A Syntactic and Morphological Analyzer for a Text-to-Speech System

Thomas Russi
Institute of Electronics, Swiss Federal Institute of Technology (ETH)
CH 8092 Zürich, Switzerland, e-mail: russi@strati.ethz.ch

Abstract

This paper presents a system which analyzes an input text syntactically and morphologically and converts the text from the graphemic to the phonetic representation (or vice versa). We describe the grammar formalism used and report a parsing experiment which compared eight parsing strategies within the framework of chart parsing. Although the morphological and syntactic analyzer has been developed for a text-to-speech system for German, it is language independent and general enough to be used for dialog systems, NL-interfaces or speech recognition systems.

1 Introduction

In order to convert text to speech one must first derive an underlying abstract linguistic representation for the text. There are at least two reasons why a direct approach (e.g., letter-to-sound rules) is inadequate. Firstly, rules for pronouncing words must take into consideration morpheme structure, e.g., <sch> is pronouced differently in the German words "lösch+en" ("extinguish") and "Hös+chen" (diminutive of "trousers"), and syntactic structure, e.g., to solve noun-verb ambiguities such as "Sucht" ("addiction") and "sucht" ("to search"). Secondly, sentence duration pattern and fundamental frequency contour depend largely on the structure of the sentence.

While most commercial, but also some laboratory text-to-speech (TTS) systems use letter-to-sound rules without taking into account the morphological structure of a word, recently developed systems [1,2,3] incorporate morphological analysis. Although the influence of syntax on prosody is widely acknowledged [2,3], most TTS systems lack syntax analysis [1,3] or use some kind of phrase-level parsing [2] to obtain information on the syntactic structure of a sentence. This is motivated more by current technological limitations than by linguistic insights.We are convinced that in order to achieve highly intelligible and natural-sounding speech, not only the phonological and morphological but also the syntactic, semantic and even discourse structure of a text must be taken into account - although this is not yet feasible. As a step toward such a model, we have developed a morphological and syntactic analyzer that is based on simple but powerful formalisms which are linguistically well-motivated and computationally effective.

2 Morphological and Syntactic Analysis

In our TTS system, morphological analysis consists of three stages: segmentation, parsing and generation. The segmentation module finds possible ways to partition the input string into dictionary entries (morphs). Spelling changes, e.g., schwa-insertion or elision, are covered by morphographemic rules. The parsing module of the morphological analysis uses a word grammar to accept or reject combinations of dictionary entries and to percolate features from the lexicon to the syntactic analyzer. The generation module of the morphological analysis generates the phonetic transcription by concatenating the phonetic strings, which are stored as part of each morph entry, and by applying morphophonetic rules. The syntactic analysis is based on a sentence grammar and a parser that takes as input the result of the morphological analyzer. It assigns to each sentence its surface syntactic structure. The syntactic structure of the sentence and the phonetic transcription of each word are used at a later stage to determine prosodic features such as duration pattern and fundamental frequency contour.

3 UTN Formalism

Morphographemic and morphophonetic rules are written in a Kimmo-style formalism [4]. Unlike the original two-level model, a word grammar is used to parse the lexical strings and to determine the category of the overall word formed by several morphs. To express word and sentence grammars, we have developed a grammar formalism, called Unification-based Transition Networks (UTN). Its skeleton are nondeterministic recursive transition networks (RTNs), which are equivalent to context-free grammars. A transition network specifies the linear precedence and immediate dominance relation

within a constituent. Each label of a transition denotes a preterminal, a constituent or an ϵ-transition. As opposed to labels in RTNs, which are monadic, labels in UTNs are complex categories (features matrices). Each transition contains a set of attribute equations, which specify the constraints that must be satisfied between complex categories in a network. Our notation of attribute equations is very similar to that commonly used in unification-based rule formalisms such as PATR [5]. The UTN formalism is fully declarative. It is based on concatenation and recursion, which is reflected in the topology of the networks, and unification, which is used for matching, equality testing and feature passing. Although the UTN formalism is somewhat similar to ATNs [6], it is much more concise and elegant because of its simplicity and declarativeness. The implementation of several grammars for German syntax and morphosyntax revealed that transition networks are well-suited to design[1] and test grammars. We believe that this formalism meets the general criteria of linguistic naturalness and mathematical power. In addition, the parsing experiment reported below shows that efficient parsers can be implemented for the UTN formalism.

4 Parsing

The design of our TTS system requires efficient parsing algorithms and a flexible parser environment to compare several search and rule invocation strategies. Active chart parsing [8] is well-suited for that purpose. We have implemented a general chart parser that can be parameterized for several search and rule invocation strategies. The aim of the experiment reported below was to investigate to what extent a parser can be directed by using the FIRST, FOLLOW and REACHABILITY relations [9,8] and combinations thereof, thereby reducing the number of edges, the number of applications of the fundamental rule and parsing time.

4.1 Rule Invocation Strategies

We compared eight parsing strategies, i.e., four top-down (T1 to T4) and four bottom-up (B1 to B4) strategies. The top-down strategies are variants of Earley's algorithm, the bottom-up strategies variants of the left-corner algorithm [9]. T1, a pure top-down strategy, implements Earley's algorithm without lookahead. Strategy T2, a directed top-down strategy, uses the FIRST relation to test whether the next input symbol is in the FIRST set of the active edge each time an empty active edge is created. Strategy T3, a top-down strategy with lookahead, uses the FOLLOW set to test whether the next input symbol belongs to the FOLLOW set of the inactive edge each time an inactive edge is created. Strategy T4 combines the selectivity of strategy T2 and lookahead of strategy T3. Strategy B1 implements a left-corner algorithm [9]. Strategy B2 is a left-corner parser directed by a top-down filter based on the REACHABILITY relation [10]. Strategy B3 implements a left-corner algorithm with lookahead similar to that of strategy T3, while strategy B4 adds a top-down filter and lookahead to the left-corner algorithm.

[1] To compare the UTN formalism with rule-based formalisms, we translated several grammars to transition networks. As an example, the grammar GIII found in Tomita's book [7] with about 220 rules was translated to a strongly equivalent network grammar of 37 transition networks. We got the impression that it is easier to write and modify a network grammar of several dozen networks (that can be displayed and edited graphically) than one of several hundreds of rules.

Str	AE	IE	TOT	FR	TIME
T1	122024	6390	128414	61432	1.44
T2	94966	6386	101352	58433	1.17
T3	120067	5885	125952	59148	1.43
T4	93009	5881	98890	56149	1.16
B1	126406	11198	137604	87422	1.25
B2	89763	7049	96812	62885	1.02
B3	123344	10080	133424	82608	1.23
B4	87586	6408	93994	60121	1.00

Table 1: Parsing sentence set SI with grammar GI

4.2 Grammars and Test Sets

For the experiment presented here, we used a grammar (GI) for German syntax[2] that has been developed for our TTS system and a grammar (GII) for English syntax[3] (GII) to compare our results with those of other experiments ([7,11,10]). Our sentence sets consist of 35 German sentences (set SI, with an average sentence length of 9.8 words) and 39 English sentences (set SII, with an average sentence length of 15.3 words) from Tomita [7], pp. 185-189.

4.3 Results

Tables 1 and 2 show the results of parsing sets SI and SII with grammars GI and GII, respectively. We measured for all strategies (T1 to B4) the number of active (AE) and inactive (IE) edges, the total number of edges (TOT = AE+IE) and parsing time[4] (TIME). Since the UTN formalism is based on unification, a time- and space-consuming operation, we also indicate the number of applications of the fundamental rule (FR) to show the relation between parsing strategy and FR applications.

[2] This grammar consists of 48 networks, 770 transitions, 1246 unification equations and describes a substantial part of German syntax.

[3] This grammar is a strongly equivalent network grammar of Tomita's grammar GIII.

[4] Parsing time is indicated relative to the fastest algorithm.

Str	AE	IE	TOT	FR	TIME
T1	91578	16946	108524	54689	1.50
T2	69160	16946	86106	54689	1.08
T3	76288	13880	90168	44226	1.35
T4	55173	13880	69053	44226	1.00
B1	210021	49372	259393	168871	3.00
B2	99001	22797	121798	75509	1.56
B3	169299	40022	209321	138232	2.68
B4	84984	19415	104399	65016	1.55

Table 2: Parsing sentence set SII with grammar GII

Our experiments confirm the results of Shann and Wirén [10,11] that parsing efficiency depends heavily on the grammar, the language, the grammar formalism and the sentence set. Nevertheless, by carefully tuning a parsing strategy, a significant increase in efficiency is gained.

Undirected top-down parsing performs better than undirected bottom-up. This coincides with the results of Wirén. Directed strategies[5] outperform undirected strategies with respect to parsing time and memory. This holds for top-down and bottom-up strategies.

Previous experiments [11,10,7] did not investigate the influence of lookahead in top-down parsing. However, using lookahead (the FOLLOW relation) significantly reduces the number of edges, the number of applications of the fundamental rule and parsing time.

Directed top-down parsing with lookahead is as fast as left-corner parsing with top-down filtering and lookahead. The difference between the two strategies is statistically insignificant when considering all experiments conducted with all German grammars and several sentence sets. However, it is uncertain to what extent this statement can be generalized to other types of grammars and languages. Based on the results of our experiments, both strategies (T4 and B4) are suited as main strategies in our TTS system.

5 Concluding Remarks

We have presented a language-independent model for syntactic and morphological analysis. Special emphasis has been laid on the description of the UTN formalism and a parser experiment which compared different rule invocation strategies. The analyzer is fully implemented in Common Lisp and its application in a text-to-speech system has significantly improved the quality of the synthetic speech. Since the grapheme-to-phoneme conversion is bidirectional, our approach may also be promising for speech recognition.

[5] The algorithm of Tomita can be considered a maximally directed chart-parser that uses the FIRST and FOLLOW relation to construct an LR-table at compile time.

References

[1] A. Pounder and M. Kommenda. Morphological Analysis for a German Text-to-Speech System. In *Proceedings of the 11th International Conference on Computational Linguistics*, pages 263–268, 1986.

[2] D.H. Klatt. Review of text-to-speech conversion for English. *Journal of the Acoustical Society of America*, 82(3):737–793, September 1987.

[3] W Daelemans. Grafon: A Grapheme-to-Phoneme Conversion System for Dutch. In *Proceedings of the 12th International Conference on Computational Linguistics*, pages 133–138, 1988.

[4] K. Koskenniemi. *Two-level Morphology: A General Computational Model for Word-Form Recognition and Production.* PhD thesis, University of Helsinki, 1983.

[5] S. M. Shieber. *An Introduction to Unification-Based Approaches to Grammar.* CSLI Lecture Notes 4, Center for the Study of Language and Information, 1986.

[6] M. Bates. The Theory and Practice of Augmented Transition Network Grammars. In L. Bolc, editor, *Natural Language Communication with Computers*, pages 191–259, Springer Verlag, 1978.

[7] M. Tomita. *Efficient Parsing for Natural Language.* Kluwer Academic Publishers, 1986.

[8] M. Kay. Algorithm schemata and data strucures in syntactic processing. In St. Allén, editor, *Text Processing: Text Analysis and Generation, Text Typology and Attribution*, pages 327–358, Almqvist and Wiksell International, Stockholm, Sweden, 1982.

[9] A.V. Aho and J.D. Ullman. *The Theory of Parsing, Translation, and Compiling. Automatic Computation*, Prentice-Hall Inc., Englewood Cliffs, N.Y., 1972.

[10] P. Shann. The selection of a parsing strategy for an on-line machine translation system in a sublanguage domain. A new practical comparison. In *Proc. of the International Workshop on Parsing Technologies*, pages 264–276, Carnegie Mellon University, 1989.

[11] M. Wirén. A comparison of rule-invocation strategies in context-free chart parsing. In *ACL Proceedings, Third European Conference*, pages 226–233, Association for Computational Linguistics, 1987.