Word-Lattice Based Spoken-Document Indexing with Standard Text Indexers

Roger Peng Yu, Kit Thambiratnam, and Frank Seide

Microsoft Research Asia, Beijing Sigma Center, 49 Zhichun Rd., 100080 Beijing, P.R.C.
{rogeryu,kit,fseide}@microsoft.com

ABSTRACT

Indexing the spoken content of audio recordings requires the use of automatic speech recognition, which is as of today not reliable. Unlike indexing text, we cannot reliably know from a speech recognizer whether a word is present or not at a given point in the audio; we can only obtain a probability for it. Making correct use of these probabilities can significantly improve spoken-document search accuracy.

First, we will describe how to improve accuracy for "web-search style" (AND/phrase) queries into audio by utilizing *speech recognition alternates* (competing word hypotheses during recognition) and *word posterior probabilities* (confidence scores), based on *word lattices.*

Then, we will present an end-to-end approach to doing so *using standard text indexers*, which by design cannot handle probabilities and unaligned alternates. We present a sequence of approximations that transform the numeric lattice-matching problem into a symbolic text-based one that can be implemented by a commercial full-text indexer.

Experiments on a 170-hour lecture set show an accuracy improvement by 30-60% for phrase searches and by 130% for two-term AND queries, compared to indexing linear text.

General Terms

Audio Indexing, Word Lattice, Posterior, Full-Text Indexing

1. INTRODUCTION

The tremendous progress in audio compression and storage technologies and the pervasive adoption of computing at work and in our daily lives has fostered a dramatic increase of the use of digital media not only on the Internet but also in the enterprise, such as online lecture videos, archived meetings or conference calls, and voicemail. Tools are needed to efficiently manage digital audio assets—audio or video recordings with intrinsic value to the enterprise. The key problem is keyword search into the spoken audio content.

On the Internet, the problem is somewhat alleviated because audio/video search engines often have metadata at their disposal, such as anchor text, surrounding text, closed captions, etc. For enterprise audio, we often have no such luxury: such information is commonly not available. We cannot escape to process the audio itself—by speech recognition to *index the spoken content.* However, typical enterprise audio is still a challenge for today's speech-recognition technology, which achieves word accuracies of only 50-70% [1, 2, 3].

SIGIR'08, July 20–24, 2008, Singapore.
ACM 978-1-60558-164-4/08/07.

In this paper, we will first describe how to improve accuracy for "web-search style" AND and phrase queries into audio. Unlike indexing text, we cannot reliably know from a speech recognizer whether a word is present or not at a given point in the audio; we can only obtain a probability for it. To this end, we will show how the *posterior probability that an audio document contains all query terms and phrases* can be computed using *speech recognition alternates* (competing word hypotheses during recognition) and confidence scores (word posterior probabilities), based on *word lattices.*

The document-level posteriors are then compared to a threshold representing the minimum precision required by the user. Experiments on a 170-hour lecture set show an accuracy improvement by 30-60% for multi-word phrase searches and by 130% for two-term AND queries, compared to indexing linear text.

Secondly, we will present an end-to-end approach to doing so *using standard text indexers*—which by their design cannot handle probabilities and unaligned alternates.

Text search engines are complex systems involving substantial investments—in both development and deployment—and also in order to seamlessly search text and audio/video materials. For this reason, maximizing their reuse is highly desireable. In fact, text indexers can "easily" be extended to index lattices (few extra bits of storage per hit, few lines of code). However, for the same reason, even minor changes can be costly and bear significant risk.

We present a sequence of approximations to transform the numeric lattice-matching problem into a symbolic text-based one implementable by a commercial full-text indexer.

This paper is organized as follows. The next section gives an overview about the concepts and prior work on spoken-document search. Then, we establish the theoretical foundation of word-lattice-based speech indexing in section 3. Section 4 introduces the proposed method to index word lattices with STIs, and section 5 presents experimental results. Section 6 concludes the paper.

2. SPOKEN-DOCUMENT SEARCH: OVERVIEW AND PRIOR WORK

Audio and video is among the least accessible content for search engines. Today's popular Internet video-search engines work by indexing meta-data, such as title and description texts, obtained from the media files themselves and also from hyperlinks in the form of anchor text or descriptive text surrounding a link. Also, for professionally produced TV content, closed captions often exist (although often not surfaced in Internet releases).

The meta-data approach is suitable for many types of Internet audio/video content such as podcasts and the Internet

release channel for TV content. However, in many scenarios such as "low-production value" enterprise recordings like phone-conference recordings or video-taped presentations, we often have no such luxury, sufficient meta-data is commonly not available, and we must consider the *spoken content of the audio itself*. The audio must be "cracked open" using automatic speech recognition.

Literature roughly distinguishes two categories of spoken-document search tasks:

- *Spoken Document Retrieval* (SDR): to find audio *documents that are relevant* to the query;
- *Spoken Term Detection* (STD): to locate *individual occurrences of query keywords*. This is also known as "keyword spotting".

2.1 Spoken-Document Retrieval (SDR)

"SDR" is often understood with the connotation of the TREC (Text REtrieval Conference) SDR benchmarks [4]. Here, queries were relatively long topic descriptions similar to TREC text benchmarks, and the task was to retrieve relevant broadcast news stories from a large broadcast-news archive.

The mainstream approach to this problem was straightforward: Transcribe the audio into text using large-vocabulary speech recognition, and then use known techniques for text retrieval on the transcripts, such as the vector-space model with weighting functions of the likes of TF.IDF or BM25.

The robustness of this approach was surprising to some: Even with transcription word-error rates as high as 35%, retrieval accuracy was on par compared to using manually created ground-truth transcriptions. Recognition did not lead to catastrophic failures due to the redundancy in the long queries and news segments. As a result, some researchers consider SDR a "solved problem" [4].

2.2 "Web-style" Search and Spoken-Term Detection (STD)

However, the kind of search that most computer users are familiar with—web-search engines—is different. In web search, queries are commonly short, 2–3 words, and unlike SDR vector-space matching, for a document to match, *all query terms must be present* (aside from query modifications like stemming and spelling normalization). Relevance ranking is applied only to the subset of documents that match the boolean condition specified by the query.

Thus, to create an audio-search engine that accommodates the expectation of web users, we are first faced with the problem of selecting documents that contain all query terms (a classic STD problem). For the remainder of this paper, we want to focus on this problem (while leaving the challenge of relevance weighting—arguably a non-speech problem—to our colleagues in the IR community).

2.3 Word Spotting vs. Large-Vocabulary Continuous Speech Recognition (LVCSR)

To detect query keywords in audio, a full transcription is actually not necessary. *Word spotting systems* use a recognition dictionary that consists only of the query terms and a "garbage model." The garbage model is to match all speech except the query terms themselves.

The key challenge of word spotters lies in the garbage model and the calibration between "garbage" and keyword scores. An alternative are "anti-models" [5]—models for the "complement" of each phoneme of the language trained on "complementary" phonemes or phoneme classes. Another challenge is that the vocabulary is defined at search time, so word-spotting systems don't lend themselves to indexing.

On the other hand, *large-vocabulary continuous-speech recognizers (LVCSR)* deploy vocabularies of up to 200,000 words and more. For the purpose of STD, all of these words except for the few query terms constitute a huge garbage model. Generally, LVCSR systems are significantly more accurate due to their use of word-M-gram language models—as long as all query terms are not only included in the vocabulary but also reasonably well-represented in the language model.

As a rule of thumb, word spotters without language model tend to suffer from poor Precision, while LVCSR systems are Recall-challenged, especially for rare query terms missing in the vocabulary or underrepresented in the language model. We found search accuracy for (phonetic lattice-based) word spotters to deteriorate about twice as fast with the log of the database size as LVCSR, rendering it unusable beyond a few hundreds of hours. Word spotters tend to be more suitable for *known-item searches* in smaller databases like personal meeting recordings, audio notes, or voicemail, while LVCSR-based systems are often the choice for *ad-hoc* search applications on large audio databases such as Internet video or lecture archives. We therefore focus on the LVCSR approach.

2.4 The Unknown-Word Problem

The unknown-word problem is not the topic of this paper, we therefore only want to briefly review this area (the interested reader shall be referred to [6]). If the query terms are not known at indexing time, this means that the recognition dictionary is defined at search time. An alternative to garbage-model based word spotters described above is *vocabulary-independent indexing using subword units*. One approach is to translate LVCSR transcriptions into phonetic or word-fragment sequences [7] or phonetic M-grams [8] and [9] and index those. These approaches are technically not STD, as some words are only partially matched. Accuracy improvements for OOV words are modest.

Another approach is to search *phoneme lattices*. A speech recognizer whose "dictionary" consists of the phonemes of the language recognizes the audio to generate phoneme hypotheses, which are kept in a lattice structure. At search time, the lattices are searched for phoneme sequences comprising the query keyword [10, 11]. [12] proposed a similar approach called "phonetic search track." Accuracy can be improved by hybrids combining phonetic and LVCSR [13, 14]. All of these are extensions of word spotting where the bulk of the speech-recognition effort is off-loaded to a preprocessing stage. Further search speed-up can be achieved by approximate indexing, e.g. of phoneme M-grams [6, 15].

2.5 Word Lattices Improve Accuracy

A major challenge remains the poor speech-to-text accuracy of state-of-the-art speech recognizers for general spoken content. With accuracies of only 50–70% [1, 2, 3], "naïve" methods based on speech-to-text transcripts suffer from poor Recall—If all query terms are required to be found for a document to match, a single query keyword missed causes the entire document to be missed.

Significant improvements can be achieved by taking the probabilistic nature of speech recognition into account [14, 16, 13, 17, 6]. The use of *word confidence scores* helps to reduce false positives, allows ranking the most reliable hits first, and by thresholding, users can adjust between "known-item" (high Recall) and "ad-hoc" (high Precision) type search. Secondly, using *alternative recognition candidates* instead of linear speech-to-text output significantly improves Recall for multi-word queries.

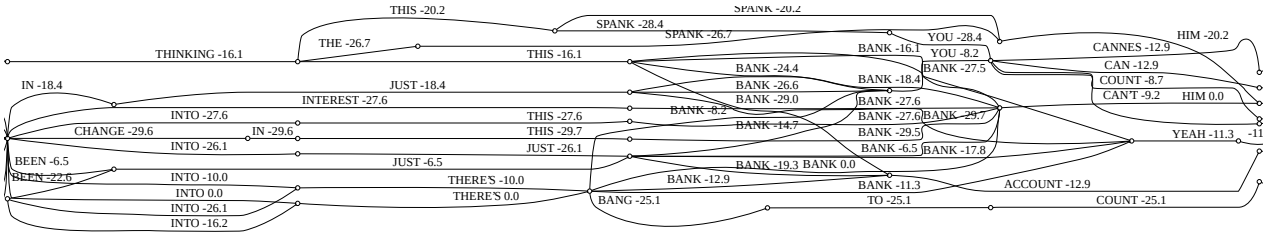


Figure 1: Word-lattice example for the word sequence "...into this bank account."

All required probabilities can be naturally computed from *word lattices*—compact representations of word candidates as directed acyclic graphs (DAG) that are augmented with recogniton probabilities and time information. Fig. 1 shows an example word lattice.

2.6 Indexing Word Lattices

Indexing word lattices is, *in principle*, no problem. For example, [14] proposed a direct inversion of raw lattices from the speech recognizer in the same way as text is indexed. No information is lost, and accuracy is the same as for directly searching the lattice.

A challenge is to *reuse existing* text search engines. This is highly desireable because search engines are complex systems involving substantial investments in both development and deployment, and also in order to seamlessly search text and audio material. However, standard full-text indexers cannot index lattices. This is because text indexers assign a unique word position to each word—thus they cannot represent the complex time structure of partially overlapping alternative recognition hypotheses. There is also no facility to store speech-recognition confidence scores. Lastly, word lattices can be as large as 100 times the size of text or more—beyond the range typical text search engines are optimized for.

[16] proposed a posterior-probability based approximate representation in which word hypotheses are merged w.r.t. word position, which is treated as a hidden variable. It more easily integrates with text search engines, but only achieves a small reduction of index entries and loses time information for individual hypotheses. An alternative are *confusion networks* ("sausages"), a method to align a speech lattice with its top-1 transcription [18]. Sausages are a parsimonious approximation of lattices. However, they do not lend themselves naturally for indexing due to the presence of a large number of null links,

In [17] we have proposed the "TMI" method (Time-based Merging for Indexing) that achieves significant reduction of lattice size by merging of similar hypotheses, and in [19] we introduced the "TALE" method (Time-Anchored Lattice Expansion) to map the complex time structure onto the word-position concept of text indexers, similar to [16] and [18]. The paper at hand extends this work in that we address the problem of storing posteriors and performing probabilistic searches using symbolic representation.

3. WORD POSTERIORS AND LATTICES

In this section, we want to formally specify the problem we are trying to solve, then motivate the use of word-posterior probabilities, and lastly show how we use word lattices to compute posteriors effiently.

In summary, we first limit the scope to the common "web-search" style queries where all query terms must be found in matching documents, without or with phrase constraints. We then argue that an important quantity is the *posterior probability that an audio document D fulfills the boolean condition* given by the query Q. This quantity is simply the product over the per-query-term probabilities of a query term q to *occur at least once* in D, which in turn can be easily computed from all locations where query term q was hypothesized and the corresponding *hit posterior probabilities*. Lastly, hit posteriors can be naturally computed from *word lattices*, which many state-of-the-art speech recognition packages can readily generate. The remainder of this section details all of this.

3.1 AND and Phrase Queries

The type of search query that is most familiar to us Internet users is a query that consists simply of a set or sequence of words, possibly with quotation marks to denote phrases.

Such a query is taken to specify the *boolean condition* that all of the query terms are required to be present in returned documents, and that, additionally, terms in quotes must match in precise sequence.

An Internet search engine first identifies all documents that fulfill the boolean condition, and then ranks them by "relevance" to the query using elaborate term and document weighting schemes.

In this paper, we will exclusively focus on such *AND/phrase queries*. Further, for the purpose of this paper, we shall not be concerned with relevance weighting, but simply *consider all audio documents that match the boolean condition equally relevant.*

Yet, there is value in ranking—because we can not reliably decide whether the boolean condition is fulfilled. We can only work with the *probability that an audio document matches the condition.* In the next subsection, we want to motivate that it is optimal to rank by exactly this probability.

3.2 Probability for Matching the Query

We formally specify our objective as follows: An (audio) document D shall be deemed "relevant" to a query Q if D fulfills the boolean condition specified by Q. We denote this as $R(Q, D) = 1$ for a relevant document, 0 otherwise.

Because the true transcription of the audio to be indexed is unknown, $R(Q, D)$ is not directly known. All we are provided with by the speech recognizer is

- hypotheses (W, T) on what might have been said (with $W = (w_1, w_2, ..., w_N)$ = hypothesized transcription of the audio database and $T = (t_1, t_2, ..., t_{N+1})$ = associated word time boundaries), and
- associated posterior probabilities $P(WT|O)$ (O for observation denoting the totality of all audio), denoting the probability that transcription hypothesis (W, T) was indeed the underlying event that caused the generation of the audio we observed.

With this, we can compute the *posterior probability that document D matches query Q* as:

$$\begin{aligned} P(R(Q,D)|O) &= E_{WT|O}\left\{R_{WT}(Q,D)\right\} \\ &= \sum_{WT} P(WT|O) \cdot R_{WT}(Q,D) \end{aligned} \quad (1)$$

where $R_{WT}(\cdot)$ shall be relevance *w.r.t. the hypothesized transcription/alignment*. The expectation is taken w.r.t. $P(WT|O)$ as provided by the recognizer.

By comparing this probability against a pre-determined threshold $P_{\min}$, we can select the documents that match the query with a probability of $P_{\min}$ or higher, where the threshold would be chosen so that a target Precision will be reached.

3.3 Hit Posteriors

We now incorporate our specific query type under consideration—all individual query terms $q \in Q$ shall be present in D. Under the assumption that the query terms are independent, we can rewrite $P(R(Q,D)|O)$ as:

$$P(R(Q,D)|O) \;=\; \prod_{q \in Q} P(R(q,D)|O)$$

where $P(R(q,D)|O)$ is the posterior probability that the *individual query term* q occurs in D (at least once).

To deal with phrases (multiple words "in quotes" that must match in precise sequence), we shall consider multi-word phrases simply as single query terms, i.e. a query term q can consist of a sequence of words that are required to match in precise sequence.

To further break down $P(R(q,D)|O)$, we define the *hit posterior* $P(*, t_s, q, t_e, *|O)$ to denote the posterior probability for the hypothesis that query term q occurred covering the precise time range $t_s...t_e$. It is computed by marginalizing out over all transcription hypotheses that contain q with precise time boundaries t_s and t_e:

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

Note that hit posteriors are not concerned with the context in which a query keyword is hypothesized to have occurred. Thus, Eq. (2) marginalizes out all contexts. Also, there are often multiple hypotheses for the same query keyword with slightly dissimilar time boundaries. For our purposes, small boundary variations should be considered "estimation noise" and be marginalized out as well. Therefore, in our practical implementation, we merge all hypotheses that overlap into the one with the highest word posterior.

With this, the probability that all terms of Q occur at least once in D can be easily computed as:

$$P(R(Q,D)|O) \;=\; \prod_{q \in Q} \left[1 - \prod_{(t_s,q,t_e)} (1 - P(*, t_s, q, t_e, *|O)) \right] \qquad (3)$$

where the product is taken over all time ranges (t_s, t_e) for which the recognizer hypothesized query term q. This is assumes that all those hits are independent, i.e. non-overlapping and sufficiently far enough from each other.

The remainder of this section will show how hit posteriors can computed efficiently and easily from word lattices, which we are going to introduce next.

3.4 Word Lattice Definition

Figure 1 shows an example word lattice. A lattice is a compact representation of speech recognition word hypotheses as a directed acyclic graph (DAG). For every word in the audio, the lattice contains multiple alternative hypotheses that were considered during the recognition process, including word matching probabilities and time boundaries.

In this DAG, each arc a represents a word hypothesis, while nodes n represent transition conditions (time and context) between word hypotheses.[1] Any route through the graph from the unique start node n_{start} to the unique end node n_{end} constitutes a hypothesis to explain the entire utterance, and its utterance-level hypothesis probability can be derived from the weights along the route.

We define a lattice as $\mathcal{L} = (\mathcal{N}, \mathcal{A}, n_{\text{start}}, n_{\text{end}})$. Each node $n \in \mathcal{N}$ has an associated time $t[n]$ and possibly an acoustic and/or language-model context condition. Arcs are 4-tuples $a = (S[a], E[a], W[a], P_{\text{arc}}[a])$. $S[a]$, $E[a] \in \mathcal{N}$ denote the start and end node of the arc. $W[a]$ is the word identity.

We use a unique variant of lattices that we call *posterior lattices*, where arc weights are *word posterior probabilities* $P_{\text{arc}}[a]$. The word posterior for arc a is the posterior probability of the hypothesis that word $W[a]$ occurred covering the time range $t[S[a]]...t[E[a]]$, in the context of arcs entering $S[a]$ and arcs leaving $E[a]$. $P_{\text{arc}}[a]$ is the aggregate posterior probability of all routes through the graph containing arc a.

3.5 Relationship of Word and Hit Posteriors

Hit posteriors $P(*, t_s, q, t_e, *|O)$ (Eq. (2)) can be easily computed from arc posteriors $P_{\text{arc}}[a]$. For *single-word queries* ($|q| = 1$), the hit posterior $P(*, t_s, q, t_e, *|O)$ is simply the arc-posterior sum over all arcs a that match q in a given time range, i.e. $q = W[a]$, $t_s = t[S[a]]$, and $t_e = t[E[a]]$:

$$P(*, t_s, q, t_e, *|O) \;=\; \sum_{\substack{WT:\exists a: t[S[a]] = t_s \wedge t[E[a]] = t_e \\ \wedge W[a] = q}} P_{\text{arc}}[a] \qquad (4)$$

This is slightly relaxed in that we merge all hypotheses that overlap into the one with the highest word posterior (see 3.3).

For multi-word queries $q = (q_1, ..., q_K)$, the hit posterior $P(*, t_s, q, t_e, *|O)$ can be computed as the sum over all connected routes $\pi = (a_1, ..., a_K)$ that begin at time t_s, end at t_e, and constitute the token sequence q, as follows:

$$P(*, t_s, q, t_e, *|O) \;=\; \sum_{\substack{\pi=(a_1,\cdots,a_K): \\ t[S[\pi]]=t_s \\ \wedge t[E[\pi]]=t_e \\ \wedge W[\pi]=q}} P_{\text{arc}}[a_1] \prod_{k=2}^{K} P_{\text{arc|node}}[a_k|S[a_k]].$$

with the "conditional" $P_{\text{arc|node}}[a_K|S[a_K]]$ defined as:

$$P_{\text{arc|node}}[a|S[a]] \;=\; \frac{P_{\text{arc}}[a]}{\sum_{a':S[a']=S[a]} P_{\text{arc}}[a']} \;=\; \frac{P_{\text{arc}}[a]}{P_{\text{node}}[S[a]]}$$

$P_{\text{node}}[n]$ is a convenience quantity that we call the *node posterior*. It does have a "physical meaning" in that it is the posterior for the hypothesis that the correct route passes through node n, and it can be computed both from incoming and outgoing arcs.

$$P_{\text{node}}[n] \;=\; \sum_{a:S[a]=n} P_{\text{arc}}[a] \;=\; \sum_{a:E[a]=n} P_{\text{arc}}[a] \qquad (5)$$

It can be precomputed and stored with the lattice nodes. In practical terms, however, we find that ignoring it (replacing it by a constant) does not negatively impact results.

3.6 Relationship to "Likelihood Lattices"

Posterior lattices are not directly generated by speech recognizers. Instead, recognizers output what we call "likelihood lattices" weighted by acoustic emission likelihoods (e.g. computed by Hidden Markov Models, HMM) and language-model probabilities [20]. Most speech-recognition packages can generate these.

[1] In alternative definitions of lattices found in literature, nodes represent words and arcs word transitions.

The primary advantage of posterior lattices is that posteriors are close to what we need, resilient to approximations like quantization and merging of similar alternates, and they allow comparing arcs with different time durations and temporal splitting e.g. compound words. Further, the node posteriors is uncritical and can be ignored.

For reference, we want to explain the relationship between the common likelihood lattice with our posterior lattices. The difference is that arc weights, instead of arc posteriors, are recognition scores, in particular log-linear combinations of an acoustic emission likelihood $p_{\text{ac}}(W[a]|S[a], E[a])$ computed with a Hidden Markov Model (HMM) and a language-model probability $P_{\text{LM}}(W[a]|S[a])$. Let us denote the arc weights of the Likelihood Lattice as $q_{\text{LL}}[a]$. Then,

$$q_{\text{LL}}[a] \quad = \quad p_{\text{ac}}(a)^{1/\lambda} \cdot P_{\text{LM}}(a)$$

with *language-model weight* λ.

With this definition, arc and node posteriors can be calculated with the well-known forward-backward recursion [21].

$$P_{\text{arc}}[a] \;=\; \frac{\alpha_{S[a]} \cdot q_{\text{LL}}[a] \cdot \beta_{E[a]}}{\alpha_{n_{\text{end}}}} \;\;;\;\; P_{\text{node}}[n] = \frac{\alpha_n \cdot \beta_n}{\alpha_{n_{\text{end}}}}, \qquad (6)$$

with the *forward probability* α_n defined as the sum over partial path recognition scores for all paths leading from the lattice start node to n, and *backward probability* β_n likewise as the sum over partial paths leading from n to the end of the lattice:

$$\alpha_n \;=\; \sum_{\substack{\pi: S[\pi]=n_{\text{start}} \\ \wedge E[\pi]=n}} \prod_{a\in\pi} q_{\text{LL}}[a] \;=\; \sum_{a:E[a]=n} q_{\text{LL}}[a] \cdot \alpha_{S[a]} \qquad (7)$$

$$\beta_n \;=\; \sum_{\substack{\pi: S[\pi]=n \\ E[\pi]=n_{\text{end}}}} \prod_{a\in\pi} q_{\text{LL}}[a] \;=\; \sum_{a:S[a]=n} q_{\text{LL}}[a] \cdot \beta_{E[a]} \qquad (8)$$

$\pi = (a_1, a_2, ...)$ denotes a connected sequence of arcs. Eq. (5) follows from Eqs. (6–8).

4. INDEXING WORD LATTICES WITH STANDARD FULL-TEXT INDEXERS

We have defined the spoken-document search task as retrieving all audio documents D that contain all query terms/phrases $q \in Q$, such that a certain Precision is met. We have further derived how, mathematically, this is achieved by by determining and thresholding against the posterior probability that an audio document contains all query terms/phrases from word posterior probabilities obtained from word lattices.

In this section, we now want to show how this can be approximately achieved in an efficient manner that by *reusing existing text indexers* as much as possible. When attempting to use text indexers to index word lattices, one encounters two blocking issues:

- *Word-position concept:* Text indexers locate word occurrences by word-position counts in the running text, for phrase matching and proximity-based relevance scoring. However, the concept of word positions is ill-defined to represent the temporal nature of word hypotheses, as word hypotheses are not generally word-aligned with each other. As seen in Fig. 1, word hypotheses may partially overlap, and even the number of words spanning a period of time is often not the same across hypotheses;
- *No posteriors (confidence scores):* For text indexers, there is no uncertainty whether a word occurred at a given position or not, so they do not store posteriors with word occurrences. At search time, the ranker considers only the "existence" of query words or phrases.

Besides, one encounters the practical problem that lattices can be really large, up to 100 times or more the size of a text transcript, beyond the range that most text indexers are designed and optimized for. Thus, *size reduction* will also be a desirable property of our method.

There is no exact solution for mapping word lattices onto a text index. We attempt to solve this conundrum by a series of transformations and approximations that transform the *numeric problem* of retrieving documents that match the query with a certain probability into a *symbolic text-based* one that can be implemented by a commercial full-text indexer.

In summary, the "TMI" method (Time-based Merging for Indexing) is first used to significantly reduce the lattice size by merging multiple similar word hypotheses that likely represent the same spoken word. After TMI, the graph becomes much more compact such that representing its connectivity in the word index would require just 2–3 additional bits of storage for each word hit. Yet, we cannot always assume that even such small modification can be done. To address this case, we then use the "TALE" method (Time-Anchored Lattice Expansion) to approximately map the complex time structure of the lattice onto word positions, while retaining phrase constraints as much as possible. Next, assuming that likewise we cannot extend the index to store the lattice's posterior probabilities directly, we quantize the probabilities and—this is where it becomes tricky—represent them *symbolically* by "decorating" the word tokens. Last, we show how the resulting index can be queried *symbolically* to return documents that contain all (possibly quoted) query terms with a given minimum probability, and provide an example runtime measurement using a commercial full-text engine.

4.1 Size Reduction: Time-based Merging for Indexing (TMI)

TMI—Time-based Merging for Indexing—is a technique for merging lattice arcs that likely represent the same spoken word [17]. A single word often generates many competing lattice arcs that differ in acoustic or language-model context conditions and exact time boundaries. Both effects can be seen in Fig. 1. In the problem at hand, hypotheses representing the same event are not competitors but should rather be considered jointly as a single hypothesis[2].

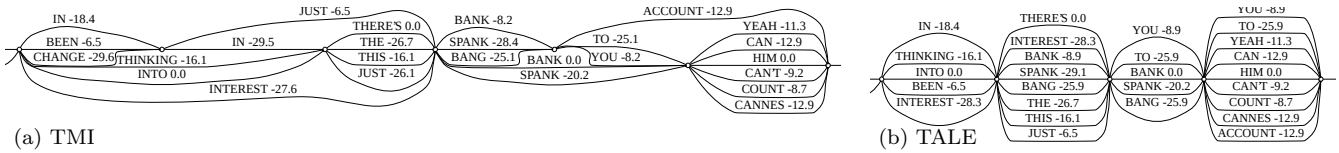
The TMI process is a heuristic approximation that aims at

- reducing redundancy/competitors that aren't (multiple hypotheses for the same query keyword with slightly dissimilar time boundaries);
- significantly reducing the total number of hypotheses to be indexed; and
- reducing storage per item,

while *retaining all word sequences* to avoid false negatives, but allowing a small amount of false positives.

The basic idea is to cluster lattice nodes with similar times, and to approximate word hypotheses by arcs between node clusters rather than individual nodes. The clustering criterion is a simple heuristic: two consecutive nodes can be clustered together *unless* that would create a loop (a hypothesis starting and ending in the same clustered node); and amongst the manifold of clusterings that satisfy this, the one leading

[2]For the case of phrase matches, context and precise time boundaries does matter in theory, but has little impact on actual accuracy.



(a) TMI (b) TALE

Figure 2: The lattice of Fig. 1 after TMI (a) and TALE (b) processing.

to the smallest number of clusters is considered the optimal solution. It can be found using dynamic programming:

- sort nodes $n_1 ... n_N$ in ascending time
- for each node n_i, determine m_i: the maximum node that n_i can be grouped with without causing a loop
- set cluster counts $C_0 \leftarrow 1$; $C_i \leftarrow \infty \, \forall \, i > 0$
- set backpointers $B_i \leftarrow n_i \, \forall \, i > 0$
- for $i = 1...N$: // DP recursion
  - for $j = i...m_i$: if $C_{i-1} + 1 \leq C_j$:
    - $C_j \leftarrow C_{i-1} + 1$
    - $B_j \leftarrow i$ // cluster $\{n_i ... n_j\}$
- $k \leftarrow N$; while $k \neq 0$: // trace back, merge nodes
  - create new node cluster $\{B_k ... n_k\}$, relink arcs
  - $k \leftarrow B_k - 1$
- merge arcs that connect the same two clusters with same word by summing up their posteriors

We call this "Time-based Merging for Indexing," as it effectively clusters nodes with similar time points, although node times are only used for node sorting. The process retains the *expected term frequency* $E_{WT|O}\{\mathrm{TF}_W(w)\}$ (with $\mathrm{TF}_W(w)$ being the word count of word w in the hypothesized transcript W) and keeps all phrases. It also introduces additional paths, but they are restricted to not cause insertions or deletions of full words. Our experiments show no loss of accuracy from false positives (these random combinations are unlikely to be valid phrases one would search for).

The final arc merging significantly reduces lattice size, into the range of operating characteristics that text engines are optimized for (further reduction is possible by pruning). Fig. 2(a) shows the lattice in Fig. 1 after TMI processing.

TMI also dramatically reduces the numeric range of arc spans ($E[a] - S[a]$) to a few bits only. E.g., a raw example lattice with 4502 nodes and 30806 arcs had a maximum span $\max_a\{E[a] - S[a]\} = 1303$, requiring 11 bits of storage (average: 371). TMI processing reduced it to 785 arcs and a maximum span of 8 (3 bits) with only 7% of the arcs above 4. Most text indexers provide a few bits per-word hit, to store attributes like whether a word occurred in a headline or was bold-faced in the text [22], which could be repurposed to store this information. Phrase-matching code would have to be modified to take this information into account.

4.2 Word-Position Mapping: Time-Anchored Lattice Expansion (TALE)

Not always can we assume that we can make even small modifications to an existing text-indexing engine. If we cannot even extend the index to store the few bits of arc-span information required to correctly match phrases, we will need a further approximation. The goal is to "get away" without storing the span, yet doing phrase matching as accurately as still possible. A method we call "Time-Anchored Lattice Expansion" (TALE) addresses this problem [19].

TALE approximates a lattice so that words are aligned to word positions, forming a "sausage"-like lattice. The standard phrase matcher requires words belonging to phrases to be in consecutive word positions. TALE addresses this by aligning some words to multiple slots (overgeneration). It is impossible to guarantee that all possible phrases are retained while keeping phrase posteriors and keeping the index small. We have to set priorities.

TALE aims to retain the expected term frequencies $E_{w|O}\{\mathrm{TF}_W(w)\}$; keep time points of individual hits accurate enough to allow playback; and have all phrases *up to three words* in consecutive word positions. The following method satisfies these criteria while keeping the index size reasonable.

First, we define the δ-conditional probability: probability that word w happens as the δ-th path token after a given node n:

$$P(w|n,\delta,O) \;=\; \frac{\sum\limits_{\substack{\forall n_i, w_i: \\ i=1\ldots\delta \wedge w_\delta = w}} P(*\text{-}n\text{-}w_1\text{-}n_1\text{-}\ldots\text{-}w\text{-}n_\delta\text{-}*|O)}{P(*\text{-}n\text{-}*|O)}$$

We then choose time anchor points $t_0 ... t_T$ to define word-position slots (t_i, t_{i+1}), e.g. the time boundaries of the best path, and align each node n to the closest slot, denoted by $i = s_n$. We call this "binning."

We can now compute the probability distribution for slot i of words w being the δ-th token of a phrase:

$$P_\delta(w|i,O) \;=\; \sum_{\forall n: s_n + \delta = i} P(*\text{-}n\text{-}*|O) \cdot P(w|n,\delta,O)$$

We call this the "Δ_δ-Expansion." Time information is retained by the anchor points.

To guarantee to retain all M-word phrases in consecutive slots, we interpolate multiple Δ_δ-Expansions:

$$P(w|i,O) \;=\; \sum_{\delta=1}^{M} \lambda_\delta \cdot P_\delta(w|i,O)$$

with $\sum \lambda_\delta = 1$. It can be proven that merging Δ_0, Δ_1 and Δ_2 retains all 3-word phrases. The weights λ_Δ would ideally be optimized on a development set to maximize overall accuracy, but it is not necessary: Experiments show that using equal weights yields almost as good result as full lattice. Fig. 2 (b) shows the lattice in Fig. 1 after TALE processing.

4.3 Storing Posteriors: Quantization and Symbolic Representation

The remaining problem is to store posteriors in the index. We found posteriors to be resilient to quantization, and 16 levels are sufficient, thus requiring 4 extra bits per hit. If storing this in the index is not possible, we must use a trick and "decorate" the word symbols themselves to encode the quantized posterior value. For example, a word "computer" with quantized posterior level 8 is stored in the index as the decorated token `spoken:computer@8`. (The prefix distinguishes spoken words from regular text.)

4.4 Symbolic Querying: Min/Max Approximation

With the above steps, we are now not only able to store lattices in the text index, but also to query for all documents matching the query approximately with a minimum probability $P_{\min}$—using *a fully symbolic query.*

We achieve that by—rather crudely—approximating the

products in Eq. (3) by minimum and maximum operations:

$$\begin{aligned} P(R(Q,D)|O) &= \prod_{q\in Q}(1-\prod_{(t_s,q,t_e)}(1-P(*,t_s,q,t_e,*|O))) \\ &\approx \min_{q\in Q}(1-\min_{(t_s,q,t_e)}(1-P(*,t_s,q,t_e,*|O))) \\ &= \min_{q\in Q}\max_{(t_s,q,t_e)} P(*,t_s,q,t_e,*|O) \end{aligned}$$

Experiments show only little degradation of accuracy from this. Then, cutting off document relevance probabilities at given threshold $P_{\min}$ can be expressed as:

$$\begin{aligned} &P(R(Q,D)|O) > P_{\min} \\ &\quad \Leftrightarrow \quad \forall q \in Q, \exists (t_s,q,t_e), P(*,t_s,q,t_e,*|O) > P_{\min} \end{aligned}$$

It is easy to see that this can be implemented as a combination of symbolic AND and OR clauses.

4.5 System Architecture

Fig. 3 shows the suggested system architecture. Raw lattices from the recognizer are TMI-compacted, pruned, saved in a forward index, and fed to the text indexer, which ingests it through a custom ingestion plug-in. Text indexers commonly expose a plug-in interface to ingest proprietary binary file formats such as PDF. Our custom plug-in applies TALE processing to bin lattices into positions, quantizes the word posteriors, and outputs the decorated words to the index.

At query time, the user-requested minimum precision $P_{\min}$ is mapped to a confidence threshold and encoded into the user query by the query construction module. For efficiency, we encode only the lowest allowed confidence level in the query itself, and use a custom FORMSOF expansion plug-in to expand it to include all higher levels as well. FORMSOF expansion is intended to implement stemming, but it is commonly expandable through plug-ins for language support. FORMSOF expansion is more efficient than explicit expansion with OR operators, especially for phrases.

The documents returned by the full-text indexer are rendered by the user interface module. An important UI feature are transcription snippets surrounding keyword hits. These are generated from the TMI lattices stored in the forward index. Associated time information can be used for navigation.

5. RESULTS

5.1 Setup

We have evaluated our method on the 170-hour MIT iCampus lecture set [3] with 160 lectures. The data set came presegmented into 65927 sentences, with an average sentence about 10 seconds. A speaker-independent Large Vocabulary

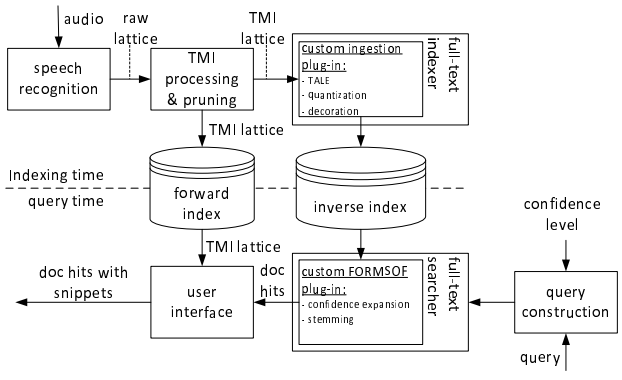


Figure 3: System Architecture.

Table 1: Spoken-document search results for phrase, single-word, and two-term AND queries. Shown are mean average precision (mAP) and Recall at a certain Precision cutoff (R_P) for P=75% and 50%. "Index size" is index entries per spoken word. "Relative improvement" is from "STT transcript with confidence" to the last setup. All numbers in percent.

query type:	phrase		single-word		x AND y		index
configuration	mAP	R_{75}	mAP	R_{50}	mAP	R_{75}	size
STT transcript	42.6	43.4	44.3	45.2	26.1	26.1	1.0
raw lattice	67.2	52.9	56.3	46.1	63.3	61.6	1617
+TMI	69.6	55.0	56.3	46.0	66.2	62.9	46.2
+pruning	67.1	55.6	55.6	46.0	60.4	60.9	9.9
+TALE	67.6	55.5	55.3	46.2	61.5	61.1	11.5
+min approx.	67.3	54.1	55.2	46.1	61.2	58.4	11.5
+quantization	66.2	54.2	52.1	46.1	60.9	57.4	11.5
rel. impr.	**+55%**	**+25%**	**+18%**	**+2%**	**×2.3**	**×2.2**	-

Continuous Speech Recognition (LVCSR) system was used to generate word lattices. The acoustic model was trained on the 1700-hour Switchboard "Fisher" telephone-speech set [2]. Due to limited LM data for lectures, we partitioned the test set into 10 parts, and recognized each part with an LM trained on the transcripts of the remaining 9 parts, keeping training and test sets disjunct. The Word Error Rate (WER) for the test set is 46.6%.

We evaluate our method with multi-word (phrase), single-word, and two-term AND queries (x AND y where x and y can be single terms or phrases). The keyword set is synthetic and consists of noun phrases chosen from the transcripts such that for each query there are at most two matching lectures.

5.2 Spoken-Document Search Experiments

In this section, we investigate the quality of the document-level posteriors through ranking experiments, and the effect of our various approximations. We use two accuracy metrics:

- mAP: mean average precision, where documents are ranked by their document-level matching posterior as defined in Eq. (3);
- R_{75}/R_{50}: R_P is the document Recall at Precision P (after the fact), with P=75% for most query sets, and P=50% for single words as there were insufficient data points at P =75%.

Table 1 shows the results. The first and second result rows compare accuracies for the "naïve" approach, indexing speech-to-text (STT) plain-text transcripts, with raw lattice indexing. To be able to compare transcript results with lattices, we attached posteriors from the lattice to each transcript word. Significant improvements are observed from indexing only STT transcripts to indexing lattices.

The next row shows the result for TMI processing. TMI reduces the lattice size over 30 times, from around 1600 arcs per spoken word to 46. We admittedly use *very* rich lattices here – in earlier experiments [17] with smaller lattices of about 25 arcs per word, TMI achieved about a 5-fold size reduction. Compared with the "raw lattice," TMI does not lead to an accuracy loss–in fact it improves accuracy by 2–3 points for multi-word queries: By creating additional paths, TMI has recovered a few phrases. For single-word queries, TMI is very close to the raw lattice as expected.

TMI processing merges arcs with same words and similar time range together, so it is more accurate for pruning arcs with low posteriors. In the next row, we prune the lattice by removing all arcs with logarithmic posteriors below -8.0. The result (in the next row "+ pruning") shows a 2–6 points accuracy loss for multi-word queries and AND queries for mAP (while R_P is almost not affected). I.e., unlike false positives from creating new paths, false negatives are expensive. The loss is much smaller for the high-precision R_{75} metric.

Table 2: Recall (R) and precision (P) in percent when using a cutoff threshold targeting precision 50% for single-word queries and 75% for phrase and AND queries. STT transcripts are not cut off. The last row is with all the approximation we have made (TMI+pruning+TALE+MinMax approximation+quantization).

query type:	phrase		single word		x AND y	
setup	P	R	P	R	P	R
STT transcript	80.9	43.4	41.7	48.6	97.9	26.1
raw lattice	75.6	54.7	50.4	46.1	75.0	61.0
all approximations	75.9	55.9	49.5	45.8	75.0	58.1

The next row shows the further TALE processing result. The is a small size increase as TALE over-generates arcs, while the accuracy is almost not affected.

The next rows show results for the Min/Max approximation and posterior quantization. Posteriors are quantized to 15 levels. We observe largest accuracy loss for single-word queries (3 points).

Overall, except the R_{50} for single-word queries, the proposed setups significantly outperform "naïve" indexing of speech-to-text transcripts, with the best improvement on AND queries from 26% to 62%.

5.3 Cutoff Experiments for Spoken Documents Search

In this setup, we want to evaluate the accuracy for the primary task of this paper—retrieving documents that contain all query terms with a probability greater than a given threshold. We consider 75% a realistic choice for the target Precision—with anything less, users may consider the system broken! For single-word queries, we had to reduce it to 50%.

The cutoff threshold should be tuned on a development set, but in our case we split our test set in two and applied a threshold measured on one half to the other half.

Table 2 compares three setups: STT transcripts, raw lattices, and the proposed method with all approximations. For phrase queries and AND queries, STT transcript suffers from a poor Recall despite high Precision. Raw lattices achieve significantly better Recall through using alternates, while the precision is within the target range. For single-word queries, STT transcript without confidence cannot achieve the targeted Precision level of 50%, while using lattices achieves this while trading Recall. Again, the loss caused by the proposed approximations is less than 3 points.

5.4 Runtime Performance

As a feasibility test, the method in this paper has been implemented in Microsoft SQL Server 2005. The system architecture follows Fig. 3. For example, searching for the four-word query *"barack obama" "hillary clinton"* in an audio database containing 5000 hours (different from our experimental database used above) indexed as described in section 4 with minimum quantized probability level of 10 takes 290 ms using the following symbolic query:

```
SELECT fileid FROM files
    WHERE CONTAINS (aib, 'FORMSOF (INFLECTIONAL,
        "spoken:barack@10 spoken:obama@10")
    AND FORMSOF (INFLECTIONAL,
        "spoken:hillary@10 spoken:clinton@10")')
```

6. CONCLUSION

We have examined the problem of indexing the spoken content of audio recordings. Unlike indexing text, today's speech recognition technology does not allow to reliably know whether a word is present or not at a given point in the audio; one can only obtain a probability for it. Simply indexing the speech-to-text transcripts results in poor accuracy.

In this paper, we have significantly improved search accuracy for "web-search style" (AND/phrase) queries by making correct use of speech-recognition probabilities—in particular by utilizing recognition alternates and word posterior probabilities (confidence scores) based on word lattices.

We have further presented an end-to-end approach to doing so with *standard full-text indexers*, despite the fact that by design text indexers cannot handle probabilities and unaligned alternates. We presented a sequence of approximations that transform the numeric lattice-matching problem into a symbolic text-based one that can be implemented by a commercial full-text indexer (Microsoft SQL Server 2005).

Experiments on a 170-hour lecture set have shown a relative accuracy improvement of 30-130% compared to indexing linear text.

7. REFERENCES

[1] M. Padmanabhan, G. Saon, J. Huang, B. Kingsbury, and L. Mangu, Automatic Speech Recognition Performance on a Voicemail Transcription Task. *IEEE Trans. on Speech and Audio Processing, Vol. 10, No. 7, 2002.*

[2] G. Evermann, H. Y. Chan, M. J. F. Gales, B. Jia, X. Liu, D. Mrva, K. C. Sim, L. Wang, P. C. Woodland, K. Yu, Development of the 2004 CU-HTK English CTS Systems Using More Than Two Thousand Hours of Data. *Proc. Fall 2004 Rich Transcription Workshop (RT-04), 2004.*

[3] J. Glass, T. J. Hazen, L. Hetherington, C. Wang, Analysis and Processing of Lecture Audio data: Preliminary investigation. *Proc. HLT-NAACL'2004 Workshop: Interdisciplinary Approaches to Speech Indexing and Retrieval*, Boston, 2004.

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

[5] T. Kawahara, C. H. Lee, B. H. Juang, Combining key-phrase detection and subword-based verification for flexible speech understanding, *Proc. ICASSP'1997*, Munich,1997.

[6] P. Yu, K. J. Chen, C. Y. Ma, F. Seide, Vocabulary-Independent Indexing of Spontaneous Speech, IEEE Transactions on Speech and Audio Processing, Vol.13, No.5.

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

[8] P. Schäuble et al. First experiences with a system for content based rtrieval of information from speech recordings. *Proc. IJCAI'95.*

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

[10] D. A. James and S. J. Young, A fast lattice-based approach to vocabulary-independent wordspotting. *Proc. ICASSP'04*, 2004.

[11] F. Seide, P. Yu, *et al*, Vocabulary-Independent Search in Spontaneous Speech. *Proc. ICASSP'04*, Montreal, 2004.

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

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

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

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

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

[17] Z. Y. Zhou, P. Yu, C. Chelba, F. Seide, Towards Spoken-Document Retrieval for the Internet: Lattice Indexing For Large-Scale Web-Search Architectures. *Proc. HLT'06*, 2006.

[18] L. Mangu, E. Brill, A. Stolcke, Finding Consensus in Speech Recognition: Word Error Minimization and Other Applications of Confusion Networks. *Computer, Speech and Language*, 14(4).

[19] F. Seide, P. Yu, Y. Shi, Towards Spoken-Document Retrieval for the Enterprise: Approximate Word-Lattice Indexing with Text Indexers. *Proc. ASRU'2007*, Kyoto, 2007.

[20] D. Jurafsky and J. Martin, An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Second Edition, Pearson Prentice Hall, 2008.

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

[22] S. Brin and L. Page, The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30.