Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Philadelphia, July 2002, pp. 273-280. Association for Computational Linguistics.

Extracting Clauses for Spoken Language Understanding in Conversational Systems

Narendra K. Gupta
AT&T Labs-Research
180 Park Avenue
Florham Park, NJ 07932
ngupta@research.att.com

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

Abstract

Spontaneous human utterances in the context of human-human and human-machine dialogs are rampant with dysfluencies, and speech repairs. Furthermore, when recognized using a speech recognizer, these utterances produce a sequence of words with no identification of clausal units. Such long strings of words combined with speech errors pose a difficult problem for spoken language parsing and understanding. In this paper, we address the issue of editing speech repairs as well as segmenting user utterances into clause units with a view of parsing and understanding spoken language utterances. We present generative and discriminative models for this task and present evaluation results on the human-human conversations obtained from the Switchboard corpus.

1 Introduction

Spoken language understanding in human-computer dialog systems must accommodate the characteristic features of human verbal communications. Most notable of such features are a) ungrammaticality b) presence of dysfluencies (Meteer and others, 1995) like repeats, restarts, and explicit/implied repairs c) absence of essential punctuation marks e.g. end of sentence and comma separated enumerations d) unpredictable word errors introduced by speech recognizers. These features make the word strings resulting from recognition or transcription of speech to be syntactically and semantically incoherent.

Current spoken dialog systems circumvent these problems by not attempting to parse the utterance but instead applying classification techniques to classify the entire input directly into a limited number of actions that the dialog system can perform. Such techniques work well when there are small number of actions, such as in the case of call routing systems (Gorin et al., 1997). They do not scale well for tasks that require very large number of classes, e.g. problem-solving tasks, or when fine-grained analysis of the user's utterance is needed. To develop deeper semantic representations of the utterances, we need to identify speech repairs as well as segment the input utterance into clauses – self-contained, syntactic units embodying a single concept in the form of a single subject-predicate set. In this paper, we present generative and discriminative models for the task of detecting sentence boundaries, identifying speech repairs and editing them out and identifying coordinating conjunctions to break the sentences into clausal units. Throughout this paper we will refer to this as a *clausifier*. We present some experiments on the human-human conversations obtained from the Switchboard corpus. The issue of parsing and understanding of the resulting clausal units requires further research and will be addressed in a later paper.

The task of identifying sentence boundaries, speech repairs and dysfluencies have been a fo-

cus of spoken language parsing research for several years (Bear et al., 1992; Seneff, 1992; Heeman, 1997; Ruland et al., 1998; Core and Schubert, 1999). Most of the previous approaches cope with dysfluencies and speech repairs in the parser by providing ways for the parser to skip over syntactically ill-formed parts of an utterance. In more recent work (Stolcke and Shriberg, 1996; Charniak and Johnson, 2001), the problem of speech parsing is viewed as a two step process. A preprocessing step is used to identify speech repairs before parsing begins. The approach we present in this paper, is similar to (Charniak and Johnson, 2001) with a few differences. We do not constrain speech edits and restarts to conform to a particular structure. Further, we also segment the utterance into clauses which we believe would be easier to interpret. Finally, we use plain word strings with no punctuation marks since this is typically the output of a speech recognizer.

The layout of the paper is as follows. In Section 2, we will define the task and illustrate with an example the encoding of the task that makes it suitable for training models for annotation. In Section 3, we discuss the two approaches for clausification - n-gram approach and discriminative approach. The experiments and evaluation results are presented in Section 4.

2 Task Definition

Clausifier takes as input the result of recognition of a user's utterance and generates clauses as its output. The clausifier annotates its input with tags that help in segmenting it into clauses. The <s> tag is used to indicate sentence boundaries, strings within [and] are to be edited out and strings between {c and } indicate coordinating conjunctions. These tags are then interpreted to retrieve the set of clauses. This interpretation involves deleting the words within [and] and replacing <s> and the words enclosed within {c and } with a line feed. An example[1] illustrating the input, the annotated output and the set of clauses resulting from interpreting the output is shown below. [2]

Clausifier Input:

```
yes I got the bill and and eh I have
a question about I was surprised I
got a phone call with in I mean for
er $time_amount is what the the bill
said and you know you charged me eh
$time_amount plus tax so eh ...
```

Clausifier annotated Output:

```
yes <s> I got the bill [ and ] {c and
} [ eh ] I have a question <s> [ about
] I was surprised <s> I got a phone
call [ with in ] [ I mean ] for [ er ]
$time_amount is what [ the ] the bill
said {c and } [ you know ] you charged
me [ eh ] $time_amount plus tax [ so eh ]
...
```

Clausifier parsed Output:

- `yes`
- `I got the bill`
- `I have a question`
- `I was surprised`
- `I got a phone call for $time_amount is what the bill said`
- `you charged me $time_amount plus tax`

In order to train models for the clausifier, we have encoded the sentence boundary, edit and conjunction information as tags following a word. If there is sentence boundary before a word, it is tagged as "Segment". Edit and conjunction tags also contain span information; <Edit1> is for edit of one word to the left,

[1] $time_amount in this example is a named entity.

[2] The example input is taken from a telephone conversation of a customer speaking to an operator. Due to the unavailability of annotated data for this domain, we report experimental results in this paper for the Switchboard corpus.

<Edit2> is for edit of two words to the left and so on. A similar encoding is used for coordinating conjunctions. A word boundary that has neither of these tags is tagged as "No Action".[3]

Encoding the problem:

```
yes <s> I got the bill and <Edit1>
and <Conj1> eh <Edit1> I have a
question <s> about <Edit1> I was
surprised <s> I got a phone call
with in <Edit2> I mean <Edit2> for
er <Edit1> $time_amount is what the
<Edit1> the bill said and <Conj1>
you know <Edit2> you charged me eh
<Edit1> $time_amount plus tax so eh
<Edit2> ...
```

3 Clausifier Method

The task of annotating the input can be viewed as a tagging problem. Each word of the input is tagged with one of a few tags that indicate the type of annotation following the word. In particular, we consider the presence of sentence boundary tag <s> and its absence <nos> as two possible tags to associate with each word. We can then use an n-gram tagging model (similar to (Church, 1988)) as shown in equation 1 to retrieve the best tag sequence for a given input sentence. We follow the same notation as in (Church, 1988).

$$P(T) = argmax_T P(w_i|t_i) * P(t_i|t_{i-1}, t_{i-2}) \quad (1)$$

Such an n-gram based sentence boundary detection method was presented in (Stolcke and Shriberg, 1996) who also point out that the advantage of n-gram based method is its natural integration within the language model of the speech recognizer. However, increasing the conditioning context in an n-gram increases the number of parameters combinatorially and estimating these parameters reliably becomes an issue. We present a discriminative classification approach to clausifier which allows us to add larger number of features, in contrast to the generative n-gram model.

[3]An alternate way of encoding is to use the "Inside-Outside-Boundary" tags for each word as is typically done for chunking of noun groups.

3.1 Classifier

We used a machine-learning tool called Boostexter, which is based on the boosting family of algorithms first proposed in (Schapire, 1999). The basic idea of boosting is to build a highly accurate classifier by combining many "weak" or "simple" base classifiers, each one of which may only be moderately accurate. To obtain these base classifiers, it is assumed that a base learning algorithm is available that can be used as a black-box subroutine. The collection of base classifiers is iteratively constructed. On each iteration t, the base learner is used to generate a base classifier h_t. Besides supplying the base learner with training data, the boosting algorithm also provides a set of nonnegative weights w_t over the training examples. Intuitively, the weights encode how important it is that h_t correctly classifies each training example. Generally, the examples that were most often misclassified by the preceding base classifiers will be given the most weight so as to force the base learner to focus on the "hardest" examples. As described in (Schapire and Singer, 1999), Boostexter uses *confidence rated* classifiers h that, rather than providing a binary decision of -1 or +1, output a real number $h(x)$ whose sign (-1 or +1) is interpreted as a prediction, and whose magnitude $|h(x)|$ is a measure of "confidence." The output of the final classifier f is $f(x) = \sum_{t=1}^{T} h_t(x)$, i.e. the sum of confidence of all classifiers h_t. The real-valued predictions of the final classifier f can be converted into probabilities by passing them through a logistic function; that is, we can regard the quantity

$$\frac{1}{1 + e^{-f(x)}}$$

as an estimate of the probability that x belongs to class +1. In fact, the boosting procedure is designed to minimize the negative conditional log likelihood of the data under this model, namely:

$$\sum_i \ln(1 + e^{-y_i f(x_i)})$$

The extension of Boostexter to the multiclass problem is described in (Schapire and Singer,

1999). With this extension Boostexter is able to assign a weight to each class. In our application we are interested in a single decision at each word boundary and hence we select the class with the highest weight. It has been our experience that Boostexter does not overfit the training data. Among others it is able to deal with independent variables of type text, and is ideally suited for our task. In order to deal with text, Boostexter extracts ngrams (sparse and connected) as features. For our applications we only use unigram features.

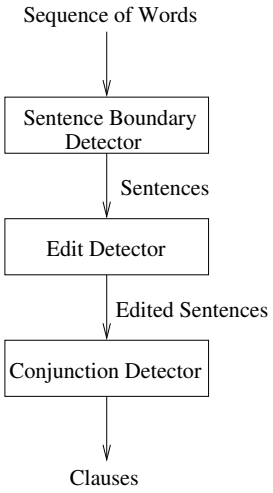

Figure 1: The architecture for the clausifier

Figure 1 shows the three different components of the clausifier one each for the tasks of detecting sentence boundaries, detecting speech repairs and detect coordinating conjunctions. We believe that this pipeline architecture which makes annotations in three stages is more suited than a single classifier that predicts all the annotations in a single step. We justify this architecture by observing that typically edits and speech repairs do not go across sentence boundaries.

3.1.1 Features used in the classifier

All the classifiers are designed to classify a word boundary into either a sentence boundary tag, an edit tag or a conjunction tag. A set of features of a word boundary are used as independent variables. In our experiments we used 18 features listed in Table 1. The first four rows consist of feature templates representing 12 features for the left and right word and part-of-speech contexts.

4 Experiments and Evaluation

In this section, we present the results of several experiments for the baseline n-gram model for sentence boundary detection and the classifier models for detecting the three components of the clausifier.

4.1 Data Preparation

For this work, we used the transcribed and annotated human-human conversations present in the Switchboard corpus. Annotation scheme for this data is described in (Meteer and others, 1995). In this data, since sentences can span over turns, we collapsed each side (side A and side B) of a dialog as a single string of words with turn markings. This resulted in total of 2242 annotated strings of words, each containing all the utterances of one participant in a dialog. We removed all annotations[4] except for a) the sentence boundary b) turn boundary c) explicit edits d) fillers e) discourse markers f) restarts and repairs g) coordinating conjunctions. Restarts and repairs together with fillers, explicit edits and discourse markers were considered as edit tags. Some of the strings that had complex nested restarts and repairs were dropped from consideration. This left us with 2115 annotated strings with approximately 660,000 words (excluding annotations). These strings were randomly partitioned in two sets; 1923 strings (610,000 words) for training various classifiers and 192 strings (58,686 words) for testing them. All the results presented in this paper are based on this training and test data.

4.2 Evaluation Metrics

We evaluate various models on the test set and compare the resulting annotation against a hand-annotated version of the test set. We report recall and precision scores on each of the

[4]In the annotated data we found that in many cases square brackets ([) and curly braces ({) were not balanced. Ignoring such strings would not leave us with enough data. We therefore wrote a heuristic program that balanced such cases.

$word_{-i}$	$i \in \{1,2,3\}$ words to the left. 'bos' if there is no word
$word_i$	$i \in \{1,2,3\}$ words to the right. 'eos' if there is no word
pos_{-i}	$i \in \{1,2,3\}$ parts of speech of three words to the left
pos_i	$i \in \{1,2,3\}$ parts of speech of three words to the right
w1	1 if $word_1$=$word_{-1}$ otherwise 0
w2	# of words common in 2 left and 2 right words
w3	# of words common in 3 left and 3 right words
p1	1 if pos_1=pos_{-1} otherwise 0
p2	# of pos common in 2 left and 2 right pos
p3	# of pos common in 3 left and 3 right pos

Table 1: Features used for the classifiers

individual tags as well as the total error rate and the baseline error rate for each tagging task. This baseline error rate is calculated by using a classifier that assigns each example the tag that occurs most frequently in the data.

Since we are eventually interested in parsing and understanding the resulting clauses, we also report recall and precision after each of the annotations are interpreted (i.e. after utterances are split at sentence boundaries, after edits are deleted and after utterances are split at conjunctions.). These scores are reported under the "Sentence" column of each model's performance table. Like other recall and precision numbers sentence level recall indicates the proportion of clauses in the input that are correctly identified in the clausifier output, and precision indicates the proportion of the output clauses that are in the input.

4.3 N-gram model: Baseline Model

Table 2 shows the results of using a trigram model, similar to (Stolcke and Shriberg, 1996) for sentence boundary detection on the data described above. In our experiments, instead of using the true part-of-speech tags as was done in (Stolcke and Shriberg, 1996), we used the result of tagging from an n-gram part-of-speech tagger (similar to (Church, 1988)). In addition to providing recall and precision scores on the individual segmentation decision, we also provide sentence level performance. Notice that segmentation precision and recall of approximately 80% and 52% turn into sentence level precision and recall of 50% and 32% respectively. We also noticed that including POS improves the performance by approximately 1%.

4.4 Classifier Models

Training data for the classifiers was prepared by labeling each word boundary with one of the tags described in Section 2 and features shown in Table 1. Apart from training individual classifiers for sentence boundary, edit and conjunction classification, we also trained a combined classifier which performs all the three tasks in one step and does not make any independence assumptions as shown in Figure 1.

4.5 Combined Classifier

Table 3 shows the performance of a combined classifier that predicts a combined tag for each of the components. The tagset for the combined classifier is the cross-product of the tagsets for segmentation,edits and conjunctions. Since this classifier makes all the decisions, the output of this classifier can be directly used to generate clauses from the input strings of words. As expected this classifier outperforms the N-gram based classifier both at segmentation decision level and at sentence level (compare column 8 and 9 of table 3 with column 3 and 4 of table 2 respectively).

4.6 Individual Classifiers

Tables 4,5,6 show the performance of the three classifiers used in the cascade shown in Figure 1. In these tables sentence level performances are with respect to their own inputs and outputs. Over all sentence level performance is shown in

	No Action	Segment	Sentence
Counts	57454	10284	10654
Recall (%)	98.13(98.02)	52.79(52.26)	32.55(31.53)
Precision(%)	92.07(91.98)	83.47(79.36)	50.94(49.29)
Total Error (%)	9.23(9.93)		
Baseline Error (%)	15.18		

Table 2: Segmentation Performance Using Trigram Model. Performance without part-of-speech information is shown in parenthesis.

	No Action	Edit One	Edit Two	Edit Three	One Conj	Two Conjs	Segment	Sentence
Counts	51126	3704	895	130	2391	258	9756	10789
Recall(%)	96.41	69.68	49.94	8.46	79.26	48.06	77.95	51.2
Prec(%)	93.10	78.69	67.12	50.00	80.71	73.37	85.18	50.06
Total Error	9.38%							
Baseline Error	25.22%							

Table 3: Performance of a classifier that predicts a combined tag

Table 7. These tables show that cascaded classifiers are significantly more accurate at making individual decisions which results in higher recall and precision at sentence level (compare column of table 7 with column 9 of table 3).

5 Sensitivity Analysis

In this section, we investigate the effect of various features on the performance of the Segmentation, Edit and Conjunction classifiers. Table 8 shows the results for each classifier using only words, only parts-of-speech (POS) tags, words and POS tags and words, POS tags combined with the similarity measure. It is not surprising to note that using only POS tags to predict sentence boundaries, edits and conjunctions results in a higher error rate compared to using only words. Adding POS features to the words-only model does not improve the performance of these classifiers. This is to be expected since the generalization provided by the POS tags is not really needed for these tasks, as there are not many unseen contexts even when using words contextual features. However, we suspect supertags (Bangalore and Joshi, 1999) can capture long-distance effects (eg. subcategorization frame of preceding verb) which could improve the segmentation performance.

It is however surprising to note that the similarity features, which had been designed to specifically capture patterns in speech repairs does not contribute as much to the performance (Words+POS+Similarity=3.5% as compared to Words Only=3.9%). By separating discourse markers (eg. you know, well, so), explicit edit terms (eg. I mean, sorry, excuse me), and fillers (eg. um, uh) from the set of Edit tags, we are left only with restarts and repairs. We trained a classifier for identifying only these tags and the sensitivity results are shown in the fourth column of Table 8. Note that the baseline performance of Restarts and Repairs is much lower than that of Edits indicating that it is an easier task than identifying Edits. Incorporating the similarity feature reduces the error rate for identifying restarts by 37% over a model which uses only words. (Words+POS+Similarity=1.7% as compared to Words Only=2.7%). This suggests that an additional classifier to identify discourse markers and explicit edit terms would be beneficial.

	No Action	Segment	Sentence
Counts	57980	10371	10561
Recall (%)	97.37	77.57	56.22
Precision (%)	96.04	84.07	60.83
Total Error	5.63%		
Baseline Error	15.05%		

Table 4: Performance of classifier for sentence boundary detection

	No Action	Edit One	Edit Two	Edit Three	Edit Four	Sentence
Counts	53936	4074	1042	125	58	10400
Recall(%)	99.09	76.12	62.57	9.60	1.72	81.58
Precision(%)	97.23	88.50	84.13	48.00	100.0	80.34
Total Error	3.48%					
Baseline Error	9.02%					

Table 5: Performance of the classifier to identify presence and the span of an edit.

	No Action	One Conj	Two Conj	Sentence
Counts	49682	1997	121	10789
Recall (%)	99.70	91.69	88.43	95.13
Precision (%)	99.66	92.94	84.25	94.46
Total Error	0.64%			
Baseline Error	4.10%			

Table 6: Performance of a classifier to identify presence and the span of a conjunction.

	Sentence Level
Counts	10789
Recall (%)	53.87
Precision (%)	55.85

Table 7: End to End Performance

	Segment	Edit	Conjunction	Restarts and Repairs
Baseline (%)	15.0	9.0	4.1	2.8
Words Only	5.6	3.9	0.6	2.7
Pos Only	9.9	7.3	1.6	2.8
Words+POS	5.6	3.9	0.6	2.7
Words+Pos+Similarity	5.6	3.5	0.6	1.7

Table 8: Sensitivity analysis of the features used for classification

6 Conclusions

In this paper, we have presented a clausifier which would be used as a preprocessor in the context of speech parsing and understanding system. The clausifier contains three classifiers that are trained to detect sentence boundaries, speech repairs and coordinating conjunctions. These models have been trained and tested on Switchboard corpus and provide an end-to-end recall and precision of 54% and 56% respectively for the task of clause identification. We have shown that classifier models clearly outperform the n-gram models, and that a combined model does not perform as well as a model that makes individual predictions. We believe that the sentence level performance can be improved further by improving the training data quantity and quality. In the Switchboard corpus we found that average turn length is 6, and that the turn boundaries are very strong indicator of the sentence boundaries. This makes it hard for the classifier to learn other discriminating features. We plan to use this system to iteratively annotate additional data with longer turn lengths (customer-operator telephone conversations), manually correct it and retrain the models described in this paper.

References

Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2).

J. Bear, J. Dowding, and E. Shriberg. 1992. Integrating multiple knowledge sources for detection and correction of repairs in human-computer dialog. In *Proceedings of ACL*, pages 56–63.

E. Charniak and M. Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proceedings of NAACL*, Pittsburgh, PA.

Kenneth Ward Church. 1988. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *2nd Applied Natural Language Processing Conference*, Austin, Texas.

M.G. Core and L.K. Schubert. 1999. A syntactic framework for speech repairs and other disruptions. In *Proceedings of ACL*, pages 413–420.

A. L. Gorin, G. Riccardi, and J. H Wright. 1997. How May I Help You? *Speech Communication*, 23:113–127.

Peter Heeman. 1997. *Speech Repairs, Intonation Boundaries and Discourse Markers: Modeling Speakers' Utterances.* Ph.D. thesis, University of Rochester.

M. Meteer et al. 1995. Dysfluency annotation stylebook for the switchboard corpus. In *Distributed by LDC.*

T. Ruland, C.J. Rupp, J. Spilker, H. Weber, and K.L. Worm. 1998. Making the most of multiplicity: A multi-parser multi-strategy architecture for the robust processing of spoken language. Technical report, DFKI, Verbmobil report 230.

R.E. Schapire and Y. Singer. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, December.

R.E. Schapire. 1999. A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence.*

Stephanie Seneff. 1992. A relaxation method for understanding spontaneous speech utterances. In *Proceedings, Speech and Natural Language Workshop*, San Mateo, CA.

A. Stolcke and E. Shriberg. 1996. Statistical language modeling for speech disfluencies. In *Proceedings of ICASSP*, Atlanta, GA.