Lattice Parsing to Integrate Speech Recognition and Rule-Based Machine Translation

Selçuk Köprü
AppTek, Inc.
METU Technopolis
Ankara, Turkey
skopru@apptek.com

Adnan Yazıcı
Department of Computer Engineering
Middle East Technical University
Ankara, Turkey
yazici@metu.edu.tr

Abstract

In this paper, we present a novel approach to integrate speech recognition and rule-based machine translation by lattice parsing. The presented approach is hybrid in two senses. First, it combines structural and statistical methods for language modeling task. Second, it employs a chart parser which utilizes manually created syntax rules in addition to scores obtained after statistical processing during speech recognition. The employed chart parser is a unification-based active chart parser. It can parse word graphs by using a mixed strategy instead of being bottom-up or top-down only. The results are reported based on word error rate on the NIST HUB-1 word-lattices. The presented approach is implemented and compared with other syntactic language modeling techniques.

1 Introduction

The integration of speech and language technologies plays an important role in speech to text translation. This paper describes a unification-based active chart parser and how it is utilized for language modeling in speech recognition or speech translation. The fundamental idea behind the proposed solution is to combine the strengths of unification-based chart parsing and statistical language modeling. In the solution, all sentence hypotheses, which are represented in word-lattice format at the end of automatic speech recognition (ASR), are parsed simultaneously. The chart is initialized with the lattice and it is processed until the first sentence hypothesis is selected by the parser. The parser also utilizes the scores assigned to words during the speech recognition process. This leads to a hybrid solution.

An important benefit of this approach is that it allows one to make use of the available grammars and parsers for language modeling task. So as to be used for this task, syntactic analyzer components developed for a rule-based machine translation (RBMT) system are modified. In speech translation (ST), this approach leads to a perfect integration of the ASR and RBMT components. Language modeling effort in ASR and syntactic analysis effort in RBMT are overlapped and merged into a single task. Its advantages are twofold. First, this allows us to avoid unnecessary duplication of similar jobs. Secondly, by using the available components, we avoid the difficulty of building a syntactic language model all from the beginning.

There are two basic methods that are being used to integrate ASR and rule-based MT systems: First-best method and the N-best list method. Both techniques are motivated from a software engineering perspective. In the First-best approach (Figure 1.a), the ASR module sends a single recognized text to the MT component to translate. Any ambiguity existing in the recognition process is resolved inside the ASR. In contrast to the First-best approach, in the N-best List approach (Figure 1.b); the ASR outputs N possible recognition hypotheses to be evaluated by the MT component. The MT picks the first hypothesis and translates it if it is grammatically correct. Otherwise, it moves to the second hypothesis and so on. If none of the available hypotheses are syntactically correct, then it translates the first one.

We propose a new method to couple ASR and rule-based MT system as an alternative to the ap-

Proceedings of the 12th Conference of the European Chapter of the ACL, pages 469–477,
Athens, Greece, 30 March – 3 April 2009.

proaches mentioned above. Figure 1 represents the two currently in-use coupling methods followed by the new approach we introduce (Figure 1.c). In the newly proposed technique, which we call the N-best word graph approach, the ASR module outputs a word graph containing all N-best hypotheses. The MT component parses the word graph, thus, all possible hypotheses at one time.

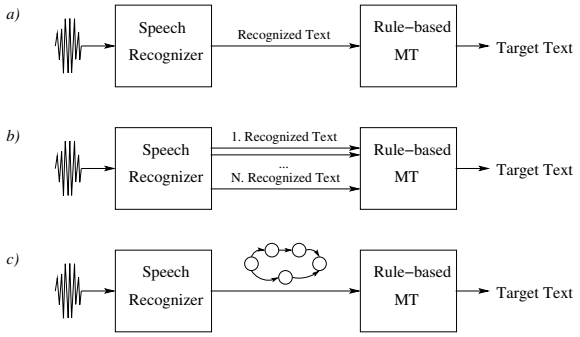

Figure 1: ASR and rule-based MT coupling: a) First-best b) N-best list c) N-best word graph.

While integrating the SR system with the rule-based MT system, this study uses word graphs and chart parsing with new extensions. Parsing of word lattices has been a topic of research over the past decade. The idea of chart parsing the word graph in SR systems has been previously used in different studies in order to resolve ambiguity. Tomita (1986) introduced the concept of word-lattice parsing for the purpose of speech recognition and used an LR parser. Next, Paeseler (1988) used a chart parser to process word-lattices. However, to the best of our knowledge, the specific method for chart parsing a word graph introduced in this paper has not been previously used for coupling purposes.

Recent studies point out the importance of utilizing word graphs in speech tasks (Dyer et al., 2008). Previous work on language modeling can be classified according to whether a system uses purely statistical methods or whether it uses them in combination with syntactic methods. In this paper, the focus is on systems that contain syntactic approaches. In general, these language modeling approaches try to parse the ASR output in word-lattice format in order to choose the most probable hypothesis. Chow and Roukos (1989) used a unification-based CYK parser for the purpose of speech understanding. Chien et al. (1990) and Weber (1994) utilized probabilistic context free grammars in conjunction with unification grammars to chart-parse a word-lattice. There are various differences between the work of Chien et al. (1990) and Weber (1994) and the work presented in this paper. First, in the previously mentioned studies, the chart is populated with the same word graph that comes from the speech recognizer without any pruning, whereas in our approach the word graph is reduced to an acceptable size. Otherwise, the efficiency becomes a big challenge because the search space introduced by a chart with over thousands of initial edges can easily be beyond current practical limits. Another important difference in our approach is the modification of the chart parsing algorithm to eliminate spurious parses.

Ney (1991) deals with the use of probabilistic CYK parser for continous speech recognition task. Stolcke (1995) summarizes extensively their approach to utilize probabilistic Earley parsing. Chappelier et al. (1999) gives an overview of different approaches to integrate linguistic models into speech recognition systems. They also research various techniques of producing sets of hypotheses that contain more "semantic" variability than the commonly used ones. Some of the recent studies about structural language modeling extract a list of N-best hypotheses using an N-gram and then apply structural methods to decide on the best hypothesis (Chelba, 2000; Roark, 2001). This contrasts with the approach presented in this study where, instead of a single sentence, the word-lattice is parsed. Parsing all sentence hypotheses simultaneously enables a reduction in the number of edges produced during the parsing process. This is because the shared word hypotheses are processed only once compared to the N-best list approach, where the shared words are processed each time they occur in a hypothesis. Similar to the current work, other studies parse the whole word-lattice without extracting a list (Hall, 2005). A significant distinction between the work of Hall (2005) and our study is the parsing algorithm. In contrast to our chart parsing approach augmented by unification based feature structures, Charniak parser is used in Hall (2005)'s along with PCFG.

The rest of the paper is organized as follows: In the following section, an overview of the proposed language model is presented. Next, in Section 3, the parsing process of the word-lattice is described in detail. Section 4 describes the exper-

iments and reports the obtained results. Finally, Section 5 concludes the paper.

2 Hybrid language modeling

The general architecture of the system is depicted in Figure 2. The HTK toolkit (Woodland, 2000) word-lattice file format is used as the default file format in the proposed solution. The word-lattice output from ASR is converted into a finite state machine (FSM). This conversion enables us to benefit from standard theory and algorithms on FSMs. In the converted FSM, non-determinism is removed and it is minimized by eliminating redundant nodes and arcs. Next, the chart is initialized with the deterministic and minimal FSM. Finally, this chart is used in the structural analysis.

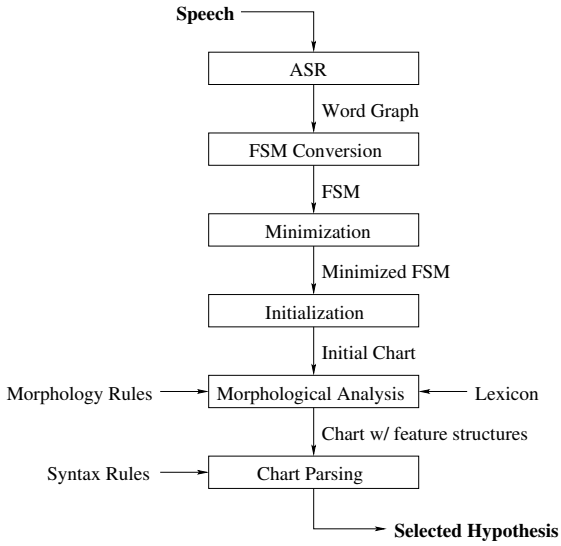

Figure 2: The hybrid language model architecture.

Structural analysis of the word-lattice is accomplished in two consecutive tasks. First, morphological analysis is performed on the word level and any information carried by the word is extracted to be used in the following stages. Second, syntactic analysis is performed on the sentence level. The syntactic analyzer consists of a chart parser in which the rules modeling the language grammar are augmented with functional expressions.

3 Word Graph Processing

The word graphs produced by an ASR are beyond the limits of a unification-based chart parser. A small-sized lattice from the NIST HUB-1 data set (Pallett et al., 1994) can easily contain a couple of hundred states and more than one thousand arcs. The largest lattice in the same data set has 25 000 states and almost 1 million arcs. No unification-based chart parser is capable of coping with an input of this size. It is unpractical and unreasonable to parse the lattice in the same form as it is output from the ASR. Instead, the word graph is pruned to a reasonable size so that it can be parsed according to acceptable time and memory limitations.

3.1 Word graph to FSM conversion

The pruning process starts by converting the time-state lattice to a finite state machine. This way, algorithms and data structures for FSMs are utilized in the following processing steps. Each word in the time-state lattice corresponds to a state node in the new FSM. The time slot information is also dropped in the recently built automata. The links between the words in the lattice are mapped as the FSM arcs.

In the original representation, the word labels in the time-state lattices are on the nodes, and the acoustic scores and the statistical language model scores are on the arcs. Similarly, the words are also on the nodes. This representation does not fit into the chart definition where the words are on the arcs. Therefore, the FSM is converted to an arc labeled FSM. The conversion is accomplished by moving back the word label on a state to the incoming arcs. The weights on the arcs represent the negative logarithms of probabilities. In order to find the weight of a path in the FSM, all weights on the arcs existing on that path are added up.

The resulting FSM contains redundant arcs that are inherited from the word graph. Many arcs correspond to the same word with a different score. The FSM is *nondeterministic* because, at a given state, there are different alternative arcs with the same word label. Before parsing the converted FSM, it is essential to find an equivalent finite automata that is *deterministic* and that has as few nodes as possible. This way, the work necessary during parsing is reduced and efficient processing is ensured.

The *minimization* process serves to shrink down the FSM to an equivalent automata with a suitable size for parsing. However, it is usually the case that the size is not small enough to meet the time and memory limitations in parsing. N-best list selection can be regarded as the last step in constricting the size. A subset of possible hypotheses is selected among many that are contained in the *mini-*

mized FSM. The selection mechanism favors only the best hypotheses according to the scores present in the FSM arcs.

3.2 Chart parsing

The parsing engine implemented for this work is an active chart parser similar to the one described in Kay (1986). The language grammar that is processed by the parser can be designed top-down, bottom-up or in a combined manner. It employs an *agenda* to store the edges prior to inserting to the chart. Edges are defined to be either *complete* or *incomplete*. Incomplete edges describe the rule state where one or more syntactic categories are expected to be matched. An incomplete edge becomes complete if all syntactic categories on the right-hand-side of the rule are matched.

Parsing starts from the rules that are associated to the lexical entries. This corresponds to the bottom-up parsing strategy. Moreover, parsing also starts from the rules that build the final symbol in the grammar. This corresponds to the top-down parsing strategy. Bottom-up rules and top-down rules differ in that the former contains a non-terminal that is marked as the *trigger* or *central element* on the left-hand-side of the rule. This central element is the starting point for the execution of the bottom-up rule. After the central element is matched, the extension continues in a bidirectional manner to complete the missing constituents. Bottom-up incomplete edges are described with *double-dotted* rules to keep track of the beginning and end of the matched fragment.

The anticipated edges are first inserted into the agenda. Edges popped out from the agenda are processed with the *fundamental rule* of chart parsing. The agenda allows the reorganization of the edge processing order. After the application of the fundamental rule, new edges are predicted according to either bottom-up or top-down parsing strategy. This strategy is determined by how the current edge has been created.

3.3 Chart initialization

The chart initialization procedure creates from an input FSM, which is derived from the ASR word lattice, a valid chart that can be parsed in an active chart parser. The initialization starts with filling in the *distance* value for each node. The *distance* of a node in the FSM is defined as the number of nodes on the *longest* path from the start state to the current state. After the *distance* value is set for all nodes in the FSM, an *edge* is created for each arc. The *edge* structure contains the *start* and *end* values in addition to the *weight* and *label* data fields. These position values represent the edge location relative to the beginning of the chart. The starting and ending node information for the arc is also copied to the edge. This node information is later utilized in chart parsing to eliminate spurious parses. The number of edges in the chart equals to the number of edges in the input FSM at the end of initialization.

Consider the simple FSM $\mathcal{F}_1$ depicted in Figure 3, the corresponding two-dimensional chart and the related hypotheses. The chart is populated with the converted word graph before parsing begins. Words in the same column can be regarded as a single lexical entry with different senses (e.g., '*boy*' and '*boycott*' in column 2). Words spanning more than one column can be regarded as idiomatic entries (e.g. '*escalated*' from column 3 to 5). Merged cells in the chart (e.g., '*the*' and '*yesterday*' at columns 1 and 6, respectively) are shared in both sentence hypotheses.

$\mathcal{F}_1$:

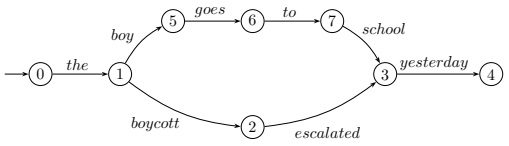

Chart:

0	1	2	3	4	5	6
0 *the* 1	1 *boy* 5	5 *goes* 6	6 *to* 7	7 *school* 3	3 *yesterday* 4	
	1 *boycott* 2	2 *escalated* 3				

Hypotheses:

- *The boy goes to school yesterday*
- *The boycott escalated yesterday*

Figure 3: Sample FSM $\mathcal{F}_4$, the corresponding chart and the hypotheses.

3.4 Extended Chart Parsing

In a standard active chart parser, the chart depicted in Figure 3 could produce some spurious parses. For example, both of the complete edges in the initial chart at location [1-2] (i.e. '*boy*' and '*boycott*) can be combined with the word '*goes*', although '*boycott goes*' is not allowed in the original word graph. We have eliminated these kinds of spuri-

ous parses by making use of the *arcstart* and *arcfinish* values. These labels indicate the starting and ending node identifiers of the path spanned by the edge in subject. The application of this idea is illustrated in Figure 4. Different from the original implementation of the fundamental rule, the procedure has the additional parameters to define starting and ending node identifiers. Before creating a new incomplete edge, it is checked whether the node identifiers match or not.

When we consider the chart given in Figure 3, '$_1$ *boycott* $_2$' and '$_5$ *goes* $_6$' cannot be combined according to the new *fundamental rule* in a parse tree because the ending node id, i.e. 2, of the former does not match the starting node id, i.e. 5, of the latter. In another example, '$_0$ *the* $_1$' can be combined with both '$_1$ *boy* $_5$' and '$_1$ *boycott* $_2$' because their respective node identifiers match. After the two edges, '*boycott*' and '*escalated*', are combined and a new edge is generated, the starting node identifiers for the entire edge will be as in '$_1$ *boycott escalated* $_3$'.

The utilization of the node identifiers enables the two-dimensional modeling of a word graph in a chart. This extension to chart parsing makes the current approach word-graph based rather than confusion-network based. Parse trees that conflict with the input word graph are blocked and all the processing resources are dedicated to proper edges. The chart parsing algorithm is listed in Figure 4.

```
input: grammar, word–graph
output: chart

algorithm CHART–PARSE(grammar, word–graph)
   INITIALIZE (chart, agenda, word–graph)
   while agenda is not empty
      edge ← POP (agenda)
      PROCESS–EDGE (edge)
   end while
end algorithm

procedure PROCESS–EDGE (A → B • α • C, [j, k], [n_s, n_e])
   PUSH (chart, A → B • α • C, [j, k], [n_s, n_e])
   FUNDAMENTAL–RULE (A → B • α • C, [j, k], [n_s, n_e])
   PREDICT (A → B • α • C, [j, k], [n_s, n_e])
end procedure

procedure FUNDAMENTAL–RULE (A → B • α • C, [j, k], [n_s, n_e])
   if B = βD      // edge is incomplete
      for each (D → •δ•, [i, j], [n_r, n_s]) in chart
         PUSH (agenda, (A → β • Dα • C, [i, k], [n_r, n_e]) )
      end for
   end if
   if C = Dγ      // edge is incomplete
      for each (D → •δ•, [k, l], [n_e, n_f]) in chart
         PUSH (agenda, (A → B • αD•γ, [j, l], [n_s, n_f]) )
      end for
   end if
   if B is null and C is null // edge is complete
      for each (D → βA • γ • δ, [k, l], [n_e, n_f]) in chart
         PUSH (agenda, (D → β • Aγ • δ, [j, l], [n_s, n_f]) )
      end for
      for each (D → β • γ • Aδ, [i, j], [n_r, n_s]) in chart
         PUSH (agenda, (D → β • γA • δ, [i, k], [n_r, n_e]) )
      end for
   end if
end procedure

procedure PREDICT (A → B • α • C, [j, k], [n_s, n_e])
   if B is null and C is null // edge is complete
      for each D → βAγ in grammar where A is trigger
         PUSH (agenda, (D → β • A • γ, [j, k], [n_s, n_e]) )
      end for
   else
      if B = βD // edge is incomplete
         for each D → γ in grammar
            PUSH (agenda, (D → γ•, [j, j], [n_s, n_s]) )
         end for
      end if
      if C = Dγ // edge is incomplete
         for each D → γ in grammar
            PUSH (agenda, (D → •γ, [k, k], [n_e, n_e]) )
         end for
      end if
   end if
end procedure
```

Figure 4: Extended chart parsing algorithm used to parse word graphs. Fundamental rule and predict procedures are updated to handle word graphs in a bidirectional manner.

3.5 Unification-based chart parsing

The grammar rules are implemented using *Lexical Functional Grammar* (LFG) paradigm. The primary data structure to represent the features and values is a directed acyclic graph (dag). The system also includes an expressive Boolean formalism, used to represent functional equations to access, inspect or modify features or feature sets in the dag. Complex feature structures (e.g. lists, sets, strings, and conglomerate lists) can be associated with lexical entries and grammatical categories using inheritance operations. Unification is used as the fundamental mechanism to integrate information from lexical entries into larger grammatical constituents.

The constituent structure (c-structure) represents the composition of syntactic constituents for a phrase. It is the term used for parse tree in LFG. The functional structure (f-structure) is the representation of grammatical functions in LFG. Attribute-value-matrices are used to describe f-structures. A sample c-structure and the corresponding f-structures in English are shown in Figure 5. For simplicity, many details and feature values are not given. The dag containing the information originated from the lexicon and the information extracted from morphological analysis is shown on the leaf levels of the parse tree in Figure 5. The final dag corresponding to the root node is built during the parsing process in cascaded unification operations specified in the grammar rules.

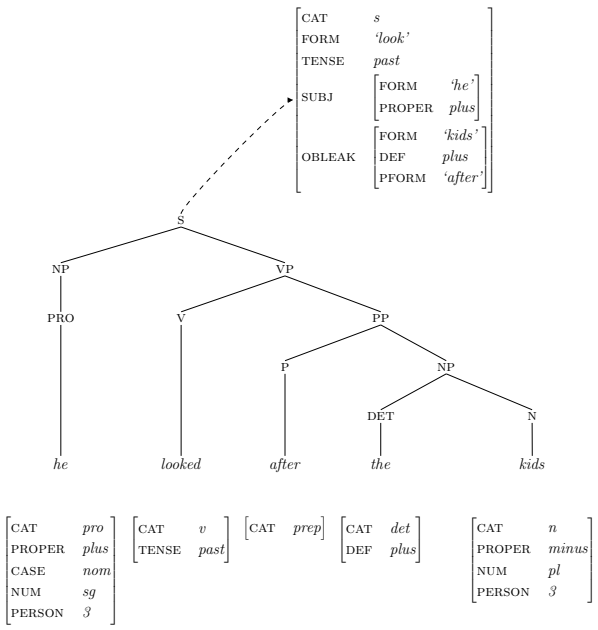

Figure 5: The c-structure and the associated f-structures.

3.6 Parse evaluation and recovery

After all rules are executed and no more edges are left in the agenda, the chart parsing process ends and parse evaluation begins. The chart is searched for complete edges with the final symbol of the grammar (e.g. SBAR) as their category. Any such edge spanning the entire input represents the full parse. If there is no such edge then the parse recovery process takes control.

If the input sentence is ambiguous, then, at the end of parsing, there will multiple parse trees in the chart that span the entire input. Similarly, a grammar built with insufficient constraints can lead to multiple parse trees. In this case, all possible edges are evaluated for *completeness* and *coherence* (Bresnan, 1982) starting from the edge with the highest weight. A parse tree is *complete* if all the functional roles (SUBJ, OBJ, SCOMP etc.) governed by the verb are actually present in the c-structure; it is *coherent* if all the functional roles present are actually governed by the verb. The parse tree that is evaluated as *complete* and *coherent* and has the highest weight is selected for further processing.

In general, a parsing process is said to be successful if a parse tree can be built according to the input sentence. The building of the parse tree fails when the sentence is ungrammatical. For the goal of MT, however, a parse tree is required for the transfer stage and the generation stage even if the input is not grammatical. Therefore, for any input sentence, a corresponding parse tree is built at the end of parsing.

If parsing fails, i.e. if all rules are exhausted and no successful parse tree has been produced, then the system tries to recover from the failure by creating a tree like structure. Appropriate complete edges in the chart are used for this purpose. The idea is to piece together all partial parses for the input sentence, so that the number of constituent edges is minimum and the score of the final tree is maximum. While selecting the constituents, overlapping edges are not chosen.

The recovery process functions as follows:

- The whole chart is traversed and a complete edge is inserted into a candidate list if it has the highest score for that start-end position. If two edges have the same score, then the farthest one to the leaf level is preferred.

- The candidate list is traversed and a combination with the minimum number of constituents is selected. The edges with the widest span get into the winning combination.

- The c-structures and f-structures of the edges in the winning combination are joined into a whole c-structure and f-structure which represent the final parse tree for the input.

4 Experiments

The experiments carried out in this paper are run on word graphs based on 1993 benchmark tests for the ARPA spoken language program (Pallett et al., 1994). In the large-vocabulary continuous speech recognition (CSR) tests reported by Pallett et al. (1994), Wall Street Journal-based CSR corpus material was made use of. Those tests intended to measure basic speaker-independent performance on a 64K-word read-speech test set which consists of 213 utterances. Each of the 10 different speakers provided 20 to 23 utterances. An acoustic model and a trigram language model is trained using Wall Street Journal data by Chelba (2000) who also generated the 213 word graphs used in the current experiments. The word graphs, referred as HUB-1 data set, contain both the acoustic scores and the trigram language model scores. Previously, the same data set was used in other

studies (Chelba, 2000; Roark, 2001; Hall, 2005) for language modeling task in ASR.

4.1 N-best list pruning

The 213 word graphs in the HUB-1 data set are pruned as described in Section 3 in order to prepare them for chart parsing. AT&T toolkit (Mohri et al., 1998) is used for determinization and minimization of the word graphs and for n-best path extraction. Prior to feeding in the word graphs to the FSM tools, the acoustic model and the trigram language model in the original lattices are combined into a single score using Equation 1, where $\mathcal{S}$ represents the combined score of an arc, $\mathcal{A}$ is the acoustic model (AM) score, $\mathcal{L}$ is the language model (LM) score, α is the AM scale factor and β is the LM scale factor.

$$\mathcal{S} = \alpha\mathcal{A} + \beta\mathcal{L} \quad (1)$$

Figure 6 depicts the word error rates for the first-best hypotheses obtained heuristically by using $\alpha = 1$ and β values from 1 to 25. The lowest WER (13.32) is achieved when α is set to 1 and β to 15. This result is close with the findings from Hall (2005) who reported to use 16 as the LM scale factor for the same data set. WER score for LM-only was 26.8 where in comparison the AM-only score was 29.64. The results imply that the language model has more predicting power over the acoustic model in the HUB-1 lattices. For the rest of the experiments, we used 1 and 15 as the acoustic model and language model scale factors, respectively.

4.2 Word graph accuracy

Using the scale factors found in the previous section we built N-best word graphs for different N values. In order to measure the word graph accuracy we constructed the FSM for reference hypotheses, $\mathcal{F}_{Ref}$, and we took the intersection of all the word graphs with the reference FSM. Table 1 lists the word graph accuracy rate for different N values. For example, an accuracy rate of 30.98 denotes that 66 word graphs out of 213 contain the correct sentences. The accuracy rate for the original word graphs in the data set (last row in Table 1) is 66.67 which indicates that only 142 out of 213 contain the reference sentence. That is, in 71 of the instances, the reference sentence is not included in the untouched word graph. The accurate rates express the maximum sentence error rate (SER) that can be achieved for the data set.

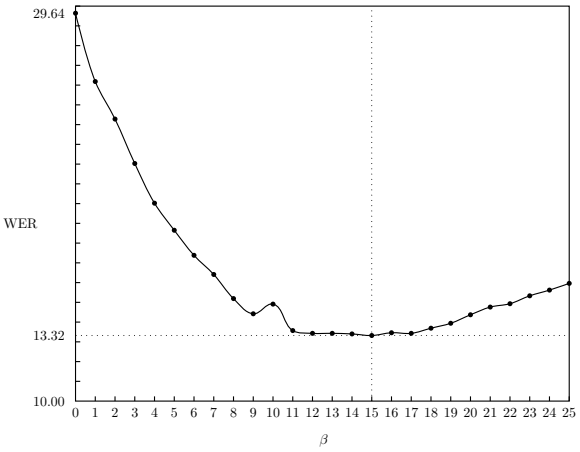

Figure 6: WER for HUB-1 first-best hypotheses obtained using different language-model scaling factors and $\alpha = 1$. The unsteadiness of the WER for $\beta = 10$ needs further investigation.

Table 1: Word graph accuracy for different N values in the data set with 213 word graphs.

N	Accuracy	N	Accuracy
1	30.98	60	59.15
10	51.17	70	59.15
20	56.34	80	59.15
30	58.22	90	60.10
40	59.15	100	60.10
50	59.15	*full*	66.67

4.3 Linguistic Resources

The English grammar used in the chart parser contained 20 morphology analysis rules and 225 syntax analysis rules. All the rules and the unification constraints are implemented in LFG formalism. The number of rules to model the language grammar is quite few compared to probabilistic CFGs which contain more than 10 000 rules. The monolingual analysis lexicon consists of 40 000 lexical entries.

4.4 Chart parsing experiment

We conducted experiments to compare the performance for N-best list parsing and N-best word graph parsing. Compared to the N-best list approach, in N-best word graph parsing approach, the shared edges are processed only once for all hypotheses. This saves a lot on the number of

Table 2: Number of complete and incomplete edges generated for the NIST HUB-1 data set using different approaches.

Approach	Hypotheses	Complete edges	Incomplete edges
N-best list	4869	798 K	12.125 M
	1	164	2490
N-best	4869	150.8 K	1.662 M
word graph	1	31	341

complete and incomplete edges generated during parsing. Hence, the overall processing time required to analyze the hypotheses are reduced. In an N-best list approach, where each hypothesis is processed separately in the analyzer, there are different charts and different parsing instances for each sentence hypothesis. Shared words in different sentences are parsed repeatedly and same edges will be created at each instance.

Table 2 represents the number of complete and incomplete edges generated for the NIST HUB-1 data set. For each hypothesis, 164 complete edges and 2490 incomplete edges are generated on the average in the N-best list approach. In the N-best word graph approach, the average number of complete edges and incomplete edges reduced to 31 and 341, respectively. The decrease is 81.1% in complete edges and 86.3% in incomplete edges for the NIST HUB-1 data set. The profit introduced in the number of edges by using the N-best word graph approach is immense.

4.5 Language modeling experiment

In order to compare this approach to previous language modeling approaches we used the same data set. Table 3 lists the WER for the NIST HUB-1 data set for different approaches including ours. The N-best word graph approach presented in this paper scored 12.6 WER and still needs some improvements. The English analysis grammar that was used in the experiments was designed to parse typed text containing punctuation information. The speech data, however, does not contain any punctuation. Therefore, the grammar has to be adjusted accordingly to improve the WER. Another common source of error in parsing is because of unnormalized text.

Table 3: WER taken from Hall and Johnson (2003) for various language models on HUB-1 lattices in addition to our approach presented in the fifth row.

Model	WER
Charniak Parser (Charniak, 2001)	11.8
Attention Shifting	11.9
(Hall and Johnson, 2004)	
PCFG (Hall, 2005)	12.0
A* decoding (Xu et al., 2002)	12.3
N-best word graph *(this study)*	**12.6**
PCFG (Roark, 2001)	12.7
PCFG (Hall and Johnson, 2004)	13.0
40m-word trigram	13.7
(Hall and Johnson, 2003)	
PCFG (Hall and Johnson, 2003)	15.5

5 Conclusions

The primary aim of this research was to propose a new and efficient method for integrating an SR system with an MT system employing a chart parser. The main idea is to populate the initial chart parser with the word graph that comes out of the SR component.

This paper presents an attempt to blend statistical SR systems with rule-based MT systems. The goal of the final assembly of these two components was to achieve an enhanced Speech Translation (ST) system. Specifically, we propose to parse the word graph generated by the SR module inside the rule-based parser. This approach can be generalized to any MT system employing chart parsing in its analysis stage. In addition to utilizing rule-based MT in ST, this study used word graphs and chart parsing with new extensions.

For further improvement of the overall system, our future studies include the following: 1. Adjusting the English syntax analysis rules to handle spoken text which does not include any punctuation. 2. Normalization of the word arcs in the input lattice to match words in the analysis lexicon.

Acknowledgments

Thanks to Jude Miller and Mirna Miller for providing the Arabic reference translations. We also thank Brian Roark and Keith Hall for providing the test data, and Nagendra Goel, Cem Bozşahin, Ayşenur Birtürk and Tolga Çiloğlu for their valuable comments.

References

J. Bresnan. 1982. Control and complementation. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 282–390. MIT Press, Cambridge, MA.

J.-C. Chappelier, M. Rajman, R. Aragues, and A. Rozenknop. 1999. Lattice parsing for speech recognition. In *TALN'99*, pages 95–104.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics.

Ciprian Chelba. 2000. *Exploiting Syntactic Structure for Natural Language Modeling*. Ph.D. thesis, Johns Hopkins University.

Lee-Feng Chien, K. J. Chen, and Lin-Shan Lee. 1990. An augmented chart data structure with efficient word lattice parsing scheme in speech recognition applications. In *Proceedings of the 13th conference on Computational linguistics*, pages 60–65, Morristown, NJ, USA. Association for Computational Linguistics.

Lee-Feng Chien, K. J. Chen, and Lin-Shan Lee. 1993. A best-first language processing model integrating the unification grammar and markov language model for speech recognition applications. *IEEE Transactions on Speech and Audio Processing*, 1(2):221–240.

Yen-Lu Chow and Salim Roukos. 1989. Speech understanding using a unification grammar. In *ICAASP'89: Proc. of the International Conference on Acoustics, Speech, and Signal Processing*, pages 727–730. IEEE.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proceedings of ACL-08: HLT*, pages 1012–1020, Columbus, Ohio, June. Association for Computational Linguistics.

Keith Hall and Mark Johnson. 2003. Language modelling using efficient best-first bottom-up parsing. In *ASR'03: IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 507–512. IEEE.

Keith Hall and Mark Johnson. 2004. Attention shifting for parsing speech. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 40, Morristown, NJ, USA. Association for Computational Linguistics.

Keith Hall. 2005. *Best-First Word Lattice Parsing: Techniques for Integrated Syntax Language Modeling*. Ph.D. thesis, Brown University.

Martin Kay. 1986. Algorithm schemata and data structures in syntactic processing. *Readings in natural language processing*, pages 35–70.

C. D. Manning and H. Schütze. 2000. *Foundations of Statistical Natural Language Processing*. The MIT Press.

Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. 1998. A rational design for a weighted finite-state transducer library. In *WIA '97: Revised Papers from the Second International Workshop on Implementing Automata*, pages 144–158, London, UK. Springer-Verlag.

Hermann Ney. 1991. Dynamic programming parsing for context-free grammars in continuous speech recognition. *IEEE Transactions on Signal Processing*, 39(2):336–340.

A. Paeseler. 1988. Modification of Earley's algorithm for speech recognition. In *Proceedings of the NATO Advanced Study Institute on Recent advances in speech understanding and dialog systems*, pages 465–472, New York, NY, USA. Springer-Verlag New York, Inc.

David S. Pallett, Jonathan G. Fiscus, William M. Fisher, John S. Garofolo, Bruce A. Lund, and Mark A. Przybocki. 1994. In *HLT '94: Proceedings of the workshop on Human Language Technology*, pages 49–74, Morristown, NJ, USA. Association for Computational Linguistics.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Andreas Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201.

Masaru Tomita. 1986. An efficient word lattice parsing algorithm for continuous speech recognition. *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '86.*, 11:1569–1572.

Hans Weber. 1994. Time synchronous chart parsing of speech integrating unification grammars with statistics. In *Proceedings of the Eighth Twente Workshop on Language Technology*, pages 107–119.

Phil Woodland. 2000. *HTK Speech Recognition Toolkit*. Cambridge University Engineering Department, `http://htk.eng.cam.ac.uk`.

Peng Xu, Ciprian Chelba, and Frederick Jelinek. 2002. A study on richer syntactic dependencies for structured language modeling. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 191–198. Association for Computational Linguistics.