ROBUST INFORMATION EXTRACTION FROM SPOKEN LANGUAGE DATA

David D. Palmer†‡ *Mari Ostendorf*† *John D. Burger*‡

†Electrical and Computer Engineering Department, Boston University, Boston, MA 02215
‡The MITRE Corporation, Bedford, MA 01730

10.21437/Eurospeech.1999-168

ABSTRACT

In this paper we address the problem of information extraction from speech data, particularly improving robustness to automatic recognition errors. We describe a baseline probabilistic model that uses word-class smoothing in a phrase n-gram language model. The model is adjusted to the error characteristics of a speech recognizer by inserting error tokens in the training data and by using word confidences in decoding to account for possible errors in the recognition output. Experiments show improved performance when training and test conditions are matched.

1. INTRODUCTION

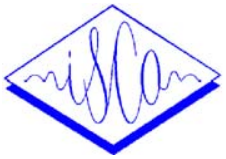
Extracting linguistic structure such as proper names, noun phrases, and verb phrases is an important first step in many systems aimed at automatic language understanding. While significant progress has been made on this problem, most of the work has focused on "clean" textual data such as newswire texts, in which cues such as capitalization and punctuation are important for obtaining high accuracy results. However, there are many data sources where these cues are not reliable, such as in spoken language data or single-case text. Spoken language sources in particular pose additional problems because of disfluencies and speech recognition errors. Previous approaches that have addressed spoken language have consisted largely of applying an existing text-based system to speech data, ignoring the fact that information lost due to recognition errors when moving from text to speech can be regained in part via word confidence prediction.

In [3], we introduced a new probabilistic model for the problem of information extraction (IE) for spoken language data and demonstrated state-of-the-art results, even for high error rate tasks. We will review the model in Section 2. One significant contribution of our system is its robustness in processing errorful speech transcriptions. In this paper we explore the robustness issue further by addressing an important problem in developing IE systems for noisy data: mismatch between the training data and the test data. For speech data, in particular, the training data consists of "clean" human transcriptions of spoken language, while the test data consists of "noisy" machine transcriptions, with many word errors.

A key component of our information extraction approach is a bigram language model built from the training data, so a failure to take into account the training-test language differences associated with recognition errors will lead to suboptimal performance. If we can model word errors accurately in both the training and test data, the language model built from the training data should better match the test data, and we should be able to improve overall performance. In Section 3 we describe our approach to this problem: to make use of the word-level confidence score, which is the posterior probability that the word output by an automatic speech recognition (ASR) system is correct. The confidence defines a binary distribution for the correct word and a generic error token, which leads to a modified (mixture) language model and requires training with errorful data. We describe methods for inserting error tokens into the training data in Section 4.

The specific information extraction task we address in this work is name recognition (identifying names of persons, locations, and organizations), as well as identification of temporal and numeric expressions. Also known as named entities (NEs), these phrases can be useful to identify in many language understanding tasks, such as co-reference resolution and summarization. NE identification in written documents has been examined extensively under the auspices of the Message Understanding Conferences sponsored by DARPA, and performance of name recognizers, or named entity taggers, on written documents such as Wall Street Journal articles is comparable to human performance (usually 94-96% accuracy[1]). DARPA has recently expanded the scope of its information extraction evaluations to include named entity recognition in speech data. Sections 2 and 5 describe experimental results on the 1998 DARPA broadcast news information extraction task. Questions raised by these results are discussed in Section 6.

[1]Named entity system performance is typically reported in terms of the *F-measure*, which is the harmonic mean of recall and precision. All NE system accuracies we report will be in terms of the F-measure.



6th European Conference on
Speech Communication and Technology
(EUROSPEECH'99)
Budapest, Hungary, September 5-9, 1999

2. BASELINE MODEL

Our basic framework, described fully in [3], builds on the work of BBN's Identifinder system [1, 2], which uses a hidden state sequence to represent phrase structure and state-conditioned word bigram probabilities as observation distributions in a probabilistic model similar to a hidden Markov model (HMM). The BBN model incorporates non-overlapping features about the words, such as punctuation and capitalization, in a bigram back-off to handle infrequent or unobserved words. Viterbi decoding is used to find the most likely sequence of phrase labels in a test corpus. The Identifinder approach has resulted in high performance on many text-based tasks, including English and Spanish newswire texts.

As in the BBN model, we use a hidden state sequence to represent phrase structure, with the assumption that each word in a document is emitted by a state in the model. The problem is thus framed as one of decoding the hidden state sequence $s_1^L = (s_1, \ldots, s_L)$ most likely to have produced the known word sequence $w_1^L = (w_1, \ldots, w_L)$, or

$$\begin{aligned} \hat{s}_1^L &= \operatorname*{argmax}_{s_1^L} P(s_1, \ldots, s_L | w_1, \ldots, w_L) \\ &= \operatorname*{argmax}_{s_1^L} P(s_1, \ldots, s_L, w_1, \ldots, w_L), \end{aligned}$$

using the fact that multiplying by $P(w_1^L)$ does not change the most likely sequence. The search can be simplified by assuming that the state at time t is dependent only on the state and observation at time $t-1$, in which case we arrive at the following formulation that can be thought of as a phrase n-gram:

$$\hat{s}_1^L = \operatorname*{argmax}_{s_1^L} \left\{ \prod_{t=1}^{L} P(w_t | w_{t-1}, s_t) P(s_t | s_{t-1}, w_{t-1}) \right\}.$$

As in an HMM, there are two main terms in the model: the state-dependent observation model $P(w_t|w_{t-1}, s_t)$ and the state transition probabilities $P(s_t|s_{t-1}, w_{t-1})$. The observation model can be thought of as a state-dependent bigram language model. Note that this model is not strictly an HMM, since both distributions violate the traditional conditional independence assumptions in the dependence on the previous word, w_{t-1}. In addition, the HMM forward-backward training algorithm is not needed here, since the state topology is defined such that the state sequence is fully observable from labeled training data.

A key difference with respect to the BBN system is in the language model component, $P(w_t|w_{t-1}, s_t)$, in which infrequent data is handled using a class-based smoothing technique [4], i.e.

$$P(w_t | w_{t-1}, s_t) = \sum_k P(w_t | w_{t-1}, c_k, s_t) P(c_k | w_{t-1}, s_t),$$

where c_k ranges over the possible linguistic classes of word w_t. Unlike the feature-dependent back-off in [1, 2], class smoothing allows for ambiguity (overlap) of word classes. Thus, we can incorporate information from place and name word lists, as well as simple part-of-speech labels, and account for the fact that some words can be used in multiple classes. Limiting the dependence on the actual words in the text and on features derived from their orthographic realization also allows the technique to handle word errors more robustly. Sparse data issues are addressed using linear interpolation with lower order statistics associated with the linguistic classes.

The baseline model yielded high performance (71-81% accuracy) on broadcast news speech data with a wide range of word error rates (WERs) ranging from 13% to 28%, which can be compared to 88% accuracy on reference transcriptions (WER 0%). The results from the Hub-4 evaluation indicated that class-based smoothing in a bigram language model produces a performance increase over standard bigram language models. This increase was consistent over the range of word error rates present in the evaluation. As in the speech recognition experiments reported in [4], additional information extraction experiments reported in [3] showed that when training and test data reflect domain differences (either topical or temporal), the class-based smoothing produced higher performance on reference transcriptions of speech data, compared to standard bigram models.

3. DECODING WITH CONFIDENCES

Most current speech recognition systems produce a confidence score for each word. This word-level confidence score can be thought of as the posterior probability that the word output by an ASR system is actually correct. As such, we can use the confidence information to further improve the robustness of our information extraction system.

3.1. Mixture Language Models

For any given bigram produced by the recognizer, there are four possibilities: both words in the bigram are correct, only the first is correct, only the second is correct, or both are incorrect. These four possibilities are represented by the lattice shown in Figure 1, in which an incorrect word is replaced by the error token ϵ.

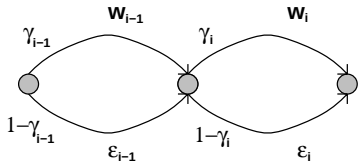


Figure 1: Bigram lattice with error tokens.

If we consider the word confidence score γ_i as the probability that word w_i is correct, then the probability of taking path w_i in the lattice is γ_i. Similarly, the probability of taking the error path ϵ_i is equal to the quantity $1 - \gamma_i$, the probability that the ASR system produced an error and that output word w_i is incorrect.

We can thus take the error tokens into account by factoring the word confidences into the calculation of the state-dependent bigram probability, $P(w_i|w_{i-1}, s_i)$. Where this probability previously was defined as a single bigram language model probability, the lattice produces a bigram probability as a mixture of four component language model probabilities:

$$\begin{aligned} P(w_i|w_{i-1}, s_i) &= \gamma_i\gamma_{i-1}P_{LM}(w_i|w_{i-1}, s_i) \\ &+ \gamma_i(1-\gamma_{i-1})P_{LM}(w_i|\epsilon_{i-1}, s_i) \\ &+ (1-\gamma_i)\gamma_{i-1}P_{LM}(\epsilon_i|w_{i-1}, s_i) \\ &+ (1-\gamma_i)(1-\gamma_{i-1})P_{LM}(\epsilon_i|\epsilon_{i-1}, s_i), \end{aligned}$$

where the mixture weights are provided by the word confidences of the component bigrams, γ_i and γ_{i-1}.

3.2. Confidence Thresholds

An alternative method of integrating the word confidence information into our model at decoding time involves using confidence thresholds. In this method, as in the baseline system, a single state-dependent language model probability is calculated. However, a word in the bigram is replaced with the error token whenever its confidence value is less than a threshold value T. For example, if $\gamma_i < T$ and $\gamma_{i-1} > T$, the probability would be given by $P(w_i|w_{i-1}, s_i) = P(\epsilon_i|w_{i-1}, s_i)$. In contrast to the mixture model, the ϵ token in the language model is used only for words with low confidences, as defined by the threshold value T.

4. TRAINING WITH ERROR TOKENS

In order for the word confidences to be used at decoding time in the manner described in Section 3, the training data (and thus the language model) must also contain errors in the form of the ϵ token. We experimented with two methods of inserting the error token, based on simulated and actual errors, as described next.

4.1. Simulated Errors

Given a set of ASR data which is independent of the reference transcript (from which we normally train the system), we can simulate word errors in the transcript based on the distribution of errors in the ASR data. Hypothesizing that the frequencies of errors made by the ASR system vary depending on the type of word being recognized and its length, we define several categories of words, based on the number of syllables and whether it was a function or content word. Examples of our word categories are *one-syllable function word*, *two-syllable content word*, and *out-of-vocabulary word* (a word not in the ASR system lexicon). For a set of ASR data, we count how often the ASR system produced substitution, deletion and insertion errors for words in each category. From these counts, we estimate the probability that the recognizer will produce the different error types.

Using the error probabilities, we simulated errors in the training data by randomly generating errors according to the training word categories. When substitutions and insertions were generated, the training word was replaced with one or more ϵ tokens. When a deletion was generated, the word was simply deleted from the transcript. The language model was trained from this data, with simulated ϵ tokens replacing some words but retaining the part-of-speech tags from the reference transcript.

4.2. Actual Errors

To reduce the mismatch between training and test data further, given a set of ASR data which overlaps with the labeled training data, we can identify actual recognition errors in the data and replace these with the ϵ token. To explore this possibility, we obtained from Dragon a large set of ASR broadcast news data for the same broadcasts that had been annotated with NE information for the Hub-4 evaluation. By aligning the NE annotated files with the ASR data, we were able to replace the substitution and insertion errors in the training data with the ϵ token, producing errorful ASR data with NE and part-of-speech annotation for use in training the language models.

5. EXPERIMENTS

In the first experiment, we used training data with simulated errors and the mixture language model. Recognizer output and confidences were provided by SRI. While we had hypothesized that this combination of simulated errors and word confidences would improve the performance of the information extraction system,

this was not the case. Instead, it resulted in a slightly improved precision but a significantly lower recall, and thus an F-measure decrease of nearly 11%. We repeated the simulated error experiment using the recognizer output and confidences provided by Dragon; the WERs of the respective recognizers were roughly comparable. This experiment resulted in an even larger F-measure decrease of 63%; again the precision improved, but the recall decreased 76%!

The degradation in performance could have been due to a mismatch between the types of simulated and actual errors, to a mismatch associated with training with "hard" errors and testing with "soft" errors (confidence weighted), and/or to a bias in the confidence estimate that makes these scores poor weights in a mixture model. To better understand this, we ran further experiments with simulated vs. actual errors on the Dragon corpus, and the same effect was observed with the actual errors. While we have not ruled out the mismatch factor, there is at least a bias problem; experiments with the mixture method indicate that lower average confidence scores result in lower overall recall. The average confidence value in the Dragon data was 0.70, while the average value in the SRI data was 0.93, which explains the larger recall decrease in the second experiment.

We also conducted further experiments using actual errors in training. The results are summarized in Table 1. The first row of Table 1 represents the baseline score obtained by training the system on the reference transcript and decoding the test data with the baseline model (no confidence). Row 2 shows, not surprisingly, that a significant degradation is observed when training with incorrect words. The remaining rows give the result when the ϵ token is inserted in the training data, but different models are used in decoding. Using the baseline model in decoding with the error token in training, row 3, corresponds to throwing away all n-grams with error tokens. Note that throwing away these n-grams is much better than using the errors. Substituting the test word tokens with an ϵ token when confidence is low does give improved performance, with suitable choice of the confidence threshold.

Table 1: Results for different training/decoding strategies using training data marked with recognition errors.

Training	Decoding	F-measure
reference	baseline	68
ASR	baseline	63
reference + ϵ	baseline	67
reference+ ϵ	ϵ subst. ($T = 0.5$)	63
reference+ ϵ	ϵ subst. ($T = 0.2$)	70

6. DISCUSSION

In summary, we have described different approaches for adjusting a probabilistic NE model to better match the error characteristics of an ASR system for IE from speech data. While the baseline system has been shown to be relatively robust to speech recognition errors, experimental results here show that improved performance can be achieved by annotating actual recognition errors in the training data and likely errors in the test data based on word confidence prediction.

The results also raise some unresolved questions. While it is tempting to conclude that the error token replacement approach is the decoding method of choice for use of confidences, we note that the result may simply be a consequence of the fact that the substitution approach most closely matches the training scenario (replacing incorrect words with the error token). Improved performance may be obtained by doing Viterbi decoding over the lattice built from the entire word sequence or using the forward algorithm to compute the mixture distribution. It would also be useful to compensate for any bias in the confidence estimate learned from analysis of training data confidences. In addition, improving the simulated error data is of interest, because of the lower training costs.

Acknowledgements. We thank Andreas Stolcke of SRI International and Steven Wegmann of Dragon Systems for making their ASR data available for these experiments.

7. REFERENCES

[1] D. Bikel, S. Miller, R. Schwartz, R. Weischedel, "NYMBLE: A High-Performance Learning Name Finder," *Proc. ANLP97,* pp. 194–201, 1997.

[2] D. Bikel, R. Schwartz, R. Weischedel, "An Algorithm that Learns What's in a Name," *Machine Learning,* 1999.

[3] D. Palmer, J. Burger, and M. Ostendorf, "Information Extraction from Broadcast News Speech Data," *Proc. DARPA Broadcast News Workshop,* 1999.

[4] R. Iyer and M. Ostendorf, "Transforming Out-of-Domain Estimates to Improve In-Domain Language Models," *Proc. Eurospeech,* vol. 4, pp. 1975–1978, 1997.