

COMBINING STOCHASTIC AND GRAMMAR-BASED LANGUAGE PROCESSING WITH FINITE-STATE EDIT MACHINES

Michael Johnston, Srinivas Bangalore

AT&T Labs - Research
180 Park Ave, Florham Park, NJ 07932

johnston@research.att.com, srini@research.att.com

ABSTRACT

Multimodal grammars provide an expressive formalism for rapid bootstrapping of finite-state mechanisms for multimodal integration and understanding. These mechanisms align speech and gesture inputs, readily scale to processing of lattice inputs, and enable recovery from speech and gesture recognition errors through mutual compensation. However, in common with other handcrafted mechanisms, they can be brittle with respect to unexpected, erroneous, or disfluent inputs. In this paper, we show how the robustness of stochastic language models can be combined with the expressiveness of multimodal grammars by adding a finite-state edit machine to the multimodal language processing cascade. We evaluate the effectiveness of the approach in a multimodal conversational system (MATCH) which provides restaurant and subway information on a speech and pen enabled mobile device.

1. INTRODUCTION

Multimodal grammars provide an expressive mechanism for quickly creating language processing capabilities for multimodal interfaces supporting input modes such as speech and pen [1, 2]. They support composite multimodal inputs by aligning speech input (words) and gesture input (represented as sequence of gesture symbols) while expressing the relation between the speech and gesture input and their combined semantic representation. In [3], we have shown that such grammars can be compiled into finite-state transducers enabling effective processing of lattice input from speech and gesture recognition and mutual compensation for errors and ambiguities.

However, like other approaches based on hand-crafted grammars, multimodal grammars can be brittle with respect to extra-grammatical or erroneous input. For example, one limitation is that a language model directly projected from the speech portion of a hand-crafted multimodal grammar will not be able to recognize any strings that are not accepted by the grammar. A corpus-driven stochastic language model (SLM) with smoothing can be built in order

to overcome this limitation. This corpus can be the data collected from using a multimodal system or data sampled from the multimodal grammar [4, 5]. Although the corpus-driven language model might recognize a user's utterance correctly, the recognized utterance may not be assigned a semantic representation since it may not be in the multimodal grammar. [4] introduced the idea of using an additional stage in the finite-state multimodal language processing cascade in which the recognized string is edited to match the closest string that can be accepted by the grammar. Essentially the idea is that, if the recognized string cannot be parsed, to determine which in-grammar string it is most like. For example, in Figure 1, the recognized string is mapped to the closest string in the grammar by deletion of the words *restaurants* and *in*.

ASR:	show cheap restaurants thai places in in chelsea
Edits:	show cheap ϵ thai places in ϵ chelsea
Grammar:	show cheap thai places in chelsea

Fig. 1. Editing Example

In this paper, we develop further this edit-based approach to finite-state multimodal language understanding and show how when appropriately tuned it can provide a substantial improvement in concept accuracy. We present evaluation results on data from the MATCH multimodal conversational system [6]. In Sections 2 and 3, we briefly describe the MATCH application and the finite-state approach to multimodal language understanding. Section 4 describes a series of progressively more sophisticated edit machines. Section 5 describes our experimental evaluation, Section 6 discusses the benefits of our hybrid stochastic/grammar-based approach and its relation to previous work, and Section 7 concludes the paper.

2. THE MATCH APPLICATION

MATCH (Multimodal Access To City Help) is a working city guide and navigation system that enables mobile users to access restaurant and subway information for New York

City and Washington, D.C. [6]. The user interacts with an interface displaying restaurant listings and a dynamic map showing locations and street information. The inputs can be speech, drawing/pointing on the display with a stylus, or synchronous multimodal combinations of the two modes. The user can ask for the review, cuisine, phone number, address, or other information about restaurants and subway directions to locations. The system responds with graphical labels on the display, synchronized with synthetic speech output. For example, if the user says *phone numbers for these two restaurants* and circles two restaurants as in Figure 2 [a], the system will draw a callout with the restaurant name and number and say, for example *Time Cafe can be reached at 212-533-7000*, for each restaurant in turn (Figure 2 [b]).



Fig. 2. MATCH Example

3. FINITE-STATE MULTIMODAL UNDERSTANDING

Our approach to integrating and interpreting multimodal inputs [6] is an extension of the finite-state approach previously proposed [1, 3, 2]. In this approach, a declarative multimodal grammar captures both the structure and the interpretation of multimodal and unimodal commands. The grammar consists of a set of context-free rules. The multimodal aspects of the grammar become apparent in the terminals, each of which is a triple $W:G:M$, consisting of speech (words, W), gesture (gesture symbols, G), and meaning (meaning symbols, M). The multimodal grammar encodes not just multimodal integration patterns but also the syntax of speech and gesture, and the assignment of meaning, here represented in XML. The symbol SEM is used to abstract over specific content such as the set of points delimiting an area or the identifiers of selected objects [6]. In Figure 3, we present a small simplified fragment from the MATCH application capable of handling information seeking requests such as *phone for these three restaurants*. The epsilon symbol (ϵ) indicates that a stream is empty in a given terminal.

In the example above where the user says *phone for these two restaurants* while circling two restaurants (Figure 2 [a]), assume the speech recognizer returns the lattice in Figure 4 (Speech). The gesture recognition component

CMD	\rightarrow	$\epsilon:\epsilon:\langle cmd \rangle$	INFO	$\epsilon:\epsilon:\langle /cmd \rangle$		
INFO	\rightarrow	$\epsilon:\epsilon:\langle type \rangle$	TYPE	$\epsilon:\epsilon:\langle /type \rangle$		
		for: $\epsilon:\epsilon$	$\epsilon:\epsilon:\langle obj \rangle$	DEICNP	$\epsilon:\epsilon:\langle /obj \rangle$	
TYPE	\rightarrow	phone: $\epsilon:\epsilon$:phone		review: $\epsilon:\epsilon$:review		
DEICNP	\rightarrow	DDETPL	$\epsilon:\epsilon$:area: $\epsilon:\epsilon$	sel: $\epsilon:\epsilon$	NUM	HEADPL
DDETPL	\rightarrow	these: $G:\epsilon$		those: $G:\epsilon$		
HEADPL	\rightarrow	restaurants:rest: $\langle rest \rangle$	$\epsilon:\epsilon$:SEM:SEM	$\epsilon:\epsilon:\langle /rest \rangle$		
NUM	\rightarrow	two:2: ϵ		three:3: ϵ ... ten:10: ϵ		

Fig. 3. Multimodal grammar fragment

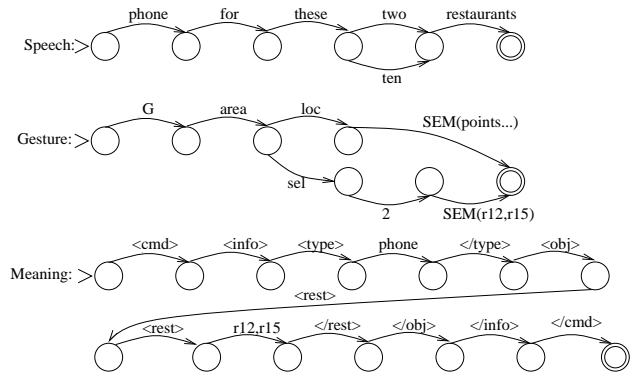


Fig. 4. Multimodal Example

also returns a lattice (Figure 4, Gesture) indicating that the user's ink is either a selection of two restaurants or a geographical area. In Figure 4 (Gesture) the specific content is indicated in parentheses after SEM . This content is removed before multimodal parsing and integration and replaced afterwards. For detailed explanation of our technique for abstracting over and then re-integrating specific gestural content and our approach to the representation of complex gestures see [6, 7]. The multimodal grammar (Figure 3) expresses the relationship between what the user said, what they drew with the pen, and their combined meaning, in this case Figure 4 (Meaning). The meaning is generated by concatenating the meaning symbols and replacing SEM with the appropriate specific content: $\langle cmd \rangle \langle info \rangle \langle type \rangle$ phone $\langle /type \rangle \langle obj \rangle \langle rest \rangle [r12,r15] \langle /rest \rangle \langle /obj \rangle \langle /info \rangle \langle /cmd \rangle$.

For use in our system, the multimodal grammar is compiled into a cascade of finite-state transducers [1, 3, 6]. As a result, processing of lattice inputs from speech and gesture processing is straightforward and efficient. In general a context-free grammar can be approximated by an FSA [8, 9]. The sample grammar presented here is in fact regular so no approximation is needed in this case. The transition symbols of the approximated FSA are the terminals of the context-free grammar and in the case of the multimodal CFG as defined above, these terminals contain three components, W , G and M . The resulting three-tape finite

state device composes two input streams into a single output semantic representation stream. While a three-tape finite-state automaton is feasible in principle [10], currently available tools for finite-state language processing [11, 12] only support finite-state transducers (FSTs) (two tapes). In our implementation, the three tape device is simulated using a pair of transducers: $G:W$, which is used to align speech and gesture, and $G_W:M$, which takes a composite alphabet of speech and gesture symbols as input and outputs meaning. The gesture lattice G and speech lattice W are composed with $G:W$ and the result is factored into an FSA $G.W$ which is composed with $G_W:M$ to derive the meaning lattice M (See [3, 1] for more details).

4. ROBUST EDIT-BASED UNDERSTANDING

A corpus trained SLM with smoothing is more effective at recognizing what the user says, but this will not help system performance if coupled directly to a grammar-based understanding system which can only assign meanings to in-grammar utterances. In order to overcome the possible mismatch between the user’s input and the language encoded in the multimodal grammar (λ_g), we introduce a weighted finite-state edit transducer to the multimodal language processing cascade. This transducer coerces the set of strings (\mathcal{S}) encoded in the lattice resulting from ASR (λ_S) to closest strings in the grammar that can be assigned an interpretation. We are interested in the string with the least costly number of edits ($argmin$) that can be assigned an interpretation by the grammar¹. This can be achieved by composition (\circ) of transducers followed by a search for the least cost path through a weighted transducer as shown below.

$$s^* = argmin_{s \in \mathcal{S}} \lambda_S \circ \lambda_{edit} \circ \lambda_g \quad (1)$$

We first describe the machine introduced in [4] (*Basic Edit*) then go on to describe a smaller edit machine with higher performance (*4-edit*) and an edit machine which incorporates additional heuristics (*Smart edit*).

4.1. Basic edit

Our baseline, the edit machine described in [4], is essentially a finite-state implementation of the algorithm to compute the Levenshtein distance. It allows for unlimited insertion, deletion, and substitution of any word for another (Figure 5). The costs of insertion, deletion, and substitution are set as equal, except for members for classes such as price (*expensive*), cuisine (*turkish*) etc., which are assigned a higher cost for deletion and substitution.

¹We note that the closest string according to the edit metric may not be the closest string in meaning

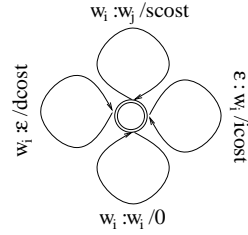


Fig. 5. Basic Edit Machine

4.2. 4-edit

Basic edit is effective in increasing the number of strings that are assigned an interpretation [4] but is quite large (15mb, 1 state, 978120 arcs) and adds an unacceptable amount of latency (5s on average). In order to overcome this performance problem we experimented with revising the topology of the edit machine so that it allows only a limited number of edit operations (at most four) and removed the substitution arcs, since they give rise to $O(|\Sigma|^2)$ arcs. For the same grammar, the resulting edit machine is about 300K with 4 states and 16796 arcs. The topology of the *4-edit* machine is shown in Figure 6.

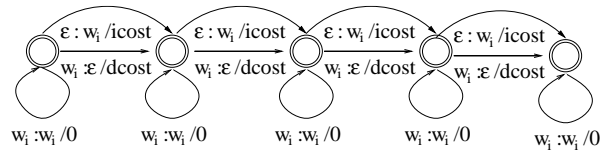


Fig. 6. 4-edit machine

4.3. Smart edit

Smart edit is a 4-edit machine which incorporates a number of additional heuristics and refinements to improve performance:

1. **Deletion of SLM-only words:** Arcs were added to the edit transducer to allow for free deletion of any words in the SLM training data which are not found in the grammar. For example, *listings in thai restaurant listings in midtown* \rightarrow *thai restaurant in midtown*.
2. **Deletion of doubled words:** A common error observed in SLM output was doubling of monosyllabic words. For example: *subway to the cloisters* recognized as *subway to to the cloisters*. Arcs were added to the edit machine to allow for free deletion of any short word when preceded by the same word.
3. **Extended variable weighting of words:** Insertion and deletion costs were further subdivided from two to three classes: a low cost for ‘dispensable’ words, (e.g. *please, would, looking, a, the*), a high cost for

special words (slot fillers, e.g. *chinese, cheap, downtown*), and a medium cost for all other words, (e.g. *restaurant, find*).

4. **Auto completion of place names:** It is unlikely that grammar authors will include all of the different ways to refer to named entities such as place names. For example, if the grammar includes *metropolitan museum of art* the user may just say *metropolitan museum*. These changes can involve significant numbers of edits. A capability was added to the edit machine to complete partial specifications of place names in a single edit. This involves a closed world assumption over the set of place names. For example, if the only *metropolitan museum* in the database is the *metropolitan museum of art* we assume that we can insert *of art* after *metropolitan museum*. The algorithm for construction of these auto-completion edits enumerates all possible substrings (both contiguous and non-contiguous) for place names. For each of these it checks to see if the substring is found in more than one semantically distinct member of the set. If not, an edit sequence is added to the edit machine which freely inserts the words needed to complete the placename. Figure 7 illustrates one of the edit translations that is added for the place name *metropolitan museum of art*. The algorithm which generates the autocomplete edits also generates new strings to add to the place name class for the SLM (expanded class). In order to limit overapplication of the completion mechanism substrings starting in prepositions (*of art* \rightarrow *metropolitan museum of art*) or involving deletion of parts of abbreviations are not considered for edits (*b c building* \rightarrow *n b c building*).

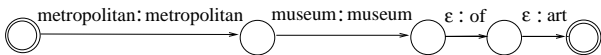


Fig. 7. Auto-completion Edits

Note that the application-specific structure and weighting of SmartEdit (3,4 above) can be derived automatically: 4. runs on the placename list for the new application and the classification in 3. is primarily determined by which words correspond to fields in the underlying application database.

5. EXPERIMENTAL EVALUATION

To evaluate the system, we collected a corpus of multimodal utterances for the MATCH domain in a laboratory setting from a set of sixteen first time users (8 male, 8 female). A total of 833 user interactions (218 multimodal / 491 speech-only / 124 pen-only) resulting from six sample task scenarios were collected and annotated for speech transcription, gesture, and meaning [13]. These scenarios involved finding

restaurants of various types and getting their names, phone numbers, addresses, or reviews, and getting subway directions between locations. The data collected was conversational speech where the users gestured and spoke freely.

Since we are concerned here with editing errors out of disfluent, misrecognized or unexpected speech, we report results on the 709 inputs that involve speech (491 unimodal speech and 218 multimodal). Since there are only a small number of scenarios performed by all users, we partitioned the data six ways by scenario. This ensures that the specific tasks in the test data for each partition are not also found in the training data for that partition. For each scenario we built a class-based trigram language model using the other five scenarios as training data. Averaging over the six partitions, ASR sentence accuracy was 49%² and word accuracy was 73.4%.

For the purpose of evaluating concept accuracy, we developed an approach similar to [14, 15] in which computing concept accuracy is reduced to comparing strings representing meaning. We extract a sorted flat list of attribute value pairs that represents the core contentful concepts of each command from the XML output. The example in Figure 4 yields the following meaning representation for concept accuracy: `cmd:info type:phone object:selection`.

In order to evaluate the concept accuracy provided by each of the different edit machines, for each partition of the data we first composed the output from speech recognition with the edit machine and the multimodal grammar, flattened the meaning representation, and computed the *exact string match accuracy* between the flattened meaning representation and the reference meaning representation. We then averaged the concept string accuracy over all six partitions. We also calculated for each machine the average time in seconds taken to process the input on a Linux server (Intel Xeon PIII 1.4GHz with 4GB RAM). The results are tabulated in Figure 8. The columns show the concept string accuracy, the relative improvement over the the baseline of no edits, and the average time taken to compose the edit machine with the input.

We first examined concept accuracy on editing the one best output from speech recognition. Compared to the baseline of 38.9% concept accuracy without edits (No Edits), *Basic Edit* gave a relative improvement of 32%, yielding 51.5% concept accuracy. Interestingly, in addition to reducing latency by 90%, *4-edit* also improved concept accuracy (53%) compared to *Basic Edit*. The heuristics in *Smart Edit* brought the concept string accuracy to 60.2%, a 55% improvement over the baseline. Smart Edit (exp) shows the concept accuracy of *Smart Edit* running on input

²Less than the 57.1% sentence accuracy with the 10-fold random partitioning we used in our earlier work [4], this is to be expected though since with random partitioning there would have been instances of the same scenario in the training data for that partition.

	ConAcc	Rel Impr	Avg Time
No edits	38.9%	0%	0s
Basic edit	51.5%	32%	5.25s
4-edit	53.0%	36%	0.53s
Smart edit	60.2%	55%	0.68s
Smart edit (exp)	59.7%	54%	0.68s
Smart edit (exp,lattice)	62.0%	59%	1.08s
Smart edit (lattice)	63.2%	62%	1.06s

Fig. 8. Results of 6-fold cross validation

from an ASR model with the expanded classes required for auto completion of place names. Inspection of individual partitions showed that while the expanded classes did allow for the correction of errors on place names, the added perplexity in the ASR model from expanding classes resulted in errors elsewhere and an overall drop in concept accuracy of 0.5% compared to *Smart Edit* without expanded classes.

We then examined the performance of *Smart edit* on lattice input (Smart edit(lattice)). In terms of concept accuracy, taking lattice input improved performance from 60.2% to 63.2%. Lattice input also improved the performance of Smart edit with the expanded classes from 59.7% to 62%.

Overall, compared to *Basic edit* from [4], the revisions to the edit machine provided a 23% relative improvement in accuracy (51.5% to 63.2%) and an 80% reduction in average processing time. Note that the runtime system latency can be avoided for in-grammar utterances, by only applying the edit transduction as a fall back for utterances which fail to compose directly with the multimodal grammar. In our test data, 38.9% of the utterances were in-grammar and so would not require the edit transduction.

Application-specific information is utilized to construct the edit machines described above and set the weights of edit operations. We are currently investigating methods to learn edit operations and their weights using data-driven techniques [16]. However, given the limited data available in multimodal applications we believe that a combination of data-driven and hand-tuned edit operations and weighting will be the most effective.

6. RELATED WORK AND DISCUSSION

Although we are not aware of any attempts to address the issue of robust understanding specifically in the context of multimodal systems, this issue has been of great interest in the context of speech-only conversational systems [17, 18, 19, 20]. In these approaches, grammar coverage problems and parsing of extra-grammatical utterances are typically addressed by retrieving fragments from the parse chart and operations that combine fragments to derive a meaning for the recognized utterance. In contrast, we have presented an approach that achieves robust multimodal utterance under-

standing using an edit-based transducer within a finite-state-based interpreter which can apply directly to lattice inputs and does not need to combine fragments from a parser. The issue of combining rule-based and data-driven approaches has recently received more attention [21, 22, 23]. The issue of recovering from ASR errors using an edit post process is also addressed in [24]. Our approach differs in that, rather than simply trying to make the recognized word string more like the reference, our goal is to determine the appropriate meaning for the user input, even for inputs which are recognized perfectly but lie outside the understanding grammar. Also, our approach differs in that it is represented as a finite-state transducer making it straightforward and computationally inexpensive to constrain search based on the whole set of strings accepted by the grammar and to process lattice input from ASR.

If, as we are finding, the combination of stochastic recognition models with grammar-based understanding can provide the kind of robust performance needed for usable systems, the question which arises is whether to pursue this hybrid approach or train the multimodal understanding component itself, using techniques such as the multimodal classifier described in [4]. There are a number of advantages to the hybrid approach:

1. The multimodal grammar is highly expressive and supports definition of either simple or complex functions relating input strings to meaning representations.
2. The multimodal grammar provides an alignment between speech and gesture input and enables multimodal integration of content from different modes.
3. It is straightforward to add new commands to the grammar or change the representation of existing commands. Only the ASR model needs retraining, and data for the ASR model can be migrated from another related domain or derived through grammar sampling [4].
4. Most importantly, the task of annotating and aligning speech data with meaning representation is complex and expensive. In contrast, in this approach, if data is available, all that is needed is transcription of the speech. If no data is available then grammar sampling can be used and no annotation of data is needed.

7. CONCLUSION

We have shown how finite-state edit machines enable integration of stochastic speech recognition with handcrafted multimodal grammars. The resulting multimodal understanding system is significantly more robust achieving 63.2% concept accuracy on data collected from novice first time users of a multimodal conversational system. This is a substantial 62% relative improvement in performance compared to 38.9% concept accuracy without edits, and brings finite-state multimodal understanding towards the performance level needed for usable systems. Also, this level of performance

was achieved with a significantly smaller edit machine that requires 80% less processing time on average.

Acknowledgments: Thanks to Patrick Ehlen, Mazin Gilbert, Helen Hastie, Candy Kamm, Amanda Stent, Marilyn Walker, and Steve Whittaker for their contributions to MATCH. Thanks to Narendra Gupta for his comments.

8. REFERENCES

- [1] M. Johnston and S. Bangalore, "Finite-state multimodal parsing and understanding," in *Proceedings of COLING*, Saarbrücken, Germany, 2000.
- [2] M. Johnston and S. Bangalore, "Finite-state multimodal integration and understanding," *Journal of Natural Language Engineering*, vol. 11, no. 2, pp. 159–187, 2005.
- [3] S. Bangalore and M. Johnston, "Tight-coupling of multimodal language processing with speech recognition," in *Proceedings of ICSLP*, Beijing, China, 2000.
- [4] S. Bangalore and M. Johnston, "Balancing data-driven and rule-based approaches in the context of a multimodal conversational system," in *Proceedings of HLT-NAACL*, 2004.
- [5] S. Bangalore and M. Johnston, "Robust multimodal understanding," in *Proceedings of ICASSP*, Montreal, 2004, pp. 417–420.
- [6] M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor, "MATCH: An architecture for multimodal dialog systems," in *Proceedings of ACL*, Philadelphia, 2002.
- [7] M. Johnston, S. Bangalore, A. Stent, G. Vasireddy, and P. Ehlen, "Multimodal language processing for mobile information access," in *Proceedings of ICSLP*, Denver, Colorado, 2002.
- [8] F. C.N. Pereira and M. D. Riley, "Speech recognition by composition of weighted finite automata," in *Finite State Devices for Natural Language Processing*, E. Roche and Y. Schabes, Eds. MIT Press, Cambridge, Massachusetts, 1997.
- [9] M-J. Nederhof, "Regular approximations of CFLs: A grammatical view," in *Proceedings of International Workshop on Parsing Technology*, Boston, 1997.
- [10] A.L. Rosenberg, "On n-tape finite state acceptors," *FOCS*, pp. 76–81, 1964.
- [11] M. Mohri, F. C. N. Pereira, and M. Riley., *A rational design for a weighted finite-state transducer library*, Number 1436 in Lecture notes in computer science. Springer, Berlin ; New York, 1998.
- [12] G. van Noord, "FSA utilities: A toolbox to manipulate finite-state automata," in *Automata Implementation. Lecture Notes in Computer Science 1260*, D. Wood, D. Raymond, and S. Yu, Eds. Springer Verlag, 1997.
- [13] P. Ehlen, M. Johnston, and G. Vasireddy, "Collecting mobile multimodal data for MATCH," in *Proceedings of ICSLP*, Denver, Colorado, 2002.
- [14] A. Ciaramella, "A Prototype Performance Evaluation Report," Tech. Rep. WP8000-D3, Project Esprit 2218 SUNDIAL, 1993.
- [15] M. Boros, W. Eckert, F. Gallwitz, G. Görz, G. Hanrieder, and H. Niemann, "Towards Understanding Spontaneous Speech: Word Accuracy vs. Concept Accuracy," in *Proceedings of ICSLP*, Philadelphia, 1996.
- [16] E. S. Ristad and P. N. Yianilos, "Learning string-edit distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 5, pp. 522–532, 1998.
- [17] J. Dowding, J. M. Gawron, D. E. Appelt, J. Bear, L. Cherny, R. Moore, and D. B. Moran, "GEMINI: A natural language system for spoken-language understanding," in *Proceedings of ACL*, 1993, pp. 54–61.
- [18] S. Seneff, "A relaxation method for understanding spontaneous speech utterances," in *Proceedings, Speech and Natural Language Workshop*, CA, 1992.
- [19] J. Allen, D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent, "An architecture for a generic dialogue shell," *JNLE*, vol. 6, no. 3, 2000.
- [20] A. Lavie, *GLR*: A Robust Grammar-Focused Parser for Spontaneously Spoken Language*, Ph.D. thesis, Carnegie Mellon University, 1996.
- [21] Y.Y. Wang, M. Mahajan, and X. Huang, "Unified Context-Free Grammar and N-Gram Model for Spoken Language Processing," in *Proceedings of ICASSP*, 2000.
- [22] M. Rayner and B. A. Hockey, "Transparent combination of rule-based and data-driven approaches in speech understanding," in *Proceedings of EAACL*, 2003.
- [23] Y. Wang and A. Acero, "Combination of cfg and n-gram modeling in semantic grammar learning," in *Proceedings of Eurospeech*, Geneva, 2003.
- [24] E. K. Ringger and J. F. Allen, "A fertility channel model for post-correction of continuous speech recognition," in *Proceedings of ICSLP*, 1996.