the temporal classification. w_i is a word other than temporal ones that has the preceding temporal word be its modifier, and w_j is not such a word to make a compound noun. TN, TA and NT are syntactic tags. A word tagged with TN would modify a succeeding noun like *bam*(night), *banghag*(vacation). A word attached with TA would modify a predicate and one with NT means it is not a temporal word. Actually, individual FSTs are combined into one and rules for tagging of temporal words are put over the FST. The rule is applied according to the priority by frequency in case more than one output are possible for a context. Namely, it is a rule-based system where the rules are extracted from corpus.

4.2 Chunking

After the FST of temporal expressions adds to words syntactic tags such as TN and TA, chunking is conducted with results from outputs by the FST. As we said earlier, chunking in Korean is relatively easy only if the temporal expression would be successfully recognized. Actually, our chunker is also based on the finite state machine. The following is an example for chunking rules.

$\langle NP_{chunk} \rangle \rightarrow \langle NP \rangle \mid \langle TNP \rangle$
$\langle NP \rangle \rightarrow \langle N \rangle^* \langle NP \rangle \mid \langle N \rangle^* \langle NU \rangle^* \langle UN \rangle$
$\langle TNP \rangle \rightarrow \langle TN \rangle^* \langle N \rangle^* \langle NP \rangle$

Here, N is a noun without any postposition, NP is a noun with a postposition, TN is a temporal noun recognized as modifying a succeeding noun, NU is a number and UN is a unit noun. After temporal tagging, the chunker transforms 'NT' into N, NP, etc. according to morphological constituents and their POS. Briefly, the rule says that an NP chunk is made from either NP or temporal NP. An NP would be constructed with one or more nouns and their modifiee or with a noun quantified. A TNP, which is related with time, is made from nouns modified by temporal words which would be identified by the FST. By identification of temporal expression and chunking, the following example sentence is chunked as below.

- *jinan*(last) *yeoreum*(summer) *banghag-e*(in vacation) *uri-neun*(we/SUBJ) *keompyuteo*(computer) *se*(three) *dae-reul*(unit/OBJ) *sassda*(bought)
  → We bought three computers in the last summer vacation.
- $jinan_{TN}$ $yeoreum_{TN}$ $banghag\text{-}e_{NP}$ $uri\text{-}neun_{NP}$ $keompyuteo_{N}$ se_{NU} $dae\text{-}reul_{NP}$ $sassda_{V}$
- $[jinan_{TN}$ $yeoreum_{TN}$ $banghag\text{-}e_{NP}]_{NC}$ $[uri\text{-}neun_{NP}]_{NC}$ $[keompyuteo_{N}$ se_{NU} $dae\text{-}reul_{NP}]_{NC}$ $sassda_{V}$

5 Experimental Results

For the experiment about temporal expression, we extracted 300 sentences containing temporal expressions from ETRI POS corpus. Table 2 shows the re-

	precision	recall
rate (%)	97.5	90.56

Table 2: Results of identifying temporal expression

	no chunking	using chunking
avg. # of cand	4.8	3.3

Table 3: Reduction of candidates resulted from chunking

sults from identifying temporal expressions and disambiguating their syntactic functions. From the result in the table we see that the method is very effective in that it very accurately identifies all the temporal expressions and assigns them syntactic tags.

And, Table 2 shows the reduction resulted from chunking after temporal expression identification. We take into consideration the average number of head candidates for each word since our parser is dependency based one. The test was conducted on the first file (about 800 sentences) of KAIST treebank (Choi *et al.* , 1994). The number was reduced by 31% in candidates compared to the system with no chunking, which makes parsing efficient.

Most of errors were caused by the case where temporal words have different syntactic roles under the same context. In this case, the global context such as the whole sentence or intersentential information or sometimes very sophisticated processing is needed to resolve the problem. For instance, '82 *nyeon*(year) *hyeonjae-yi*(now/GEN)' could be used two-way. If the speech time is the year 1982, then *hyeonjae-yi* are combined with 82 *nyeon* to represent time. Otherwise, 82 does not modify *hyeonjae-yi*, which cannot be recognized only with the local context. Nevertheless, the system is promising in that generally it can improve efficiency without losing accuracy which is crucial for the practical system.

6 Conclusions

In this paper, we presented a method for identification of temporal expressions and their syntactic functions based on FST and lexical data extracted from corpus. Since temporal words have the syntactic ambiguity when used in a sentence, it is important to identify the syntactic function as well as the temporal expression itself.

For the purpose, we manually extracted lexical co-occurrences from large corpus and it was possible as the number of temporal nouns is tractable enough to manipulate lexical data by hand. As shown in the result, lexical co-occurrences are crucial for disambiguating the syntactic function of the temporal expression. Besides, the finite state approach provided an efficient model for temporal expression processing. Combined with the chunker, it helped remarkably lessen, by pruning irrelevant candidates, intermediate structures generated while parsing.

References

Abney, S. 1991. Parsing By Chunks. In Berwick, Abney, and Tenny, editors, *Principle-Based Parsing.* Boston: Kluwer Academic Publishers.

Choi, K. S., Han, Y. S., Han, Y. G., and Kwon, O. W. 1994. KAIST Tree Bank Project for Korean: Present and Future Development. In *Proceedings of the International Workshop on Sharable Natural Language Resources.*

Ciravegna, F. and Lavelli, A. 1997. Controlling Bottom-Up Chart Parsers through Text Chunking. In *Proceedings of the 5th International Workshop on Parsing Technology.*

Collins, M. J. 1996. A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of the 34th Annual Meeting of the ACL.*

Elgot, C. C. and Mezei, J. E. 1965. On relations defined by generalized finite automata. *IBM Journal of Research and Development*, 9, 47–65.

Gross, M. 1993. Local Grammars and their Representation by Finite Automata. *Data, Description, Discourse: Papers on the English language in hornour of John McH Sinclair*, Michael Hoey (ed). London: HarperCollins Publishers.

KAIST. KAIST Concordance Program. URL http://csfive.kaist.ac.kr/kcp/.

Mohri, M. 1997. Finite-state Transducers in language and Speech Processing. *Computational Linguistics*, Vol 23, No (2).

Ramshaw, L. A. and Marcus, M. P. 1995. Text Chunking Using Transformation-Based Learning. In *Proceedings of the ACL Workshop on Very Large Corpora.*

Roche, E. and Schabes, Y. 1995. Deterministic Part-of-Speech Tagging with Finite-State Transducers. *Computational Linguistics*, Vol 21, No (2).

Roche, E. and Schabes, Y. 1997. *Finite-State Language Processing.* The MIT Press.

Skut, W. and Brants, T. 1999. Chunk Tagger. In *Proceedings of ESSLLI-98 Workshop on Automated Acquisition of Syntax and Parsing.*

Sproat, R. W., Shih, W., Gale, W. and Chang, N. 1994. A Stochastic Finite-State Word-segmentation Algorithm for Chinese. In *Proceedings of the 32nd Annual Meeting of ACL*

Yoon, J., Choi, K. S. and Song, M. 1999. Three Types of Chunking in Korean and Dependency Analysis Based on Lexical Association. In *Proceedings of ICCPOL '99.*