Towards Spoken-Document Retrieval for the Internet: Lattice Indexing For Large-Scale Web-Search Architectures

Zheng-Yu Zhou[*], **Peng Yu, Ciprian Chelba**[+], and **Frank Seide**
[*]Chinese University of Hong Kong, Shatin, Hong Kong
Microsoft Research Asia, 5F Beijing Sigma Center, 49 Zhichun Road, 100080 Beijing
[+]Microsoft Research, One Microsoft Way, Redmond WA 98052
zyzhou@se.cuhk.edu.hk, {rogeryu,chelba,fseide}@microsoft.com

Abstract

Large-scale web-search engines are generally designed for linear text. The linear text representation is suboptimal for audio search, where accuracy can be significantly improved if the search includes alternate recognition candidates, commonly represented as word lattices.

This paper proposes a method for indexing word lattices that is suitable for large-scale web-search engines, requiring only limited code changes.

The proposed method, called Time-based Merging for Indexing (TMI), first converts the word lattice to a posterior-probability representation and then merges word hypotheses with similar time boundaries to reduce the index size. Four alternative approximations are presented, which differ in index size and the strictness of the phrase-matching constraints.

Results are presented for three types of typical web audio content, podcasts, video clips, and online lectures, for phrase spotting and relevance ranking. Using TMI indexes that are only five times larger than corresponding linear-text indexes, phrase spotting was improved over searching top-1 transcripts by 25-35%, and relevance ranking by 14%, at only a small loss compared to unindexed lattice search.

1 Introduction

Search engines have become the essential tool for finding and accessing information on the Internet. The recent runaway success of podcasting has created a need for similar search capabilities to find audio on the web. As more news video clips and even TV shows are offered for on-demand viewing, and educational institutions like MIT making lectures available online, a need for audio search arises as well, because the most informative part of many videos is its dialogue.

There is still a significant gap between current web audio/video search engines and the relatively mature text search engines, as most of today's audio/video search engines rely on the surrounding text and metadata of an audio or video file, while ignoring the actual audio content. This paper is concerned with technologies for searching the audio content itself, in particular how to represent the speech content in the index.

Several approaches have been reported in the literature for 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, unpredictable, highly variable acoustic conditions, non-native and accented speaker, informal talking style, and unlimited-domain language cause word-error rates to be much higher (40-60%). Directly searching such inaccurate speech recognition transcripts suffers from a poor recall.

A successful way for dealing with high word error rates is the use of recognition alternates (lattices) (Saraclar, 2004; Yu, 2004; Chelba, 2005). For example, (Yu, 2004) reports a 50% improvement of FOM (Figure Of Merit) for a word-spotting task in voice-mails, and (Yu, HLT2005) adopted the approach for searching personal audio collections, using a hybrid word/phoneme lattice search.

Web-search engines are complex systems involving substantial investments. For extending web search to audio search, the key problem is to find a (approximate)

representation of lattices that can be implemented in a state-of-the-art web-search engine with as little changes as possible to code and index store and without affecting its general architecture and operating characteristics.

Prior work includes (Saraclar, 2004), which proposed a direct inversion of raw lattices from the speech recognizer. No information is lost, and accuracy is the same as for directly searching the lattice. However, raw lattices contain a large number of similar entries for the same spoken word, conditioned on language-model (LM) state and phonetic cross-word context, leading to inefficient usage of storage space.

(Chelba, 2005) 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 easily integrates with text search engines, as the resulting index resembles a normal text index in most aspects. However, it trades redundancy w.r.t. LM state and context for uncertainty w.r.t. word position, and only achieves a small reduction of index entries. Also, time information for individual hypotheses is lost, which we consider important for navigation and previewing.

(Mangu, 2000) presented a method to align a speech lattice with its top-1 transcription, creating so-called "confusion networks" or "sausages." Sausages are a parsimonious approximation of lattices, but due to the presence of null links, they do not lend themselves naturally for matching phrases. Nevertheless, the method was a key inspiration for the present paper.

This paper is organized as follows. The next section states the requirements for our indexing method and describes the overall system architecture. Section 3 introduces our method, and Section 4 the results. Section 5 briefly describes a real prototype built using the approach.

2 Indexing Speech Lattices, Internet Scale

Substantial investments are necessary to create and operate a web search engine, in software development and optimization, infrastructure, as well as operation and maintainance processes. This poses constraints on what can practically be done when integrating speech-indexing capabilities to such an engine.

2.1 Requirements

We have identified the following special requirements for speech indexing:

- realize best possible accuracy – speech alternates must be indexed, with scores;
- provide time information for individual hits – to facilitate easy audio preview and navigation in the UI;
- encode necessary information for phrase matching – phrase matching is a basic function of a search engine and an important feature for document ranking. This is non-trivial because boundaries of recognition alternates are generally not aligned.

None of these capabilities are provided by text search engines. To add these capabilities to an existing web engine, we are facing practical constraints. First, the structure of the index store cannot be changed fundamentally. But we can reinterpret existing fields. We also assume that the index attaches a few auxiliary bits to each word hit. E.g., this is done in (early) Google (Brin, 1998) and MSN Search. These can be used for additional data that needs to be stored.

Secondly, computation and disk access should remain of similar order of magnitude as for text search. Extra CPU cycles for phrase-matching loops are possible as long as disk access remains the dominating factor. The index size cannot be excessively larger than for indexing text. This precludes direct inversion of lattices (and unfortunately also the use of phonetic lattices).

Last, while local code changes are possible, the overall architecture and dataflow cannot be changed. E.g., this forbids the use of a two-stage method as in (Yu, HLT2005).

2.2 Approach

We take a three-step approach. First, following (Chelba, 2005), we use a posterior-probability representation, as posteriors are resilient to approximations and can be quantized with only a few bits. Second, we reduce the inherent redundancy of speech lattices by merging word hypotheses with same word identity and similar time boundaries, hence the name "Time-based Merging for Indexing" (TMI). Third, the resulting hypothesis set is represented in the index by reinterpreting existing data fields and repurposing auxiliary bits.

2.3 System Architecture

Fig. 1 shows the overall architecture of a search engine for audio/video search. At indexing time, a media decoder first extracts the raw audio data from different formats of audio found on the Internet. A music detector prevents music from being indexed. The speech is then fed into a large-vocabulary continuous-speech recognizer (LVCSR), which outputs word lattices. The lattice indexer converts the lattices into the TMI representation, which is then merged into the inverted index. Available textual metadata is also indexed.

At search time, all query terms are looked up in the index. For each document containing all query terms (determined by intersection), individual hit lists of each query term are retrieved and fed into a phrase matcher to identify full and partial phrase hits. Using this information, the ranker computes relevance scores. To achieve acceptable response times, a full-scale web engine would split this process up for parallel execution on multiple servers. Finally the result presentation module will create snippets

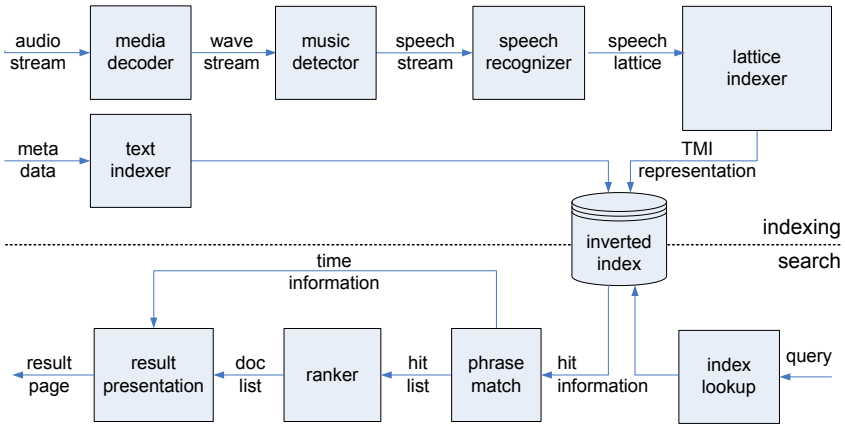

Figure 1: System Architecture.

for the returned documents and compose the result page. In audio search, snippets would contain time information for individual word hits to allow easy navigation and preview.

3 Time-based Merging for Indexing

Our previous work (Yu, IEEE2005) has shown that in a word spotting task, ranking by phrase posteriors is in theory optimal if (1) a search hit is considered relevant if the query phrase was indeed said there, and (2) the user expects a ranked list of results such that the accumulative relevance of the top-n entries of the list, averaged over a range of n, is maximized. In the following, we will first recapitulate the lattice notation and how phrase posteriors are calculated from the lattice. We then introduce time-based merging, which leads to an approximate representation of the original lattice. We will describe two strategies of merging, one by directly clustering word hypotheses (arc-based merging) and one by grouping lattice nodes (node-based merging).

3.1 Posterior Lattice Representation

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_{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.[1]

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 $a = (S[a], E[a], I[a], w[a])$. $S[a]$, $E[a] \in \mathcal{N}$ denote the start and end node of the arc. $I[a]$ is the word identity. Last, $w[a]$ shall be a weight assigned to the arc by the recognizer. Specifically, $w[a] = p_{\text{ac}}(a)^{1/\lambda} \cdot P_{\text{LM}}(a)$ with acoustic likelihood $p_{\text{ac}}(a)$, LM probability P_{LM}, and LM weight λ.

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

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 $w[\pi] = \prod_{k=1}^{K} w[a_k]$.

Based on this, we define *arc posteriors* $P_{\text{arc}}[a]$ and *node posteriors* $P_{\text{node}}[n]$ as

$$P_{\text{arc}}[a] = \frac{\alpha_{S[a]} \cdot w[a] \cdot \beta_{E[a]}}{\alpha_{n_{\text{end}}}} \quad ; \quad P_{\text{node}}[n] = \frac{\alpha_n \cdot \beta_n}{\alpha_{n_{\text{end}}}},$$

with *forward-backward probabilities* α_n, β_n defined as:

$$\alpha_n = \sum_{\pi: S[\pi]=n_{\text{start}} \wedge E[\pi]=n} w[\pi] \quad ; \quad \beta_n = \sum_{\pi: S[\pi]=n \wedge E[\pi]=n_{\text{end}}} w[\pi]$$

α_n and β_n can be conveniently computed using the well-known forward-backward recursion, e.g. (Wessel, 2000).

With this, an alternative equivalent representation is possible by using word posteriors as arc weights. The *posterior lattice* representation stores four fields with each edge: $S[a]$, $E[a]$, $I[a]$, and $P_{\text{arc}}[a]$, and two fields with each node: $t[n]$, and $P_{\text{node}}[a]$.

With the posterior lattice representation, the phrase posterior of query string Q is computed as

$$P(*, t_s, Q, t_e, *|O) = \sum_{\substack{\pi=(a_1,\cdots,a_K): \\ t[S[\pi]]=t_s \wedge t[E[\pi]]=t_e \wedge I[\pi]=Q}} \frac{P_{\text{arc}}[a_1] \cdots P_{\text{arc}}[a_K]}{P_{\text{node}}[S[a_2]] \cdots P_{\text{node}}[S[a_K]]}. \quad (1)$$

This posterior representation is lossless. Its advantage is that posteriors are much more resiliant to approximations than acoustic likelihoods. This paves the way for lossy approximations aiming at reducing lattice size.

3.2 Time-based Merging for Indexing

First, (Yu, HLT2005) has shown that node posteriors can be replaced by a constant, with no negative effect on

search accuracy. This approximation simplifies the denominator in Eq. 1 to p_{node}^{K-1}.

We now merge all nodes associated with the same time points. As a result, the connection condition for two arcs depends only on the boundary time point. This operation gave the name Time-based Merging for Indexing.

TMI stores arcs with start and end time, while discarding the original node information that encoded dependency on LM state and phonetic context. This form is used, e.g., by (Wessel, 2000). Lattices are viewed as sets of *items* $h = (ts[h], dur[h], I[h], P[h])$, with $ts[h]$ being the start time, $dur[h]$ the time duration, $I[h]$ the word identity, and $P[h]$ the posterior probability. Arcs with same word identity and time boundaries but different start/end nodes are merged together, their posteriors being summed up.

These item sets can be organized in an inverted index, similar to a text index, for efficient search. A text search engine stores at least two fields with each word hit: word position and document identity. For TMI, two more fields need to be stored: duration and posterior. Start times can be stored by repurposing the word-position information. Posterior and duration go into auxiliary bits. If the index has the ability to store side information for documents, bits can be saved in the main index by recording all time points in a look-up table, and storing start times and durations as table indices instead of absolute times. This works because the actual time values are only needed for result presentation. Note that the TMI index is really an extension of a linear-text index, and the same code base can easily accomodate indexing both speech content and textual metadata.

With this, multi-word phrase matches are defined as a sequence of items $h_1 ... h_K$ matching the query string ($Q = (I[h_1], \cdots, I[h_K])$) with matching boundaries ($ts[h_i] + dur[h_i] = ts[h_{i+1}]$). The phrase posterior is calculated (using the approximate denominator) as

$$P(*, t_s, Q, t_e, *|O) \approx \sum \frac{P[h_1] \cdots P[h_K]}{p_{\text{node}}^{K-1}}, \quad (2)$$

summing over all item sequences with $t_s = ts[h_1]$ and $t_e = ts[h_K] + dur[h_K]$.

Regular text search engines can not directly support this, but the code modification and additional CPU cost is small. The major factor is disk access, which is still linear with the index size.

We call this index representation "TMI-base." It provides a substantial reduction of number of index entries compared to the original lattices. However, it is obviously an approximative representation. In particular, there are now conditions under which two word hypotheses can be matched as part of a phrase that were not connected in the original lattice. This approximation seems sensible, though, as the words involved are still required to have precisely matching word boundaries. In fact it has been shown that this representation can be used for direct word-error minimization during decoding (Wessel, 2000).

Table 1: Test corpus summary.

test set	duration	#segments	#keywords (#multi-word)	WER [%]
podcasts	1.5h	367	3223 (1709)	45.8
videos	1.3h	341	2611 (1308)	50.8
lectures	169.6h	66102	96 (74)	54.8

For further reduction of the index size, we are now relaxing the merging condition. The next two sections will introduce two alternate ways of merging.

3.3 Arc-Based Merging

A straightforward way is to allow tolerance of time boundaries. Practically, this is done by the following bottom-up clustering procedure:

- collect arcs with same word identity;
- find the arc a^* with the best posterior, set the resulting item time boundary same as a^*;
- merge all overlapping arcs a satisfying $t[S[a^*]] - \triangle_1 \leq t[S[a]] \leq t[S[a^*]] + \triangle_1$ and $t[E[a^*]] - \triangle_1 \leq t[E[a]] \leq t[E[a^*]] + \triangle_1$;
- repeat with remaining arcs.

We call this method "TMI-arc" to denote its origin from direct clustering of arcs.

Note that the resulting structure can generally not be directly represented as a lattice anymore, as formally connected hypotheses now may have slightly mismatching time boundaries. To compensate for this, the item connection condition in phrase matching needs to be relaxed as well: $ts[h_{i+1}] - \triangle_1 \leq ts[h_i] + dur[h_i] \leq ts[h_{i+1}] + \triangle_1$.

The storage cost for each TMI-arc item is same as for TMI-base, while the *number* of items will be reduced.

3.4 Node-Based Merging

An alternative way is to group ranges of time points, and then merge hypotheses whose time boundaries got grouped together.

The simplest possibility is to quantize time points into fixed intervals, such as 250 ms. Hypotheses are merged if their quantized time boundaries are identical. This method we call "TMI-timequant."

Besides reducing index size by allowing more item merging, TMI-timequant has another important property: since start times and duration are heavily quantized, the number of bits used for storing the information with the items in the index can be significantly reduced.

The disadvantage of this method is that loops are frequently being generated this way (quantized duration of 0), providing sub-optimal phrase matching constraints.

To alleviate for this problem, we modify the merging by forbidding loops to be created: Two time points can be

Table 2: Lattice search accuracy on different dataset.

setup	best path			raw lattice		
keywords	all	sing.	mult.	all	sing.	mult.
Phrase spotting, FOM[%]						
podcasts	55.0	59.9	50.1	69.5	74.7	64.2
videos	47.0	50.6	43.0	64.4	67.4	61.1
lectures	65.5	69.5	47.1	77.0	80.8	58.8
Relevance ranking, mAP[%]						
lectures	52.6	52.7	52.6	61.6	66.4	60.2

grouped together if (1) their difference is below a threshold (like 250 ms); and (2) if there is no word hypothesis starting and ending in the same group. As a refinement, the second point is relaxed by a pruning threshold in that hypotheses with posteriors below the threshold will not block nodes merging.

Amongst the manifold of groupings that satisfy these two conditions, the one leading to the smallest number of groups is considered the optimal solution. It can be found using dynamic programming:

- line up all existing time boundaries in ascending order, $t_i < t_{i+1}, i = 1, \cdots, N$;
- for each time point t_i, find out the furthest time point that it can be grouped with given the constraints, denoting its index as $T[t_i]$;
- set group count $C[t_0] = 1$; $C[t_i] = \infty$, $i > 0$;
- set backpointer $B[t_0] = -1$; $B[t_i] = t_i$, $i > 0$;
- for $i = 1, \cdots, N$:
  - for $j = i+1, \cdots, T[t_i]$: if $C[t_{j+1}] > C[t_i]+1$:
    - $C[t_{j+1}] = C[t_i] + 1$;
    - $B[t_{j+1}] = t_i$;
- trace back and merge nodes:
  - set $k = N$, repeat until $k = -1$:
    - group time points from $B[t_k]$ to t_{k-1};
    - $k = B[t_k]$.

This method can be applied to the TMI-base representation, or alternatively directly to the posterior lattice. In this case, the above algorithm needs to be adapted to operate on nodes rather than time points. The above method is called "TMI-node."

If, as mentioned before, times and durations are stored as indexes into a look-up table, TMI-node is highly space efficient. In most cases, the index difference between end and start point is 1, and in practical terms, the index difference can be capped by a small number below 10.

4 Results

4.1 Setup

We have evaluated our system on three different corpora, in an attempt to represent popular types of audio currently found on the Internet:

- podcasts: short clips ranging from mainstream media like ABC and CNN to non-professionally produced edge content;
- video clips, acquired from MSN Video;
- online lectures: a subset of the MIT iCampus lecture collection (Glass, 2004).

In relation to our goal of web-scale indexing, the podcast and video sets are miniscule in size (about 1.5 hours each). Nevertheless they are suitable for investigating the effectiveness of the TMI method w.r.t. phrase spotting accuracy. Experiments on relevance ranking were conducted only on the much larger lecture set (170 hours).

For the iCampus lecture corpus, the same set of queries was used as in (Chelba, 2005), which was collected from a group of users. Example keywords are *computer science* and *context free grammar*. On the other two sets, an automatic procedure described in (Seide, 2004) was used to select keywords. Example keywords are *playoffs, beach Florida*, and *American Express financial services.*

A standard speaker-independent trigram LVCSR system was used to generate raw speech lattices. For video and podcasts, models were trained on a combination of telephone conversations (Switchboard), broadcast news, and meetings, downsampled to 8 kHz, to accomodate for a wide range of audio types and speaking styles. For lectures, an older setup was used, based on a dictation engine without adaptation to the lecture task. Due to the larger corpus size, lattices for lectures were pruned much more sharply. Word error rates (WER) and corpus setups are listed in Table 1. It should be noted that the word-error rates vary greatly within the podcast and video corpora, ranging from 30% (clean broadcast news) to over 80% (accented reverberated speech with a cheering crowd).

Each indexing method is evaluated by a phrase spotting task and a document retrieval task.

4.1.1 Phrase Spotting

We use the "Figure Of Merit" (FOM) metric defined by NIST for word-spotting evaluations. In its original form, FOM is the detection/false-alarm curve averaged over the range of [0..10] false alarms per hour per keyword. We generalized this metric to spotting of phrases, which can be multi-word or single-word. A multi-word phrase is matched if all of its words match in order.

Since automatic word alignment can be troublesome for long audio files in the presence of errors in the reference transcript, we reduced the time resolution of the FOM metric and used the sentence as the basic time unit. A phrase hit is considered correct if an actual occurence of the phrase is found in the same sentence. Multiple hits of the same phrase within one sentence are counted as a single hit, their posterior probabilities being summed up for ranking.

The segmentation of the audio files is based on the reference transcript. Segments are on average about 10 seconds long. In a real system, sentence boundaries are of course unknown, but previous experiments have shown

Table 3: Comparison of different indexing methods. Only results for multi-words queries are shown, because results for single-word queries are identical across lattice-indexing methods (approximately identical in the case of pruning.)

dataset	podcasts		videos		lectures		
	FOM [%]	size	FOM [%]	size	FOM [%]	mAP [%]	size
bestpath	50.1	1.1	43.0	1.0	47.1	52.6	1.0
raw lattice	64.2	527.6	61.1	881.7	58.8	60.2	23.3
$P_{\text{node}} = \text{const}$	64.3	527.6	61.1	881.7	58.8	60.3	23.3
no pruning							
TMI-base	65.3	55.2	62.6	78.8	58.8	60.2	7.7
TMI-arc	62.9	16.1	58.5	20.7	57.9	60.1	4.4
TMI-timequant	66.7	15.4	64.2	19.5	58.8	60.3	4.5
TMI-node	66.5	20.7	63.4	27.6	58.7	59.7	4.4
PSPL	68.9	182.0	66.2	212.0	58.7	61.0	21.2
pruned to about 5 entries per spoken word							
TMI-base	62.1	5.6	54.1	5.1	57.0	60.3	4.5
TMI-arc	60.7	4.6	53.6	5.0	57.9	60.1	4.4
TMI-timequant	63.1	4.7	57.1	5.1	58.8	60.3	4.5
TMI-node	63.7	4.6	57.7	5.1	58.7	59.7	4.4
PSPL	57.3	6.0	49.8	5.8	53.6	61.0	4.4

that the actual segmentation does not have significant impact on the results.

4.1.2 Relevance Ranking

The choice and optimization of a relevance ranking formula is a difficult problem that is beyond the scope of this paper. We chose a simple document ranking method as described in (Chelba, 2005):

Given query $Q = (q_1, \cdots, q_L)$, for each document D, expected term frequencies (ETF) of all sub-strings $Q_{[i,j]} = (q_i, \cdots, q_j)$ are calculated:

$$\text{ETF}(Q_{[i,j]}|D) = \sum_{t_s,t_e} P(*, t_s, Q_{[i,j]}, t_e, *|O, D) \quad (3)$$

A document is returned if all query words are present. The relevance score is calculated as

$$S(D, Q) = \sum_{i=1}^{L} \sum_{j=i}^{L} w_{j-i} \log[1 + \text{ETF}(Q_{[i,j]}|D)] \quad (4)$$

where the weights w_ℓ have the purpose to give higher weight to longer sub-strings. They were chosen as $w_\ell = 1 + 1000 \cdot \ell$, no further optimization was performed.

Only the lecture set is used for document retrieval evaluation. The whole set consists of 169 documents, with an average of 391 segments in each document. The evaluation metric is the mean average precision (mAP) as computed by the standard `trec_eval` package used by the TREC evaluations (NIST, 2005). Since actual relevance judgements were not available for this corpus, we use the output of a state-of-the-art text retrieval engine on the ground truth transcripts as the reference. The idea is that if human judgements are not available, the next best thing to do is to assess how close our spoken-document retrieval system gets to a text engine applied to reference transcripts. Although one should take the absolute mAP scores with a pinch of salt, we believe that comparing the relative changes of these mAP scores is meaningful.

4.2 Lattice Search and Best Path Baseline

Table 2 lists the word spotting and document retrieval result of direct search in the original raw lattice, as well as for searching the top-1 path. Results are listed separately for single- and multi-word queries. For the phrase-spotting task, a consistent about 15% improvement is observed on all sets, re-emphasizing the importance of searching alternates. For document retrieval, the accuracy (mAP) is also significantly improved from 53% to 62%.

4.2.1 Comparing Indexing Methods

Table 3 compares different indexing methods with respect to search accuracy and index size. We only show results for multi-words queries results, as it can be shown that results for single-word queries must be identical. The index size is measured as index entries per spoken word, i.e. it does not reflect that different indexing methods may require different numbers of bits in the actual index store.

In addition to four types of TMI methods, we include an alternative posterior-lattice indexing method in our comparison called PSPL (position-specific posterior lattices) (Chelba, 2005). A PSPL index is constructed by enumerating all paths through a lattice, representing each path as a linear text, and adding each text to the index, each time starting over from word position 1. Each word hypothesis on each path is assigned the posterior probability of the entire path. Instances of the same word occuring at the same text position are merged, accumulating their posterior probabilities. This way, each index entry represents the posterior probability that a word occurs at a particular position in the actual spoken word sequence. PSPL is an attractive alternative to the work presented in

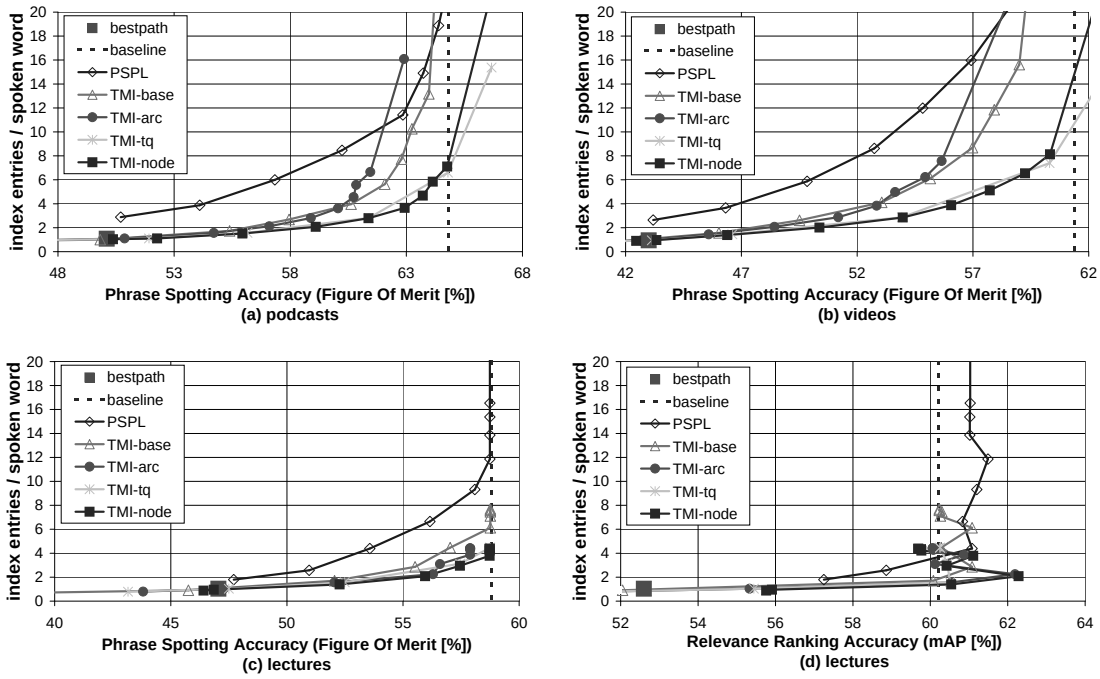

Figure 2: Index size vs. accuracy for different pruning thresholds for word-spotting on (a) podcasts, (b) videos, (c) lectures, and (d) relevance ranking for lectures.

this paper because it continues to use the notion of a word position instead of time, with the advantage that existing implementations of phrase-matching conditions apply without modification.

The results show that, comparing with the direct raw-lattice search, all indexing methods have only slight impact on both word spotting and document retrieval accuracies. Against our expectation, in many cases *improved* accuracies are observed. These are caused by creating additonal paths compared to the original lattice, improving recall. It is not yet clear how to exploit this in a systematic manner.

W.r.t. storage efficiency, the TMI merging methods have about 5 times less index entries than the original lattice for lectures (and an order of magnitude less for podcasts and videos that were recognized with rather wasteful pruning thresholds). This can be further improved by pruning.

4.2.2 Pruning

Index size and accuracy can be balanced by pruning low-scoring index entries. Experiments have shown that the optimal pruning strategy differs slightly from method to method. For the TMI set, the index is pruned by removing all entries with posterior probabilities below a certain fixed threshold. In addition, for TMI-node we enforce that the best path is not pruned. For PSPL, an index entry at a particular word position is removed if its posterior is worse by a fixed factor compared to the best index entry for the *same* word position. This also guarantees that the best path is never pruned.

Fig. 2 depicts the trade-off of size and accuracy for different indexing methods. TMI-node provides the best trade-off. The last block of Table 3 shows results for all indexing methods when pruned with the respective pruning thresholds adjusted such that the number of index entries is approximately five times that for the top-1 transcript. We chose this size because reducing the index size still has limited impact on accuracy (0.5-points for podcasts, 3.5 for videos, and none for lectures) while keeping operating characteristics (storage size, CPU, disk) within an order of magnitude from text search.

5 The System

The presented technique was implemented in a research prototype shown in Fig. 3. About 780 hours of audio documents, including video clips from MSN Video and audio files from most popular podcasts, were indexed. The index is disk-based, its size is 830 MB, using a somewhat wasteful XML representation for research convenience. Typically, searches are executed within 0.5 seconds.

The user interface resembles a typical text search engine. A media player is embedded for immediate within-page playback. Snippets are generated for previewing the search results. Each word in a snippet has its original time point associated, and a click on it positions the media player to the corresponding time in the document.

6 Conclusion

We targeted the paper to the task of searching audio content from the Internet. Aiming at maximizing reuse of existing web-search engines, we investigated how best to

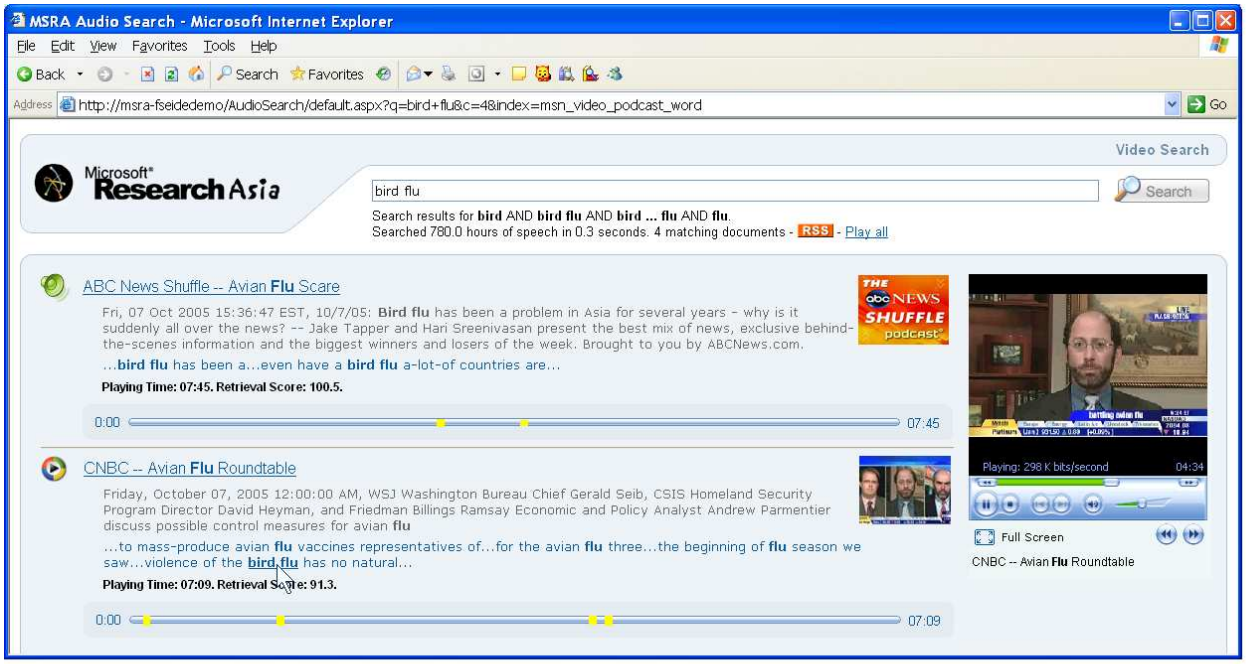

Figure 3: Screenshot of the video/audio-search prototype. For each document, in addition to the title and description text from meta-data, the system displays recognition-transcript snippets around the audio hits, e.g. "... **bird flu** has been a ..." in the first document. Clicking on a word in a snippet starts playing back the video at that position using the embedded video player.

represent important lattice properties – recognition alternates with scores, time boundaries, and phrase-matching constraints – in a form suitable for large-scale web-search engines, while requiring only limited code changes.

The proposed method, Time-based Merging for Indexing (TMI), first converts the word lattice to a posterior-probability representation and then merges word hypotheses with similar time boundaries to reduce the index size. Four approximations were presented, which differ in size and the strictness of phrase-matching constraints.

Results were presented for three typical types of web audio content – podcasts, video clips, and online lectures – for phrase spotting and relevance ranking. Using TMI indexes that are only five times larger than corresponding linear-text indexes, accuracy was improved over searching top-1 transcripts by 25-35% for word spotting and 14% for relevance ranking, very close to what is gained by a direct search of unindexed lattices.

Practical feasibility has been demonstrated by a research prototype with 780 hours indexed audio, which completes searches within 0.5 seconds.

To our knowledge, this is also the first paper to report speech recognition results for podcasts.

7 Acknowledgements

The authors wish to thank Jim Glass and T. J. Hazen at MIT for providing the iCampus data.

References

S. Brin and L. Page, The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems, 30(1-7):107-117*.

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

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

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.

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):373-400*.

MSN Video. `http://video.msn.com`.

The TREC evaluation package. `http://www-lpir.nist.gov/projects/trecvid/trecvid.tools/trec_eval`.

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

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

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

P. Yu, K. J. .Chen, L. Lu, F. Seide, Searching the Audio Notebook: Keyword Search in Recorded Conversations. *Proc. HLT'2005*, Vancouver, 2005.

P. Yu, K. J. Chen, C. Y. Ma, F. Seide, Vocabulary-Independent Indexing of Spontaneous Speech, IEEE transaction on Speech and Audio Processing, Vol.13, No.5, Special Issue on Data Mining of Speech, Audio and Dialog.

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