Edit Machines for Robust Multimodal Language Processing

Srinivas Bangalore
AT&T Labs-Research
180 Park Ave
Florham Park, NJ 07932
srini@research.att.com

Michael Johnston
AT&T Labs-Research
180 Park Ave
Florham Park, NJ 07932
johnston@research.att.com

Abstract

Multimodal grammars provide an expressive formalism for multimodal integration and understanding. However, hand-crafted multimodal grammars can be brittle with respect to unexpected, erroneous, or disfluent inputs. Spoken language (speech-only) understanding systems have addressed this issue of lack of robustness of hand-crafted grammars by exploiting classification techniques to extract fillers of a frame representation. In this paper, we illustrate the limitations of such classification approaches for multimodal integration and understanding and present an approach based on edit machines that combine the expressiveness of multimodal grammars with the robustness of stochastic language models of speech recognition. We also present an approach where the edit operations are trained from data using a noisy channel model paradigm. We evaluate and compare the performance of the hand-crafted and learned edit machines in the context of a multimodal conversational system (MATCH).

1 Introduction

Over the years, there have been several multimodal systems that allow input and/or output to be conveyed over multiple channels such as speech, graphics, and gesture, for example, *put that there* (Bolt, 1980), CUBRICON (Neal and Shapiro, 1991), QuickSet (Cohen et al., 1998), SmartKom (Wahlster, 2002), Match (Johnston et al., 2002). Multimodal integration and interpretation for such interfaces is elegantly expressed using multimodal grammars (Johnston and Bangalore, 2000). These grammars support composite multimodal inputs by *aligning* speech input (words) and gesture input (represented as sequences of gesture symbols) while expressing the relation between the speech and gesture input and their combined semantic representation. In (Bangalore and Johnston, 2000; Johnston and Bangalore, 2005), 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, erroneous and disfluent input. For speech recognition, a corpus-driven stochastic language model (SLM) with smoothing or a combination of grammar-based and n-gram model (Bangalore and Johnston, 2004; Wang et al., 2002) can be built in order to overcome the brittleness of a grammar-based language model. Although the corpus-driven language model might recognize a user's utterance correctly, the recognized utterance may not be assigned a semantic representation by the multimodal grammar if the utterance is not part of the grammar.

There have been two main approaches to improving robustness of the understanding component in the spoken language understanding literature. First, a parsing-based approach attempts to recover partial parses from the parse chart when the input cannot be parsed in its entirety due to noise, in order to construct a (partial) semantic representation (Dowding et al., 1993; Allen et al., 2001; Ward, 1991). Second, a classification-based approach views the problem of understanding as extracting certain bits of information from the input. It attempts to classify the utterance and identifies substrings of the input as slot-filler values to construct a frame-like semantic representation. Both approaches have shortcomings. Although in the first approach, the grammar can encode richer semantic representations, the method for combining the fragmented parses is quite ad hoc. In the second approach, the robustness is derived from training classifiers on annotated data, this data is very expensive to collect and annotate, and the semantic representation is fairly limited. Furthermore, it is not clear how to extend this approach to apply on lattice input – an important requirement for multimodal processing.

An alternative to these approaches is to edit the recognized string to match the closest string that can be accepted by the grammar. Essentially the idea is that, if the recognized string cannot be parsed, then we 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

Figure 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 also explore learning edits from data and present an approach of modeling this process as a machine translation problem. We learn a model to translate from out of grammar or misrecognized language (such as 'ASR:' above) to the closest language the system can understand ('Grammar:' above). To this end, we adopt techniques from statistical machine translation (Brown et al., 1993; Och and Ney, 2003) and use statistical alignment to learn the edit patterns. Here we evaluate these different techniques on data from the MATCH multimodal conversational system (Johnston et al., 2002) but the same techniques are more broadly applicable to spoken language systems in general whether unimodal or multimodal.

The layout of the paper is as follows. In Sections 2 and 3, we briefly describe the MATCH application and the finite-state approach to multimodal language understanding. In Section 4, we discuss the limitations of the methods used for robust understanding in spoken language understanding literature. In Section 5 we present our approach to building hand-crafted edit machines. In Section 6, we describe our approach to learning the edit operations using a noisy channel paradigm. In Section 7, we describe our experimental evaluation.

2 MATCH: A Multimodal 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. (Johnston et al., 2002). 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]).

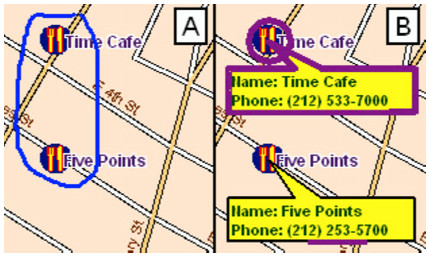

Figure 2: MATCH Example

3 Finite-state Multimodal Understanding

Our approach to integrating and interpreting multimodal inputs (Johnston et al., 2002) is an extension of the finite-state approach previously proposed in (Bangalore and Johnston, 2000; Johnston and Bangalore, 2005). 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 (Johnston et al., 2002). 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 also returns a lattice (Figure 4, Gesture) indicating that the user's ink

CMD → ϵ:ϵ:<cmd> INFO ϵ:ϵ:</cmd>
INFO → ϵ:ϵ:<type> TYPE ϵ:ϵ:</type>
for:ϵ:ϵ ϵ:ϵ:<obj> DEICNP ϵ:ϵ:</obj>
TYPE → phone:ϵ:phone | review:ϵ:review
DEICNP → DDETPL ϵ:area:ϵ ϵ:sel:ϵ NUM HEADPL
DDETPL → these:G:ϵ | those:G:ϵ
HEADPL → restaurants:rest:<rest>ϵ:*SEM*:*SEM*
ϵ:ϵ:</rest>
NUM → two:2:ϵ | three:3:ϵ ... ten:10:ϵ

Figure 3: Multimodal grammar fragment

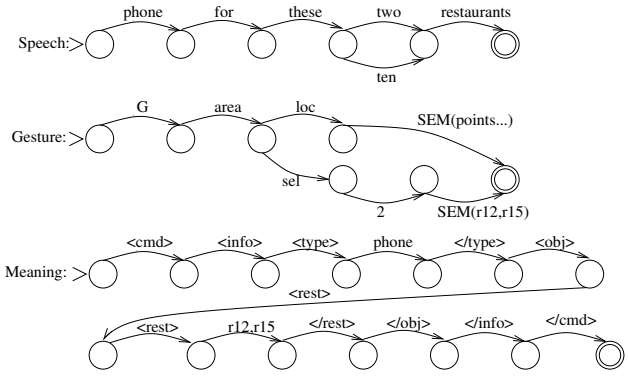

Figure 4: Multimodal Example

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 (Johnston et al., 2002). 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: *<cmd> <info> <type>* phone *</type> <obj> <rest> [r12,r15] </rest> </obj> </info> </cmd>*.

For use in our system, the multimodal grammar is compiled into a cascade of finite-state transducers (Johnston and Bangalore, 2000; Johnston et al., 2002; Johnston and Bangalore, 2005). As a result, processing of lattice inputs from speech and gesture processing is straightforward and efficient.

3.1 Meaning Representation for Concept Accuracy

The hierarchically nested XML representation above is effective for processing by the backend application, but is not well suited for the automated determination of the performance of the language understanding mechanism. We adopt an approach, similar to (Ciaramella, 1993; Boros et al., 1996), in which the meaning representation, in our case XML, is transformed into a sorted flat list of attribute-value pairs indicating the core contentful concepts of each command. The example above yields:

`cmd : info type : phone object : selection.` (1)

This allows us to calculate the performance of the understanding component using the same string matching metrics used for speech recognition accuracy. *Concept Sentence Accuracy* measures the number of user inputs for which the system got the meaning completely right (this is called Sentence Understanding in (Ciaramella, 1993)).

4 Robust Understanding

Robust understanding has been of great interest in the spoken language understanding literature. The issue of noisy output from the speech recognizer and disfluencies that are inherent in spoken input make it imperative for using mechanisms to provide robust understanding. As discussed in the introduction, there are two approaches to addressing robustness – partial parsing approach and classification approach. We have explored the classification-based approach to multimodal understanding in earlier work. We briefly present this approach and discuss its limitations for multimodal language processing.

4.1 Classification-based Approach

In previous work (Bangalore and Johnston, 2004), we viewed multimodal understanding as a sequence of classification problems in order to determine the *predicate* and *arguments* of an utterance. The meaning representation shown in (1) consists of an predicate (the command attribute) and a sequence of one or more argument attributes which are the parameters for the successful interpretation of the user's intent. For example, in (1), `cmd : info` is the predicate and `type : phone object : selection` is the set of arguments to the predicate.

We determine the predicate (c^*) for a N token multimodal utterance (S_1^N) by maximizing the posterior probability as shown in Equation 2.

$$c^* = \underset{c}{argmax}\ Pr(c \mid S_1^N) \quad (2)$$

We view the problem of identifying and extracting arguments from a multimodal input as a problem of associating each token of the input with a specific tag that encodes the label of the argument and the span of the argument. These tags are drawn from a tagset which is constructed by

extending each argument label by three additional symbols I, O, B, following (Ramshaw and Marcus, 1995). These symbols correspond to cases when a token is inside (I) an argument span, outside (O) an argument span or at the boundary of two argument spans (B) (See Table 1).

User Utterance	cheap thai upper west side
Argument Annotation	<price> cheap < /price> <cuisine> thai </cuisine> <place> upper west side </place>
IOB Encoding	cheap_price<B> thai_cuisine<B> upper_place<I> west_place<I> side_place<I>

Table 1: The {I,O,B} encoding for argument extraction.

Given this encoding, the problem of extracting the arguments is a search for the most likely sequence of tags (T^*) given the input multimodal utterance S_1^N as shown in Equation (3). We approximate the posterior probability $Pr(T \mid S_1^N)$ using independence assumptions as shown in Equation (4).

$$T^* = \underset{T}{argmax} \ Pr(T \mid S_1^N) \quad (3)$$

$$\approx \underset{T}{argmax} \prod_i Pr(t_i \mid S_{i-n}^{i+n}, t_{i-1}, t_{i-2}) \quad (4)$$

Owing to the large set of features that are used for predicate identification and argument extraction, we estimate the probabilities using a classification model. In particular, we use the Adaboost classifier (Freund and Schapire, 1996) wherein a highly accurate classifier is build by combining many "weak" or "simple" base classifiers f_i, each of which may only be moderately accurate. The selection of the weak classifiers proceeds iteratively picking the weak classifier that correctly classifies the examples that are misclassified by the previously selected weak classifiers. Each weak classifier is associated with a weight (w_i) that reflects its contribution towards minimizing the classification error. The posterior probability of $Pr(c \mid x)$ is computed as in Equation 5.

$$Pr(c \mid x) = \frac{1}{(1 + e^{-2 * \sum_i w_i * f_i(x)})} \quad (5)$$

4.2 Limitations of this approach

Although, we have shown that the classification approach works for unimodal and simple multimodal inputs, it is not clear how this approach can be extended to work on lattice inputs. Multimodal language processing requires the integration and joint interpretation of speech and gesture input. Multimodal integration requires alignment of the speech and gesture input. Given that the input modalities are both noisy and can receive multiple within-modality interpretations (e.g. a circle could be an "O" or an area gesture); it is necessary for the input to be represented as a multiplicity of hypotheses, which can be most compactly represented as a lattice. The multiplicity of hypotheses is also required for exploiting the mutual compensation between the two modalities as shown in (Oviatt, 1999; Bangalore and Johnston, 2000). Furthermore, in order to provide the dialog manager the best opportunity to recover the most appropriate meaning given the dialog context, we construct a lattice of semantic representations instead of providing only one semantic representation.

In the multimodal grammar-based approach, the alignment between speech and gesture along with their combined interpretation is utilized in deriving the multimodal finite-state transducers. These transducers are used to create a gesture-speech aligned lattice and a lattice of semantic interpretations. However, in the classification-based approach, it is not as yet clear how alignment between speech and gesture would be achieved especially when the inputs are lattice and how the aligned speech-gesture lattices can be processed to produce lattice of multimodal semantic representations.

5 Hand-crafted Finite-State Edit Machines

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 ($\lambda_{\mathcal{S}}$) 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 grammar[1]. 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.

$$s^* = \underset{s \in \mathcal{S}}{argmin} \ \lambda_{\mathcal{S}} \circ \lambda_{edit} \circ \lambda_g \quad (6)$$

We first describe the edit machine introduced in (Bangalore and Johnston, 2004) (*Basic Edit*) then go on to describe a smaller edit machine with higher performance (*4-edit*) and an edit machine

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

which incorporates additional heuristics (*Smart edit*).

5.1 Basic edit

Our baseline, the edit machine described in (Bangalore and Johnston, 2004), 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 of classes such as price (*cheap, expensive*), cuisine (*turkish*) etc., which are assigned a higher cost for deletion and substitution.

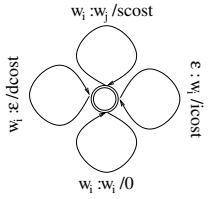

Figure 5: Basic Edit Machine

5.2 4-edit

Basic edit is effective in increasing the number of strings that are assigned an interpretation (Bangalore and Johnston, 2004) 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 $O(|\sum|^2)$ arcs. For the same grammar, the resulting edit machine is about 300K with 4 states and 16796 arcs and the average latency is (0.5s). The topology of the *4-edit* machine is shown in Figure 6.

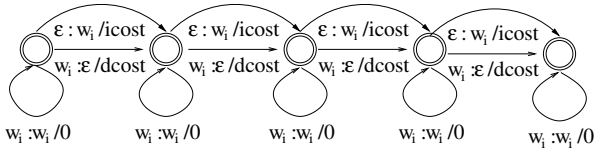

Figure 6: 4-edit machine

5.3 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* → *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 transductions 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 over-application of the completion mechanism substrings starting in prepositions (*of art* → *metropolitan museum of art*) or involving deletion of parts of abbreviations are not considered for edits (*b c building* → *n b c building*).

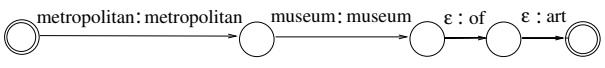

Figure 7: Auto-completion Edits

The average latency of *SmartEdit* is 0.68s. 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.

6 Learning Edit Patterns

In the previous section, we described an edit approach where the weights of the edit operations have been set by exploiting the constraints from the underlying application. In this section, we discuss an approach that learns these weights from data.

6.1 Noisy Channel Model for Error Correction

The edit machine serves the purpose of *translating* user's input to a string that can be assigned a meaning representation by the grammar. One of the possible shortcomings of the approach described in the preceding section is that the weights for the edit operations are set heuristically and are crafted carefully for the particular application. This process can be tedious and application-specific. In order to provide a more general approach, we couch the problem of error correction in the noisy channel modeling framework. In this regard, we follow (Ringger and Allen, 1996; Ristad and Yianilos, 1998), however, we encode the error correction model as a weighted Finite State Transducer (FST) so we can directly edit ASR input lattices. Furthermore, unlike (Ringger and Allen, 1996), the language grammar from our application filters out edited strings that cannot be assigned an interpretation by the multimodal grammar. Also, while in (Ringger and Allen, 1996) the goal is to translate to the reference string and improve recognition accuracy, in our approach the goal is to translate in order to get the reference meaning and improve concept accuracy.

We let S_g be the string that can be assigned a meaning representation by the grammar and S_u be the user's input utterance. If we consider S_u to be the noisy version of the S_g, we view the decoding task as a search for the string S_g^* that maximizes the following equation.

$$S_g^* = \underset{S_g}{argmax}\ P(S_u, S_g) \tag{7}$$

We then use a Markov approximation (trigram for our purposes) to compute the joint probability $P(S_u, S_g)$.

$$S_g^* = \underset{S_g}{argmax} \prod P(S_u^i, S_g^i | S_u^{i-1}, S_u^{i-2}, S_g^{i-1}, S_g^{i-2}) \tag{8}$$

where $S_u = S_u^1 S_u^2 \ldots S_u^n$ and $S_g = S_g^1 S_g^2 \ldots S_g^m$.

In order to compute the joint probability, we need to construct an alignment between tokens (S_u^i, S_g^i). We use the viterbi alignment provided by GIZA++ toolkit (Och and Ney, 2003) for this purpose. We convert the viterbi alignment into a *bilanguage* representation that pairs words of the string S_u with words of S_g. A few examples of bilanguage strings are shown in Figure 8. We compute the joint n-gram model using a language modeling toolkit (Goffin et al., 2005). Equation 8 thus allows us to edit a user's utterance to a string that can be interpreted by the grammar.

show:show me:me the:ϵ map:ϵ of:ϵ midtown:midtown
no:ϵ find:find me:me french:french restaurants:around downtown:downtown
I:ϵ need:ϵ subway:subway directions:directions

Figure 8: A few examples of bilanguage strings

6.2 Deriving Translation Corpus

Since our multimodal grammar is implemented as a finite-state transducer it is fully reversible and can be used not just to provide a meaning for input strings but can also be run in reverse to determine possible input strings for a given meaning. Our multimodal corpus was annotated for meaning using the multimodal annotation tools described in (Ehlen et al., 2002). In order to train the translation model we build a corpus that pairs the reference speech string for each utterance in the training data with a target string. The target string is derived in two steps. First, the multimodal grammar is run in reverse on the reference meaning yielding a lattice of possible input strings. Second, the closest string in the lattice to the reference speech string is selected as the target string.

6.3 FST-based Decoder

In order to facilitate editing of ASR lattices, we represent the edit model as a weighted finite-state transducer. We first represent the joint n-gram model as a finite-state acceptor (Allauzen et al., 2004). We then interpret the symbols on each arc of the acceptor as having two components – a word from user's utterance (input) and a word from the edited string (output). This transformation makes a transducer out of an acceptor. In doing so, we can directly compose the editing model with ASR lattices to produce a weighted lattice of edited strings. We further constrain the set of

edited strings to those that are interpretable by the grammar. We achieve this by composing with the language finite-state acceptor derived from the multimodal grammar as shown in Equation 5. Figure 9 shows the input string and the resulting output after editing with the trained model.

Input: I'm trying to find african restaurants
that are located west of midtown
Edited Output: find african around west midtown

Input: I'd like directions subway directions from
the metropolitan museum of art to the empire state building
Edited Output: subway directions from the
metropolitan museum of art to the empire state building

Figure 9: Edited output from the MT edit-model

7 Experiments and Results

To evaluate the approach, 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 (Ehlen et al., 2002). 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% and word accuracy was 73.4%.

In order to evaluate the understanding performance 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 (as described in Section 3.1), and computed the *exact string match accuracy* between the flattened meaning representation and the reference meaning representation. We then averaged this concept sentence accuracy measure over all six partitions.

	ConSentAcc
No edits	38.9%
Basic edit	51.5%
4-edit	53.0%
Smart edit	60.2%
Smart edit (lattice)	**63.2%**
MT-based edit (lattice)	51.3%
Classifier	34.0%

Figure 10: Results of 6-fold cross validation

The results are tabulated in Figure 10. The columns show the concept sentence accuracy (ConSentAcc) and the relative improvement over the the baseline of no edits. Compared to the baseline of 38.9% concept sentence accuracy without edits (No Edits), *Basic Edit* gave a relative improvement of 32%, yielding 51.5% concept sentence accuracy. *4-edit* further improved concept sentence accuracy (53%) compared to *Basic Edit*. The heuristics in *Smart Edit* brought the concept sentence accuracy to 60.2%, a 55% improvement over the baseline. Applying Smart edit to lattice input improved performance from 60.2% to 63.2%.

The *MT-based edit* model yielded concept sentence accuracy of 51.3% a 31.8% improvement over the baseline with no edits, but still substantially less than the edit model derived from the application database. We believe that given the lack of data for multimodal applications that an approach that combines the two methods may be most effective.

The *Classification* approach yielded only 34.0% concept sentence accuracy. Unlike *MT-based edit* this approach does not have the benefit of composition with the grammar to guide the understanding process. The low performance of the classifier is most likely due to the small size of the corpus. Also, since the training/test split was by scenario the specifics of the commands differed between training and test. In future work will explore the use of other classification techniques and try combining the annotated data with the grammar for training the classifier model.

8 Conclusions

Robust understanding is a crucial feature of a practical conversational system whether spoken or multimodal. There have been two main approaches to addressing this issue for speech-only dialog systems. In this paper, we present an alternative approach based on edit machines that is more suitable for multimodal systems where generally very little training data is available and data

is costly to collect and annotate. We have shown how edit machines enable integration of stochastic speech recognition with hand-crafted multimodal understanding grammars. The resulting multimodal understanding system is significantly more robust 62% relative improvement in performance compared to 38.9% concept accuracy without edits. We have also presented an approach to learning the edit operations and a classification-based approach. The *Learned edit* approach provides a substantial improvement over the baseline, performing similarly to the *Basic edit* machine, but does not perform as well as the application-tuned *Smart edit* machine. Given the small size of the corpus, the classification-based approach performs less well. This leads us to conclude that given the lack of data for multimodal applications a combined strategy may be most effective. Multimodal grammars coupled with edit machines derived from the underlying application database can provide sufficiently robust understanding performance to bootstrap a multimodal service and as more data become available data-driven techniques such as *Learned edit* and the *classification-based* approach can be brought into play.

References

C. Allauzen, M. Mohri, M. Riley, and B. Roark. 2004. A generalized construction of speech recognition transducers. In *ICASSP*, pages 761–764.

J. Allen, D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent. 2001. Towards Conversational Human-Computer Interaction. *AI Magazine*, 22(4), December.

S. Bangalore and M. Johnston. 2000. Tight-coupling of multimodal language processing with speech recognition. In *Proceedings of ICSLP*, pages 126–129, Beijing, China.

S. Bangalore and M. Johnston. 2004. Balancing data-driven and rule-based approaches in the context of a multimodal conversational system. In *Proceedings of HLT-NAACL*.

Robert A. Bolt. 1980. "put-that-there":voice and gesture at the graphics interface. *Computer Graphics*, 14(3):262–270.

M. Boros, W. Eckert, F. Gallwitz, G. Gŏrz, G. Hanrieder, and H. Niemann. 1996. Towards Understanding Spontaneous Speech: Word Accuracy vs. Concept Accuracy. In *Proceedings of ICSLP*, Philadelphia.

P. Brown, S.D. Pietra, V.D. Pietra, and R. Mercer. 1993. The Mathematics of Machine Translation: Parameter Estimation. *Computational Linguistics*, 16(2):263–312.

A. Ciaramella. 1993. A Prototype Performance Evaluation Report. Technical Report WP8000-D3, Project Esprit 2218 SUNDIAL.

Philip R. Cohen, M. Johnston, D. McGee, S. L. Oviatt, J. Pittman, I. Smith, L. Chen, and J. Clow. 1998. Multimodal interaction for distributed interactive simulation. In M. Maybury and W. Wahlster, editors, *Readings in Intelligent Interfaces*. Morgan Kaufmann Publishers.

J. Dowding, J. M. Gawron, D. E. Appelt, J. Bear, L. Cherny, R. Moore, and D. B. Moran. 1993. GEMINI: A natural language system for spoken-language understanding. In *Proceedings of ACL*, pages 54–61.

P. Ehlen, M. Johnston, and G. Vasireddy. 2002. Collecting mobile multimodal data for MATCH. In *Proceedings of ICSLP*, Denver, Colorado.

Y. Freund and R. E. Schapire. 1996. Experiments with a new boosting alogrithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156.

V. Goffin, C. Allauzen, E. Bocchieri, D. Hakkani-Tur, A. Ljolje, S. Parthasarathy, M. Rahim, G. Riccardi, and M. Saraclar. 2005. The at&t watson speech recognizer. In *Proceedings of ICASSP*, Philadelphia, PA.

M. Johnston and S. Bangalore. 2000. Finite-state multimodal parsing and understanding. In *Proceedings of COLING*, pages 369–375, Saarbrücken, Germany.

M. Johnston and S. Bangalore. 2005. Finite-state multimodal integration and understanding. *Journal of Natural Language Engineering*, 11(2):159–187.

M. Johnston, S. Bangalore, G. Vasireddy, A. Stent, P. Ehlen, M. Walker, S. Whittaker, and P. Maloor. 2002. MATCH: An architecture for multimodal dialog systems. In *Proceedings of ACL*, pages 376–383, Philadelphia.

J. G. Neal and S. C. Shapiro. 1991. Intelligent multi-media interface technology. In J. W. Sullivan and S. W. Tyler, editors, *Intelligent User Interfaces*, pages 45–68. ACM Press, Addison Wesley, New York.

F.J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

S. L. Oviatt. 1999. Mutual disambiguation of recognition errors in a multimodal architecture. In *CHI '99*, pages 576–583. ACM Press, New York.

L. Ramshaw and M. P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third Workshop on Very Large Corpora*, MIT, Cambridge, Boston.

E. K. Ringger and J. F. Allen. 1996. A fertility channel model for post-correction of continuous speech recognition. In *ICSLP*.

E. S. Ristad and P. N. Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532.

W. Wahlster. 2002. SmartKom: Fusion and fission of speech, gestures, and facial expressions. In *Proceedings of the 1st International Workshop on Man-Machine Symbiotic Systems*, pages 213–225, Kyoto, Japan.

Y. Wang, A. Acero, C. Chelba, B. Frey, and L. Wong. 2002. Combination of statistical and rule-based approaches for spoken language understanding. In *Proceedings of the ICSLP*, Denver, Colorado, September.

W. Ward. 1991. Understanding spontaneous speech: the phoenix system. In *ICASSP*.