Generation of Sentence Parse Trees Using Parts of Speech

Tunga Güngör[1]

[1] Boğaziçi University, Computer Engineering Department, Bebek,
34342 İstanbul, Turkey
Gungort@boun.edu.tr

Abstract. This paper proposes a new corpus-based approach for deriving syntactic structures and generating parse trees of natural language sentences. The parts of speech (word categories) of words in the sentences play the key role for this purpose. The grammar formalism used is more general than most of the grammar induction methods proposed in the literature. The approach was tested for Turkish language using a corpus of more than 5,000 sentences and successful results were obtained.

1 Introduction

In this paper, we propose a corpus-based approach for deriving the syntactic structures of sentences in a natural language and forming parse trees of these sentences. The method is based on a concept which we name as *proximity*. The parts of speech (word categories) of words in the sentences play the key role in determining the syntactic relationships within sentences. The data about the order and frequency of word categories are collected from a corpus and are converted to proximity measures for word categories and sentences. Then these data are used to obtain probable parse trees for a given sentence.

It is well-known that grammars of natural languages are highly complicated and powerful. There have been several efforts for obtaining suitable grammars for particular languages that can generate most (if not all) of the sentences in those languages. The grammars defined manually for this purpose have limited success. The difficulty lies in the resistance of natural languages against syntactic formalizations. It is not known exactly what are the syntactic hierarchies inherent in the sentences. In fact, it is very easy to define a grammar that can generate all sentences in a language, but such a "general" grammar also generates non-sentences. Thus, forming grammars that can include sentences and at the same time exclude non-sentences is the difficult part of this task.

In order to overcome the difficulties posed by such rule-based approaches in processing natural languages, corpus-based approaches (collected under the name "statistical natural language processing") have begun to emerge recently [1,2]. They embody the assumption that human language comprehension and production works with representations of concrete past language experiences, rather than with abstract

grammatical rules. There are several studies on statistical natural language processing. A nice approach is *data oriented parsing* model [3,4,5]. This model necessitates annotated corpora in which parse trees of sentences are explicit. The idea is building new sentences by composing fragments of corpus sentences.

An interesting field where corpus-based approaches are used is grammar induction (learning). This usually means in the literature learning probabilistic context-free grammars (PCFGs). As stated in [1], the simplest method (and the basic idea) is generating all possible rules, assigning them some initial probabilities, running a training algorithm on a corpus to improve the probability estimates, and identifying the rules with high probabilities as the grammar of the language. However, this method is unrealistic as there is no bound on the number of possible rules, and even with constraints on the number of rules, often the number is so large that it becomes impractical in terms of computation time. The solution usually applied is restricting the rule types. In [6,7,8,9], some methods which use dependency grammars or Chomsky-normal-form grammars are presented.

2 Outline of the Method

Given a sentence, first the categories of the words in the sentence are determined. The sentence is at first considered as a single unit, which is formed of the sequence of these categories. It is then analyzed how this sequence can be divided into subsequences. For nontrivial sentences, the number of possible subsequences is quite large, and in general it grows exponentially with the length of the sentence. Among the possible subsequences, the best one is found according to the data in the corpus. The result is a set of smaller sentences. For each sentence in this set, the same process is repeated until the original sentence is partitioned into single categories.

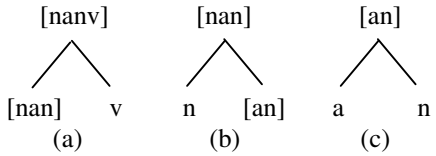

Fig. 1. Partitions for the sentence "adam siyah şapkayı beğendi"

The process is illustrated for the following simple Turkish sentence in Figure 1:

adam siyah şapkayı beğendi
(n) (a) (n) (v)
man black hat+acc like+pst+3sg
(the man liked the black hat)

(The word categories that appear in this work are: a for adjective, d for adverb, n for noun, and v for verb.) We represent the sentence as [nanv] in the form of a single sequence of word categories. Suppose that, after all alternative subsequences are evaluated, dividing into two groups as [nan] and v yields the best result, as shown

in Figure 1.a. These two subsequences are considered as new (smaller) sentences. The process is over for the second one since it is formed of a single category. The other sentence ([nan]) is analyzed and divided into subsequences n and [an] (Figure 1.b). Finally, the only subsequence left ([an]) is partitioned into a and n, as shown in Figure 1.c. (In fact, an analysis need not be performed for a subsequence of length two, since there is only one way it can be partitioned.) The process ends up with all the subsequences having a single category. By combining the phases of this process, we can obtain a tree structure as the result of the analysis of the sentence. This is shown in Figure 2.

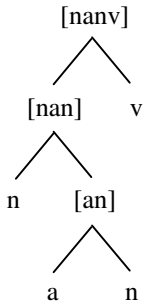

Fig. 2. Combination of partitions for the sentence "adam siyah şapkayı beğendi"

As can be seen, the tree formed after the analysis of a sentence is very similar to a parse tree of the sentence. By denoting the root node with S and the intermediate nodes with special symbols X_i, $i \geq 1$, and extending the leaf nodes with the words in the sentence, it is converted to a parse tree. The symbols X_i denote syntactic constituents like NP, VP. Since the parse tree of a sentence is built with respect to a grammar for the language, it is possible to extract the grammatical rules inherent in the tree. This grammar induction process is beyond the scope of this research. But, it can be solved with a simple mechanism when the parse trees are already available. Our aim here is limited to obtaining probable parses for sentences.

3 The Grammar Formalism

The grammar type underlying the parse trees in this research is a restricted context-free grammar: Each rule is in the form $N \rightarrow \alpha$, where N is a nonterminal symbol, α is a string of nonterminal and terminal symbols, and the number of symbols in α is greater than one. The only restriction we impose on a context-free grammar comes from the last part of this definition; a rule can not derive a single nonterminal or terminal symbol. We call a rule $N \rightarrow A$, where N is a nonterminal and A is a nonterminal or terminal, a *1-1 rule* and the corresponding derivation a *1-1 derivation*. In our grammar, 1-1 rules are not allowed. Note that the grammar type we employ is more general than those in [6,8,9], which restrict themselves to dependency and Chomsky-normal-form grammars.

The number of parse trees that can be generated by this grammar is exponential in nature. For a sentence formed of n words (categories), there are 2^{n-1}-1 alternative derivations in the first level. If we continue enumerating the alternative derivations until each category is a leaf node, we obtain a large number of parse trees.

A few words are in order about the restriction we impose on the grammar. The reason of this restriction is to limit the number of parse trees that can be generated to a finite (albeit, very large) number and also to decrease the computation time. If 1-1 derivations are allowed when enumerating all the parse trees of a sentence, there will obviously be an infinite number of trees. However, in the case that 1-1 derivations are not used, the number of categories in a node will always be less than that of its parent node, and thus the depth of the tree will be finite.

4 Parse Tree Generation

The method makes use of a corpus containing individual sentences. For each sentence, the categories of the words in the sentence are found first and then the number of each consecutive two-category, three-category, etc. combinations are stored. We call each such category combination a *category string*. In other words, for a sentence of n categories $[c_1c_2\ldots c_n]$, the category strings are as follows: $[c_1c_2]$, $[c_2c_3]$, …, $[c_{n-1}c_n]$, $[c_1c_2c_3]$, $[c_2c_3c_4]$, …, $[c_{n-2}c_{n-1}c_n]$, …, $[c_1c_2\ldots c_{n-1}]$, $[c_2c_3\ldots c_n]$, $[c_1c_2\ldots c_n]$. This calculation is performed for each sentence in the corpus and the numbers are totalled. The result gives us an indication about the frequency of consecutive use of word categories. As can be guessed, the frequencies of short category strings are usually greater than those of long category strings, since short category strings already appear within some long ones. We will denote the frequency of a category string $[c_ic_{i+1}\ldots c_j]$, i<j, with $Freq(c_i,c_{i+1},\ldots,c_j)$.

Definition: Given a sentence of n words $[c_1c_2\ldots c_i\ldots c_j\ldots c_n]$, n>1, $1\le i,j\le n$, i<j, the *category proximity* of the category string $[c_ic_{i+1}\ldots c_j]$, $CP(c_i,c_{i+1},\ldots,c_j)$, indicates the closeness of the categories c_i, c_{i+1}, …, c_j to each other and is defined as follows:

$$CP(c_i, c_{i+1},..., c_j) = \frac{Freq(c_1, c_2,..., c_n)}{Freq(c_i, c_{i+1},..., c_j)} \quad . \tag{1}$$

$CP(c_i,\ldots,c_j)$ is a measure of the strength of the connection between the categories c_i, …, c_j when considered as a single group. Small value of CP indicates stronger connection. If $CP(c_i,\ldots,c_j)$ is small, it is more likely that $[c_i\ldots c_j]$ forms a syntactic constituent.

Figure 3 compares a small CP value with a large CP value. For visualization, we represent CPs as distances between relevant nodes on a tree; that is, $CP(c_i,\ldots,c_j)$ is the distance between nodes c_i and c_j. In Figure 3.a, $CP(c_i,\ldots,c_j)$ is a small number (relative to Figure 3.c), which means that the categories c_i, …, c_j are close to each other (i.e. this category combination is a frequently occurring one). Thus they have a tendency to form a syntactic constituent, as shown in Figure 3.b. (Note that the branches in Figure 3.a do not indicate a derivation – this is emphasized by using dotted lines. They are used only to visualize the CPs on a figure. Also note that, since the situation is explained for one group of categories c_i, …, c_j, we do not take the other categories

(c_1, .., c_{i-1}, c_{j+1}, .., c_n) into account. The CP values of other categories will in fact affect the partitioning in Figure 3.b.) On the other hand, Figure 3.c shows a case where CP(c_i,..,c_j) is large. In this case, we say that the category combination c_i, .., c_j does not occur frequently. They do not tend to form a syntactic constituent; rather they tend to be partitioned as separate branches in the tree, as shown in Figure 3.d.

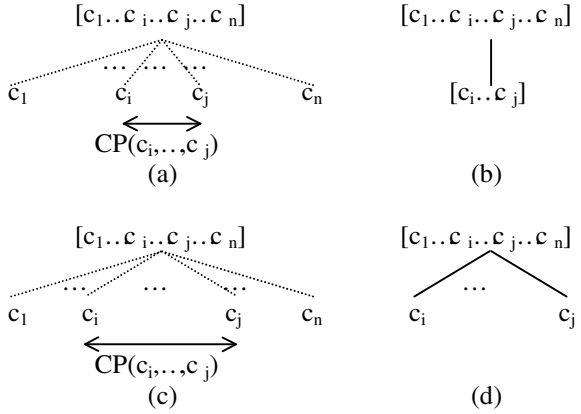

Fig. 3. Comparison of CP values

As an example, consider the following sentence:

birdenbire odaya girdi
(d) (n) (v)
suddenly room+dat enter+pst+3sg
(he/she suddenly entered the room)

Suppose that Freq(d,n)=100, Freq(n,v)=1000, and Freq(d,n,v)=50. That is, the adverb-noun combination is followed by a verb half of the time, and the noun-verb combination occurs frequently but it is rarely preceded by an adverb. Then, the category proximity measures are as follows: CP(d,n)=0.5, CP(n,v)=0.05. We see the situation in Figure 4. The figure suggests that the noun and the verb can form a syntactic constituent.

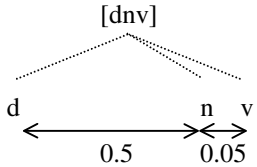

Fig. 4. CP values for the sentence 'birdenbire odaya girdi"

Definition: Given a sentence of n words [$c_1c_2..c_n$], n>1, the *sentence proximity* of the sentence, SP(c_1,c_2,..,c_n), indicates the overall closeness of the categories in the sentence and is defined in terms of category proximities:

$$SP(c_1, c_2, ..., c_n) = \sum_{i=1}^{n-1} CP(c_i, c_{i+1}) \ . \qquad (2)$$

Similar to category proximity, $SP(c_1,..,c_n)$ is a measure of the strength of the connection between the categories in the sentence. The difference lies in the range of categories it affects. Instead of determining how probable it is for a particular group of categories $c_i, .., c_j$ within the sentence to form a syntactic constituent, it increases or decreases these probabilities for all category combinations in the sentence. Small value of SP is a bias in favour of more syntactic constituents.

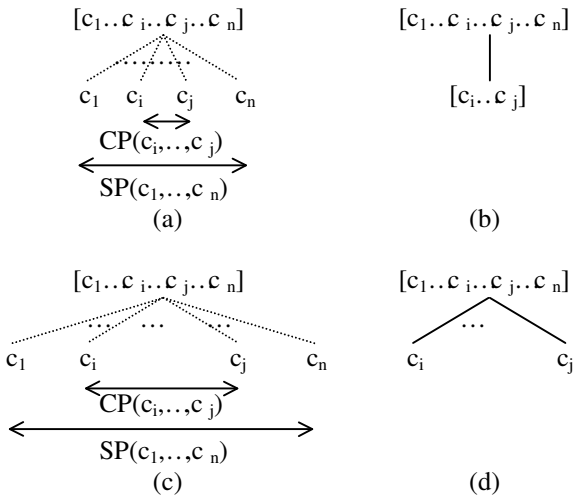

Fig. 5. Comparison of SP values

Figure 5 compares a small SP value with a large SP value. Assume that the ratio $\frac{CP(c_i,...,c_j)}{SP(c_1,...,c_n)}$ (and the ratios for category strings other than $[c_i..c_j]$) are the same in Figures 5.a and 5.c. In this case, the category proximity measure is not sufficient to differentiate the different syntactic relationships among the categories (as will be clear later) – it will force the same syntactic constituents to be built in both sentences. However, in reality, the fact that category combinations occur more frequently is a sign of more syntactic relationships (since they did not occur in that sentence 'by chance'). This effect is provided by the sentence proximity measure. Figure 5.b shows that the categories $c_i, .., c_j$ of Figure 5.a form a syntactic constituent, whereas Figure 5.d shows that the categories $c_i, .., c_j$ of Figure 5.c tend to be partitioned as separate categories.

The two proximity concepts are used together in order to produce a parse tree for a sentence. Suppose that we have a sentence of n words $[c_1c_2..c_n]$, n>1. The category proximity values for all category strings in the sentence (except $CP(c_1,..,c_n)$) are calculated. These values may be in conflict with each other. For instance, $CP(c_1,c_2)$ and $CP(c_2,c_3)$ may be small, forcing the corresponding categories to make a group, but

$CP(c_1,c_2,c_3)$ may be large, having an opposite effect. The idea is extracting the real (or, best) proximity figures inherent in these data. This is accomplished by taking the initial CP values of category strings of length two (i.e. $CP(c_i,c_{i+1})$, $1 \le i < n$) into consideration, applying the effects of other CP values on these, and arriving at final CP values of category strings of length two. These values denote the real proximities for each pair of categories.

For this purpose, the following linear programming problem is formulated and solved: (The equations have n-1 variables $x_1, x_2, .., x_{n-1}$ whose values are sought. x_i, $1 \le i < n$, corresponds to $CP(c_i,c_{i+1})$. $p_{i,j}$ and $n_{i,j}$, $1 \le i \le n-2$, $1 \le j \le n-1$, $i+j \le n$, stand for positive and negative slack variables, respectively. The goal is obtaining actual $CP(c_i,c_{i+1})$ values (i.e. x_i's) such that the sum of the slack variables is minimum.)

$$\min p_{1,1}+..+p_{1,n-1}+p_{2,1}+..+p_{2,n-2}+..+p_{n-2,1}+p_{n-2,2}+ n_{1,1}+..+n_{1,n-1}+n_{2,1}+..+n_{2,n-2}+..+n_{n-2,1}+n_{n-2,2}$$

subject to

$$x_1+p_{1,1}-n_{1,1} = CP(c_1,c_2)$$
$$x_2+p_{1,2}-n_{1,2} = CP(c_2,c_3)$$
$$\vdots$$
$$x_{n-1}+p_{1,n-1}-n_{1,n-1} = CP(c_{n-1},c_n)$$
$$x_1+x_2+p_{2,1}-n_{2,1} = CP(c_1,c_2,c_3)$$
$$\vdots$$
$$x_{n-2}+x_{n-1}+p_{2,n-2}-n_{2,n-2} = CP(c_{n-2},c_{n-1},c_n)$$
$$\vdots$$
$$x_1+..+x_{n-2}+p_{n-2,1}-n_{n-2,1} = CP(c_1,..,c_{n-1})$$
$$x_2+..+x_{n-1}+p_{n-2,2}-n_{n-2,2} = CP(c_2,..,c_n)$$

Let $CP'(c_i,c_{i+1})$, $1 \le i < n$, denote the actual category proximity values obtained and $SP'(c_1,..,c_n)$ $(= \sum_{i=1}^{n-1} CP'(c_i, c_{i+1}))$ the actual sentence proximity value. The tree structure formed with these actual values will be called the *actual tree*. As mentioned in Section 3, the category string $[c_1..c_n]$ can be partitioned in 2^{n-1}-1 ways. We call each such partition a *partition tree*. The task is finding the most probable partition tree. To this effect, the actual tree is compared with each partition tree, a score is calculated for each, and the one with the smallest score is chosen.

Definition: Given an actual tree and a partition tree P of n words $[c_1c_2..c_n]$, $n>1$, the *sentence proximity* of the partition tree, $SP_P(c_1,c_2,..,c_n)$, is equal to the sentence proximity of the actual tree. That is,

$$SP_P(c_1,c_2,..,c_n) = SP'(c_1,c_2,..,c_n) . \quad (3)$$

Definition: Given a partition tree P of n words $[c_1c_2..c_n]$, $n>1$, let the m partitions, $1<m\le n$, be $(c_1,...,c_{i_1})$, $(c_{i_1+1},...,c_{i_2})$,..., $(c_{i_{m-1}+1},...,c_{i_m})$ $(1 \le i_1 < i_2 < .. < i_m = n)$. Then, the *category proximity* of two consecutive categories, $CP_P(c_i,c_{i+1})$, $1 \le i < n$, in the tree, is defined as follows:

$$CP_P(c_i, c_{i+1}) = \begin{cases} 0 & \text{, if } c_i \text{ and } c_{i+1} \text{ are in the same partition} \\ \dfrac{SP_P(c_1,...,c_n)}{m-1} & \text{, otherwise} \end{cases} . \quad (4)$$

Intuitively, we consider the distance (proximity value) between the first and last branches of a partition tree as equal to the same distance in the actual tree and then divide this distance to the number of branches minus one to obtain an equal distance between each pair of branches.

Having obtained the actual tree, it is compared with each possible partition tree in order to find the most similar one. In fact, the actual tree is the most realistic tree in terms of showing the syntactic relationships in the sentence. However, since such "fuzzy" derivations can not take part in sentence parse trees, we must represent it with a suitable partition tree.

Definition: Given an actual tree of n words [$c_1 c_2 .. c_n$], n>1, the *cumulative category proximity* of a category c_i, 1<i<n, CCP′(c_i), is the total of the category proximity values between the first and the c_i^{th} categories. That is,

$$CCP'(c_i) = \sum_{j=1}^{i-1} CP'(c_j, c_{j+1}) . \quad (5)$$

The cumulative category proximity for a partition tree P, $CCP_P(c_i)$, is defined analogously. Note that CCP′(c_1)=0 and CCP′(c_n)=SP′(c_1,..,c_n); but these border values will not be used in the following derivations.

Definition: Given an actual tree and a partition tree P of n words [$c_1 c_2 .. c_n$], n>2, the *similarity score* between the two trees, SS_P, is defined as follows:

$$SS_P = \sum_{i=2}^{n-1} abs[CCP'(c_i) - CCP_P(c_i)] * cg(c_i) . \quad (6)$$

where *abs* is the absolute value function and cg(c_i) is the category grouping value:

$$cg(c_i) = \begin{cases} 1 & \text{, if } c_i \text{ forms a partition by itself} \\ SP'(c_1,...,c_n) & \text{, otherwise} \end{cases} . \quad (7)$$

Intuitively, the similarity score between an actual tree and a partition tree indicates the total of the amount of the distances traversed when "moving" the branches of the actual tree in order to make the actual tree identical to the partition tree. Small value of SS_P means more similarity between the trees, as the distance traversed will be less.

The category grouping value serves for the effect of sentence proximity mentioned before (Figure 5). Suppose that a category c_i is included within a partition of length greater than one, as in Figure 5.b, so cg(c_i)=SP′(c_1,..,c_n). Then, an actual tree with a smaller SP′ value (Figure 5.a) than another actual tree with a larger SP′ value (Figure 5.c) will be more similar to that partition tree, since cg(c_i) is a multiplicative factor in equation (6). In other words, the former one will bias in favour of those partition trees in which c_i appears within a group among all the possible partition trees, whereas the latter one will bias in favour of partition trees in which c_i forms a separate partition.

After the most similar partition tree is chosen, each partition with length greater than two is considered as a new sentence and the whole process is repeated. As explained in Section 2, the collection of all the most similar partition trees then forms the parse tree of the sentence.

5 Implementation of the Method

The proposed approach was implemented for Turkish. A corpus of general text containing about 5,700 sentences was compiled. The average length (number of words) of the sentences is 18.6. The corpus includes long sentences having as many as 50 words. Word categories are derived by using the spelling checker program explained in [10]. The frequencies of all category strings in the corpus are collected and stored in a database.

The method was applied to several sentences and parse trees were generated. Below we present the details of a short sentence only due to lack of space. The sentence was taken from a newspaper:

ülkedeki demokratik gelişmeler yetersizdir
(n) (a) (n) (v)
country+loc democratic progress+pl adequate+neg+cop
(democratic progresses in the country are not adequate)

Table 1. Calculations for the example sentence (first iteration)

<table>
<tr><td>Freq(n,a) = 5,992
Freq(a,n) = 6,973
Freq(n,v) = 6,639
Freq(n,a,n) = 3,036
Freq(a,n,v) = 865
Freq(n,a,n,v) = 367
(a)</td><td colspan="2">CP(n,a) = 0.061
CP(a,n) = 0.053
CP(n,v) = 0.055
CP(n,a,n) = 0.121
CP(a,n,v) = 0.424
SP(n,a,n,v) = 0.169
(b)</td></tr>
<tr><td colspan="3">min p1+n1+p2+n2+p3+n3+p4+n4+p5+n5
subject to
x1+p1-n1=0.061
x2+p2-n2=0.053
x3+p3-n3=0.055
x1+x2+p4-n4=0.121
x2+x3+p5-n5=0.424
(c)</td></tr>
<tr><td>CP′(n,a) = 0.061
CP′(a,n) = 0.060
CP′(n,v) = 0.365
SP′(n,a,n,v) = 0.486
(d)</td><td>CP_P(n,a) = 0
CP_P(a,n) = 0
CP_P(n,v) = 0.486
SP_P(n,a,n,v) = 0.486
(e)</td><td>CCP′(a) = 0.061
CCP′(n) = 0.121
CCP_P(a) = 0
CCP_P(n) = 0
SS_P = 0.088
(f)</td></tr>
</table>

Table 2. Calculations for the example sentence (second iteration)

<table>
<tr><td>Freq(n,a) = 5,992
Freq(a,n) = 6,973
Freq(n,a,n) = 3,036
(a)</td><td colspan="2">CP(n,a) = 0.507
CP(a,n) = 0.435
SP(n,a,n) = 0.942
(b)</td></tr>
<tr><td colspan="3">min p1+n1+p2+n2
subject to
x1+p1-n1=0.507
x2+p2-n2=0.435
(c)</td></tr>
<tr><td>CP′(n,a) = 0.507
CP′(a,n) = 0.435
SP′(n,a,n) = 0.942
(d)</td><td>CP_P(n,a) = 0.471
CP_P(a,n) = 0.471
SP_P(n,a,n) = 0.942
(e)</td><td>CCP′(a) = 0.507
CCP_P(a) = 0.471
SS_P = 0.036
(f)</td></tr>
</table>

For each iteration of the process, we give the calculations in a table. The calculations involve the actual tree and the most probable partition tree P. Calculations for other partition trees are not shown due to the large number of possible trees. Each table consists of the following data: category string frequencies (part a), initial category proximities and sentence proximity (part b), linear programming problem (part c), actual category proximities and sentence proximity (part d), category proximities and sentence proximity for the partition tree P (part e), and cumulative category proximities and the similarity score between the actual tree and the partition tree P (part f).

Table 1 contains the data and the results for the category string [nanv] and Table 2 for the category string [nan]. The final parse tree is shown in Figure 6.

An observation about the computational complexity of the method is worth mentioning here. The program was executed on a Pentium 4 1.3 Ghz machine. The execution time is very low for sentences containing at most 15-20 words. It takes about 4-5 seconds parsing such sentences. The computation time seems very promising when we consider the size of the search space – that is, we are working with a nearly unrestricted context-free grammar formalism. The reason is pruning the search space at each iteration and taking only the best partitioning into account.

6 Conclusions

In this paper, we proposed a new method for generating parse trees of natural language sentences. Due to the limitations of rule-based techniques in formalizing natural languages, statistical techniques are gaining popularity. This was the direction pursued in this research. The method is based on the information inherent in the word categories (parts of speech) of words within the sentences. By using the frequency and order of these categories, a method was formulated to make the syntactic relationships in sentences explicit.

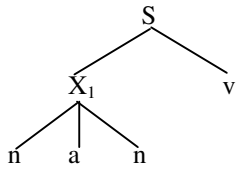

Fig. 6. Parse tree for the example sentence

The parse trees that the method produces are equivalent to those that can be generated by a little restricted context-free grammar. The grammar formalism used is more general than those used by other similar approaches.

The approach was tested for Turkish using a corpus of about 5,700 sentences. Although an exact evaluation is not possible since there does not exist a complete grammar for the language, the results are successful. The parse trees produced by the program for about half of the sentences seem correct. One strength of the method is its ability to generate plausible parses for complex sentences. But parses which can not capture the syntactic relationships inside the sentences or that result in slightly misplaced constituents were also produced. As the size of the corpus increases, we may expect better results.

An attractive area for future research is extracting a grammar using these parse trees. This will be an important contribution if it becomes possible to obtain a robust grammar, since no comprehensive grammars have been written yet. It may also provide feedback to rule-based grammar studies.

Acknowledgement

This work was supported by the Boğaziçi University Research Fund, Grant no. 02A107.

References

1. Charniak, E.: Statistical Language Learning. MIT, Cambridge MA (1997)
2. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT, Cambridge MA (2002)
3. Bod, R.: Data Oriented Parsing. In: Computational Linguistics in the Netherlands. Amsterdam The Netherlands (1991) 26-39
4. Bod, R.: Beyond Grammar: An Experience-Based Theory of Language. CSLI Publications, Stanford (1998)
5. Kaplan, R.: A Probabilistic Approach to Lexical-Functional Analysis. In: Conference and Workshop on Lexical Functional Grammar. CSLI Publications, Stanford (1996)
6. Carroll, G.: Learning Probabilistic Grammars for Language Modelling. Ph.D. Thesis. Brown University, Providence RI (1995)
7. Carroll, G., Charniak, E.: Learning Probabilistic Dependency Grammars from Labelled Text. AAAI Fall Symposium on Probabilistic Approaches to Natural Language. Cambridge MA (1992) 25-32

8. Pereira, F., Schabes, Y.: Inside-Outside Reestimation from Partially Bracketed Corpora. In: Annual Meeting of the Association for Computational Linguistics. Newark Deleware (1992) 128-135
9. Briscoe, T., Waegner, N.: Robust Stochastic Parsing Using the Inside-Outside Algorithm. AAAI Workshop on Statistically-Based NLP Techniques. San Jose California (1992) 30-53
10. Güngör, T.: Computer Processing of Turkish: Morphological and Lexical Investigation. Ph.D. Thesis. Boğaziçi University, İstanbul Turkey (1995)