TOWARDS SPOKEN-DOCUMENT RETRIEVAL FOR THE ENTERPRISE: APPROXIMATE WORD-LATTICE INDEXING WITH TEXT INDEXERS

Frank Seide, Peng Yu, And Yu Shi

Microsoft Research Asia, 5F Beijing Sigma Center, 49 Zhichun Rd., 100080 Beijing, P.R.C.
{fseide,rogeryu,yushi}@microsoft.com

ABSTRACT

Enterprise-scale search engines are generally designed for linear text. Linear text is suboptimal for audio search, where accuracy can be significantly improved if the search includes alternate recognition candidates, commonly represented as word lattices. We propose two methods to enable text indexers to approximately index lattices with little or no code change: "TMI" (Time-based Merging for Indexing) aims at lattice-index size reduction, and the "sausage"-like "TALE" (Time-Anchored Lattice Expansion) approximation requires no indexer-code or data-format changes at all. On four enterprise-type data sets (meetings, phone calls, lectures, and voicemail), TMI and TALE improve accuracy by 30-60% for multi-word phrase searches and by 130% for two-term AND queries, compared to indexing linear text.

Index Terms— Audio indexing, lattice, posterior, keyword spotting.

1. INTRODUCTION

The tremendous progress in audio compression and storage technologies and the pervasive adoption of e-mail and the Intra/Internet has fostered a dramatic increase of the use of digital media in the enterprise, such as online lecture videos, archived meetings or conference calls, and voicemail. Tools are needed to efficiently manage digital enterprise audio assets – audio or video recordings with intrinsic value to the enterprise – particularly search engines.

On the Internet, audio/video search engines can work well by relying on the anchor text, surrounding text, closed captions, and metadata of an audio or video file. For enterprise audio, such information is commonly not available – the audio itself must be processed 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].

To maximize search accuracy, the probabilistic nature of speech recognition must be taken into account [4]. A significant improvement can be achieved through incorporating word confidence scores and alternative recognition candidates by searching *word lattices* instead of linear speech-to-text output [5, 6, 7, 8]. Word lattices are a compact representation of word candidates and their score and time information.

The goal of this paper is to make word-lattice indexing feasible for the enterprise, in particular: (1) search alternates with scores for best possible search accuracy on typical enterprise audio; (2) indexed search for a similarly instant search experience as for text; (3) reuse as much as possible existing enterprise-search code bases, because enterprise-scale search engines are complex systems involving substantial investments in both development and deployment.

This poses two challenges: (a) raw lattices can be as large as 100 times the size of text or more; and (b) although word lattices can theoretically be indexed by the same principles as text, existing text indexing engines cannot do that because their notion of word positions cannot represent alternates with different time boundaries or spanning multiple words.

This paper addresses these challenges by investigating the question: How can word lattices be represented or approximated such that they can be indexed with existing text indexers with little or no modification? We propose two ways: The first, called TMI (Time-based Merging for Indexing [8]), is a *node-clustering procedure for reducing lattice size* significantly, down to operating points that today's text indexers are optimized for, without loss of accuracy. It does require small changes to phrase-matching code and a few extra bits in the index data structure. The second method named TALE (Time-Anchored Lattice Expansion) allows lattice indexing *without core-code change at all* by approximating the lattice further into a "sausage"-like structure, and overgenerating entries to keep M-gram phrases. Both methods are evaluated with multi and single-word queries on four enterprise-type production-size audio data sets, the largest being 300 hours.

This paper is organized as follows. Next, we review prior work, followed by a description of the overall system architecture in section 3 and a discussion of lattices for indexing in section 4. Section 5 introduces the TMI and TALE method, and section 6 presents experimental results.

2. PRIOR WORK

Several approaches have been reported in the literature for indexing spoken words in audio recordings. In the TREC (Text REtrieval Conference) Spoken-Document Retrieval (SDR) benchmarking on audio-retrieval of broadcast-news clips, most systems apply text-based information retrieval to approximate speech-recognition transcripts. They achieve retrieval accuracy similar to using human reference transcripts, and ad-hoc retrieval for broadcast news was concluded to be a

"solved problem" [9]. This scenario differs significantly from ours with its significantly lower word-error rates (20%) and high redundancy of news segments and queries that prevented recognition errors to cause catastrophic failures.

Prior work on lattice indexing includes [5], 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, but no attempt to reduce lattice size is made besides beam pruning. [6] 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, but only achieves a small reduction of index entries and loses time information for individual hypotheses. Also related to our topic are "confusion networks" (or "sausages"), a method to align a speech lattice with its top-1 transcription [10]. Sausages are a parsimonious approximation of lattices, but due to the presence of null links, they do not lend themselves naturally for matching phrases or indexing.

3. SYSTEM ARCHITECTURE

Fig. 1 shows the overall architecture of a search engine for audio/video search. At indexing time, a media decoder extracts the raw audio data from different formats of audio found in the enterprise. This is then fed into a large-vocabulary continuous-speech recognizer (LVCSR), which outputs word lattices. The lattices are processed with our TMI or TALE method and merged into the inverted index.

At search time, list of hits of all query terms are retrieved from the index and intersected (including phrase matching) to determine documents that contain all query terms. The ranker computes relevance scores, and a result presentation module creates snippets for the returned documents and compose the result page, which would contain time information for individual word hits to allow easy navigation and preview.

4. WORD-LATTICE BASED SEARCH

Before presenting the TMI and TALE methods in section 5, we want to review word-lattice based search. First, we discuss how and where lattice indexing can help accuracy – and where not; and then formally define lattices in the form we use.

4.1. Benefit of Lattices

At word accuracies of 50-70%, recognition results are highly uncertain. The idea of lattice indexing is to retain alternative word candidates that the recognizer also considered, with their associated probabilities – Alternates improves Recall, whereas probabilities, representing confidence, improve Precision. For example, while a text represents information like "word w was spoken at a certain point of the recording," a lattice represents "soft" statements like "It appears w_1 was spoken, but expect that only to be correct with a probability of P_1, and by the way it could also have been w_2 with probability P_2 or w_3 with P_3 etc." Figure 2 shows an example.

For *known-item searches* – retrieving one or more known audio document – the dominant metric is Recall, and we have

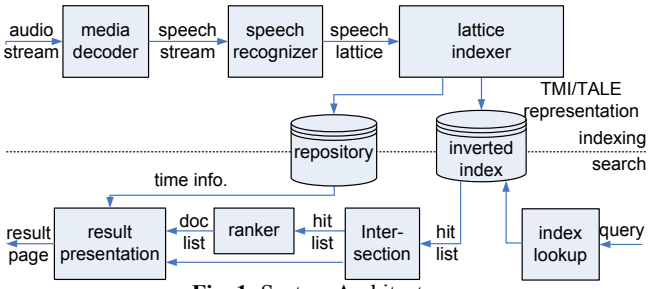

Fig. 1. System Architecture.

shown in [4] that it is optimal (w.r.t. a specific accuracy metric) to list all hits that match the query ranked by the hits' query posterior probabilities, i.e. confidence scores.

For *ad-hoc search* – retrieval of relevant documents, typically unknown, from a large database that may contain multiple acceptable answers – the dominant criterion is Precision. A system with Precision below, say, 70-80% will likely be perceived as broken. Requiring Precision of 70-80% has an interesting consequence: It requires a confidence threshold above 0.5 – but because confidences are probabilities and sum up to 1.0, no second best recognition alternate could ever be retrieved! Thus, it seems that in high-Precision ad-hoc scenarios, indexing alternates cannot help to improve Recall (while Precision can still benefit from confidence scores.)

We can experimentally confirm this for single-word queries. However, most queries are *multi-word queries* (e.g. Internet web queries consist of 2.35 words on average [11]), and query terms are strongly correlated: Hits are significantly more like to be correct than single-word queries when they all appear in the lattice, even more so if they occur in sequence (phrase match). Indeed, we find that even at confidence thresholds far below 0.5, the required precision is retained. For *multi-word (phrase, AND) queries*, including alternates does lead to a significant improvement of Recall even at a high-Precision requirement.

4.2. Lattice Definition

A word lattice is $\mathcal{L} = (\mathcal{N}, \mathcal{A}, n_{\text{enter}}, n_{\text{exit}})$ is a weighted directed acyclic graph (DAG) where arcs $\mathcal{A}$ represent word hypotheses with recognizer scores, and nodes $\mathcal{N}$ the connections between them, encoding times and possibly context conditions.[1] n_{enter} and $n_{\text{exit}} \in \mathcal{N}$ are the unique initial and final node, respectively. The recognizer score of a word hypothesis is used as the arc weight:

$$q_{n_s,w,n_e} \quad = \quad p^{\frac{1}{\lambda}}(O(t_{n_s}...t_{n_e})|n_s, w, n_e) \cdot P(w|n_s)$$

where $p(O(t_{n_s}...t_{n_e})|n_s, w, n_e)$ is the likelihood for acoustic observation $O(t_{n_s}...t_{n_e})$ given word w, its time boundaries (t_s, t_e), and its cross-word triphone context (n_s, n_e). $P(w|n_s)$ is the language-model (LM) probability of word w to follow its LM history (encoded in n_s). λ is the well-known LM weight.[2] Consider $q_{n_s,w,n_e} = 0$ for non-existent arcs.

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

[2] Despite its name, the function of the LM weight is now widely considered to be to flatten acoustic emission probabilities. This matters when sums

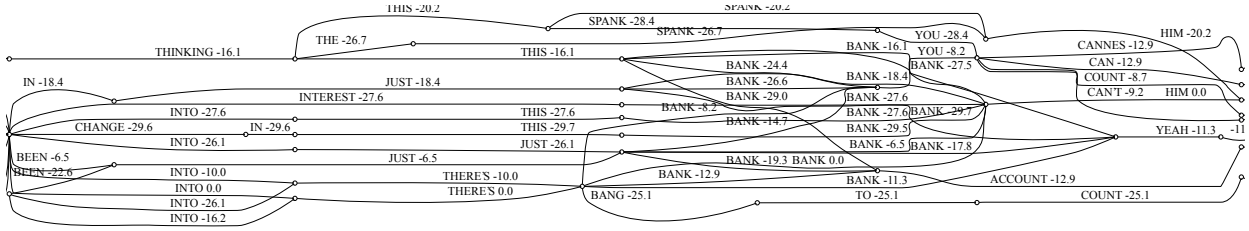

Fig. 2. Word-lattice example for the word sequence "...into this bank account."

The lattice representation allows to answer one question of interest: Given our observed audio recording O, what is the probability $P(*\text{-}t_s\text{-}w\text{-}t_e\text{-}*|O)$ that a particular word w was spoken at a particular time $t_s...t_e$? This quantity is called the *word posterior probability*. Despite its name, it is defined over *paths*, and $*\text{-}t_s\text{-}w\text{-}t_e\text{-}*$ shall denote the set of paths that contain w with boundaries t_s and t_e. To compute it, we sum over all nodes (n_s, n_e) with the given time points (t_s, t_e)[3]:

$$P(*\text{-}t_s\text{-}w\text{-}t_e\text{-}*|O) \quad = \sum_{\substack{(n_s,n_e):\\ t_{n_s}=t_s \wedge t_{n_e}=t_e}} P(*\text{-}n_s\text{-}w\text{-}n_e\text{-}*|O)$$

where the *arc posterior* $P(*\text{-}n_s\text{-}w\text{-}n_e\text{-}*|O)$ is computed as:

$$P(*\text{-}n_s\text{-}w\text{-}n_e\text{-}*|O) \quad = \quad \frac{\alpha_{n_s} \cdot q_{n_s,w,n_e} \cdot \beta_{n_e}}{\beta_{n_{\text{enter}}}}$$

and the *forward probability* α_{n_s} and *backward probability* β_{n_e} represent the sum over all paths from sentence start n_{enter} to n_s and n_e to sentence end n_{exit}, respectively. They can be computed conveniently with the forward-backward recursion [12]. $\beta_{n_{\text{enter}}}$ is the total probability over all paths.

Relevance-ranking formulas often use the *term frequency* TF_w (per-document keyword occurrence). Its expected value can be computed from the lattice as:

$$\begin{aligned} E_{w|O}\{\text{TF}_w\} \quad &= \sum_{\substack{\forall m, n_0 \ldots n_m:\\ n_0=n_{\text{enter}} \wedge\\ n_m=n_{\text{exit}}}} P(n_0\text{-}w_1\text{-}n_1\text{-}...\text{-}w_m\text{-}n_m) \cdot \sum_{\forall i: w_i=w} 1 \\ &= \sum_{\forall n,n'} P(*\text{-}n\text{-}w\text{-}n'\text{-}*|O) \end{aligned}$$

One would also want to answer the same question for multi-word sequences $(w_1 w_2 ... w_m)$, not only to support explicitly quoted phrase queries, but also because sequence matches are significantly more accurate, and query terms often occur in sequence (implicit phrases). The *phrase posterior* $P(*\text{-}t_s\text{-}w_1...w_m\text{-}t_e\text{-}*|O)$ can be computed by summing over all m-arc paths with the given boundaries t_s and t_e:

$$P(*\text{-}t_s\text{-}w_1...w_m\text{-}t_e\text{-}*|O) = \sum_{\substack{\forall m, n_0 \ldots n_m:\\ t_{n_0}=t_s \wedge\\ t_{n_m}=t_e}} P(*\text{-}n_0\text{-}w_1\text{-}n_1\text{-}...\text{-}w_m\text{-}n_m\text{-}*|O)$$

$$P(*\text{-}n_0\text{-}w_1\text{-}n_1\text{-}...\text{-}w_m\text{-}n_m\text{-}*|O) \quad = \quad \frac{\alpha_{n_0} \prod_{i=1}^{m} q_{n_{i-1},w_i,n_i} \beta_{n_m}}{\beta_{n_{\text{enter}}}}$$

of path probabilities are taken instead of just determining the best path.

[3]In most applications, one would also relax the time boundaries, e.g. extending the sum to include alternate boundaries with significant overlap.

In this paper, we actually use an equivalent, more convenient representation, which we call the *posterior lattice*. In a posterior lattice, arc weights are not q_{n_s,w,n_e} but directly the precomputed arc posteriors $P(*\text{-}n_s\text{-}w\text{-}n_e\text{-}*|O)$. This representation still allows exact computation of phrase posteriors:

$$P(*\text{-}n_0\text{-}w_1\text{-}n_1\text{-}...\text{-}w_m\text{-}n_m\text{-}*|O) \quad = \quad \frac{\prod_{i=1}^{m} P(*\text{-}n_{i-1}\text{-}w_i\text{-}n_i\text{-}*|O)}{\prod_{i=1}^{m-1} P(*\text{-}n_i\text{-}*|O)}$$

$$\text{with} \quad P(*\text{-}n\text{-}*|O) \quad = \quad \frac{\alpha_n \beta_n}{\beta_{n_{\text{enter}}}} \quad = \quad \sum_{\forall n'} \sum_{\forall w} P(*\text{-}n\text{-}w\text{-}n'\text{-}*|O)$$

We call the new term $P(*\text{-}n\text{-}*|O)$ *node posterior*. The primary advantage of the posterior representation is that posteriors are resilient to approximations like aggressive quantization and merging of alternates with non-identical time boundaries, and they allow comparing arcs with different time durations and temporal splitting e.g. compound words. Further, the node posteriors turn out to be uncritical and can be replaced by a constant in our scenario.

5. INDEXING LATTICES WITH TEXT INDEXERS WITH NO OR LITTLE CHANGES

This section addresses the main question of this paper: How to represent or approximate word lattices such that they can be indexed with existing text indexers with little or no modification?

Lattices can be indexed according to the same principles as text. An inverse index stores for every term a list of its occurences, including document id, relative *word position*, and supplementary *information for ranking* (e.g. font size [13]). Documents can be retrieved efficiently by intersecting the lists of all query terms. For the typical "Google-style" query we are all familiar with, all query terms must be present at least once in a document, and for phrases, query terms must occur in consecutive word positions.

To transform a text indexer into a lattice indexer, "word position" must be changed to store start and end node (data structure change) and the phrase matcher must use that information (code change); the word posterior has to be stored e.g. as part of the supplementary "ranking information" and must be used by the ranker. Node times are needed only for the result display, and are not stored in the inverted index.

Having posteriors used by the ranker can be achieved by a trick. Typical indexer designs use the "ranking information" as an abstract type index into a weight table [13]. To use posteriors in ranking, we'd just need to change the weight table accordingly. Thus, the remaining issue is phrase matching.

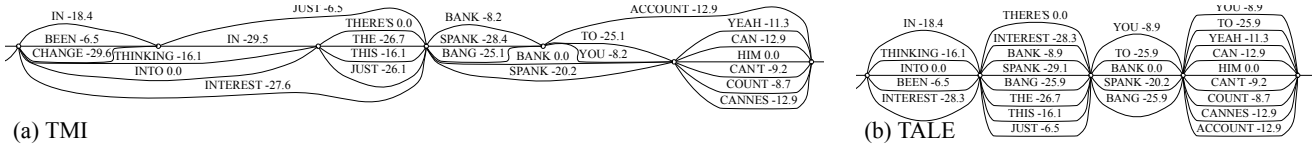

Fig. 3. The lattice of Fig. 2 after TMI (a) and TALE (b) processing.

If the code and data-format changes are permissible, the remaining challenge is lattice size. The TMI method (Time-based Merging for Indexing) addresses this problem. If even the above changes are not possible, further approximations are needed to map nodes to word positions. This is addressed by the TALE method (Time-Anchored Lattice Expansion).

5.1. TMI – Time-based Merging for Indexing

As Fig. 2 shows, a single word often has multiple lattice arcs with different acoustic or language-model context conditions and slightly off time boundaries. The objective of the TMI method, which was first introduced in [8], is to significantly reduce the lattice size by exploiting this redundancy, while retaining all word sequences to avoid false negatives.

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 simple: 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 condition, the one leading 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 $E_{w|O}\{\mathrm{TF}_w\}$ and keeps all phrases. It also introduces additional paths, but they are restricted to not cause insertions or deletions of full words, and our experiments show no loss of accuracy from false positives (these random combinations are unlikely to be valid phrases that 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), and the resulting number of nodes is only slightly above the number of spoken words, so text indexers' mechanisms to store word positions, which may involve compression, are suited to store start nodes n_s. Indexing TMI lattices still requires small code and data-structure modifications to store end nodes (n_e–n_s is barely ever above 8: three bits are sufficient) and interpret it for phrase matching. Scores are stored in the ranking information and do not require code change.

5.2. TALE – Time-Anchored Lattice Expansion

The objective of TALE is to approximate lattice indexing where changes to the core indexer code or data format are *not* an option. In this case, words must be aligned to word positions, forming a sausage-like lattice. The standard phrase matcher requires words belonging to phrases to be in consecutive word positions, i.e. some words must be aligned to multiple slots (overgeneration). It is impossible to guarantee retaining all possible phrases while keeping phrase posteriors and keeping the index small. We have to set priorities.

TALE aims to retain the expected term frequencies $E_{w|O}\{\mathrm{TF}_w\}$ (they matter for ranking); 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 conditional probability that word w happens as the Δ-th path token after a given node n:

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

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 "P_Δ-Expansion." It is easy to show that $E_{w|O}\{\mathrm{TF}_w\}$ remains unchanged for all w. The time information are retained by the anchor points.

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

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

with $\sum \alpha_\Delta = 1$. The weights α_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. We name this method "TALE" (Time-Anchored Lattice Expansion)

Table 1. Summary of data sets used.

set and scenario	dur. [h]	#documents	av. dur [min]	WER [%]
lectures (online learning)	170	169	60.4	46.6
1:1 calls (call mining)	310	2435	7.6	37.5
meetings (archive)	99	104	57.1	45.7
voicemails (personal)	30	3934	0.5	35.6

6. EXPERIMENTAL RESULTS

6.1. Setup

We evaluate our method on four different data sets totaling 600 hours of audio from popular speech corpora. They were chosen to represent typical enterprise scenarios – online learning: MIT iCampus *lectures* [3]; call mining: Switchboard-I *telephone conversations*; *meeting archive* scenario: the training set of Rich Transcription 2004 Spring Meeting Recognition Evaluation; and personal search: *voicemails* (IBM Voicemail I+II). Table 1 summarizes the dataset information.

Raw lattices were generated with a speaker-independent LVCSR system. Its acoustic model was trained on the 1700-hour Switchboard "Fisher" telephony-speech set (does not include Switchboard-I) [2]. The trigram language model (LM) for recognizing Switchboard-I conversations was trained on Fisher transcripts. Due to limited LM data for voicemails and 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 disjunct. For meetings, LMs were trained on transcripts of a collection of broadcast news, voicemails, and telephone conversations, plus the meetings transcripts in the same 10-part held-out manner. Word-error rates (WER) range from 35 to 47%.

We evaluate our method on keyword search without relevance weighting. We include multi-word phrase queries, single-word queries, and two-term AND queries (each term can be single or multi-word). 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 documents.

We use three different accuracy metrics:

- FOM: The NIST Figure Of Merit defined as the detection/false-alarm curve averaged over [0..10] false alarms per keyword per h hours. Instead of the original $h = 1$, we use h=data set duration from Table 1.
- mAP: mean average precision, ranking documents by the probability that the single or multi-word query Q occurs in the document at least once:

$$P(Q \text{ in } doc) \quad = \quad 1 - \prod_{\forall \text{hits } h \text{ for } Q} (1 - P(h|doc))$$

  and for AND queries:

$$P(X \text{ AND } Y \text{ in } doc) \quad = \quad P(X \text{ in } doc) \cdot P(Y \text{ in } doc)$$

- $\mathrm{R}_{75}/\mathrm{R}_{50}$: document Recall at Precision 75%; and at 50% for single-word queries, for which it proved difficult to get enough observation points above 75%.

Both mAP and R_{75} rely on a joint ranking among all query terms. Therefore we normalize the word and phrase posteriors using three 2-state Hidden-State Maximum-Entropy models [14] for single words, two-word phrase, and longer phrases, respectively. This improves AND-query results by up to 5 points.

6.2. Speech-To-Text Transcript vs. Lattice Indexing

Table 2 shows the results. The first and second result rows compare accuracies for the simplest 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. (Without that, there is only one Precision/Recall point, for which the results are very similar to the R_{P} result.)

The first three result columns show accuracies for phrase queries. From STT transcript to raw lattice, a significant improvement is observed. For FOM and mAP, relative improvements are 63 and 57%, respectively, and 28% for R_{75}. For the single-word queries (next three columns), improvements for FOM and mAP are in the 30% range, and none for R_{50}. The next three columns show AND-query results. Both mAP and R_{75} increase by a solid 2.4 times. Indeed, indexing alternates helps significantly for known-item searches (FOM, mAP), while for ad-hoc queries with high precision ($\mathrm{R}_{75}/\mathrm{R}_{50}$) it helps for multi-word queries. The results are shown for only one data set, iCampus lectures. For the other data sets, the picture is very consistent.

6.3. Lattice Reduction – TMI

In the next three rows, we compare the effect of the TMI size-reduction method. First, TMI reduces the lattice size (last column) over 30 times, from around 1600 arcs per spoken word to 46. We admittedly use *very* rich lattices here – in earlier experiments [8] 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 hit – 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.

Because the lattice size is still above our target of 10 arcs per spoken word, we prune the lattice by removing all arcs with posteriors below -8. The result (in the next row "TMI with pruning") shows a 2-6 points accuracy hit for multi-word queries for FOM and mAP. 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.

"TMI with pruning" realizes nearly all the benefit of indexing "raw lattices" while only requiring about 10 arcs per spoken word. Table 2 also shows the pruning results for the other three data sets. The results are very consistent.

6.4. Lattice Expansion – TALE

For TALE, we first compare binning of words into word slots *without* and with the TALE expansion. Not using the TALE expansion causes a 5-point accuracy hit for multi-word

Table 2. Search results for phrase queries, single-word queries, and two-term queries of the form X AND Y where X and Y may be phrases. Shown are Figure of Merit (FOM) per keyword hit, per-document mean average precision (mAP), and per-document Recall at a certain Precision cut-off (R_P) for P=75% and 50% for multi and single-word queries, respectively. "Index size" is index entries per spoken word. "Relative improvement" is from "STT transcript with confidence" to the "with pruning."

query type:	phrase queries			single-word queries			X AND Y queries			index
configuration	FOM	mAP	R_{75}	FOM	mAP	R_{50}	FOM	mAP	R_{75}	size
iCampus Lectures (Online Learning)										
STT transcript with confidence	40.6	42.7	43.4	36.4	44.2	45.2	42.8	26.1	26.1	1.0
raw lattice	66.1	67.2	55.7	49.0	55.9	45.4	55.6	63.3	61.6	1617
TMI (node merging w/o loops)	68.4	69.6	58.0	48.7	55.9	45.4	56.1	66.3	63.9	46.2
TMI with pruning	65.7	67.1	58.3	48.2	55.4	45.4	55.0	60.4	61.0	9.9
⇒ relative improvement over STT	**+62%**	**+57%**	**+34%**	**+32%**	**+25%**	**+1%**	**+30%**	**×2.3**	**×2.3**	-
simple binning of word hypotheses	63.9	65.4	55.5	47.8	55.9	45.4	54.0	61.1	58.9	19.9
TALE (binning with expansion)	68.5	70.2	57.6	48.4	55.8	45.5	56.2	67.7	63.7	35.5
TALE with pruning	65.8	67.6	57.7	48.1	55.4	45.5	55.2	61.5	61.7	11.5
⇒ relative improvement over STT	**+62%**	**+58%**	**+33%**	**+32%**	**+25%**	**+1%**	**+29%**	**×2.4**	**×2.4**	-
Switchboard 1:1 Telephone Conversations (Call Mining)										
STT transcript with confidence	50.5	52.1	52.8	45.8	50.0	52.7	53.2	33.2	33.2	1.0
TMI with pruning	78.1	78.9	70.9	59.5	63.3	55.1	70.2	71.1	71.3	8.0
TALE with pruning	75.8	76.7	66.9	59.3	61.9	53.8	68.6	68.6	68.8	9.3
LDC Meetings (Meeting Archives)										
STT transcript with confidence	37.7	40.1	40.5	40.0	45.1	40.6	41.9	25.7	25.7	1.0
TMI with pruning	65.0	67.0	60.1	54.0	58.3	38.6	57.6	60.1	58.9	9.5
TALE with pruning	65.2	67.4	58.0	54.9	58.3	39.4	58.2	60.7	58.4	11.3
LDC Voicemails (Personal Search)										
STT transcript with confidence	44.4	44.5	45.5	37.9	37.5	37.9	51.0	26.7	26.7	1.0
TMI with pruning	68.5	67.2	62.1	50.8	47.5	38.6	64.5	56.6	56.9	8.0
TALE with pruning	68.6	67.2	61.5	51.5	47.5	38.6	65.0	57.0	57.3	9.4
Cross-corpus Averages										
TMI av. relative improvement over STT	**+62%**	**+56%**	**+35%**	**+34%**	**+26%**	**+0%**	**+32%**	**×2.2**	**×2.2**	-
TALE av. relative improvement over STT	**+61%**	**+57%**	**+38%**	**+33%**	**+27%**	**+1%**	**+32%**	**×2.2**	**×2.2**	-

queries. This shows again the cost of false negatives. However, that the lattice size of TALE is nearly twice that of simple binning. Besides, accuracies for TALE are very similar to TMI, except for Switchboard we observe a 2-4 points drop.

7. CONCLUSION

We have presented two methods to enable text indexers to approximately index lattices with little or no code change: "TMI" (Time-based Merging for Indexing) aims at lattice-index size reduction, and the sausage-like "TALE" (Time-Anchored Lattice Expansion) approximation requires no indexer-code or data-format changes at all. The effectiveness of TMI and TALE has been shown for keyword search on four enterprise-type data sets, where both achieve accuracy improvements of 30-60% for multi-word phrase searches and 130% for two-term AND queries, compared to indexing linear text. It has been found that lattices can help to improve known-item searches, while for ad-hoc searches with high precision requirements they only benefit multi-word queries. We also found that in lattice indexing it is important to retain phrases as much as possible, while a negative impact from creation of additional paths through false positives was not observed. By enabling reuse of existing text indexers, this paper allows to capitalize on the immense investments made on enterprise text search products, and we hope that it will open a path for accelerated deployment of speech-recognition based audio and video search solutions in the enterprise.

8. 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] 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.

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

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

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

[8] 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*, New York, 2006.

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

[10] 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*.

[11] Craig Silverstein, Monika Henzinger, Hannes Marais, and Michael Moricz, Analysis of a Very Large Web Search Engine Query Log. ACM SIGIR Forum, Volume 33, Issue 1 (Fall 1999).

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

[13] 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*.

[14] P. Yu, J. Xu, G. L. Zhang, Y. C. Chang, and F. Seide, A Hidden-State Maximum Entropy Model for Word Confidence Estimation. *Proc. ICASSP'2007*, Honolulu, 2007.