

LEARNING EDIT MACHINES FOR ROBUST MULTIMODAL UNDERSTANDING

Michael Johnston, Srinivas Bangalore

AT&T Labs-Research, 180 Park Ave, Florham Park, NJ 07932, USA
{johnston, srini}@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. In previous work, we have shown 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. In this paper, we present an approach where the edits are trained from data using a noisy channel model paradigm. We evaluate this model and compare its performance against hand-crafted edit machines from our previous work in the context of a multimodal conversational system (MATCH).

1. INTRODUCTION

Multimodal interfaces allow input and/or output to be conveyed over multiple channels such as speech, graphics, and gesture [1, 2, 3, 4] Multimodal grammars provide an expressive mechanism for quickly creating language processing capabilities for multimodal interfaces supporting input modes such as speech and pen [5]. 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 [6], 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 extragrammatical or erroneous input. For recognition, 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 [7]. 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. [7] 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*. In [8], we developed further this edit-based approach to finite-state multimodal language understanding and show how when appropriately tuned based on the underlying application database it can provide a substantial improvement in concept accuracy.

In this paper, we explore learning edits from training data. This can be thought of as a machine translation problem where we want

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

to learn how 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 [9, 10] and use statistical alignment to learn the edit patterns. We evaluate this approach on data from the MATCH multimodal conversational system [11] and compare it to the handcrafted edit-based approach described in [8]. In Sections 2 and 3, we briefly describe the MATCH application and the finite-state approach to multimodal language understanding. Section 4 describes the hand-crafted edit machine approach. In Section 5, we describe our approach to learning the edit operations using a noisy channel paradigm. Section 6 describes our experimental evaluation, and Section 7 concludes the paper.

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. [11]. 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 [11] is an extension of the finite-state approach previously described in [6]. 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 [11]. 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.

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

Fig. 3. Multimodal grammar fragment

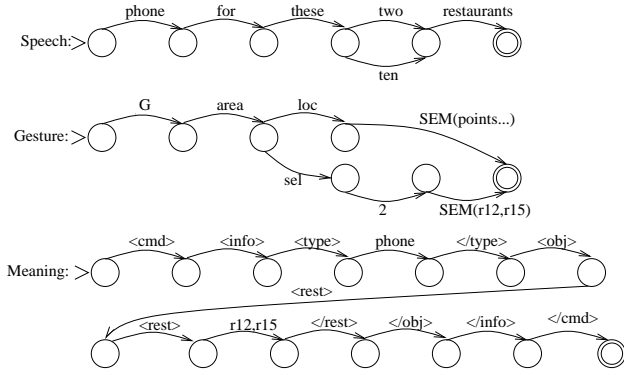


Fig. 4. Multimodal Example

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 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 [11]. 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 appro-

priate specific content: $\langle\text{cmd}\rangle \langle\text{info}\rangle \langle\text{type}\rangle \text{phone} \langle/\text{type}\rangle \langle\text{obj}\rangle \langle\text{rest}\rangle [r12,r15] \langle/\text{rest}\rangle \langle/\text{obj}\rangle \langle/\text{info}\rangle \langle/\text{cmd}\rangle$.

For use in our system, the multimodal grammar is compiled into a cascade of finite-state transducers [6, 11]. As a result, processing of lattice inputs from speech and gesture processing is straightforward and efficient.

4. HANDCRAFTED 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 (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 grammar. 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.

$$s^* = \underset{s \in S}{argmin} \lambda_s \circ \lambda_{edit} \circ \lambda_g \quad (1)$$

We first describe the machine introduced in [7] (*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*).

Our baseline (*Basic Edit*), 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 ($icost$), deletion ($dcost$), and substitution ($scost$) are set as equal, except for members of classes such as price (*expensive*), cuisine (*turkish*) etc., which are assigned a higher cost for deletion and substitution.

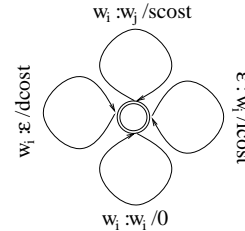


Fig. 5. Basic Edit Machine

Basic edit is effective in increasing the number of strings that are assigned an interpretation 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 revised 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 $O(|\sum l|^2)$ 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.

Our third edit machine, *Smart edit* is a 4-edit machine which incorporates a number of additional heuristics and refinements to improve performance.

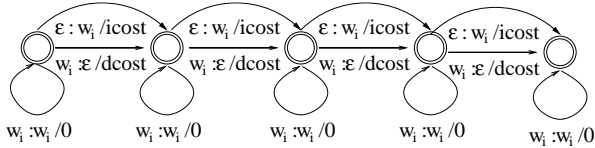


Fig. 6. 4-edit machine

- 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*.
- 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.
- Insertion (*icost*) and deletion (*dcost*) 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*).
- A capability was added to the edit machine to complete partial specifications of place names in a single edit. 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*.

Note that the application-specific structure and weighting of *Smart edit* can be derived automatically based on the underlying application database.

5. 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 (*MT edit*).

5.1. Noisy Channel Model for Error Correction

The edit machine serves the purpose of *translating* the 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. 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 [12, 13], however, we encode the error correction model as a weighted Finite State Transducer (FST) so we can directly edit ASR input lattices. Furthermore, unlike [12], the language grammar from our application filters out edited strings that cannot be assigned an interpretation by the multimodal grammar. Also, while in [12] 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. Note we formulate this as a joint probability maximization in contrast to the usual conditional probability maximization $P(S_g|S_u)$.

$$S_g^* = \underset{S_g}{\operatorname{argmax}} P(S_u, S_g) \quad (2)$$

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

$$S_g^* = \underset{S_g}{\operatorname{argmax}} \prod P(S_u^i, S_g^i | S_u^{i-1}, S_u^{i-2}, S_g^{i-1}, S_g^{i-2}) \quad (3)$$

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

In order to construct the word alignment (S_u^i, S_g^i) , we use the viterbi alignment provided by GIZA++ toolkit [10]. We convert the viterbi alignment into a *bilanguage* representation that pairs words of the string S_u with words of S_g . We compute the joint n-gram model using a language modeling toolkit [14]. Equation 3 thus allows us to edit a user’s utterance to a string that can be interpreted by the grammar.

5.2. 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 [15]. In order to train the translation model we build a corpus that pairs the transcribed 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.

5.3. 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 [16]. 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 in-terpretable by the grammar. We achieve this by composing with the language finite-state acceptor derived from the multimodal grammar as shown in Equation 1.

6. 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 [15]. 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.

	ConAcc	Rel Impr
No edits	38.9%	0%
Basic edit	51.5%	32%
4-edit	53.0%	36%
Smart edit	60.2%	55%
Smart edit (lattice)	63.2%	62%
MT edit	50.3%	29%

Fig. 7. Results of 6-fold cross validation

	ConAcc
Smart edit	67.4%
MT edit	61.1%

Fig. 8. Results of 10-fold cross validation

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%.

For the purpose of evaluating concept accuracy, we used an approach similar to [17, 18] 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 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.

The results are tabulated in Figure 7. The columns show the concept string accuracy and the relative improvement over the the baseline of no edits. 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. *4-edit* further 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. Applying Smart edit to lattice input improved performance from 60.2% to 63.2%.

The *MT edit* model yielded concept accuracy of 50.3% a 29% improvement over the baseline with no edits. It is very likely that there is insufficient training data for the *MT edit* model. Also, *MT edit* may not do well with the 6-fold split by scenario as it is more sensitive to the specifics of each scenario. To examine this further we re-ran the MT evaluation and *Smart edit* with a more typical 10-fold random split cross validation (Figure 8). The *MT edit* did substantially better with the 10-fold cross validation (61.1%) but still significantly less than the handcrafted edit model derived from the application database.

7. CONCLUSIONS

In previous work, we have shown how finite-state edit machines can dramatically improve the robustness of multimodal understanding. In our prior work the best performing edit machine incorporated heuristics derived from the underlying application database. In this paper, we take a different approach, viewing the editing as a translation process from language the system cannot handle to the closest language it can. The *MT edit* approach provides a substantial improvement over the baseline (29%), performing similarly to the *Basic edit* machine, but does not do as well as the application-tuned *Smart edit* machine. This leads us to conclude that, given the lack of data for multimodal applications, a combined strategy may be most effective. For initial deployment the underlying application database can be leveraged to build an edit machine to improve system performance. As data is collected the *MT edit* approach can be brought into play in order to further improve performance, as suggested by the 10-fold evaluation. In ongoing work we are examining techniques for combination of the data-driven and application-tuned edit approaches described here.

8. REFERENCES

- [1] E. André, "Natural language in multimedia/multimodal systems," in *Handbook of Computational Linguistics*, Ruslan Mitkov, Ed. OUP, 2002.
- [2] W. Wahlster, "SmartKom: Fusion and fission of speech, gestures, and facial expressions," in *Proceedings of the 1st International Workshop on Man-Machine Symbiotic Systems*, Kyoto, Japan, 2002, pp. 213–225.
- [3] S. L. Oviatt, "Multimodal interactive maps: Designing for human performance," in *Human-Computer Interaction*, 1997, pp. 93–129.
- [4] J. Cassell, "Embodied conversational agents: Representation and intelligence in user interface," in *AI Magazine*, 2001, vol. 22, pp. 67–83.
- [5] M. Johnston and S. Bangalore, "Finite-state multimodal parsing and understanding," in *Proceedings of COLING*, Saarbrücken, Germany, 2000, pp. 369–375.
- [6] M. Johnston and S. Bangalore, "Finite-state multimodal integration and understanding," *Journal of Natural Language Engineering*, vol. 11, no. 2, pp. 159–187, 2005.
- [7] 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.
- [8] M. Johnston and S. Bangalore, "Combining stochastic and grammar-based language processing with finite-state edit machines," in *Proceedings of ASRU 2005*, Cancun, Mexico, 2005.
- [9] P. Brown, S.D. Pietra, V.D. Pietra, and R. Mercer, "The Mathematics of Machine Translation: Parameter Estimation," *Computational Linguistics*, vol. 16, no. 2, pp. 263–312, 1993.
- [10] F.J. Och and H. Ney, "A systematic comparison of various statistical alignment models," *Computational Linguistics*, vol. 29, no. 1, pp. 19–51, 2003.
- [11] 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, pp. 376–383.
- [12] E. K. Ringger and J. F. Allen, "A fertility channel model for post-correction of continuous speech recognition," in *ICSLP*, 1996.
- [13] 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.
- [14] V. Goffin, C. Allauzen, E. Bocchieri, D. Hakkani-Tur, A. Ljolje, S. Parthasarathy, M. Rahim, G. Riccardi, and M. Saraclar, "The at&t watson speech recognizer," in *Proceedings of ICASSP*, Philadelphia, PA, 2005.
- [15] P. Ehlen, M. Johnston, and G. Vasireddy, "Collecting mobile multimodal data for MATCH," in *Proceedings of ICSLP*, Denver, Colorado, 2002.
- [16] C. Allauzen, M. Mohri, M. Riley, and B. Roark, "A generalized construction of speech recognition transducers," in *ICASSP*, 2004, pp. 761–764.
- [17] A. Ciaramella, "A Prototype Performance Evaluation Report," Tech. Rep. WP8000-D3, Project Esprit 2218 SUNDIAL, 1993.
- [18] 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.