# SEARCHING THE AUDIO NOTEBOOK: KEYWORD SEARCH IN RECORDED CONVERSATIONS

**Peng Yu, Kaijiang Chen, Lie Lu,** and **Frank Seide**
Microsoft Research Asia, 5F Beijing Sigma Center, 49 Zhichun Rd., 100080 Beijing, P.R.C.
{rogeryu,kaijchen,llu,fseide}@microsoft.com

## Abstract

MIT's Audio Notebook added great value to the note-taking process by retaining audio recordings, e.g. during lectures or interviews. The key was to provide users ways to quickly and easily access portions of interest in a recording. Several non-speech-recognition based techniques were employed. In this paper we present a system to search directly the audio recordings by key phrases. We have identified the user requirements as accurate ranking of phrase matches, domain independence, and reasonable response time. We address these requirements by a hybrid word/phoneme search in lattices, and a supporting indexing scheme. We will introduce the ranking criterion, a unified hybrid posterior-lattice representation, and the indexing algorithm for hybrid lattices. We present results for five different recording sets, including meetings, telephone conversations, and interviews. Our results show an average search accuracy of 84%, which is dramatically better than a direct search in speech recognition transcripts (less than 40% search accuracy).

## 1 Introduction

Lisa Stifelman proposed in her thesis the idea of the "Audio Notebook," where audio recordings of lectures and interviews are retained along with the notes (Stifelman, 1997). She has shown that the audio recordings are valuable to users if portions of interest can be accessed quickly and easily.

Stifelman explored various techniques for this, including user-activity based techniques (most noteworthy time-stamping notes so they can serve as an index into the recording) and content-based ones (signal processing for accelerated playback, "snap-to-grid" (=phrase boundary) based on prosodic cues). The latter are intended for situations where the former fail, e.g. when the user has no time for taking notes, does not wish to pay attention to it, or cannot keep up with complex subject matter, and as a consequence the audio is left without index. In this paper, we investigate technologies for searching the *spoken content* of the audio recording.

Several approaches have been reported in the literature for the problem of indexing spoken words in audio recordings. The TREC (Text REtrieval Conference) Spoken-Document Retrieval (SDR) track has fostered research on audio-retrieval of broadcast-news clips. Most TREC benchmarking systems use broadcast-news recognizers to generate approximate transcripts, and apply text-based information retrieval to these. They achieve retrieval accuracy similar to using human reference transcripts, and ad-hoc retrieval for broadcast news is considered a "solved problem" (Garofolo, 2000). Noteworthy are the rather low word-error rates (20%) in the TREC evaluations, and that recognition errors did not lead to catastrophic failures due to redundancy of news segments and queries.

However, in our scenario, requirements are rather different. First, word-error rates are much higher (40-60%). Directly searching such inaccurate speech recognition transcripts suffers from a poor recall. Second, unlike broadcast-news material, user recordings of conversations will not be limited to a few specific domains. This not only poses difficulties for obtaining domain-specific training data, but also implies an unlimited vocabulary of query phrases users want to use. Third, audio recordings will accumulate. When the audio database grows to hundreds or even thousands of hours, a reasonable response time is still needed.

A successful way to deal with high word error rates is the use of recognition alternates (lattices). For example, (Seide and Yu, 2004; Yu and Seide, 2004) reports a substantial 50% improvement of FOM (Figure Of Merit) for a word-spotting task in voicemails. Improvements from using lattices were also reported by (Saraclar and Sproat, 2004) and (Chelba and Acero, 2005).

To address the problem of domain independence, a subword-based approach is needed. In (Logan, 2002) the authors address the problem by indexing phonetic or word-fragment based transcriptions. Similar approaches, e.g. using overlapping $M$-grams of phonemes, are discussed in (Schäuble, 1995) and (Ng, 2000). (James and Young, 1994) introduces the approach of searching phoneme lattices. (Clements, 2001) proposes a similar idea called "phonetic search track." In previous work (Seide and Yu, 2004), promising results were obtained with phonetic lattice search in voicemails. In (Yu and



*Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 947–954, Vancouver, October 2005. 

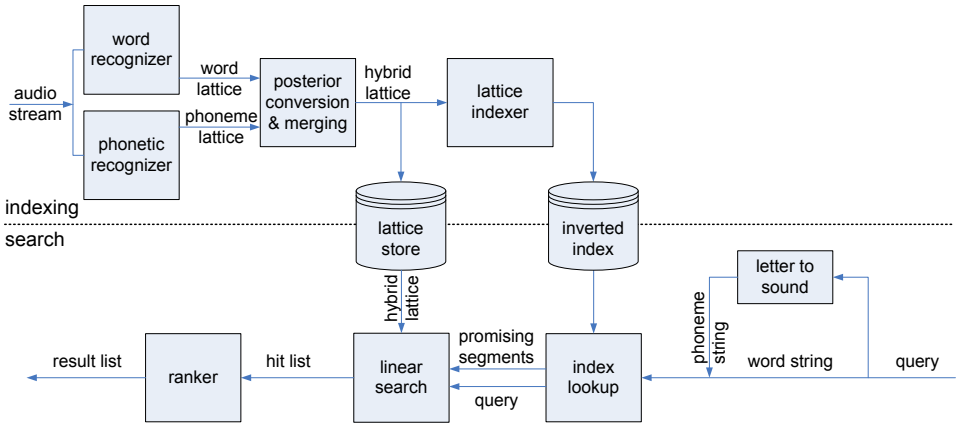


Figure 1: System architecture.

Seide, 2004), it was found that even better result can be achieved by combining a phonetic search with a word-level search.

For the third problem, quick response time is commonly achieved by indexing techniques. However, in the context of phonetic lattice search, the concept of "indexing" becomes a non-trivial problem, because due to the unknown-word nature, we need to deal with an open set of index keys. (Saraclar and Sproat, 2004) proposes to store the individual lattice arcs (inverting the lattice). (Allauzen *et al.*, 2004) introduces a general indexation framework by indexing expected term frequencies ("expected counts") instead of each individual keyword occurrence or lattice arcs. In (Yu *et al.*, 2005), a similar idea of indexing expected term frequencies is proposed, suggesting to approximate expected term frequencies by $M$-*gram phoneme language models* estimated on segments of audio.

In this paper, we combine previous work on phonetic lattice search, hybrid search and lattice indexing into a real system for searching recorded conversations that achieves high accuracy and can handle hundreds of hours of audio. The main contributions of this paper are: a real system for searching conversational speech, a novel method for combining phoneme and word lattices, and experimental results for searching recorded conversations.

The paper is organized as follows. Section 2 gives an overview of the system. Section 3 introduces the overall criterion, based on which the system is developed, Section 4 introduces our implementation for a hybrid word/phoneme search system, and Section 5 discusses the lattice indexing mechanism. Section 6 presents the experimental results, and Section 7 concludes.

## 2 A System For Searching Conversations

A system for searching the spoken content of recorded conversations has several distinct properties. Users are searching their own meetings, so most searches will be known-item searches with at most a few correct hits in the archive. Users will often search for specific phrases that they remember, possibly with boolean operators. Relevance weighting of individual query terms is less of an issue in this scenario.

We identified three user requirements:

- high recall and accurate ranking of phrase matches;
- domain independence – it should work for any topic, ideally without need to adapt vocabularies or language models;
- reasonable response time – a few seconds at most, independent of the size of the conversation archive.

We address them as follows. First, to increase recall we search *recognition alternates* based on *lattices*. Lattice oracle word-error rates[1] are significantly lower than word-error rates of the best path. For example, (Chelba and Acero, 2005) reports a lattice oracle error rate of 22% for lecture recordings at a top-1 word-error rate of 45%[2]. To utilize recognizer scores in the lattices, we formulating the ranking problem as one of risk minimization and derive that keyword hits should be ranked by their *word (phrase) posterior probabilities*.

Second, domain independence is achieved by combining large-vocabulary recognition with a phonetic search. This helps especially for proper names and specialized terminology, which are often either missing in the vocabulary or not well-predicted by the language model.

Third, to achieve quick response time, we use an $M$-gram based indexing approach. It has two stages, where the first stage is a fast index lookup to create a short-list of candidate lattices. In the second stage, a detailed lattice match is applied to the lattices in the short-list. We call the second stage *linear search* because search time grows linearly with the duration of the lattices searched.

[1]The "oracle word-error rate" of a lattice is the word error rate of the path through the lattice that has the least errors.

[2]Note that this comparison was for a reasonably well-tuned recognizer setup. Any arbitrary lattice oracle error rate can be obtained by adjusting the recognizer's pruning setup and investing enough computation time (plus possibly adapting the search-space organization).

The resulting system architecture is shown in Fig. 1. In the following three sections, we will discuss our solutions in these three aspects in details respectively.

## 3 Ranking Criterion

For ranking search results according to "relevance" to the user's query, several relevance measures have been proposed in the text-retrieval literature. The key element of these measures is weighting the contribution of individual keywords to the relevance-ranking score. Unfortunately, text ranking criteria are not directly applicable to retrieval of speech because recognition alternates and confidence scores are not considered.

Luckily, this is less of an issue in our known-item style search, because the simplest of relevance measures can be used: A search hit is assumed relevant if the query phrase was indeed said there (and fulfills optional boolean constraints), and it is not relevant otherwise.

This simple relevance measure, combined with a variant of the *probability ranking principle* (Robertson, 1977), leads to a system where phrase hits are ranked by their phrase posterior probability. This is derived through a Bayes-risk minimizing approach as follows:

1. Let the relevance be $R(Q, \mathrm{hit}^i)$ of a returned audio hit – $\mathrm{hit}^i$ to a user's query $Q$ formally defined is 1 (match) if the hit is an occurrence of the query term with time boundaries $(t_s^{\mathrm{hit}^i}, t_e^{\mathrm{hit}^i})$, or 0 if not.

2. The user expects the system to return a list of audio hits, ranked such that the *accumulative relevance* of the top $n$ hits $(\mathrm{hit}^1 ... \mathrm{hit}^n)$, averaged over a range of $n = 1...n_{\max}$, is maximal:

$$\frac{1}{n_{\max}} \sum_{n=1}^{n_{\max}} \sum_{i=1}^{n} R(Q, \mathrm{hit}^i) \quad \stackrel{!}{=} \quad \max. \qquad (1)$$

   Note that this is closely related to popular word-spotting metrics, such as the NIST (National Institute of Standards & Technology) Figure Of Merit.

To the retrieval system, the true transcription of each audio file is unknown, so it must maximize Eq. (1) in the sense of an expected value

$$E_{WT|O} \left\{ \frac{1}{n_{\max}} \sum_{n=1}^{n_{\max}} \sum_{i=1}^{n} R_{WT}(Q, \mathrm{hit}^i) \right\} \quad \stackrel{!}{=} \quad \max,$$

where $O$ denotes the totality of all audio files ($O$ for observation), $W = (w_1, w_2, ..., w_N)$ a hypothesized transcription of the entire collection, and $T = (t_1, t_2, ..., t_{N+1})$ the associated time boundaries on a shared collection-wide time axis.

$R_{WT}(\cdot)$ shall be relevance w.r.t. the hypothesized transcription and alignment. The expected value is taken w.r.t. the posterior probability distribution $P(WT|O)$ provided by our speech recognizer in the form of scored lattices. It is easy to see that this expression is maximal if the hits are ranked by their expected relevance $E_{WT|O}\{R_{WT}(Q, \mathrm{hit}^i)\}$. In our definition of relevance, $R_{WT}(Q, \mathrm{hit}^i)$ is written as

$$R_{WT}(Q, \mathrm{hit}^i) \quad = \quad \begin{cases} 1 & \exists k, l : t_k = t_s^{\mathrm{hit}^i} \\ & \quad \wedge t_{k+l} = t_e^{\mathrm{hit}^i} \\ & \quad \wedge w_k, ..., w_{k+l-1} = Q \\ 0 & \text{otherwise} \end{cases}$$

and the expected relevance is computed as

$$\begin{aligned} E_{WT|O}\{R_{WT}(Q, \mathrm{hit}^i)\} &= \sum_{WT} R_{WT}(Q, \mathrm{hit}^i) P(WT|O) \\ &= P(*, t_s^{\mathrm{hit}^i}, Q, t_e^{\mathrm{hit}^i}, *|O) \end{aligned}$$

with

$$P(*, t_s, Q, t_e, *|O) \quad = \sum_{\substack{WT: \exists k,l: t_k = t_s \wedge t_{k+l} = t_e \\ \wedge w_k, ..., w_{k+l-1} = Q}} P(WT|O). \qquad (2)$$

For single-word queries, this is the well-known *word posterior probability* (Wessel *et al.*, 2000; Evermann *et al.*, 2000). To cover multi-label phrase queries, we will call it *phrase posterior probability*.

The formalism in this section is applicable to all sorts of units, such as fragments, syllables, or words. The transcription $W$ and its units $w_k$, as well as the query string $Q$, should be understood in this sense. For a regular word-level search, $W$ and $Q$ are just *strings of words* In the context of phonetic search, $W$ and $Q$ are *strings of phonemes*. For simplicity of notation, we have excluded the issue of multiple pronunciations of a word. Eq. (2) can be trivially extended by summing up over all alternative pronunciations of the query. And in a hybrid search, there would be multiple representations of the query, which are just as pronunciation variants.

## 4 Word/Phoneme Hybrid Search

For a combined word and phoneme based search, two problems need to be considered:

- Recognizer configuration. While established solutions exist for word-lattice generation, what needs to be done for generating high-quality phoneme lattices?
- How should word and phoneme lattices be jointly represented for the purpose of search, and how should they be searched?

### 4.1 Speech Recognition

#### 4.1.1 Large-Vocabulary Recognition

Word lattices are generated by a common speaker-independent large-vocabulary recognizer. Because the speaking style of conversations is very different from, say,

your average speech dictation system, specialized acoustic models are used. These are trained on conversational speech to match the speaking style. The vocabulary and the trigram language model are designed to cover a broad range of topics.

The drawback of large-vocabulary recognition is, of course, that it is infeasible to have the vocabulary cover all possible keywords that a user may use, particularly proper names and specialized terminology.

One way to address this *out-of-vocabulary problem* is to mine the user's documents or e-mails to adapt the recognizer's vocabulary. While this is workable for some scenarios, it is not a good solution e.g. when new words are frequently introduced in the conversations themselves rather than preceding written conversations, where the spelling of a new word is not obvious and thus inconsistent, or when documents with related documents are not easily available on the user's hard disk but would have to be specifically gathered by the user.

A second problem is that the performance of state-of-the-art speech recognition relies heavily on a well-trained domain-matched language model. Mining user data can only yield a comparably small amount of training data. Adapting a language model with it would barely yield a robust language model for newly learned words, and their usage style may differ in conversational speech.

For the above reasons, we decided not to attempt to adapt vocabulary and language model. Instead, we use a fixed broad-domain vocabulary and language model for large-vocabulary recognition, and augment this system with maintenance-free *phonetic search* to cover new words and mismatched domains.

#### 4.1.2 Phonetic Recognition

The simplest phonetic recognizer is a regular recognizer with the vocabulary replaced by the list of phonemes of the language, and the language model replaced by a phoneme $M$-gram. However, such phonetic language model is much weaker than a word language model. This results in poor accuracy and inefficient search.

Instead, our recognizer uses "phonetic word fragments" (groups of phonemes similar to syllables or half-syllables) as its vocabulary and in the language model. This provides phonotactic constraints for efficient decoding and accurate phoneme-boundary decisions, while remaining independent of any specific vocabulary. A set of about 600 fragments was automatically derived from the language-model training set by a bottom-up grouping procedure (Klakow, 1998; Ng, 2000; Seide and Yu, 2004). Example fragments are /-k-ih-ng/ (the syllable -*king*), /ih-n-t-ax-r-/ (*inter*-), and /ih-z/ (the word *is*).

With this, lattices are generated using the common Viterbi decoder with word-pair approximation (Schwartz *et al.*, 1994; Ortmanns *et al.*, 1996). The decoder has been modified to keep track of individual phoneme boundaries and scores. These are recorded in the lattices, while fragment-boundary information is discarded. This way, phoneme lattices are generated.

In the results section we will see that, even with a well-trained domain-matching word-level language model, searching phoneme lattices can yield search accuracies comparable with word-level search, and that the best performance is achieved by combining both into a hybrid word/phoneme system.

### 4.2 Unified Hybrid Lattice Representation

Combining word and phonetic search is desirable because they are complementary: Word-based search yields better precision, but has a recall issue for unknown and rare words, while phonetic search has very good recall but suffers from poor precision especially for short words.

Combining the two is not trivial. Several strategies are discussed in (Yu and Seide, 2004), including using a hybrid recognizer, combining lattices from two separate recognizers, and combining the results of two separate systems. Both hybrid recognizer configuration and lattice combination turned out difficult because of the different dynamic range of scores in word and phonetic paths.

We found it beneficial to convert both lattices into posterior-based representations called *posterior lattices* first, which are then merged into a hybrid posterior lattice. Search is performed in a hybrid lattice in a unified manner using both phonetic and word representations as "alternative pronunciation" of the query, and summing up the resulting phrase posteriors.

Posterior lattices are like regular lattices, except that they do not store acoustic likelihoods, language model probabilities, and precomputed forward/backward probabilities, but *arc and node posteriors*. An arc's posterior is the probability that the arc (with its associated word or phoneme hypothesis) lies on the correct path, while a node posterior is the probability that the correct path connects two word/phoneme hypotheses through this node. In our actual system, a node is only associated with a point in time, and the node posterior is the probability of having a word or phoneme boundary at its associated time point.

The inclusion of node posteriors, which to our knowledge is a novel contribution of this paper, makes an exact computation of phrase posteriors from posterior lattices possible. In the following we will explain this in detail.

#### 4.2.1 Arc and Node Posteriors

A lattice $\mathcal{L} = (\mathcal{N}, \mathcal{A}, n_{\text{start}}, n_{\text{end}})$ is a directed acyclic graph (DAG) with $\mathcal{N}$ being the set of nodes, $\mathcal{A}$ is the set of arcs, and $n_{\text{start}}$, $n_{\text{end}} \in \mathcal{N}$ being the unique initial and unique final node, respectively. Nodes represent times and possibly context conditions, while arcs represent word or phoneme hypotheses.[3]

Each node $n \in \mathcal{N}$ has an associated time $t[n]$ and possibly an acoustic or language-model context condition. Arcs are 4-tuples $a = (S[a], E[a], I[a], w[a])$. $S[a]$, $E[a]$

[3] Alternative definitions of lattices are possible, e.g. nodes representing words and arcs representing word transitions.

$\in \mathcal{N}$ denote the start and end node of the arc. $I[a]$ is the arc label[4], which is either a word (in word lattices) or a phoneme (in phonetic lattices). Last, $w[a]$ shall be a weight assigned to the arc by the recognizer. Specifically, $w[a] = p_{\mathrm{ac}}(a)^{1/\lambda} \cdot P_{\mathrm{LM}}(a)$ with acoustic likelihood $p_{\mathrm{ac}}(a)$, language model probability $P_{\mathrm{LM}}$, and language-model weight $\lambda$.

In addition, we define *paths* $\pi = (a_1, \cdots, a_K)$ as *sequences* of connected arcs. We use the symbols $S$, $E$, $I$, and $w$ for paths as well to represent the respective properties for entire paths, i.e. the path start node $S[\pi] = S[a_1]$, path end node $E[\pi] = E[a_K]$, path label sequence $I[\pi] = (I[a_1], \cdots, I[a_K])$, and total path weight $w[\pi] = \prod_{k=1}^{K} w[a_k]$.

Finally, we define $\Pi(n_1, n_2)$ as the entirety of all paths that start at node $n_1$ and end in node $n_2$: $\Pi(n_1, n_2) = \{\pi | S[\pi] = n_1 \wedge E[\pi] = n_2\}$.

With this, the phrase posteriors defined in Eq. 2 can be written as follows.

In the simplest case, $Q$ is a single word token. Then, the phrase posterior is just the word posterior and, as shown in e.g. (Wessel *et al.*, 2000) or (Evermann *et al.*, 2000), can be computed as

$$\begin{aligned} P(*, t_s, Q, t_e, *|O) &= \frac{\sum\limits_{\substack{\pi=(a_1,\cdots,a_K) \in \Pi(n_{\mathrm{start}}, n_{\mathrm{end}}): \\ \exists l: [S[a_l]]=t_s \wedge t[E[a_l]]=t_e \wedge I[a_l]=Q}} w[\pi]}{\sum\limits_{\pi \in \Pi(n_{\mathrm{start}}, n_{\mathrm{end}})} w[\pi]} \\ &= \sum_{\substack{a \in \mathcal{A}: t[S[a]]=t_s \\ \wedge t[E[a]]=t_e \wedge I[a]=Q}} P_{\mathrm{arc}}[a] \end{aligned} \quad (3)$$

with $P_{\mathrm{arc}}[a]$ being the *arc posterior* defined as

$$P_{\mathrm{arc}}[a] = \frac{\alpha_{S[a]} \cdot w[a] \cdot \beta_{E[a]}}{\alpha_{n_{\mathrm{end}}}}$$

with the *forward/backward probabilities* $\alpha_n$ and $\beta_n$ defined as:

$$\begin{aligned} \alpha_n &= \sum_{\pi \in \Pi(n_{\mathrm{start}}, n)} w[\pi] \\ \beta_n &= \sum_{\pi \in \Pi(n, n_{\mathrm{end}})} w[\pi]. \end{aligned}$$

$\alpha_n$ and $\beta_n$ can conveniently be computed from the word lattices by the well-known forward/backward recursion:

$$\alpha_n = \begin{cases} 1.0 & n = n_{\mathrm{start}} \\ \sum\limits_{a: E[a]=n} \alpha_{S[a]} \cdot w[a] & \text{otherwise} \end{cases}$$

$$\beta_n = \begin{cases} 1.0 & n = n_{\mathrm{end}} \\ \sum\limits_{a: S[a]=n} w[a] \cdot \beta_{E[a]} & \text{otherwise.} \end{cases}$$

[4]Lattices are often interpreted as weighted finite-state acceptors, where the arc labels are the *input symbols*, hence the symbol $I$.

Now, in the general case of multi-label queries, the phrase posterior can be computed as

$$\begin{aligned} &P(*, t_s, Q, t_e, *|O) \\ &= \sum_{\substack{\pi=(a_1,\cdots,a_K): \\ t[S[\pi]]=t_s \wedge t[E[\pi]]=t_e \wedge I[\pi]=Q}} \frac{P_{\mathrm{arc}}[a_1] \cdots P_{\mathrm{arc}}[a_K]}{P_{\mathrm{node}}[S[a_2]] \cdots P_{\mathrm{node}}[S[a_K]]} \end{aligned}$$

with $P_{\mathrm{node}}[n]$, the *node posterior*[5], defined as

$$P_{\mathrm{node}}[n] = \frac{\alpha_n \cdot \beta_n}{\alpha_{n_{\mathrm{end}}}}. \quad (4)$$

#### 4.2.2 Advantages of Posterior Lattices

The posterior-lattice representation has several advantages over traditional lattices. First, lattice storage is reduced because only one value (node posterior) needs to be stored per node instead of two ($\alpha$, $\beta$)[6]. Second, node and arc posteriors have a smaller and similar dynamic range than $\alpha_n$, $\beta_n$, and $w[a]$, which is beneficial when the values should be stored with a small number of bits.

Further, for the case of word-based search, the summation in Eq. 3 can also be precomputed by merging all lattice nodes that carry the same time label, and merging the corresponding arcs by summing up their arc posteriors. In such a "pinched" lattice, word posteriors for single-label queries can now be looked up directly. However, posteriors for multi-label strings cannot be computed precisely anymore. Our experiments have shown that the impact on ranking accuracy caused by this approximation is neglectable. Unfortunately, we have also found that the same is not true for phonetic search.

The most important advantage of posterior lattices for our system is that they provide a way of combining the word and phoneme lattices into a single structure – by simply merging their start nodes and their end nodes. This allows to implement hybrid queries in a single unified search, treating the phonetic and the word-based representation of the query as alternative pronunciations.

## 5 Lattice Indexing

Searching lattices is time-consuming. It is not feasible to search large amounts of audio. To deal with hundreds or even thousands of hours of audio, we need some form of inverted indexing mechanism.

This is comparably straight-forward when indexing text. It is also not difficult for indexing word lattices. In both case, the set of words to index is known. However, indexing phoneme lattices is very different, because theoretically any phoneme string could be an indexing item.

[5]Again, mind that in our lattice formulation word/phoneme hypotheses are represented by arcs, while nodes just represent connection points. The node posterior is the probability that the correct path passes through a connection point.

[6]Note, however, that storage for the traditional lattice can also be reduced to a single number per node by weight pushing (Saraclar and Sproat, 2004), using an algorithm that is very similar to the forward/backward procedure.

We address this by our $M$-gram lattice-indexing scheme. It was originally designed for phoneme lattices, but can be – and is actually – used in our system for indexing word lattices.

First, audio files are clipped into homogeneous segments. For an audio segment $i$, we define the *expected term frequency* (ETF) of a query string $Q$ as summation of phrase posteriors of all hits in this segment:

$$\begin{aligned}\mathrm{ETF}_i(Q) &= \sum_{\forall t_s,t_e} P(*,t_s,Q,t_e,*|O^i) \\ &= \sum_{\pi\in\Pi^i:I[\pi]=Q} p[\pi]\end{aligned}$$

with $\Pi^i$ being the set of all paths of segment $i$.

At indexing time, ETFs of a list of $M$-grams for each segment are calculated. They are stored in an inverted structure that allows retrieval by $M$-gram.

In search time, the ETFs of the query string are estimated by the so-called "$M$-gram approximation". In order to explain this concept, we need to first introduce $P(Q|O^i)$ – the probability of observing query string $Q$ at any word boundary in the recording $O^i$. $P(Q|O^i)$ has a relationship with ETF as

$$\mathrm{ETF}_i(Q) = \tilde{N}_i \cdot P(Q|O^i)$$

with $\tilde{N}_i$ being the expected number of words in the segment $i$. It can also be computed as

$$\tilde{N}_i = \sum_{n\in\mathcal{N}_i} p[n],$$

where $\mathcal{N}_i$ is the node set for segment $i$.

Like the $M$-gram approximation in language-model theory, we approximate $P(Q|O^i)$ as

$$\begin{aligned}P(Q|O^i) &\approx \tilde{P}(Q|O^i) \\ &= \prod_{k=1}^{l} \tilde{P}(q_k|q_{k-M+1},\cdots,q_{k-1},O^i),\end{aligned}$$

while the right-hand items can be calculated from $M$-gram ETFs:

$$\begin{aligned}&\tilde{P}(q_k|q_{k-M+1},\cdots,q_{k-1},O^i) \\ &\quad= \frac{\mathrm{ETF}_i(q_{k-M+1},\cdots,q_k)}{\mathrm{ETF}_i(q_{k-M+1},\cdots,q_{k-1})}.\end{aligned}$$

The actual implementation uses only $M$-grams extracted from a large background dictionary, with a simple backoff strategy for unseen $M$-grams, see (Yu *et al.*, 2005) for details.

The resulting index is used in a two stage-search manner: The index itself is only used as the first stage to determine a short-list of promising segments that may contain the query. The second stage involves a linear lattice search to get final results.

Table 1: Test corpus summary.

| test set | duration | #segments | keyword set (incl. OOV) |
|---|---|---|---|
| ICSI meetings | 2.0h | 429 | 1878 (96) |
| SWBD eval2000 | 3.6h | 742 | 2420 (215) |
| SWBD rt03s | 6.3h | 1298 | 2325 (236) |
| interviews (phone) | 1.1h | 267 | 1057 (49) |
| interviews (lapel) | 1.0h | 244 | 1629 (107) |

# 6 Results

## 6.1 Setup

We have evaluated our system on five different corpora of recorded conversations:

- one meeting corpus (NIST "RT04S" development data set, ICSI portion, (NIST, 2000-2004))
- two eval sets from the switchboard (SWBD) data collection ("eval 2000" and "RT03S", (NIST, 2000-2004))
- two in-house sets of interview recordings of about one hour each, one recorded over the telephone, and one using a single microphone mounted in the interviewee's lapel.

For each data set, a keyword list was selected by an automatic procedure (Seide and Yu, 2004). Words and multi-word phrases were selected from the reference transcriptions if they occurred in at most two segments. Example keywords are *overseas*, *olympics*, and *"automated accounting system"*. For the purpose of evaluation, those data sets are cut into segments of about 15 seconds each. The size of the corpora, their number of segments, and the size of the selected keyword set are given in Table 1.

The acoustic model we used is trained on 309h of the Switchboard corpus (SWBD-1). The LVCSR language model was trained on the transcriptions of the Switchboard training set, the ICSI-meeting training set, and the LDC Broadcast News 96 and 97 training sets. No dedicated training data was available for the in-house interview recordings. The recognition dictionary has 51388 words. The phonetic language model was trained on the phonetic version of the transcriptions of SWBD-1 and Broadcast News 96 plus about 87000 background dictionary entries, a total of 11.8 million phoneme tokens.

To measure the search accuracy, we use the "Figure Of Merit" (FOM) metric defined by NIST for word-spotting evaluations. In its original form, it is the average of detection/false-alarm curve taken over the range [0..10] false alarms per hour per keyword. Because manual word-level alignments of our test sets were not available, we modified the FOM such that a correct hit is a 15-second segment that contains the key phrase.

Besides FOM, we use a second metric – "Top Hit Precision" (THP), defined as the correct rate of the best ranked hit. If no hit is returned for an existing query term, it is counted as an error. Both of these metrics are relevant measures in our known-item search.

Table 2: Baseline transcription word-error rates (WER) as well as precision (P), recall (R), FOM and THP for searching the transcript.

| test set | WER [%] | P [%] | R [%] | FOM [%] | THP [%] |
|---|---|---|---|---|---|
| ICSI meetings | 44.1 | 80.6 | 43.8 | 43.6 | 43.6 |
| SWBD eval2000 | 39.0 | 79.6 | 41.1 | 41.1 | 41.1 |
| SWBD rt03s | 45.2 | 72.6 | 36.3 | 36.3 | 36.0 |
| interviews (phone) | 57.7 | 68.8 | 31.6 | 29.3 | 31.3 |
| interviews (lapel) | 62.8 | 80.1 | 32.0 | 30.2 | 32.1 |
| average | 49.8 | 76.3 | 37.0 | 36.1 | 36.8 |

Table 3: Comparison of search accuracy for word, phoneme, and hybrid lattices.

| test set | word | phoneme | hybrid |
|---|---|---|---|
| Figure Of Merit (FOM) [%] | | | |
| ICSI meetings | 72.1 | 81.2 | 88.2 |
| SWBD eval2000 | 71.3 | 80.4 | 87.3 |
| SWBD rt03s | 66.4 | 76.9 | 84.2 |
| interviews (phone) | 60.6 | 73.7 | 83.3 |
| interviews (lapel) | 59.0 | 70.2 | 77.7 |
| average | 65.9 | 76.5 | 84.1 |
| INV words only | 69.4 | 77.0 | 84.7 |
| OOV words only | 0 | 73.8 | 73.8 |
| Top Hit Precision (THP) [%] | | | |
| ICSI meetings | 67.2 | 65.0 | 78.7 |
| SWBD eval2000 | 67.1 | 63.6 | 77.9 |
| SWBD rt03s | 59.6 | 59.1 | 71.7 |
| interviews (phone) | 55.7 | 64.4 | 73.1 |
| interviews (lapel) | 55.6 | 59.7 | 71.2 |
| average | 61.0 | 62.4 | 74.5 |
| INV words only | 64.5 | 62.4 | 75.3 |
| OOV words only | 0 | 60.5 | 60.5 |

### 6.2 Word/Phoneme Hybrid Search

Table 2 gives the LVSCR transcription word-error rates for each set. Almost all sets have a word-error rates above 40%. Searching those speech recognition transcriptions results in FOM and THP values below 40%.

Table 3 gives results of searching in word, phoneme, and hybrid lattices. First, for all test sets, word-lattice search is drastically better than transcription-only search.

Second, comparing word-lattice and phoneme-lattice search, phoneme lattices outperforms word lattices on all tests in terms of FOM. This is because phoneme lattice has better recall rate. For THP, word lattice search is slightly better except on the interview sets for which the language model is not well matched. Hybrid search leads to a substantial improvement over each (27.6% average FOM improvement and 16.2% average THP improvement over word lattice search). This demonstrates the complementary nature of word and phoneme search.

We also show results separately for known words (invocabulary, INV) and out-of-vocabulary words (OOV). Interestingly, even for known words, hybrid search leads to a significant improvement (get 22.0% for FOM and 16.7% for THP) compared to using word lattices only.

### 6.3 Effect of Node Posterior

In Section 4.2, we have shown that phrase posteriors can be computed from posterior lattices if they include both arc and node posteriors (Eq. 4). However, posterior representations of lattices found in literature only include word (arc) posteriors, and some posterior-based systems simply ignore the node-posterior term, e.g. (Chelba and Acero, 2005). In Table 4, we evaluate the impact on accuracy when this term is ignored. (In this experiment, we bypassed the index-lookup step, thus the numbers are slightly different from Table 3.)

We found that for word-level search, the effect of node posterior compensation is indeed neglectable. However, for phonetic search it is not: We observe a 4% relative FOM loss.

### 6.4 Index Lookup and Linear Search

Section 5 introduced a two-stage search approach using an $M$-gram based indexing scheme. How much accuracy is lost from incorrectly eliminating correct hits in the first (index-based) stage? Table 5 compares three setups. The first column shows results for linear search only: no index lookup used at all, a complete linear search is performed on all lattices. This search is optimal but does not scale up to large database. The second column shows index lookup *only*. Segments are ranked by the approximate $M$-gram based ETF score obtained from the index. The third column shows the two-stage results.

The index-based two-stage search is indeed very close to a full linear search (average FOM loss of 1.2% and THP loss of 0.2% points). A two-stage search takes under two seconds and is mostly independent of the database size. In other work, we have applied this technique successfully to search a database of nearly 200 hours.

### 6.5 The System

Fig. 2 shows a screenshot of a research prototype for a search-enabled audio notebook. In addition to a note-taking area (bottom) and recording controls, it includes a rich audio browser showing speaker segmentation and automatically identified speaker labels (both not scope of this paper). Results of keyword searches are shown as color highlights, which are clickable to start playback at that position.

## 7 Conclusion

In this paper, we have presented a system for searching recordings of conversational speech, particularly meet-

Table 4: Effect of ignoring the node-posterior term in phrase-posterior computation (shown for ICSI meeting set only).

| FOM | word | phoneme |
|---|---|---|
| exact computation | 72.1 | 82.3 |
| node posterior ignored | 72.0 | 79.2 |
| relative change [%] | -0.1 | -3.8 |

Table 5: Comparing the effect of lattice indexing. Shown is unindexed "linear search," index lookup only (segments selected via the index without subsequent linear search), and the combination of both.

| test set | linear search | index lookup | two-stage |
|---|---|---|---|
| Figure Of Merit (FOM) [%] | | | |
| ICSI meetings | 88.6 | 86.4 | 88.2 |
| SWBD eval2000 | 88.7 | 86.5 | 87.3 |
| SWBD rt03s | 87.3 | 85.1 | 84.2 |
| interviews (phone) | 83.8 | 81.2 | 83.3 |
| interviews (lapel) | 78.3 | 76.1 | 77.7 |
| average | 85.3 | 83.1 | 84.1 |
| Top Hit Precision (THP) [%] | | | |
| ICSI meetings | 78.8 | 70.7 | 78.7 |
| SWBD eval2000 | 78.0 | 71.4 | 77.9 |
| SWBD rt03s | 71.9 | 65.7 | 71.7 |
| interviews (phone) | 73.8 | 64.6 | 73.1 |
| interviews (lapel) | 70.8 | 65.9 | 71.2 |
| average | 74.7 | 67.7 | 74.5 |

ings and telephone conversations. We identified user requirements as accurate ranking of phrase matches, domain independence, and reasonable response time. We have addressed these by hybrid word/phoneme lattice search and a supporting indexing scheme. Unlike many other spoken-document retrieval systems, we search recognition alternates instead of only speech recognition transcripts. This yields a significant improvement of keyword spotting accuracy. We have combined word-level search with phonetic search, which not only enables the system to handle the open-vocabulary problem, but also substantially improves in-vocabulary accuracy. We have proposed a posterior-lattice representation that allows for unified word and phoneme indexing and search. To speed up the search process, we proposed $M$-gram based lattice indexing, which extends our open vocabulary search ability for large collection of audio. Tested on five different recording sets including meetings, conversations, and interviews, a search accuracy (FOM) of 84% has been achieved – dramatically better than searching speech recognition transcripts (under 40%).

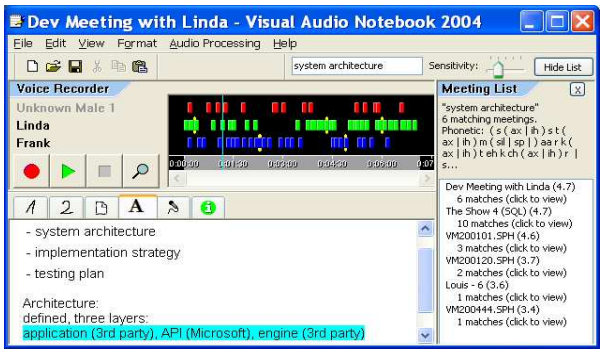


Figure 2: Screenshot of our research prototype of a search-enabled audio notebook.

## 8 Acknowledgements

The authors wish to thank our colleagues Asela Gunawardana, Patrick Nguyen, Yu Shi, and Ye Tian for sharing their Switchboard setup and models with us; and Frank Soong for his valuable feedback on this paper.

## References

C. Allauzen, M. Mohri, M. Saraclar, General indexation of weighted automata – application to spoken utterance retrieval. *Proc. HLT'2004*, Boston, 2004.

C. Chelba and A. Acero, Position specific posterior lattices for indexing speech. *Proc. ACL'2005*, Ann Arbor, 2005.

Mark Clements *et al.*, Phonetic Searching vs. LVCSR: How to find what you really want in audio archives. *Proc. AVIOS'2001*, San Jose, 2001.

G. Evermann *et al.*, Large vocabulary decoding and confidence estimation using word posterior probabilities. *Proc. ICASSP'2000*, Istanbul, 2000

J. Garofolo, TREC-9 Spoken Document Retrieval Track. National Institute of Standards and Technology, `http://trec.nist.gov/pubs/trec9/sdrt9_slides/sld001.htm`.

D. A. James and S. J. Young, A fast lattice-based approach to vocabulary-independent wordspotting. *Proc. ICASSP'1994*, Adelaide, 1994.

D. Klakow. Language-model optimization by mapping of corpora. *Proc. ICASSP'1998*.

Beth Logan *et al.*, An experimental study of an audio indexing system for the web. *Proc. ICSLP'2000*, Beijing, 2000.

Beth Logan *et al.*, Word and subword indexing approaches for reducing the effects of OOV queries on spoken audio. *Proc. HLT'2002*, San Diego, 2002.

Kenney Ng, Subword-based approaches for spoken document retrieval. PhD thesis, Massachusetts Institute of Technology, 2000.

NIST Spoken Language Technology Evaluations, `http://www.nist.gov/speech/tests/`.

S. Ortmanns, H. Ney, F. Seide, and I. Lindam, A comparison of the time conditioned and word conditioned search techniques for large-vocabulary speech recognition. *Proc. ICSLP'1996*, Philadelphia, 1996.

S. E. Robertson, The probability ranking principle in IR. Journal of Documentation 33 (1977).

M. Saraclar, R. Sproat, Lattice-based search for spoken utterance retrieval. *Proc. HLT'2004*, Boston, 2004.

P. Schäuble *et al.*, First experiences with a system for content based retrieval of information from speech recordings. *Proc. IJCAI'1995*, Montreal, 1995.

R. Schwartz *et al.*, A comparison of several approximate algorithms for finding multiple (n-best) sentence hypotheses. *Proc. ICSLP'1994*, Yokohama, 1994.

F. Seide, P. Yu, *et al.*, Vocabulary-independent search in spontaneous speech. *Proc. ICASSP'2004*, Montreal, 2004.

Lisa Joy Stifelman, The Audio Notebook. PhD thesis, Massachusetts Institute of Technology, 1997.

F. Wessel, R. Schlüter, and H. Ney, Using posterior word probabilities for improved speech recognition. *Proc. ICASSP'2000*, Istanbul, 2000.

P. Yu, F. Seide, A hybrid word / phoneme-based approach for improved vocabulary-independent search in spontaneous speech. *Proc. ICLSP'04*, Jeju, 2004.

P. Yu, K. J. Chen, C. Y. Ma, F. Seide, Vocabulary-Independent Indexing of Spontaneous Speech, to appear in IEEE transaction on Speech and Audio Processing, Special Issue on Data Mining of Speech, Audio and Dialog.