# On the Computation of Document Frequency Statistics from Spoken Corpora using Factor Automata

*Doğan Can*[1], *Shrikanth S. Narayanan*[1,2]

[1]Department of Computer Science and [2]Department of Electrical Engineering,
University of Southern California, Los Angeles, CA

dogancan@usc.edu, shri@sipi.usc.edu

## Abstract

Factor automaton is an efficient data structure for representing all factors (substrings) of a set of strings (e.g. a finite-state automaton). This notion can be generalized to weighted automata by associating a weight to each factor. In this paper, we consider the problem of computing expected document frequency (DF), and TF-IDF statistics for all substrings seen in a collection of word lattices by means of factor automata. We present an algorithm which transforms an acyclic weighted automaton, e.g. an ASR lattice, to a weighted factor automaton where the path weight of each factor represents the total weight associated by the input automaton to the set of strings including that factor at least once. We show how this automaton can be used to efficiently construct other types of weighted factor automata representing DF and TF-IDF statistics for all factors seen in a large speech corpus. Compared to the state-of-the-art in computing these statistics from spoken documents, our approach i) generalizes the statistics from single words to contiguous substrings, ii) provides significant gains in terms of average run-time and storage requirements and iii) constructs efficient inverted index structures for retrieval of such statistics. Experiments on a Turkish data set corroborate our claims.

**Index Terms**: Factor Automata, Weighted Finite-State Transducers, Lattice Indexing, Spoken Document Retrieval

## 1. Introduction

Speech retrieval (SR) is a key technology which integrates automatic speech recognition (ASR) and information retrieval (IR) to provide large scale access to spoken content. Computing document similarity is a fundamental need in SR just as in the case of text retrieval. Inverse document frequency (IDF) [1, 2] is an important term specificity measure used in almost every IR system in some form. In its most basic form, it is defined as $-\log$ of the fraction of documents that include a term. IDF computation involves computing document frequency (DF), simply defined as the number of documents that include a term. Term frequency (TF) is another important measure, which in its most basic form is defined as the number of occurrences of a term in a document. The more general class of term weighting schemes known as TF-IDF, which involves multiplying an IDF measure by a TF measure, constitutes the basis for quantifying document similarity in almost every IR system. While it is fairly straightforward to compute TF and DF measures from textual documents, the same cannot be said for spoken ones since the computation should be carried out over probability distributions over strings, e.g. ASR lattices. Approximating TF and DF measures by the statistics of 1-best ASR output is a common strategy used in SR. However, as shown by [3], proper estimation of these measures over ASR lattices provides significant gains in performance over the 1-best baseline.

Factor automaton (Section 2.3) is an efficient data structure for representing all factors (substrings) of a set of strings (e.g. a finite-state automaton). This notion can be generalized to weighted finite-state automata [4, 5] by associating a weight to each factor (Section 3). In this paper, we consider the problem of computing TF, DF, and TF-IDF measures in the expected sense for all factors seen in a collection of word sequence lattices by means of weighted factor automata. We first introduce the concept of factor weight of a string $x$, defined as the total weight associated by a weighted automaton to the set of strings which include $x$ as a substring at least once (Section 3). Then, we present a novel algorithm (Section 3.1) for transforming an acyclic probabilistic automaton, e.g. an ASR lattice, into a term probability factor automaton which is an ideal index structure for spoken utterance retrieval applications. We then show how this automaton can be used to efficiently construct other weighted factor automata representing DF (Section 3.2), IDF and TF-IDF (Section 3.3) statistics for all factors of a collection of ASR lattices. Finally we provide experiments evaluating the average run-time and storage requirements of our approach and conclude with a brief discussion (Section 4).

## 2. Preliminaries

This section introduces the definitions and notation related to weighted transducers and automata [6], as well as the definitions for TF, DF and TF-IDF measures as used in this paper.

### 2.1. Semirings

**Definition 1** *A semiring is a 5-tuple* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *where* $(\mathbb{K}, \oplus, \overline{0})$ *is a commutative monoid,* $(\mathbb{K}, \otimes, \overline{1})$ *is a monoid,* $\otimes$ *distributes over* $\oplus$ *and* $\overline{0}$ *is an annihilator for* $\otimes$.

Table 1: Common semirings.

| SEMIRING | SET | $\oplus$ | $\otimes$ | $\overline{0}$ | $\overline{1}$ |
|---|---|---|---|---|---|
| Boolean | $\{0,1\}$ | $\vee$ | $\wedge$ | 0 | 1 |
| Real | $\mathbb{R}_+ \cup \{+\infty\}$ | $+$ | $\times$ | 0 | 1 |
| Max-times | $\mathbb{R}_+ \cup \{+\infty\}$ | max | $\times$ | 0 | 1 |
| Log | $\mathbb{R} \cup \{-\infty, +\infty\}$ | $\oplus_{\log}$ | $+$ | $+\infty$ | 0 |
| Tropical | $\mathbb{R} \cup \{-\infty, +\infty\}$ | min | $+$ | $+\infty$ | 0 |

$a \oplus_{\log} b = -\log(e^{-a} + e^{-b})$

Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD / ARL) contract number W911NF-12-C-0012.

### 2.2. Weighted Finite-State Transducers and Automata

**Definition 2** *A weighted finite-state transducer $T$ over a semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is an 8-tuple $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ where: $\Sigma$, $\Delta$ are respectively the finite input and output alphabets; $Q$ is a finite set of states; $I, F \subseteq Q$ are respectively the set of initial and final states; $E \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Delta \cup \{\varepsilon\}) \times \mathbb{K} \times Q$ is a finite set of arcs; $\lambda : I \to \mathbb{K}$, $\rho : F \to \mathbb{K}$ are respectively the initial and final weight functions.*

Given an arc $e \in E$, we denote by $i[e]$ its input label, $o[e]$ its output label, $w[e]$ its weight, $p[e]$ its origin state and $n[e]$ its destination state. A path $\pi = e_1 \cdots e_k$ is an element of $E^*$ with consecutive arcs satisfying $n[e_{i-1}] = p[e_i]$, $i = 2, \ldots, k$. We extend $n$ and $p$ to paths by setting $n[\pi] = n[e_k]$ and $p[\pi] = p[e_1]$. The labeling and the weight functions can also be extended to paths by defining $i[\pi] = i[e_1] \cdots i[e_k]$, $o[\pi] = o[e_1] \cdots o[e_k]$ and $w[\pi] = w[e_1] \otimes \cdots \otimes w[e_k]$. We denote by $\Pi(q, q')$ the set of paths from $q$ to $q'$ and by $\Pi(q, x, y, q')$ the set of paths from $q$ to $q'$ with input label $x \in \Sigma^*$ and output label $y \in \Delta^*$. These definitions can be extended to subsets $S, S' \subseteq Q$, e.g.

$$\Pi(S, x, y, S') = \bigcup_{q \in S, q' \in S'} \Pi(q, x, y, q').$$

An *accepting* path in a transducer $T$ is a path in $\Pi(I, F)$. A string $x$ is accepted by $T$ if there exists an accepting path $\pi$ labeled with $x$ on the input side. $T$ is *unambiguous* if for any string $x \in \Sigma^*$ there is at most one accepting path labeled with $x$ on the input side. $T$ is *deterministic* if it has at most one initial state and at any state no two outgoing transitions share the same input label. Let $\pi$ and $\pi'$ be two paths of a transducer T with the same input and output labels: $i[\pi] = i[\pi']$ and $o[\pi] = o[\pi']$. We say that $\pi = e_1 \cdots e_n$ and $\pi' = e'_1 \cdots e'_{n'}$ are *identical* if they have the same number of transitions ($n = n'$) with the same labels: $i[e_k] = i[e'_k]$ and $o[e_k] = o[e'_k]$ for $k = 1, \ldots, n$. A transducer $T$ is said to be *normalized*, if any two paths $\pi$ and $\pi'$ with the same input and output labels are identical.

The weight associated by a transducer $T$ to any input-output string pair $(x, y) \in \Sigma^* \times \Delta^*$ is given by

$$[\![T]\!](x, y) = \bigoplus_{\pi \in \Pi(I, x, y, F)} \lambda(p[\pi]) \otimes w[\pi] \otimes \rho(n[\pi])$$

and $[\![T]\!](x, y)$ is defined to be $\overline{0}$ when $\Pi(I, x, y, F) = \varnothing$.

We denote by $s[A]$ the $\oplus$-sum of the weights of all accepting paths of $A$ when it is defined and in $\mathbb{K}$. $s[A]$ can be viewed as the shortest-distance from the initial states to the final states.

A *weighted finite-state automaton* $A$ can be defined as a weighted finite-state transducer with identical input and output labels. The weight associated by $A$ to $(x, x)$ is denoted by $[\![A]\!](x)$. Similarly, in the graphical representation of weighted automata, output labels are omitted.

A weighted automaton $A$ defined over the probability semiring $(\mathbb{R}_+, +, \times, 0, 1)$ is said to be *probabilistic* if for any state $q \in Q$, $\oplus_{\pi \in \Pi(q,q)} w[\pi]$, the sum of the weights of all cycles at $q$, is well-defined and in $\mathbb{R}_+$ and $\sum_{x \in \Sigma^*} [\![A]\!](x) = 1$.

### 2.3. Factor Automata

**Definition 3** *Given two strings $x, y \in \Sigma^*$, $x$ is a factor (substring) of $y$ if $y = uxv$ for some $u, v \in \Sigma^*$. More generally, $x$ is a factor of a language $L \subseteq \Sigma^*$ if $x$ is a factor of some string $y \in L$. The factor automaton $S(y)$ of a string $y$ is the minimal deterministic finite-state automaton recognizing exactly the set of factors of $y$.*

$S(y)$ can be built in linear time and its size is linear in the size of the input string $|y|$ [7, 8]. We denote by $S(A)$ the minimal deterministic automaton recognizing the set of factors of a finite-state automaton $A$, that is the set of factors of the strings accepted by $A$. A recent work [9] showed that the size of the factor automaton $S(A)$ is linear in the size of the input automaton $|A| = |Q| + |E|$ and provided an algorithm for the construction of $S(A)$ in linear time when $A$ is unweighted.

### 2.4. Term Frequency, Document Frequency, TF-IDF

We define the term frequency $TF(x, i)$ as the number of occurrences of a term $x$ in document $D_i$, and the document frequency $DF(x)$ as the fraction of documents including the term $x$ in a collection of documents $\{D_i | i = 1, \cdots, n\}$:

$$DF(x) = \frac{|\{i | x \in D_i\}|}{n}. \quad (1)$$

TF-IDF can be defined in a number of ways by choosing a TF and an IDF definition from a multitude of options. Here we use:

$$TFIDF(x, i) = TF(x, i) \log \frac{1}{DF(x)}. \quad (2)$$

## 3. Factor Automata of Weighted Automata

Factor automaton (FA) notion can be generalized to weighted automata by associating a weight to each factor. When the input automata are probabilistic, there are several meaningful statistics that can be used as a weight. The weighted index structure described in [4] uses expected term frequencies derived from probabilistic automata. In general, any weight satisfying the semiring conditions can be used. For instance, the timed factor transducer structure described in [5] uses (probability, start time, end time) triplets derived from time aligned ASR lattices.

The unweighted FA can be thought as a weighted automaton over the Boolean semiring where the path weights represent factor occurrence, i.e. each factor of the input automaton is assigned weight $\overline{1}$, any other string is assigned weight $\overline{0}$. The natural extension of this occurrence concept to probabilistic automata is simply the probability of occurrence (at least once). Given a probabilistic input automaton, the probability of occurrence of a factor is simply the sum of probabilities assigned to each accepting path including that factor. More formally, for a weighted finite-state automaton $A$ over the semiring $\mathcal{K}$, we define the *factor weight* $\langle\!\langle A \rangle\!\rangle(x)$ of a string $x \in \Sigma^*$ as the $\oplus$-sum of the weights of all accepting paths including $x$ as a factor:

$$\langle\!\langle A \rangle\!\rangle(x) = \bigoplus_{\substack{\pi \in \Pi(I, F) \\ \exists u, v \in \Sigma^* : \, i[\pi] = uxv}} \lambda(p[\pi]) \otimes w[\pi] \otimes \rho(n[\pi]) \quad (3)$$

Over the Boolean semiring, factor weight is simply the logical disjunction of Boolean path weights and reduces to factor occurrence. Over the real (equivalently log) semiring, factor weight corresponds to the sum ($\oplus_{\log}$-sum) of path weights, which is simply the probability ($-\log$ probability) of occurrence of a factor (at least once) when the input automaton is probabilistic.

Next section presents a recipe based on weighted transducer algorithms [6] for transforming an acyclic probabilistic automaton into a term probability (TP) factor automaton where the path weight of each factor represents its probability of occurrence, i.e. factor weight over the real semiring. TP factor automaton has an ideal weight structure for spoken utterance retrieval [4, 10] applications where the goal is to retrieve all utterances

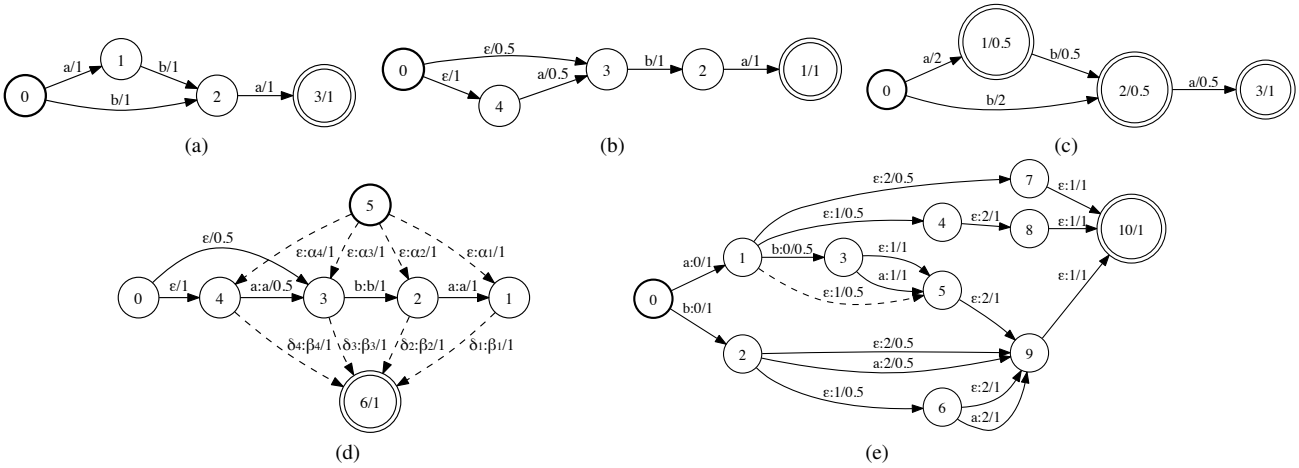


Figure 1: Construction of $S$ from $A$. (a) Input automaton ($A$) over the real semiring, (b) after preprocessing ($B$), (c) resulting factor automaton ($S$), (d) after factor generation ($T$), (e) after factor normalization and determinization over the max-times semiring ($N$).

containing a query term in a large spoken archive. We should note that, unlike TF which can be expressed as a sum over the posterior probabilities of factor occurrences [4], TP measure given by (3) is a sum over the probabilities of accepting paths including a factor. This subtle difference makes the TP computation significantly harder since we have to make sure that the accepting paths including a factor multiple times are counted only once. The main idea is to construct a mapping from any factor to the accepting paths including it and eliminating the path replications caused by factors occurring multiple times in a path before doing the probability summation.

### 3.1. Term Probability Factor Automaton

Let $A$ denote an acyclic automaton over the log (real) semiring. Figure 1a gives such an automaton over the real semiring. We can derive the TP factor automaton $S$ from $A$ in four steps:

#### *3.1.1. Preprocessing*

We assume the input is an acyclic probabilistic automaton over the log (real) semiring. When that is not the case, we can use the general weight-pushing algorithm [6] to convert input weights to posterior probabilities. We also require the input to satisfy three specific conditions which are crucial for the correctness of the algorithm: i) it should be unambiguous, ii) no two arcs coming into a state should have the same label and iii) it should be $\varepsilon$-free except for the arcs originating from the initial states. All of these conditions can be satisfied by reversing the input automaton, applying weighted epsilon removal and determinization on the reverse machine, and then reversing the resulting automaton [6]. Figure 1b illustrates the result of these operations.

#### *3.1.2. Generation*

Let $B = (\Sigma, \Sigma, Q, I, F, E, \lambda, \rho)$ denote the automaton obtained after preprocessing $A$. We generate factors of $B$ (Figure 1d) simply by creating a unique initial state $q_I \notin Q$, a unique final state $q_F \notin Q$, and two new arcs for each non-initial state $q \in Q \setminus I$: $(q_I, \varepsilon, \alpha_q, \overline{1}, q)$ and $(q, \delta_q, \beta_q, \overline{1}, q_F)$. Here, $\delta_q$ is a unique disambiguation symbol. $\alpha_q$ (or $\beta_q$) is a unique non-terminal symbol representing the prefix transducer $U_q = (\{\varepsilon\}, \Sigma, Q, I, \{q\}, E', \lambda, 1)$ (resp. suffix transducer $V_q = (\{\varepsilon\}, \Sigma, Q, \{q\}, F, E', 1, \rho)$) derived from $B$ by setting $q$ as the sole final (resp. initial) state and replacing the input symbols with $\varepsilon$, i.e. $E' = \{(p[e], i[e], \varepsilon, w[e], n[e]) | e \in E\}$. Technically, the (root) transducer $T$ obtained after these changes and the set of finite state transducers $\{U_q | q \in Q \setminus I\} \cup \{V_q | q \in Q \setminus I\}$ define a recursive transition network (RTN) [11]. Due to the unique disambiguation symbols $\{\delta_q | q \in Q \setminus I\}$ on the input side of the new arcs into the unique final state $q_F$ and the fact that $B$ is unambiguous with no two arcs coming into a state having the same label, $T$ is a functional transducer. Optionally, we can filter the generated factors by composing a finite-state grammar $G$ on the input side of $T$ [4]. The default filter rejects empty factors, i.e. any path $\pi$ in $T$ for which the only non-$\varepsilon$ input label is the disambiguation symbol at the end.

#### *3.1.3. Normalization*

We obtain a normalized transducer $N$ from $T$ by applying weighted transducer determinization, replacing the disambiguation symbols with $\varepsilon$ symbols, replacing the non-terminal symbols [11] with the corresponding prefix and suffix transducers and finally applying weighted epsilon removal. $N$ defines a mapping from each factor $x$ of $B$ to the set of paths in $B$ that include $x$ as a factor. Weight of a path $\pi$ in $N$ equals $[\![B]\!](o[\pi])$, the weight associated by $B$ to the path defined by the output symbols $o[\pi]$. Furthermore, all paths in $N$ with the same input and output labels are identical and all identical paths in $N$ have the same weight by construction since $B$ is unambiguous and the path weights in $N$ are determined solely by the output labels. We merge the identical paths in $N$ by viewing it as an acceptor, i.e. encoding input-output labels as a single label, and applying weighted automata determinization over the tropical (max-times) semiring. Since tropical semiring is idempotent, determinization simply eliminates the repeating paths which are generated when a factor occurs multiple times on an accepting path of $B$. Note that it is crucial to use an idempotent semiring at this point since otherwise repeating paths are merged by $\oplus$-summing identical path weights which in general produces a sum different from the input path weights. Figure 1e illustrates the normalized transducer (with the dashed arc) and the final result after encoded determinization (without the dashed arc).

#### *3.1.4. Optimization*

At this step, we project $N$ to its input labels (representing factors) and apply weighted $\varepsilon$-removal and minimization [6] over

the log semiring to obtain the TP factor automaton $S$. Note that we can directly apply minimization since the machine is already deterministic. Figure 1c illustrates the result of optimization.

### 3.2. Document Frequency Factor Automaton

Using TP factor automata, we can efficiently compute and store expected document frequencies for each factor of a collection of probabilistic automata, e.g. ASR lattices. For each input automaton $A_i$ of the collection $\{A_i | i = 1, \ldots, n\}$, we construct a TP factor automaton $S_i$ over the log semiring. The document frequency factor automaton $S^{\mathrm{DF}}$ of the entire collection is constructed simply by taking the union $U$ of individual TP factor automata, concatenating a simple automaton

$$N = (\{\varepsilon\}, \{\varepsilon\}, \{0,1\}, \{0\}, \{1\}, \{(0, \varepsilon, \varepsilon, -\log(n), 1)\}, 1, 1)$$

representing the size of the collection and $U$, and finally applying weighted $\varepsilon$-removal, determinization, and minimization over the log semiring.

### 3.3. Term Frequency and TF-IDF Factor Automaton

We can use the algorithm described in [4] to construct a term frequency factor automaton $S_i^{\mathrm{TF}}$ for each input automaton $A_i$ in the collection. These automata can be combined with the DF factor automaton $S^{\mathrm{DF}}$ of the entire collection to construct the TF-IDF factor automata $S_i^{\mathrm{TF,TF-IDF}}$ in which path weights represent (TF, TF-IDF) pairs. The desired relation between these automata can be expressed as:

$$[\![S_i^{\mathrm{TF,TF-IDF}}]\!](x) = ([\![S_i^{\mathrm{TF}}]\!](x), [\![S_i^{\mathrm{TF}}]\!](x) \otimes -\log [\![S^{\mathrm{DF}}]\!](x))$$

This operation can be carried out with the weighted intersection operation over a special semiring structure known as the expectation (or entropy) semiring [12, 13]. Expectation semiring is defined as follows:

$$\mathcal{E} = ((\mathbb{R} \cup \{-\infty, +\infty\}) \times (\mathbb{R} \cup \{-\infty, +\infty\}), \oplus, \otimes, (0,0), (1,0))$$

$$(x_1, y_1) \oplus (x_2, y_2) = (x_1 + x_2, y_1 + y_2)$$

$$(x_1, y_1) \otimes (x_2, y_2) = (x_1 x_2, x_1 y_2 + x_2 y_1)$$

Let $\log A$ denote the weighted automaton derived from $A$ by replacing each weight $w \in \mathbb{R}_+$ by $\log w$ and let $\Phi_1(A)$ and $\Phi_2(A)$ denote the weighted automata over the expectation semiring derived from $A$ by replacing each weight $w$ by the pair $(w, 0)$ and $(1, w)$ respectively. The factor automata representing TF, DF and (TF, TF-IDF) statistics satisfy the following identity in the expectation semiring:

$$S_i^{\mathrm{TF,TF-IDF}} = \Phi_1(S_i^{\mathrm{TF}}) \cap \Phi_2(-\log S^{\mathrm{DF}})$$

Here the second term on the right can be recognized as the factor automaton representing IDF statistics. Hence, TF-IDF computation reduces to weighted automata intersection in the expectation semiring. Consider the vector space model defined over factors, i.e. each dimension corresponds to a factor. Inner product computation in this vector space between a query factor automaton $\Phi_1(S_Q^{\mathrm{TF}})$ representing term frequencies over the expectation semiring and each factor automaton $\{S_i^{\mathrm{TF,TF-IDF}} | i = 1, \ldots, n\}$ can be carried out by intersecting the two automata and then performing a single-source shortest-distance computation [14] over the entropy semiring:

$$< \Phi_1(S_Q^{\mathrm{TF}}), S_i^{\mathrm{TF,TF-IDF}} >= s[\Phi_1(S_Q^{\mathrm{TF}}) \cap S_i^{\mathrm{TF,TF-IDF}}]$$

One application for this inner product is the computation of cosine similarity between two spoken documents, e.g. ASR lattices corresponding to a voiced query and a spoken document.

Table 2: Runtime Results: TP Factor Automata vs. Baseline

| Max length | 1 | 2 | 3 | 6 | 10 | all |
|---|---|---|---|---|---|---|
| $\log_{10}$(# factors) | 3.0 | 3.8 | 4.2 | 5.1 | 6.3 | 11.2 |
| Baseline time (s) | 5 | 15 | 32 | 311 | 5413 | - |
| TP FA time (s) | 8 | 13 | 18 | 34 | 53 | 105 |

Table 3: Factor Automata Comparison

| FA Type | UW | TF | TP | DF | TF-IDF |
|---|---|---|---|---|---|
| $\Sigma_i \lvert S_i \rvert$ (M) | 16 | 20 | 21 | 14 | 27 |
| On disk (MB) | 251 | 315 | 324 | 209 | 511 |
| Time (min) | 5 | 8 | 1093 | +2 | +1 |

## 4. Experiments and Discussion

We conducted experiments on the training subset of the Turkish language pack provided by the IARPA Babel program which includes 80 hours of conversational telephone speech. Lattices were generated with a speaker dependent DNN ASR system that was trained on the same data set using IBM's Attila toolkit. All lattices were pruned to a logarithmic beam width of 5.

Estimating document frequencies of single-word factors in a collection of lattices has previously been addressed in [3]. Their recipe for computing probability of occurrence consists of composing the input lattice with a simple finite-state filter that rejects the paths including the target word, computing the total probability of the remaining paths and complementing. Computation is carried out one factor at a time, i.e. factors are enumerated and each one is processed independently. We implemented a generalized version of this recipe which can be used with multi-word factors for our baseline results. Both this baseline algorithm and the factor automata construction algorithms in consideration were implemented using the OpenFst Library [15]. Table 2 gives a runtime comparison between the baseline and the TP factor automata construction algorithm. We randomly selected 100 lattices from our data set (total size: #states + #arcs = 81K, disk size: 1.2MB) and compared the total runtime while changing the maximum factor length with a finite-state length restriction filter [4]. Runtime complexity of the baseline method is exponential in the maximum factor length (or linear in the number of factors) due to the enumeration of factors. Proposed method takes advantage of weighted transducer algorithms to do the computation jointly for all factors.

Table 3 compares the total runtime and storage requirements for various factor automata. For these experiments, we used the entire training set which includes 88K lattices (total size: #states + #arcs = 33M, disk size: 481MB) and a maximum factor length of 3. First column (UW) represents the unweighted factor automata obtained by removing all weights from the input lattices. Storage requirements seem to be comparable for the types of factor automata in consideration. The runtime for the construction of DF factor automaton excludes the time spent for the construction of TP factor automata. Similarly the runtime for the construction of TF-IDF factor automata excludes the time spent for the construction of TF and DF factor automata. Just like TF factor automaton construction [4], TP factor automaton construction algorithm of the previous section is worst case exponential in the input size and may blow up for some input. The runtime reported in Table 3 is dominated by the time spent on a small fraction of input lattices for which the algorithm took several minutes to complete. The difference in the average runtime characteristics calls for further study.

# 5. References


[1] S. Robertson, "Understanding inverse document frequency: On theoretical arguments for idf," *Journal of Documentation*, vol. 60, p. 2004, 2004.

[2] D. Harman, "The history of idf and its influences on ir and other fields," in *Charting a New Course: Natural Language Processing and Information Retrieval*, ser. The Kluwer International Series on Information Retrieval, J. Tait, Ed. Springer Netherlands, 2005, vol. 16, pp. 69–79.

[3] D. Karakos, M. Dredze, K. W. Church, A. Jansen, and S. Khudanpur, "Estimating document frequencies in a speech corpus." in *ASRU*, D. Nahamoo and M. Picheny, Eds. IEEE, 2011, pp. 407–412.

[4] C. Allauzen, M. Mohri, and M. Saraclar, "General indexation of weighted automata: Application to spoken utterance retrieval," in *Proc. HLT-NAACL Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval*, Boston, MA, USA, 2004, pp. 33–40.

[5] D. Can and M. Saraclar, "Lattice indexing for spoken term detection," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 8, pp. 2338 –2347, nov. 2011.

[6] M. Mohri, "Weighted automata algorithms," in *Handbook of Weighted Automata*, ser. Monographs in Theoretical Computer Science. An EATCS Series, M. Droste, W. Kuich, and H. Vogler, Eds. Springer Berlin Heidelberg, 2009, pp. 213–254.

[7] A. Blumer, J. Blumer, A. Ehrenfeucht, D. Haussler, M. T. Chen, and J. Seiferas, "The smallest automaton recognising the subwords of a text," *Theoretical Computer Science*, vol. 40, pp. 31–55, 1985.

[8] M. Crochemore, "Transducers and repetitions," *Theoretical Computer Science*, vol. 45, no. 1, pp. 63–86, 1986.

[9] M. Mohri, P. Moreno, and E. Weinstein, "General suffix automaton construction algorithm and space bounds," *Theoretical Computer Science*, vol. 410, no. 37, pp. 3553 – 3562, 2009.

[10] M. Saraclar and R. Sproat, "Lattice-based search for spoken utterance retrieval," in *Proc. HLT-NAACL*, Boston, MA, USA, 2004, pp. 129–136.

[11] C. Allauzen and M. Riley, "A pushdown transducer extension for the openfst library," in *Proceedings of the 17th international conference on Implementation and Application of Automata*, ser. CIAA'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 66–77.

[12] J. Eisner, "Expectation semirings: Flexible EM for finite-state transducers," in *Proceedings of the ESSLLI Workshop on Finite-State Methods in Natural Language Processing (FSMNLP)*, G. van Noord, Ed., Helsinki, Aug. 2001, extended abstract (5 pages).

[13] C. Cortes, M. Mohri, A. Rastogi, and M. Riley, "On the computation of the relative entropy of probabilistic automata," *Int. J. Found. Comput. Sci.*, vol. 19, no. 1, pp. 219–242, 2008.

[14] M. Mohri, "Semiring frameworks and algorithms for shortest-distance problems," *Journal of Automata, Languages and Combinatorics*, vol. 7, no. 3, pp. 321–350, 2002.

[15] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: A general and efficient weighted finite-state transducer library," in *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, ser. Lecture Notes in Computer Science, vol. 4783. Springer, 2007, pp. 11–23, `http://www.openfst.org`.