ELSEVIER

Speech Communication 32 (2000) 95–109

SPEECH COMMUNICATION

www.elsevier.nl/locate/specom

# Robust information extraction from automatically generated speech transcriptions

David D. Palmer [a,b,*], Mari Ostendorf [a], John D. Burger [b]

[a] *Department of Electrical Engineering, University of Washington, Seattle, WA 98195, USA*
[b] *The MITRE Corporation, Bedford, MA 01730, USA*

**Abstract**

This paper describes a robust system for information extraction (IE) from spoken language data. The system extends previous hidden Markov model (HMM) work in IE, using a state topology designed for explicit modeling of variable-length phrases and class-based statistical language model smoothing to produce state-of-the-art performance for a wide range of speech error rates. Experiments on broadcast news data show that the system performs well with temporal and source differences in the data. In addition, strategies for integrating word-level confidence estimates into the model are introduced, showing improved performance by using a generic error token for incorrectly recognized words in the training data and low confidence words in the test data. © 2000 Elsevier Science B.V. All rights reserved.

**Zusammenfassung**

Mittels automatischer Sprachverarbeitung gewonnene Daten werden mit Hilfe eines robusten Systems bezüglich ihrer semantischen Informationen analysiert. Bestehende HMM Technologien (zur Informationsextraktion) werden zum einen durch die Verwendung einer Rang-Topologie zur Modellierung von Satzbestandteilen variabler Länge erweitert, zum anderen ermöglicht eine klassenbasierte, statistische Sprachmodellglättung optimale Verarbeitung in einem weitem Bereich von Sprachfehlerraten. Analysen von Rundfunk- und Fernsehnachrichten bestätigten sehr hohe Fehlertoleranzen gegenüber Zeit- und Quellenunterschieden in den Daten. Weiterführende Strategien zur Verbesserung der Fehlertoleranz über die Integration von Wortkonfidenzwerten in das System werden vorgestellt und anhand von Beispieldaten demonstriert. © 2000 Elsevier Science B.V. All rights reserved.

*Keywords:* Information extraction; Phrase language model; Named entity; Word recognition confidence

## 1. Introduction

Extracting entities 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 *information extraction* (IE) problem, most of the work has focused on "clean" textual data such as newswire texts, where 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, lack of explicit punctuation, and speech recognition errors. This paper addresses the problem of IE

[*] Corresponding author.

from speech, introducing a new probabilistic approach that builds on language modeling techniques to obtain robust results even from sources with high word error rates (WERs).

Previous approaches to processing spoken language data have consisted largely of applying an existing text-based system to speech data, ignoring the fact that information is lost due to recognition errors when moving from text to speech and the possibility that it can be regained in part via word confidence prediction. Our approach tackles the speech problem directly, by avoiding the use of features that are characteristic of written text and by explicitly addressing the problem of speech recognition errors through the use of smoothing techniques and word confidence information. We have developed a baseline probabilistic framework that builds on the work of BBN's Identifinder system (Bikel et al., 1997, 1999; Miller et al., 1999), 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 orthographic features of the words, such as punctuation and capitalization, in a bigram back-off to handle infrequent or unobserved words. A key difference in our approach is that infrequent data is handled using the class-based smoothing technique described in (Iyer and Ostendorf, 1997) that, unlike the feature-dependent back-off, allows for ambiguity 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. In addition, we make use of word confidences to further improve the robustness of the system to speech recognition errors. This word-level confidence score, which is produced by many current automatic speech recognition (ASR) systems, is an estimate of the posterior probability that the word output by an ASR system is actually correct. As such, we can use the confidence information to ignore or place less weight on unreliable words.

The specific IE 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 are useful in many language understanding tasks, such as coreference resolution, sentence chunking and parsing, and summarization/gisting. Named entity identification has been examined extensively through the Message Understanding Conferences (MUCs) and the recent Broadcast News information extraction evaluation, both sponsored by DARPA in the US. As a result of these formal evaluations, a standard paradigm has evolved for the evaluation of named entity systems. Working within this paradigm, we show that our system has produced high performance on broadcast news speech data with a wide range of WERs, including reference transcriptions (0% WER). When applied to transcripts generated by automatic speech recognition, the model showed high performance, despite word error rates ranging from 28% to 13%. We also show that integrating word confidence information leads to additional improvements if actual error characteristics are available for the training data.

The paper is organized as follows. Section 2 provides further details on related work in information extraction and speech understanding. Our baseline model is described in Section 3, and extensions to incorporate word confidences are described in Section 4. In Section 5, we describe experimental results obtained by applying our approach to the broadcast news IE task. We discuss remaining questions raised by the results in Section 6.

## 2. Related work

IE from text-based sources has been a focus of research for many years, and some systems have reported performance approaching human performance on the named entity task. However, IE from speech data is a very new topic of research, and most previous work has consisted of the direct application of text-based systems to speech data, with some minor adaptations. In this section we provide a brief overview of approaches to named entity extraction from text data and a description of existing NE systems for speech data.

### 2.1. Text-based named entity systems

Named entity system performance on written sources such as the Wall Street Journal and the Associated Press newswire has been evaluated in recent MUCs. Multi-lingual MUC-style evaluations have also focused on the named entity task in written Spanish, Japanese and Mandarin news data. Most of the systems with the highest performance on the written news data were manually developed and consisted of hand-crafted rule-based grammars, usually with domain-specific word lists. For example, the FASTUS system (Hobbs et al., 1997) consists of a set of large grammars for recognizing different kinds of linguistic phrases (such as named entities), where the grammars are compiled into a cascade of finite-state automata. The application of NameTag, a commercial software product in development since 1986, to the MUC NE task is described in (Krupka, 1995); the system consists of a series of language-dependent morphological analyzers and syntactic pattern matchers.

In addition to the manually developed systems, some high-performance systems have contained a trainable component that used machine learning in some form. For example, the system described in (Bennett et al., 1997) is based on decision trees, and the system in (Aberdeen et al., 1995) used the transformation-based learning approach introduced in (Brill, 1993). Bikel et al. (1997) introduced a trainable probabilistic system for named entity recognition, in which each type of entity to be recognized is represented as a separate state in a finite-state machine. A bigram language model is trained for each phrase type (i.e., for each state), and Viterbi-style decoding is then used to produce the most likely sequence of phrase labels in a test utterance. This model incorporates non-overlapping features about the words, such as punctuation and capitalization, in a bigram back-off to handle infrequent or unobserved words. The approach has resulted in high performance on many text-based tasks, including English and Spanish newswire texts.

While the best performance on text-based named entity extraction has been achieved by hand-crafted rule-based systems, these systems are based on many staff years of effort, and the manually-developed text systems do not necessarily port to noisy data such as speech. Trainable systems have rapidly reached the same performance level as hand-crafted systems, and they have the advantage of offering a short development cycle and easy adaptability to new tasks and domains.

### 2.2. Named entity systems for speech

The system developed by BBN (Bikel et al., 1997), described in the previous section, has also been successfully applied to speech data. Despite the fact that the original model relied heavily on text-based features such as punctuation and capitalization in the language model back-off, it gives good results on speech data without modifying anything but the training material (Miller et al., 1999). Our model is most closely related to this approach, and details of differences are discussed in the next section after the model is described more formally.

Another closely related statistical approach to named entity tagging in speech data was developed at Sheffield and is described in (Gotoh et al., 1999; Gotoh and Renals, 1999; Renals et al., 1999). In their model, named entity tags are treated as categories associated with words, effectively expanding the vocabulary, e.g. a word that might be both a person and a place name would be represented with two different lexical items. An n-gram language model is trained on these augmented words, using a single model for joint word/tag dependence on the history rather than the two components used in the BBN model and thus representing the class-to-class transitions implicitly rather than explicitly. A key difference between the approaches is in the back-off mechanism, which resembles a class grammar for the Sheffield system. In addition, the Sheffield approach uses a causal decoding algorithm, unlike the Viterbi algorithm which delays decisions until an entire sentence has been observed, though this is not a restriction of the model. The extended-vocabulary n-gram approach has the advantage that it is well-suited for use directly in the ASR search process (Renals and Gotoh, 1999).

Another example of a text-based system successfully adapted for use with speech data is the TextPro system (Appelt and Martin, 1999), which is similar to the FASTUS text-based system: a large manually-created grammar compiled into finite-state transducers. Adapting the grammar for speech (or, single-case text) involved the use of large lexicons to indicate which words were likely to be parts of names and manual refinement of rules to improve performance on the training data.

All three speech systems have produced results on the task of identifying NEs in speech data (Przybocki et al., 1999) that can be compared to our results described in Section 5.2.

## 3. System overview

The mathematical model we use is similar to the models described in (Bikel et al., 1997; Renals et al., 1999), with several important differences. In Sections 3.1–3.4 we will review the basic assumptions made in the model and discuss the differences between our approach and other work.

### 3.1. Probabilistic model

As in the other probabilistic state machine approaches, 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 state 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:

$$\begin{aligned}P(s_1, \ldots, s_L, w_1, \ldots, w_L) &= \prod_{t=1}^{L} P(w_t, s_t | w_{t-1}, s_{t-1}) \\ &= \prod_{t=1}^{L} P(w_t | w_{t-1}, s_t) P(s_t | s_{t-1}, w_{t-1}),\end{aligned}$$

where $w_0$ and $s_0$ are the start symbol and state, respectively. 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 (as described next).

In our system, the language model probability $P(w_t | w_{t-1}, s_t)$ is obtained via a language model using class-based smoothing, as will be described further in Section 3.3. In BBN's work (Bikel et al., 1997), which uses the same HMM-like formalism, this emission probability is obtained via a simple back-off bigram language model. Specifically, each word is deterministically assigned one of 14 non-overlapping features (such as two-digit-number, contains-digit-and-period, capitalized-word, and all-capital-letters), and the back-off distribution depends on the assigned feature. The Sheffield model (Renals et al., 1999) represents the class transition probabilities implicitly in the quantity $P(w_t, s_t | w_{t-1}, s_{t-1})$ (in our notation), by treating $s_t$ as a word attribute. For the unobserved $(w_t, s_t)$ combinations, the "back-off" is effectively a class-dependent unigram combined with the class transition probability: $P(w_t, s_t | w_{t-1}, s_{t-1}) = P(w_t | s_t) \times P(s_t | w_{t-1}, s_{t-1})$.

### 3.2. Model topology

The state topology of our model assumes that phrases can have one or more words and that any phrase type can follow any other. The different phrase types (including *other* for words that are not part of NEs) are arranged in parallel, with any phrase type being allowed to follow any other. An important characteristic of the topology is that each phrase type has two states associated with it: the first models the first word of the phrase, and the second models all successive words. Phrase boundaries can thus be reliably determined, even when two phrases of the same type occur consecutively. This two-state topology can effectively model words that are common in named entities but that occur in a certain phrase position. For example, the word "City" is common in locations such as "New York City" and "Mexico City", but it almost always occurs at the end of the phrase and would be generated by the second state of the location model.

We also model both utterance boundaries and phrase boundaries explicitly in the topology, believing that there are important contextual effects associated with both. In particular, utterance boundaries are highly correlated with sentence and major clause boundaries in written text. Utterance-initial and utterance-final states are included at the beginning and end of the network, with the insertion of a pseudo "boundary" observation at each end of a word sequence as is standard in speech recognition language modeling.

Fig. 1 shows a simplified view of the topology. Note the pairs of states for each phrase type – in this case only those for *location* and *other* are shown in detail. In general, the second of each pair can only be reached from the first (and itself, via a self-loop), while the first can be reached from any other state except the end state.

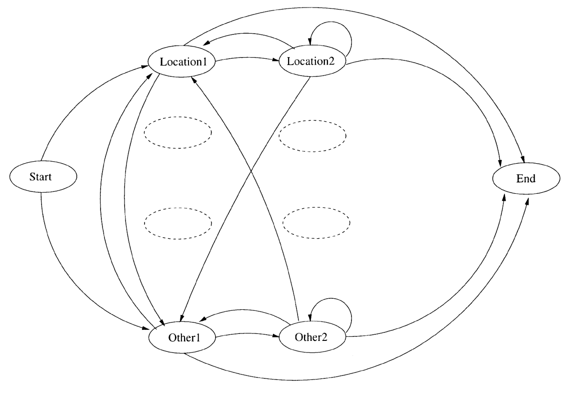

Fig. 1. Simplified model topology.

The HMM topology provides an implicit model of phrase length, as the self-loop transition probability to a state corresponds to a geometric distribution of state duration. For a certain phrase type, the probability of a phrase of length $l$ is as follows:

$$P(l) = 1 - p, \quad l = 1,$$

$$P(l) = pq^{(l-2)}(1 - q), \quad l > 1,$$

where $p$ is the transition probability between the first and second state of the phrase model, and $q$ is the self-loop probability for the second state. (Note that a benefit of the two state model is a somewhat more accurate duration model than the geometric distribution because of the parameter $p$ related to the probability of length-one phrases.) Modeling length in this manner (via the HMM state transition probabilities) is useful for identifying phrases such as NEs, since phrase types have a wide range of length distributions. For example, in the broadcast news data used in this work, the average phrase length for NEs varies from 1.4 words (for *location* phrases) to 3.4 words (for *money* phrases), while the average length of non-entities (*other* phrases) is 15.2 words.

### 3.3. State-dependent language models

In order to produce a robust model applicable to spoken language data, we wish to limit the dependence on the actual words in the text and on features derived from their orthographic realization. While orthography features – such as punctuation, capitalization, and the presence of non-alphabetic characters – provide useful information for distinguishing tokens in textual data, they are normally absent in speech data. For example, \$30.25 in text becomes "thirty dollars and twenty five cents" in speech transcriptions. In addition, this example shows that there can be several spoken words that correspond to a single orthographic token, highlighting the need for a topology to handle multi-word phrases (since defining multi-word lexical items for all such cases is impractical).

We address the need for more robust features by using the class-based smoothing technique described in (Iyer and Ostendorf, 1997), which previously has been used to develop speech recognition language models that successfully combine information from many sources. Specifically, the bigram probability is obtained by smoothing over the possible linguistic classes for the predicted word in the bigram:

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

where $c_k$ ranges over the possible linguistic classes of word $w_t$. An advantage of this method over the simple back-off solution is that it allows us to incorporate information such as simple part-of-speech labels and account for the fact that some words can be used in multiple classes. Note that this also distinguishes our approach from typical implementations of a class grammar, which assume words belong to a single class and therefore no sum is needed. When available, the standard "clean text" features such as capitalization and punctuation can also be included in this way; however, the system we discuss in this paper has achieved high performance without including clean text features in the model.

As discussed in Section 3.2, each type of phrase to be identified is represented by a pair of states in the model topology. The first state of each pair always represents the first word of the phrase, and the emission probability for the first state is determined by conditioning on the phrase start token rather than the preceding word. The emission probability for the second state is determined using both words of the bigram. Since the emission probabilities are very similar for the two states in a pair, we pool the data for training the unigram for the back-off for the two states.

Unlike data from text sources, such as newspapers, the words in transcriptions that are output by a speech recognizer may not be the "correct" words. When the words are out of the recognizer's vocabulary, the recognizer will frequently find a sequence of short high-frequency words to map to the unknown word, in which case it is impossible to recognize the spoken named entity. However, in other cases, the recognized word is a variant of the spoken word, i.e. missing a plural or the incorrect verb tense. In these instances, the POS smoothing algorithm may lead to more robust performance in the face of errors, because of increasing the probability of such unseen events. As will be shown later, we do observe improved performance due to POS smoothing in the presence of recognition errors, though not for the reference transcripts.

## 3.4. Parameter estimation

One advantage of the model topology described in Section 3.2 is the fact that, given training data hand-labeled with phrase type and extent, the state corresponding to each word is completely observable. The first word in each phrase is emitted by the first state, while all remaining words are emitted by the second state. Consequently, the maximum likelihood values of the transition parameters can be easily calculated from hand-labeled state transition counts in the training data without the need for the forward–backward algorithm. However, because the state transitions are conditioned on the previous word, the model is susceptible to sparse data problems. In all aspects of the statistical model, we use linear interpolation to compensate for sparse data, smoothing with lower order statistics. In the case of the state transitions, the interpolated formula becomes

$$\begin{aligned} &P(s_t|s_{t-1},w_{t-1}) \\ &\quad = \lambda P_{\mathrm{ML}}(s_t|s_{t-1},w_{t-1}) + (1-\lambda)P(s_t|s_{t-1}), \end{aligned}$$

where $P_{\mathrm{ML}}$ is a maximum likelihood (relative frequency) estimate taken directly from training data. The word-dependent interpolation constant $\lambda$, type C from (Witten and Bell, 1991), is given by the formula

$$\lambda = \frac{n(s_{t-1},w_{t-1})}{n(s_{t-1},w_{t-1}) + r(s_{t-1},w_{t-1})},$$

where $n()$ is the number of times a context occurs, and $r()$ the number of unique state outcomes of that context. (This interpolation formula can also be rewritten as a back-off estimate, although this is not true for interpolation methods in general.) Note that the lower order component $P(s_t|s_{t-1})$ is similarly estimated by smoothing $P_{\mathrm{ML}}(s_t|s_{t-1})$ with the first-order maximum likelihood distribution $P_{\mathrm{ML}}(s_t)$.

Sparse data problems in the state-dependent bigrams are also addressed using linear interpolation with lower order statistics:

$$P(w_t|w_{t-1}, c_k, s_t) \\ = \lambda P_{\mathrm{ML}}(w_t|w_{t-1}, c_k, s_t) + (1 - \lambda)P(w_t|c_k, s_t),$$

where the interpolation constant

$$\lambda = \frac{n(w_{t-1}, c_k, s_t)}{n(w_{t-1}, c_k, s_t) + r(w_{t-1}, c_k, s_t)}$$

is analogous to that for the state transitions. A similar equation describes the interpolation of the state- and class-dependent unigram $p(w_t|c_k, s_t)$ with the state-dependent unigram $p(w_t|s_t)$, which is in turn interpolated with the uniform distribution.

Training the language model consists of first assigning a part-of-speech tag to each word in the training data, then calculating the bigram statistics for the formula above. More specifically, the words in the training data are labeled with part-of-speech information using the MITRE part-of-speech tagger, an implementation of the rule-based part-of-speech tagger described in (Brill, 1992). The tagger assigns to each word one of 40 tags from the Penn Treebank tagset (Marcus et al., 1993). The MITRE tagger has a reported accuracy of 93–95% on single-case text with no punctuation, which is similar to the speech transcriptions we are processing. After POS tagging is completed, the training data is separated into its component "languages". For example, the *location* language model is estimated by creating a new file containing all the words (and corresponding POS tags) from *location* phrases in the training data, with each phrase treated as a separate utterance. A bigram language model is estimated for each of these training data subsets. Note that performance might be slightly improved by treating the POS tag as a hidden variable or by using the N-best tags rather than the single best tag in training. However, since the tagger is quite accurate and the tags are only used for smoothing, we felt that the small anticipated performance gains would not be worth the added training complexity.

## 4. Use of word confidences

In this section we describe methods for integrating word-level confidence information into the baseline model introduced in the previous sections. We assume that the output of a recognizer is a sequence of words $w_1, w_2, \ldots, w_n$ with an associated sequence of confidence scores $\gamma_1, \gamma_2, \ldots, \gamma_n$, where $\gamma_t$ is an estimate of the posterior probability that the word $w_t$ output by an ASR system is correct given various features associated with the speech signal at the time of that word and the model used in the recognition process. The estimation of word confidence has been the subject of much study (Siu and Gish, 1997; Gillick et al., 1997; Kemp and Schaaf, 1997; Weintraub et al., 1997); the focus here is how to *use* the confidence score and not how to estimate it.

### 4.1. Decoding strategies

The word and confidence score sequence can be interpreted as a lattice, as illustrated in Fig. 2, with an error branch (denoted with $\epsilon$) in parallel with each word. The probability of taking the word $w_t$ branch and the error branch at time $t$ are $\gamma_t$ and $1 - \gamma_t$, respectively. The confidence scores can be thought of as a low-cost alternative to storing a large word lattice annotated with acoustic and language model scores. There are several ways one could envision using such a lattice, including Viterbi decoding to find error tokens and NE labels jointly, summing over the different possible paths while finding the most likely NE sequence, and eliminating low probability branches in the lattice before doing the named entity decoding. Here we explored variants of all three alternatives, as described below.

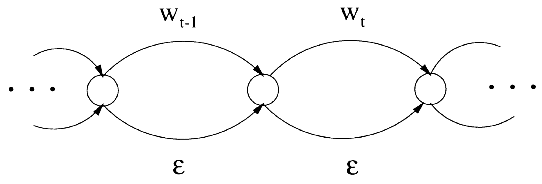

Fig. 2. NE decoding lattice with word and error tokens.

### 4.1.1. Joint error and NE state decoding

If one has the goal of jointly finding errors and named entities, then the decoding problem becomes one of finding the most likely path through the lattice above. Let $K_t \in \{0,1\}$ be an indicator of whether there is an error at time $t$. Using the notation $K$, $S$ and $W$ for the time sequence of the respective variables and $A$ to represent the acoustic (or other) evidence of an error, the decoding problem can be expressed as

$$\begin{aligned}(S,K)^* &= \operatorname*{argmax}_{S,K};p(S,K|W,A)\\ &= \operatorname*{argmax}_{S,K}\ p(S,K,W|A) \qquad (1)\\ &= \operatorname*{argmax}_{S,K}\ p(\tilde{W},S)p(K|A), \qquad (2)\end{aligned}$$

where $\tilde{W}_t = W_t$ when $K_t = 0$ and is the error token otherwise. We assume that errors are conditionally independent given the full sequence of error evidence $p(K|A) = \prod_t P(K_t|A)$, to simplify decoding, with $P(K_t = 0|A) = \gamma_t$. The other term, $p(\tilde{W},S)$, is just the named entity model described in Section 3 except that there are now error tokens in the "word" inventory.

The decoding algorithm is just a Viterbi update on a state space that is effectively doubled in size,

$$\delta_t(i,w_t) = \max_j \max_{\alpha=w_{t-1},\epsilon} \delta_{t-1}(j,\alpha) \times P(s_t = i|s_{t-1} = j,\alpha)\gamma_t P(w_t|\alpha,s_t = i), \qquad (3)$$

$$\delta_t(i,\epsilon) = \max_j \max_{\alpha=w_{t-1},\epsilon} \delta_{t-1}(j,\alpha) \times P(s_t = i|s_{t-1} = j,\alpha)(1-\gamma_t)P(\epsilon|\alpha,s_t = i), \qquad (4)$$

where $\delta_t(i,\alpha)$ is the probability of the most likely NE sequence ending at state $i$ in branch $\alpha \in \{w_t,\epsilon\}$. Note that, just as acoustic word models in ASR systems differ and thus can require different language model weight factors, it may be useful to include a $\gamma_t$ weight factor in the decoding. However, we did not explore the use of such a tuning parameter.

One problem with this model is that the error token represents a large class of possible errors, so that the phrase language model probability can be thought of as

$$p(\tilde{w}_t = \epsilon|\tilde{w}_{t-1},s_t) = \sum_{w' \neq w_t^{\mathrm{ref}}} p(w'|\tilde{w}_{t-1},s_t),$$

where $w_t^{\mathrm{ref}}$ is the reference word label at time $t$. Thus, the error token has a much larger probability than the individual words. This puts too much weight on errors, so we introduce a heuristic scaling factor that normalizes these probabilities by the size of the vocabulary. Also note that this approach can find hypothesized words that correspond to substitution or insertion errors, but it cannot detect deletion errors.

### 4.1.2. Confidence weighting

If one is not interested in knowing the location of errors, an alternative way to use the lattice is to sum over the possible error paths. The solution involves replacing the second "max" in Eqs. (3) and (4) with a sum over $\alpha$. It can be thought of as a modified Viterbi algorithm, with a forward component to capture error state evolution.

In order to assess the idea of weighting without changing the decoder, we implemented an approximation of this search that does not augment the decoder state but uses a weighted combination of bigrams in the state update to take into account ASR uncertainty, i.e.,

$$\begin{aligned}P(w_t|w_{t-1},s_t) \leftarrow [&\gamma_t\gamma_{t-1}P(w_t|w_{t-1},s_t)\\ &+ \gamma_t(1-\gamma_{t-1})P(w_t|\epsilon,s_t)\\ &+ (1-\gamma_t)\gamma_{t-1}P(\epsilon|w_{t-1},s_t)\\ &+ (1-\gamma_t)(1-\gamma_{t-1})P(\epsilon|\epsilon,s_t)]. \qquad (5)\end{aligned}$$

This approach will be referred to as the mixture approximation.

### 4.1.3. Confidence thresholding

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 $\gamma_t < T$ and $\gamma_{t-1} > T$, the probability would be given by $P(w_t|w_{t-1},s_t) = P(\epsilon|w_{t-1},s_t)$. This corresponds to pruning branches from the lattice, so that there is only a single path, specified before NE decoding begins. In contrast to using confidences as weights, the $\epsilon$ token in the language model is used only for words with low confidences, as defined by the threshold value $T$.

### 4.2. Training with error tokens

In order to evaluate hypotheses with the error token at decoding time in the manner described above, the language model (and thus the training data) must also contain errors in the form of the $\epsilon$ token. We experimented with three methods of inserting the error token, based on simulated, hypothesized and actual errors.

*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 three different error types – single word substitution, single word deletion and multiple word split (i.e. substitution plus insertion) errors – for words in each category. From these counts, we estimate the probability that the recognizer will produce the three different types of errors for each input word.

Using the error probabilities, we simulated errors in the training data by randomly generating errors according to the training word categories. A substitution caused the training word to be replaced with an $\epsilon$ token; a multiple word split resulted in two $\epsilon$ tokens. When a deletion was generated, the word was simply deleted from the transcript. (Note that this gives only an approximation of what would be observed in test data, since some types of insertions are not predicted.) The language model was trained from this data, with simulated $\epsilon$ tokens replacing some words but retaining the part-of-speech tags from the reference transcript.

*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 $\epsilon$ 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 Broadcast News IE evaluation. By aligning the NE annotated files with the ASR data, we were able to replace the actual substitution and insertion errors in the training data with the $\epsilon$ token, producing errorful ASR data with NE and part-of-speech annotation for use in training the language models.

*Hypothesized errors.* The Dragon recognizer output described above was also marked with confidence scores. Thus, we can extend the idea of a confidence threshold, as introduced in the decoding section, by replacing training words with error tokens in cases where the word confidence in the training data is below a threshold. This should hopefully produce a better match between the training and test conditions when the confidence threshold decoding method is used.

## 5. Experiments

The system described above was evaluated on the Broadcast News IE task. In addition to evaluating the system in the standard training paradigm, which we describe in the next section, we ran several contrasting experiments to determine the contributions of various system components to our NE performance, as described in the sections to follow.

### 5.1. Evaluation paradigm

As a result of the many formal DARPA-sponsored named entity evaluations, a standard evaluation paradigm has evolved. The common paradigm consists of several steps. First, a set of manually-annotated training data is prepared according to guidelines that explicitly define the entities to be identified. This training data can be used, via any combination of machine learning and manually-written rules, to develop a system which can automatically annotate unseen data. In our case, there is no manually-written component of the system; it is trained entirely via machine learning. The system is then run on one or more common "blind" test sets, and the output is scored against a manually-annotated "key". This paradigm allows the direct comparison of the performance of different systems on the same test data.

For the Broadcast News IE task which we discuss in this work, the training and evaluation data consisted of a combination of American broadcast sources from a range of dates between 1996 and 1998. The data consisted of a mixture of broadcast domains: world news summaries such as CNN's "The World Today", topical news shows such as ABC's "Nightline" and CSPAN's "Public Policy", and radio news shows such as NPR's "All Things Considered" and PRI's "Marketplace". The complete training set consisted of more than 1.6 million words of manually transcribed data annotated with about 80,000 NEs; roughly 60% of the training data had been prepared for the official Hub-4 NE evaluation (Robinson et al., 1999), and the remaining training data was prepared independently by BBN and later made available to the community. A separate evaluation set, also prepared for the Hub-4 NE evaluation, consisted of about 32,000 words and 1800 NEs. To enable comparison of system performance for output from different speech recognizers, four versions of the evaluation set were prepared, with a range of WERs from 28.3% to 0% (the reference transcript).

System performance for the NE task is evaluated using an automatic scoring program (Chinchor, 1995; Burger et al., 1998), which compares the correct phrases in the key document to the phrases identified by the system in the hypothesis document. The scoring program counts the number of phrases in the key document ($N$) and the number of phrases in the automatically generated hypothesis document ($M$), as well as identifying correct phrases ($C$), substitutions ($S$), deletions ($D$), and insertions ($I$). Using the fact that $N = C + S + D$ and $M = C + S + I$, several scores can be calculated from these counts. The most common scores are based on two measures – *recall* and *precision*. Recall is the percent of the "correct" NEs that the system identifies, $R = C/N$; precision is the percent of the phrases that the system identifies that are actually correct NE phrases, $P = C/M$. The recall and precision scores are then used to calculate the *F-measure*, which is the harmonic mean of recall and precision, $F = 2PR/(P + R)$. Human performance on the NE task has been determined to be quite high, with $F$-measures better than 98% (Robinson et al., 1999). Whereas the $F$-measure provides a performance measure for NE systems, the *slot error rate* (SER) was proposed in (Makhoul et al., 1999) as an error measure for the NE task, analogous to the word error rate metric used for evaluating speech recognition performance. The SER can also be computed directly from the raw counts and is defined as $E = (S + D + I)/N$. For speech data, both $F$-measure and SER can be calculated in three dimensions based on three categories of possible errors: the phrase label ("type"), the phrase boundaries ("extent") and the word-level phrase transcription ("content"); our results in this paper represent an equally-weighted average of the type, extent and content scores.

## 5.2. Baseline system performance

Using this experimental paradigm described in the previous section, we evaluated our system. The lexicon consisted of 63k words representing a combination of the lexicons from two large-vocabulary speech recognition systems as well as a list of words from the MITRE part-of-speech tagger.

Table 1 shows our system results for each of the four common evaluation data sets, with the training data consisting of the entire 1.6 million word corpus. For the reference transcriptions and lower error rate systems, our scores are comparable to the best reported results. [1] For the high

[1] For complete evaluation results see (Przybocki et al., 1999). Our results in this paper are slightly better than the MITRE system in the evaluation (Palmer et al., 1999), which did not have access to the full training set used here.

Table 1
Baseline system performance for a range of word error rates on the full 1999 Hub-4 NE test set

| WER (%) | *F*-measure | Slot error |
|---|---|---|
| 28.3 | 71.3 | 46.3 |
| 14.5 | 82.5 | 29.5 |
| 13.5 | 81.6 | 30.8 |
| 0 | 90.2 | 16.1 |

error rate case, our system gives slightly better results (71.3 versus 70.3 *F*-measure), suggesting that the POS smoothing technique is more robust to ASR errors. In addition, the good results (90.2% *F*-measure) on the reference transcription could likely be improved by including orthographic features such as capitalization and punctuation, since these features are present (although somewhat inconsistent) in the reference texts. As with other speech NE systems, we observe that the NE performance degrades more slowly than the WER, i.e., each recognition error does not result in an NE error.

## 5.3. Effect of training data changes

Our objective in using class-based smoothing in the bigram language model was to produce more robust models than the simple bigram model, which is dependent exclusively on word identity. In order to determine the contribution of the class-based smoothing in the language model, we ran experiments with language models trained with and without class information using different training configurations. The different training configurations included varying the amount of training data and varying the source with respect to the test data.

For the language models without class information, we retrained the language model, collapsing all POS tags to a single tag, effectively removing the class smoothing. This is roughly equivalent to other statistical NE systems in that the bigram probability is determined primarily by the word identities. However, our use of Witten–Bell interpolation differs from the feature-based back-off.

*Effect of training data size.* Miller et al. (1999) published results of experiments in which they trained their NE system with different amounts of training data, ranging from 100k words to over one million words. They found that overall performance in NE recognition increased in a log-linear fashion; that is, for each doubling in the amount of training data, the NE performance improved by a few points. We performed similar experiments with our system. We divided the 1.6 million word training set into four subsets and created different training sets from the subsets in all possible combinations. Evaluating the resulting systems on a separate test set with the class smoothing produced log-linear results over a range of 400k–1.6M words, similar to those reported in (Miller et al., 1999), for both the reference word strings and the high error rate (28.3%) ASR data. Removing the POS smoothing from the language models resulted in a consistent degradation in named entity performance (1–2% *F*-measure, absolute) for all training sets when testing on the high error rate output. No difference in performance was observed on the reference data. Thus, the POS smoothing helps with recognition errors but not reference transcriptions with reduced training data. These results indicate that it is not just the general smoothing property that is helping, but specifically the POS smoothing (in combination with standard LM smoothing of components). We conjecture that simply training with a different LM smoothing (back-off) technique without POS smoothing would not have the same positive effect.

*Effect of training data source.* In the paradigm of the previous experiments, both the training and evaluation data consisting of broadcasts from a range of dates (1996–1998) and broadcast domains. Since the class-based smoothing that we use has been the most effective in combining sparse data from multiple domains, as shown in (Iyer and Ostendorf, 1997), it is interesting to investigate whether the POS smoothing technique makes the IE model more robust to training source differences as well. Thus, we conducted two further experiments on the reference transcript data. In both experiments we defined training and test sets that would have less overlap in content than the larger training and test data used in the previous section.

In the first experiment, we defined a new training set that consisted of the CNN world news broadcast programs in the larger training set; this set comprised 63 broadcasts with 365k words. We defined an independent test set consisting of eight NPR broadcasts. The training data thus consisted entirely of broad coverage television news data while the test data consisted of topical radio news data. Comparing the performance of language models trained with and without class-based smoothing on this data indicated that the class-based smoothing improved overall *F*-measure performance by 2.3% relative.

In the second experiment, we defined a training set consisting of the files from the larger training data set that represented broadcasts from early 1996, 40 broadcasts with 200k words. We defined an independent test set consisting of the 8 broadcasts from the middle of 1998 that represented the largest time difference between broadcasts in the data. Comparing the performance of language models trained with and without class-based smoothing on this data indicated that the class-based smoothing improved overall *F*-measure performance by 1.6% relative.

## 5.4. Using confidences

The experiments with word confidences were based on output from the Dragon systems recognizer. As described in Section 4.2, the training data was a subset of 90 broadcasts (roughly 600k words) for which recognizer output was available. The recognizer word error rate on the training data was 29.4%, and the average confidence value was 0.70. The test data was a subset of the standard test set used in the previous section, consisting of three news broadcasts (about 17k words) for which word confidence scores were available. The baseline system performance for this test set was $F = 68$, where the baseline corresponds to ignoring the confidence scores in decoding and using the reference transcriptions in training.

We explored several different methods for including error tokens in the training data, as well as the different decoding methods proposed. The training variations compared (i) use of reference versus ASR transcriptions, (ii) three different ways for introducing the error token (based on confidence values, ASR errors or simulated errors), and (iii) use of copies of the training set with and without errors. The decoding algorithms (all Viterbi) differ in terms of whether they are based on a single word (and $\epsilon$) sequence hypothesis, a lattice, or the mixture approximation. The results are summarized in Table 2.

The first row of Table 2 corresponds to the baseline system, representing the score obtained by training the system on the reference transcript and decoding the test data without error tokens. The next two rows illustrate the degradation in performance associated with having errorful training data and not accounting for errors in decoding. In experiment 2, not surprisingly, a significant degradation is observed when training with incorrect words. Experiment 3, in which incorrect words are replaced with the error token in training, gives only a small loss in performance. This system

Table 2
Results on the test subset for different training/decoding strategies using the Dragon Systems recognizer and confidence output

| Expt. number | Training | | Decoding | | *F*-measure |
|---|---|---|---|---|---|
| | Source | $\epsilon$ token | Algorithm | $\epsilon$ token | |
| 1 | Reference | None | Single hyp | None | 68 |
| 2 | ASR | None | Single hyp | None | 63 |
| 3 | Reference | ASR errors | Single hyp | None | 67 |
| 4 | ASR | $T = 0.5$ | Single hyp | Subst., $T = 0.5$ | 60 |
| 5 | ASR | $T = 0.2$ | Single hyp | Subst., $T = 0.2$ | 64 |
| 6 | Reference | ASR errors | Single hyp | Subst., $T = 0.5$ | 63 |
| 7 | Reference | ASR errors | Single hyp | Subst., $T = 0.2$ | 67 |
| 8 | Reference | ASR errors | Mixture | All | 63 |
| 9 | Reference | Simulated errors | Mixture | All | 62 |
| 10 | Reference(1) + reference(2) | None(1) + ASR errors(2) | Single hyp | Subst., $T = 0.2$ | 70 |
| 11 | Reference(1) + reference(2) | None(1) + ASR errors(2) | Lattice | All | 71 |

corresponds to throwing away all training n-grams with error tokens. Note that throwing away these n-grams is much better than using the errors, i.e. using the ASR data as the training source.

The next four experiments (rows 4–7) represent different criteria for replacing words with the error token in both training and decoding. For decoding, words are replaced with the error token depending on whether the confidence score is below some threshold. Because such a threshold-based substitution will also replace correct words with the $\epsilon$ token, it requires a suitable choice of the confidence threshold. We completed systematic experiments with incrementally larger threshold values between 0.05 and 0.5 and found that a threshold of 0.2 produces the largest performance increase while a threshold of 0.5 produces the worst performance. (The $T = 0.5$ threshold corresponds to choosing the most likely label, so lower thresholds represent a conservative usage of the error token.) Note that the threshold depends on the particular confidence estimator used, each of which may have different biases in the estimates. We also found that better performance could be obtained by using actual errors to determine the $\epsilon$ tokens in training than by using the recognizer output and confidence scores. However, performance for the best case system was still below that of the baseline; i.e., confidence scores did not lead to performance improvements. Interestingly, there was no gain from using the error tokens in decoding when they are in the training data (experiments 3 versus 7).

Comparing experiment 8 to experiment 7 shows that the mixture approximation to the forward lattice decoding algorithm leads to significant degradation in performance. Summing over the error branches effectively makes the observation distributions flatter when error tokens are likely (since error tokens occur in all states). The mixture approximation effectively double counts the error branches by using the two sequence error probabilities at each time step, thereby increasing the flattening effect, which may explain the degradation in performance. Training with simulated errors (experiment 9), which was evaluated in the mixture decoding paradigm, shows only a small degradation in performance for using simulated versus actual errors.

We hypothesized that one reason for the disappointing performance of the error substitution decoding algorithm (experiment 7) was because training data was effectively lost by substituting some of the words with the error token. Therefore, we retrained the model from a doubled training set that included the unmodified reference transcriptions combined with the version using the error tokens. The modified training led to a performance improvement over the baseline (*F*-measure 70 versus 68). Using the same training method with the joint error and state lattice decoding algorithm gave the best case *F*-measure of 71. The precision and recall rates for the best case system were $P = 0.70$, $R = 0.72$, compared to $P = 0.67$, $R = 0.69$ for the baseline system.

## 6. Discussion

In this work we have introduced a robust framework for the labeling of linguistic structure in spoken language data, extending previous HMM IE models to incorporate class-based n-gram smoothing and a state topology designed for explicit modeling of variable-length phrases. Experiments show that the model is comparable to the best reported result on reference transcriptions (0% WER) and gives improved performance for data with speech recognition errors. In addition, for information extraction tasks for which the source of the training data sets differs from that of the evaluation data, the class-based smoothing can also produce higher performance on reference transcriptions of speech data. Our current system was developed with speech data in mind; consequently, we do not use capitalization or punctuation, even when they are present in the training data or reference transcripts. However, the modeling framework is extensible; it allows for the integration of any additional features, including features unique to text data, such as punctuation and capitalization. Similarly, while our initial language model implementation consists of class-based smoothing over part-of-speech categories, the classes we use are not limited to part-of-speech categories. They could also be semantically defined or automatically generated classes (via clustering), and word lists can also be used to assign more specific class labels. We also explored different approaches for adjusting the probabilistic model to better match the error characteristics of an ASR system for IE from speech data, finding that performance can be improved by introducing an error token in training and test data based on known recognition errors and word confidence prediction, respectively.

The results raise some unresolved questions. A small performance degradation was observed using simulated versus actual errors in the training data. Further exploration of error simulation would be of interest because of the lower cost of training with simulated errors. Similarly, while performance degradation was observed using ASR output in training, it may be that improved confidence estimates combined with the lattice decoding algorithm could make this a viable approach.

Finally, we note that there are connections between this work and statistical models used in other language understanding problems. Variations of phrase language models have been used in a range of speech understanding tasks, including (Seneff et al., 1992; Meteer and Rohlicek, 1993; Gorin et al., 1997; Haas et al., 1997a,b). These models strive for a more detailed understanding ("deeper", e.g. in the sense of being able to take action on a command but not in the sense of true understanding) than the shallow representation aimed for here, but cover much more restricted domains. A better understanding of where the different variations are useful would likely benefit all applications.

## Acknowledgements

We would like to thank Rukmini Iyer for her help in understanding and modifying her language modeling code. We also thank BBN for preparing and releasing additional NE training data and for providing insight into their Identifinder system. Finally, we thank Steven Wegmann of Dragon Systems for making their ASR data available for these experiments.

## References

Aberdeen, J., Burger, J., Day, D., Hirschman, L., Robinson, P., Vilain, M., 1995. MITRE: Description of the Alembic system used for MUC-6. In: Proceedings of the Sixth Message Understanding Conference (MUC-6). pp. 141–155.

Appelt, D., Martin, D., 1999. Named entity extraction from speech: Approach and results using the Textpro system. In: Proceedings of the DARPA Broadcast News Workshop. pp. 51–54.

Bennett, S., Aone, C., Lovell, C., 1997. Learning to tag multilingual texts through observation. In: Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP-2).

Bikel, D., Miller, S., Schwartz, R., Weischedel, R., 1997. NYMBLE: A high-performance learning name finder. In: Proceedings of the Applied Natural Language Processing. pp. 194–201.

Bikel, D., Schwartz, R., Weischedel, R., 1999. An algorithm that learns what's in a name. Machine Learning 34 (1/3), 211–231.

Brill, E., 1992. A simple rule-based part of speech tagger. In: Proceedings of the Third Conference on Applied Natural Language Processing. Trento, Italy, pp. 152–0155.

Brill, E., 1993. A corpus-based approach to language learning. Ph.D. Thesis, University of Pennsylvania.

Burger, J., Palmer, D., Hirschman, L., 1998. Named entity scoring for speech input. In: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics (ACL COLING 98). pp. 201–205.

Chinchor, N., 1995. MUC-5 evaluation metrics. In: Proceedings of the Fifth Message Understanding Conference (MUC-5). Baltimore, Maryland, pp. 69–78.

Gillick, L., Ito, Y., Young, J., 1997. A probabilistic approach to confidence estimation and evaluation. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processsing. Vol. 2, pp. 879–882.

Gorin, A., Riccardi, G., Wright, J., 1997. How may I help you? Speech Communication 23 (1–2), 113–127.

Gotoh, Y., Renals, S., 1999. Statistical annotation of named entities in spoken audio. In: Proceedings of the ESCA ETRW Workshop on Accessing Information in Spoken Audio. pp. 43–48.

Gotoh, Y., Renals, S., Williams, G., 1999. Named entity tagged language models. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing. pp. 513–516.

Haas, J., Nöth, E., Niemann, H., 1997a. Semantigrams – polygrams detecting meaning. In: Proceedings of the Second SQEL Workshop on Multi-Lingual Information Retrieval Dialogs (SQEL97). Plzen, pp. 65–70.

Haas, J., Hornegger, J., Huber, R., Niemann, H., 1997b. Probabilistic semantic analysis of speech. In: Paulus, E., Wahl, F., (Eds.), Mustererkennung 1997. Berlin, pp. 270–277.

Hobbs, J., Appelt, D., Bear, J., Israel, D., Kameyama, M., Stickel, M., Tyson, M., 1997. FASTUS: A cascaded finite-state transducer for extracting information from natural-language text. In: Roche, Schabes (Eds.), Finite-State Language Processing. MIT Press, Cambridge MA, pp. 381–406.

Iyer, R., Ostendorf, M., 1997. Transforming out-of-domain estimates to improve in-domain language models. In: Proceedings of the Eurospeech Conference. Vol. 4, pp. 1975–1978.

Kemp, T., Schaaf, T., 1997. Estimating confidence using word lattices. In: Proceedings of the Eurospeech Conference. pp. 827–830.

Krupka, G., 1995. SRA: Description of the SRA System as used for MUC-6. In: Proceedings of the Sixth Message Understanding Conference (MUC-6). pp. 221–235.

Makhoul, J., Kubala, F., Schwartz, R., Weischedel, R., 1999. Performance measures for information extraction. In: Proceedings of the DARPA Broadcast News Workshop. pp. 249–252.

Marcus, M., Santorini, B., Marcinkiewicz, M., 1993. Building a large annotated corpus of English: The penn treebank. Computational Linguistics 19 (2), 313–330.

Meteer, M., Rohlicek, J.R., 1993. Statistical language modeling combining n-gram and context free grammars. In: Proceedings of the International Conference on Acoustics Speech and Signal Processing. Vol. 2, pp. 173–176.

Miller, D., Schwartz, R., Weischedel, R., Stone, R., 1999. Named entity extraction from broadcast news. In: Proceedings of the DARPA Broadcast News Workshop. pp. 37–40.

Palmer, D., Burger, J., Ostendorf, M., 1999. Information extraction from broadcast news speech data. In: Proceedings of the DARPA Broadcast News Workshop. pp. 41–46.

Przybocki, M., Fiscus, J., Garofolo, J., Pallett, D., 1999. 1998 HUB-4 information extraction evaluation. In: Proceedings of the DARPA Broadcast News Workshop. pp. 13–18.

Renals, S., Gotoh, Y., Gaizauskas, R., Stevenson, M., 1999. Baseline IE-NE experiments using the SPRACH/Lasie system. In: Proceedings of the DARPA Broadcast News Workshop. pp. 47–54.

Renals, S., Gotoh, Y., 1999. Integrated transcription and identification of named entities in broadcast speech. In: Proceedings of the Eurospeech Conference. Vol. 3, pp. 1039–1042.

Robinson, P., Brown, E., Burger, J., Chinchor, N., Douthat, A., Ferro, L., Hirschman, L., 1999. Overview: Information extraction from broadcast news. In: Proceedings of the DARPA Broadcast News Workshop. pp. 27–30.

Seneff, S., Meng, H., Zue, V., 1992. Language modeling for recognition and understanding using layered bigrams. In: Proceedings of the International Conference on Spoken Language Processing. Vol. 1, pp. 317–320.

Siu, M., Gish, H., 1997. Improved estimation, evaluation, and applications of confidence measures for speech recognition. In: Proceedings of the Eurospeech Conference. pp. 831–834.

Weintraub, M., Beaufays, F., Rivlin, Z., Konig, Y., Stolcke, A., 1997. Neural-network based measures of confidence for word recognition. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing. Vol. 2, pp. 887–890.

Witten, I.H., Bell, T.C., 1991. The zero frequency estimation of probabilities of novel events in adaptive text compression IEEE Transactions on Information Theory 7 (4), 1085–1094.