IMPROVING SPOKEN LANGUAGE UNDERSTANDING USING WORD CONFUSION NETWORKS

Gokhan Tur, Jerry Wright, Allen Gorin, Giuseppe Riccardi, and Dilek Hakkani-Tür

AT&T Labs-Research,
180 Park Avenue, Florham Park, NJ, USA
{gtur,jwright,algor,dsp3,dtur}research.att.com

ABSTRACT

A natural language spoken dialog system includes a large vocabulary automatic speech recognition (ASR) engine, whose output is used as the input of a spoken language understanding component. Two challenges in such a framework are that the ASR component is far from being perfect and the users can say the same thing in very different ways. So, it is very important to be tolerant to recognition errors and some amount of orthographic variability. In this paper, we present our work on developing new methods and investigating various ways of robust recognition and understanding of an utterance. To this end, we exploit word-level confusion networks (sausages), obtained from ASR word graphs (lattices) instead of the ASR 1-best hypothesis. Using sausages with an improved confidence model, we decreased the call-type classification error rate for AT&T's *How May I Help You*^{SM} (*HMIHY*^{SM}) natural dialog system by 38%.

1. INTRODUCTION

Voice-based natural dialog systems enable customers to express what they want in spoken natural language. Such systems automatically extract the meaning from speech input and act upon what people actually say, in contrast to what one would like them to say, shifting the burden from users to the machine.

In a natural spoken dialog system, it is very important to be robust to ASR errors, since all the communication is in natural spoken language. Especially with telephone speech, the typical word error rate (WER) is around 30%. Consider the example 1-best ASR output, "*I have a question about my will.*" which erroneously selected "*will*" instead of "*bill*". Obviously, misrecognizing such a salient word will result in misunderstanding the whole utterance, although all other words are correct. Besides that, the understanding system should tolerate some amount of orthographic variability, since people say the same thing in different ways.

To this end, we exploit lattices of the utterances provided by the ASR instead of only using the best paths. A lattice is a directed graph of words, which can encode a large number of possible sentences. Using lattices as input to a spoken language understanding system is very promising because of the following:

- The oracle accuracy of lattices is much higher than for 1-best. The oracle accuracy is the accuracy of the path in a lattice closest to the transcriptions. For example, it is likely to have the correct word "*bill*" in the lattice for the above example.
- Lattices include multiple hypotheses of the recognized utterance, hence can be useful in tolerating some amount of orthographic variability.

In spoken language processing systems, lattices have been usually used for obtaining better word accuracies, by rescoring them.

In this study, we used a special kind of lattice, called word confusion networks, or sausages. Sausages consist concatenation of word sets, one for each word time interval. The general structure of the sausages and lattices are shown in Figure 1. The advantages of the sausages can be listed as follows:

- Since they force the competing words to be in the same group, they enforce the alignment of the words. This time alignment may be very useful in language processing.
- The words in the sausage come up with their posterior probabilities, which can be used as their confidence scores. This is basically the sum of the probabilities of all paths which contain that word. We use this confidence during understanding.
- Their memory sizes are about 1/100 of those of ASR lattices in our experiments, but according to our test set, they still have comparable oracle accuracy and even lower word error rate using the consensus hypothesis, which is the best path of a sausage.

In the literature sausages have been used for decreasing the word error rate or obtaining word confidences [1, 2]. As

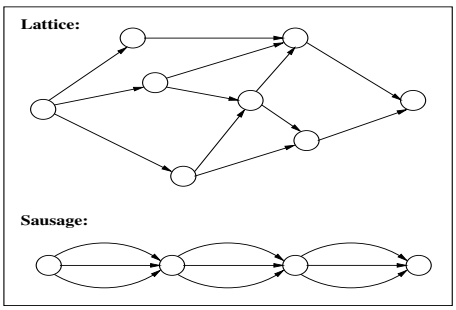

Fig. 1. Typical structures of lattices and sausages.

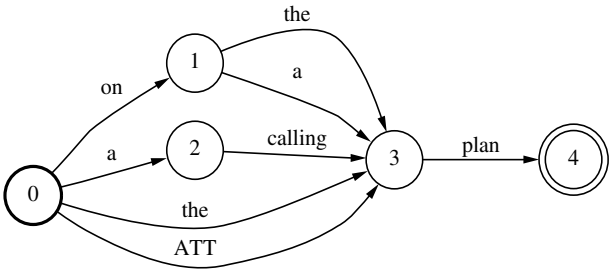

Fig. 2. An example salient grammar fragment.

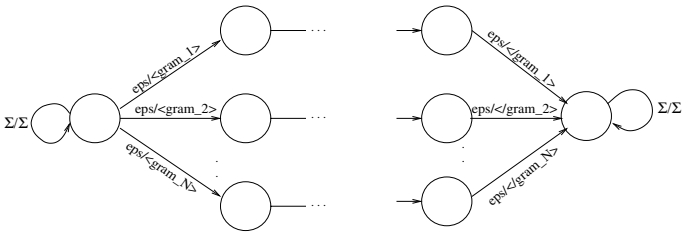

Fig. 3. FST of the SGFs. `<gram_n>` and `</gram_n>` tokens are used for marking the boundaries of a SGF. `eps` denotes for an epsilon transition.

far as we know, there is no prior work using sausages as input of a natural dialog system.

In this work, we have evaluated the idea of exploiting the sausages on the AT&T's *How May I Help You* (*HMIHY*) natural dialog system. In this system, users are asking questions about their bill, calling plans, etc. Within the SLU component, we are aiming at classifying the input telephone calls into 19 classes (call-types), such as *Billing Credit*, or *Calling Plans* [3].

In the following section, we present our algorithm. The confidence model used with the sausages is described in Section 3. Section 4 describes our experiments and results.

2. MATCHING SALIENT GRAMMAR FRAGMENTS IN SAUSAGES

The SLU component of HMIHY is a series of modules. First a preprocessor converts certain groups of words into a single token (e.g. converting tokens "A T and T" into "ATT".) Then, the salient grammar fragments are matched in the preprocessed utterance, and they are filtered and parsed for the classifier, the SLU confidence model adjusts the raw probabilities assigned to call-types using a confidence model [4].

HMIHY SLU classifier heavily depends on portions of input utterances, namely *salient phrases*, which are salient to some call-types. For example, in an input utterance like "*I would like to change oh about long distance service two one charge nine cents a minute*", the salient phrase "*cents a minute*" is strongly related to the call-type *Calling Plans*. In our previous work we have presented how we automatically acquire salient phrases from a corpus of transcribed and labeled training data and cluster them into salient grammar fragments (SGFs) in the format of finite state machines (FSMs) [4, 5]. Figure 2 shows an example salient grammar fragment.

In order to try our idea of using sausages, we change the matching module of the SLU. First we form a single finite state transducer (FST) of the SGFs given in Figure 3. This is nothing but a union of the FSTs of the SGFs which can be preceded or followed by any number of words. Note that, the boundaries of a salient phrase found are marked with `<gram_n>` and `</gram_n>` tokens, where n is the id of the salient grammar fragment, Σ denotes any word in the vocabulary, and `eps` denotes an epsilon transition, which is a transition without any input.

It is then straightforward to compose the FSM of the input utterance with this FST of the SGFs and enumerate the paths (where each contains exactly one salient phrase) in the resulting FST. For example, one path can be as follows:

"*I am* `<gram11>` *on the plan* `</gram11>` *of ...*"

We use the geometric mean of the confidence scores of the words in a salient phrase as the confidence score of the phrase. Then we use this phrase confidence score during classification.

If this composition is done naively, the number of resulting salient phrases would be huge (in the order of millions). While enumerating the phrases in the result, it produces one distinct match for each of the paths containing that phrase. This includes all the combinations of the words in the nodes of the sausage which are outside of the match. For example, the following two matches are considered to be different, although the difference is beyond the boundaries of the salient phrase:

"*I am* `<gram11>` *on the plan* `</gram11>` *of ...*"

"*I am* `<gram11>` *on the plan* `</gram11>` *on ...*"

Noting that, we are only interested in the id and boundaries of the salient phrases, we avoid this problem simply mapping all the words in the result to a single token, α. This is done easily by composing the FST of the SGFs with the

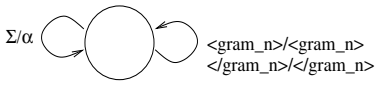

Fig. 4. FST mapping all words, Σ, to α. <gram_n> and </gram_n> tokens are used for marking the boundaries of a SGF.

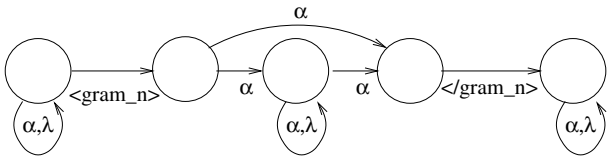

Fig. 5. FST used for avoiding epsilons (or λs) as the first or last token in a salient phrase. α denotes a word in the dictionary, and <gram_n> and </gram_n> tokens are used for marking the boundaries of a SGF.

FST shown in Figure 4.

Assuming that there is almost always an epsilon transition between two consecutive nodes of a sausage, the result contains a distinct path for each salient phrase with different number of epsilons used before or after that match. For example, the following two matches are considered to be different, although the only difference is that, the second word is eps, in other words, deleted.

"α α <gram11> α α α </gram11> α ..."

"α <gram11> α α α </gram11> α ..."

We have solved this problem by changing all the epsilons in the sausage to another token, λ, and then behaving λ as an epsilon. In order to do this, we put a λ/λ transition to every state of the FST of the SGFs, and insert λ^* expression between two tokens in the regular expressions (to allow any number of λs in-between) while forming the preprocessor. For example, the rule forming the token "*ATT*" has become as follows: $A\ \lambda^*\ (A|T|and|\lambda) +\ \lambda^*\ T\ \rightarrow\ ATT$

Because of the epsilons (or now λs) appearing as the very first or very last tokens of a salient phrase, it is possible to have the same salient phrase, occurring with different number of λs at these positions, as shown below:

"α λ <gram11> α α α </gram11> α ..."

"α <gram11> λ α α α </gram11> α ..."

In order to avoid this, we form an FST shown in Figure 5, and compose our FST of the SGFs with this. Note that, only α is allowed as the first and last token of a salient phrase. We use the FST after all these compositions as our new FST of the SGFs.

The previous preprocessor had been designed to handle a single string of words, not a lattice or sausage. Instead of modifying the existing preprocessor, we re-implemented the preprocessor in the form of a FST. Accordingly, we compose the sausage with this preprocessor first, and then with the FST of the SGFs. Figure 6 shows this general structure of matching salient grammar fragments from sausages.

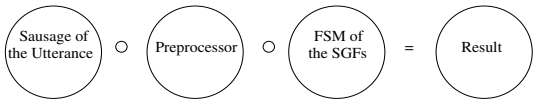

Fig. 6. Matching salient grammar fragments from sausages using a series of FST compositions. "o" denotes the composition operation.

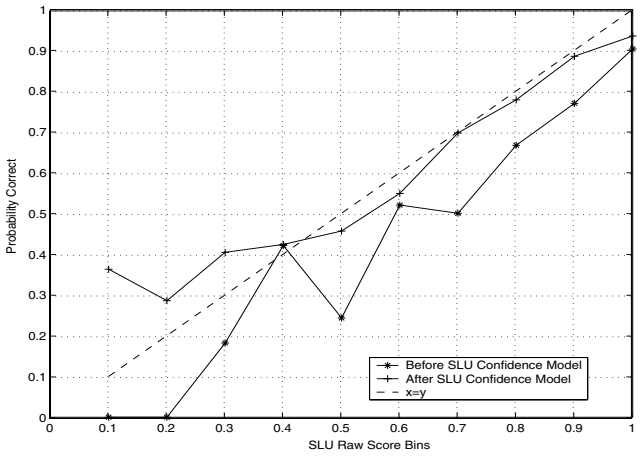

Fig. 7. Effect of the improved SLU confidence model. The aim is to adjust the raw SLU scores in order to make them as close as to the probability of being correct.

3. SLU CONFIDENCE MODEL

Given the sets of salient grammar fragments, the classifier assigns some raw scores to the call-types. In order to adjust these individual scores, p, so that they can actually give the probability of being correct, we use the following logistic regression formula using the length, l, and coverage (the percentage of the words occurring inside a salient phrase in an utterance), c, of that utterance:

$$p' = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \times p + \beta_2 \times l + \beta_3 \times c)}}$$

where the β values are learned from the training data, using Newton-Raphson Method [6]. Figure 7 shows the effect of the improved SLU confidence model for our test data.

4. EXPERIMENTS AND RESULTS

During our experiments, in order to evaluate the change in performance due to our method, we keep the ASR engine, the set of salient phrase grammars, and the SLU classifier unchanged. We used Mangu *et al.*'s algorithm to convert lattices into sausages [1]. For the HMIHY task, we present

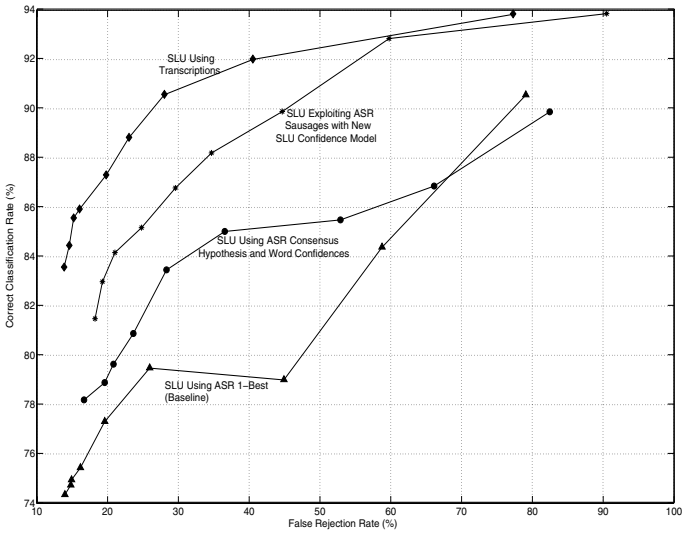

Fig. 8. Improvements in call classification performance.

our results in the form of an ROC curve, in which we vary the threshold to see the effects of false rejection and correct classification. The utterances for which call-type score is below that threshold are rejected (labeled as "Other") otherwise accepted (thus classified). The correct classification rate is the ratio of corrects among the accepted utterances. The ratio of the utterances that should be classified but are rejected is called as the false rejection rate. Hence, there is a trade-off between correct classification and false rejection by varying the rejection threshold, and the ultimate aim is to reach a correct classification rate of 100%, and false rejection rate of 0%.

The baseline performance uses the ASR 1-best output. Then we present our results, using word confidence scores obtained from the consensus hypothesis, using the whole sausage along with a new SLU confidence model, and finally using transcriptions. Note that the curve for the transcriptions uses our new SLU confidence model.

We use 1405 utterances from 1208 dialogs as our test data. All the utterances are responses to the first greeting in the dialog (e.g. "*Hello, This is AT&T. How May I Help You?*".) The word error rate for these utterances is 31.7% and the oracle accuracy of their lattices is 91.7%. The oracle accuracy of a lattice is defined as the accuracy of the path in the lattice closest to the transcription. This number decreases only to 90.6% using sausages, but note that the average size of a sausage is 100 times less than a lattice. Furthermore, the word error rate of the consensus hypothesis is 0.6% smaller, 31.1%.

As seen, first using salient phrase confidence scores, even using the consensus hypothesis of the sausage, the error rate decrease 18% on the ROC curve for a false rejection rate of 30%. This shows the importance of the confidence score of a salient phrase for our classifier. Then using the whole sausage along with the new SLU confidence model, we manage to increase this improvement to 38% for the same false rejection rate. The ROC curve using transcriptions verifies that we are now closer to the upper bound of the classifier.

5. CONCLUSIONS

We have presented algorithms for exploiting the word confusion networks (sausages) obtained from ASR lattices as input to a speech understanding system, and demonstrated its utility for AT&T's HMIHY natural dialog system. Note that this is applicable to any speech processing system using ASR 1-best as input. Exploiting the fact that sausages are nothing but finite state machines, we have converted the preprocessor and salient grammar fragments into finite state transducers and have reduced this task to consecutive compositions of finite state machines. Doing so, when compared with the performance of the baseline, 38% reduction in error rate is achieved, keeping the ASR and the SLU classifier otherwise unchanged.

6. REFERENCES

[1] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: word error minimization and other applications of confusion networks," *Computer Speech and Language*, vol. 14, no. 4, pp. 373–400, 2000.

[2] R. Gretter and G. Riccardi, "On-line learning of language models with word error probability distributions," in *Proceedings of the ICASSP*, Salt Lake City, Utah, May 2001, pp. 557–560.

[3] A. Gorin, J. Wright, G. Riccardi, A. Abella, and T. Alonso, "Semantic information processing of spoken language," in *Proceedings of the ATR Workshop on Multi-Lingual Speech Communication*, Kyoto, Japan, October 2000, pp. 13–16.

[4] J. Wright, A. Gorin, and G. Riccardi, "Automatic acquisition of salient grammar fragments for call-type classification," in *Proceedings of the Eurospeech*, Rhodes, Greece, September 1997, pp. 1419–1422.

[5] A.L. Gorin, G. Riccardi, and J.H. Wright, "How May I Help You?," *Speech Communication*, vol. 23, pp. 113–127, 1997.

[6] A. Agresti, *Categorical Data Analysis*, chapter 4, pp. 84–117, John Wiley and Sons, 1990.