Weighted finite-state transducers: the later years

Kyle Gorman
Graduate Center, City University of New York & Google

http://wellformedness.com/courses/fstp/

We are now at nearly a decade into what has been called “deep learning tsunami” (Manning, 2015). Yet weighted finite-state transducers continue to play a crucial role in industrial speech and language technologies.

A battle between two great powers?

- knowledge-based vs. data-driven
- rationalism vs. empiricism
- neats vs. scruffies
- cowboys vs. aliens

Text normalization

Many speech and language technologies map between “written” and “spoken” representations of language. *Text normalization* (Sproat et al., 2001) refers to mappings between pseudo-ideographic representations like **$4.20** to more pronounceable representations like *four dollars and twenty cents*.

Semiotic categories (Ebden and Sproat, 2014)

- Cardinal: **69** → *sixty nine*
- Date: **11/2/1985** → *November second nineteen eighty five*
- Decimal: **23.3** → *twenty three point three*
- Electronic: **kgorman@gc.cuny.edu** → *k gorman at gc dot cuny dot edu*
- Fraction: **2/5** → *two fifths*
- Measure: **12kg** → *twelve kilograms*
- Money: **$5.96** → *five dollars and ninety six cents*
- Ordinal: **69th** → *sixty ninth*
- Roman numeral: **LIV** → *fifty four*
- Telephone: **566-6123** → *five six six, six one two three*
- Time: **11:58** → *eleven fifty eight*

Wikipedia (“written” domain)

The giraffe has an extremely elongated neck, which can be up to **2 m** (**6 ft 7 in**) in length, accounting for much of the animal’s vertical height. Each cervical vertebra is over **28 cm** (**11 in**) long. They comprise **52-54 percent** of the length of the giraffe’s vertebral column, compared with the **27–33 percent** typical of similar large ungulates, including the giraffe’s closest living relative, the okapi.

Wikipedia (“spoken” domain)

The giraffe has an extremely elongated neck, which can be up to *two meters* (*six feet seven inches*) in length, accounting for much of the animal’s vertical height. Each cervical vertebra is over *twenty eight centimeters* (*eleven inches*) long. They comprise *fifty two to fifty four percent* of the length of the giraffe’s vertebral column, compared with the *twenty seven to thirty three percent* typical of similar large ungulates, including the giraffe’s closest living relative, the okapi.

Applications

- In text-to-speech synthesis, the front-end is responsible for providing pronunciations for semiotic classes.
- In automatic speech recognition:
  - the written text used to train language models are converted to spoken form.
  - spoken form transcriptions from the recognizer are converted back to written form (e.g., Shugrina, 2010; Pusateri et al., 2017).
- In information extraction, verbalizations can be used as a canonical form for spoken and the various written forms of dates, times, etc.

Machine learning for text normalization at Google

- Sentence boundary detection (Sproat and Hall, 2014)
- English abbreviation expansion (Roark and Sproat, 2014; Gorman et al., 2021)
- Grapheme-to-phoneme prediction (Jansche, 2014; Rao et al., 2015)
- Russian word stress prediction (Hall and Sproat, 2013)
- Number name generation (Gorman and Sproat, 2016; Ritchie et al., 2019)
- Letter sequence prediction (Sproat and Hall, 2014)
- Homograph disambiguation (Gorman et al., 2018)
- End-to-end research (Ng et al., 2017; Sproat and Jaitly, 2017; Zhang et al., 2019)

But...

- Yet nearly all text normalization is still done with hand-written language-specific grammars, just like 25 years ago (e.g., Sproat, 1996), not with sequence-to-sequence neural networks.
- The required native speaker-cum-computational-linguistic sophistication needed to develop and maintain these grammars is thin on the ground and this is **the** major barrier to internationalization in speech technology.

Outline

- Formalization
- OpenFst and friends
- Some new(ish) WFST algorithms:
  - general-purpose WFST optimization
  - A* shortest string decoding over non-idempotent semirings

Formalization

Figure: The humble gumball machine. (Image credit: Wikimedia Commons.)

State machines

State machines are systems whose behavior can be described solely in terms of a set of *states*—corresponding roughly to “memory”—and arcs, transitions between those states. One familiar example of a state machine—encoded in hardware rather than software—is the old-fashioned gumball machine. Each state of the gumball machine is associated with actions such as

- turning the knob,
- inserting a coin, or
- emitting a gumball.

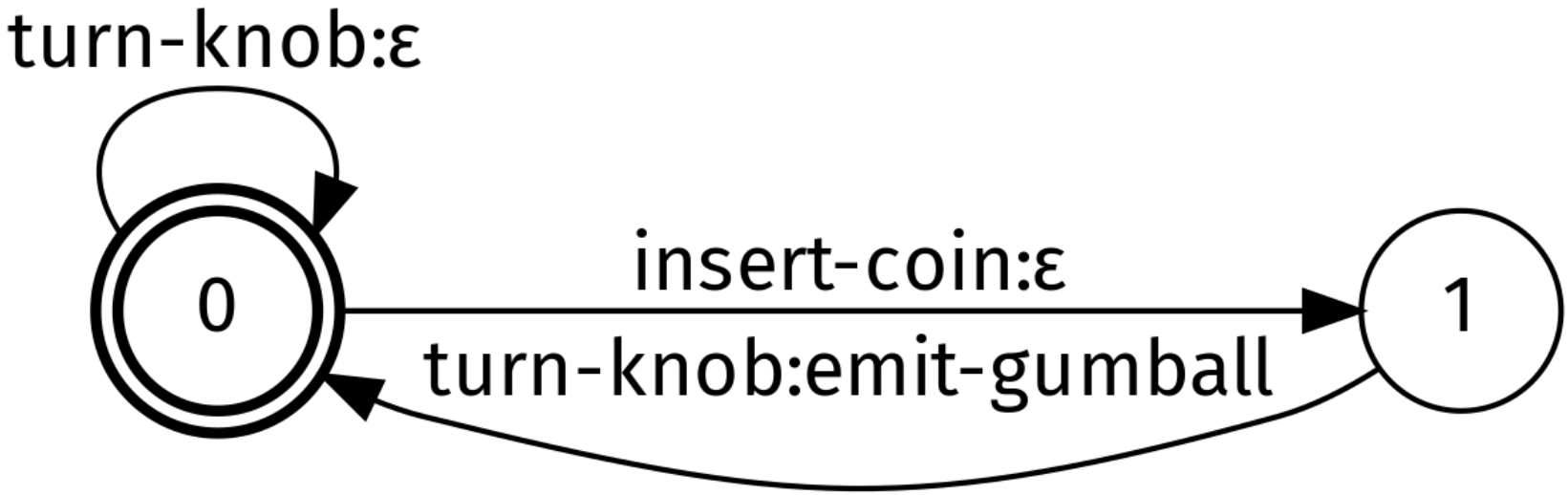


Figure: A state diagram describing a gumball machine; bold vertices indicate initial states and double-struck vertices indicate final states.

Preliminaries

Following the practice used in the OpenFst library, we define one automaton type—a single-source finite-state two-way ϵ-transducer—and derive other types as special cases.

Monoids

A *monoid* is a pair $(\mathbb{K}, \bullet)$ where $\mathbb{K}$ is a set and $\bullet$ is a binary operator over $\mathbb{K}$ with properties of:

- *closure*: $\forall a, b \in \mathbb{K} : a \bullet b \in \mathbb{K}$.
- *associativity*: $\forall a, b, c \in \mathbb{K} : (a \bullet b) \bullet c = a \bullet (b \bullet c)$.
- *identity*: $\exists e \in \mathbb{K} : e \bullet a = a \bullet e = a$.

A monoid:

- is *commutative* in the case that $\forall a, b \in \mathbb{K} : a \bullet b = b \bullet a$.

Semirings

A semiring is a five-tuple $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ such that:

- $(\mathbb{K}, \oplus)$ is a commutative monoid with identity element $\bar{0}$.
- $(\mathbb{K}, \otimes)$ is a monoid with identity element $\bar{1}$.
- $\forall a, b, c \in \mathbb{K} : a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$.
- $\forall a \in \mathbb{K} : a \otimes \bar{0} = \bar{0} \otimes a = \bar{0}$.

A semiring:

- is *zero-sum-free* if non-$\bar{0}$ elements cannot sum to $\bar{0}$; that is, $\forall a, b \in \mathbb{K} : a \oplus b$ implies $a = b = \bar{0}$.
- is *idempotent* if $\oplus$ is idempotent; that is, $\forall a \in \mathbb{K} : a \oplus a = a$.
- has the *path property* if $\forall a, b \in \mathbb{K} : a \oplus b \in \{a, b\}$.
- that has the path property is also idempotent.

Order

- The *natural order* of an idempotent semiring is a boolean operator $\preceq$ such that $\forall a, b \in \mathbb{K} : a \preceq b$ if and only if $a \oplus b = a$.
- In a semiring with the path property, the natural order is a *total* order; that is, $\forall a, b \in \mathbb{K}$, either $a \preceq b$ or $b \preceq a$.
- A semiring:
  - is *monotonic* if $\forall a, b, c \in \mathbb{K}$, $a \preceq b$ implies
    - $a \oplus c \preceq b \oplus c$,
    - $a \otimes c \preceq b \otimes c$, and
    - $c \otimes a \preceq c \otimes b$.
  - is *negative* if $\bar{1} \preceq \bar{0}$.
- In a monotonic negative semiring, $\forall a, b \in \mathbb{K} : a \preceq \bar{0}$ and $a \oplus b \preceq b$.

	$\mathbb{K}$	$\oplus$	$\otimes$	$\bar{0}$	$\bar{1}$	$\preceq$
Plus-times	$\mathbb{R}_+$	$+$	$\times$	0	1	$\geq$
Max-times	$\mathbb{R}_+$	$\max$	$\times$	0	1	$\geq$
Log	$\mathbb{R} \cup \{-\infty, +\infty\}$	$\oplus_{\log}$	$+$	$+\infty$	0	$\leq$
Tropical	$\mathbb{R} \cup \{-\infty, +\infty\}$	$\min$	$+$	$+\infty$	0	$\leq$

Table: Common monotonic negative semirings; $a \oplus_{\log} b = -\ln(e^{-a} + e^{-b})$.

Weighted finite-state transducers

A *weighted finite-state transducer* (WFST) is defined by a six-tuple $(Q, s, \Sigma, \Phi, \omega, \delta)$ and a semiring $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ where:

- Q is a finite set of states.
- $s \in Q$ is the *initial* or *start* state.
- Σ is the *input alphabet*.
- Φ is the *output alphabet*.
- $\omega \subseteq Q \times \mathbb{K}$ is the *final weight function*.
- $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Phi \cup \{\epsilon\}) \times \mathbb{K} \times Q$ is the *transition relation*.

Paths

A *path* through an transducer p is a 4-tuple consisting of:

- a *state sequence* $q[p] = q_1, q_2, \ldots, q_n, \in Q^n$
- a *weight sequence* $k[p] = k_1, k_2, \ldots, k_n \in \mathbb{K}^n$
- an *input string* $x[p] = x_1, x_2, \ldots, x_n \in \left(\Sigma \cup \{\epsilon\}\right)^n$
- an *output string* $y[p] = y_1, y_2, \ldots, y_n \in \left(\Phi \cup \{\epsilon\}\right)^n$

subject to the constraint that $\forall i \in [1, n] : \left(q_i, x_i, y_i, k_i, q_{i+1}\right) \in \delta$.

Complete paths

- A state $q \in Q$ is *final* if $\omega(q) \neq \bar{0}$.
- Let $F = \{q \mid \omega(q) \neq \bar{0}\}$ denote the set of final states.
- A path is *complete* if:
  - $(s, x_1, y_1, k_1, q_1) \in \delta$.
  - $q_n \in F$.
- The weight of a complete path is given by

$$\bar{k} = \left(\bigotimes_{k_i \in k[p]} k_i \right) \otimes \omega(q_n).$$

Properties

A transducer is:

- *acyclic* if there exists a *topological ordering*, an ordering of the states such that if there is a transition from state q to r then q is ordered before r, and *cyclic otherwise*.
- *deterministic* if for each state $q \in Q$, there is at most one transition with a given input label $x \in (\Sigma \cup \{\epsilon\})$ from that state, and *non-deterministic* otherwise.

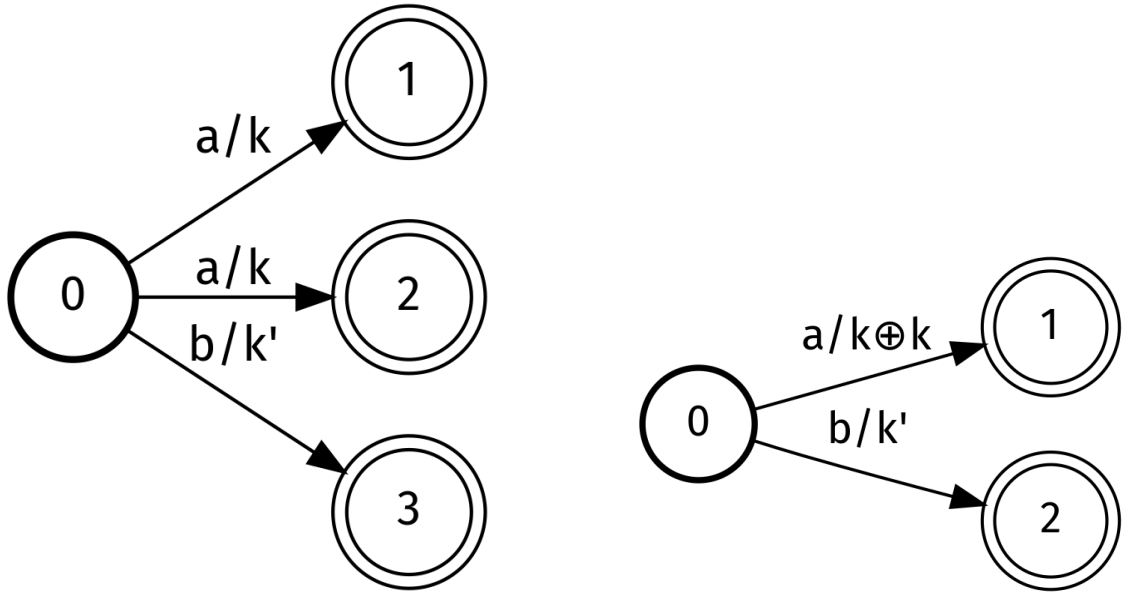


Figure: State diagrams showing a weighted NFA (left) and an equivalent DFA (right).

Weighted transduction

A transducer is said to *map* or *transduce* from string x to string y with weight $\bar{k}$ just in case there exists a path p with input string x, output string y, and complete path weight $\bar{k}$.

Special cases

- An unweighted transducer—one where all weights are either $\bar{0}$ or $\bar{1}$—corresponds to a *rational relation*.
- A weighted acceptor—one where all input and output labels match—corresponds to a weighted (e.g., probabilistic) distribution over a *regular language*.
- An unweighted acceptor corresponds to a regular language.

OpenFst and friends

OpenFst (Allauzen et al., 2007)

OpenFst is a open-source C++17 library for weighted finite state transducers developed at Google. Among other things, it is used in:

- automatic speech(-to-text) recognizers (e.g., Kaldi and many commercial products).
- text-to-speech synthesizers (as part of the “front-end”).
- input method engines (e.g., mobile text entry systems).
- many other kinds of text hacking.

Features

- One serialization format (`.fst`) is shared across all OpenFst and OpenGrm libraries.
- FSTs can be compacted; e.g., unweighted string acceptors can be stored as integer arrays.
- Collections of FSTs can be stored in FST archives (`.far`), a shardable key-value store.

OpenFst design

There are (at least) four layers to OpenFst:

- a C++ template/header library in `<fst/*.h>`
- a C++ “scripting” library in `<fst/script/*.{h,cc}>`
- CLI programs in `/usr/local/bin/*`
- a Python extension module `pywrapfst`

OpenGrm

- Baum-Welch (Gorman et al., 2021): CLI tools and libraries for performing expectation maximization on WFSTs
- NGram (Roark et al., 2012): CLI tools and libraries for building conventional n-gram language models
- Pynini (Gorman, 2016; Gorman and Sproat, 2021): Python extension module for WFST grammar development
- SFst (Allauzen and Riley, 2018): CLI tools and libraries for building *stochastic FSTs*
- Thrax (Roark et al., 2012): DSL-based compiler for WFST grammar development

General-purpose WFST optimization

Optimal for what?

There are many ways a WFST might be said to be optimal. For instance, a WFST could be optimal for:

- composition efficiency (i.e., by eliminating internal ϵ-labels or moving them later along paths).
- footprint in memory (i.e., by reducing the number of states and arcs).
- cache utilization or other application-specific use patterns.

Minimality

- An automaton is *minimal* if it expresses its (weighted) language or relation using the minimal number of states.
- Efficient algorithms exist for minimizing deterministic automata (e.g., Mohri, 2000).
- However, finding an equivalent deterministic automaton for an arbitrary WFST can be computationally expensive if not impossible.

Implementation

- Pynini `Fst` objects have a destructive instance method `optimize`.
- Thrax has a function `Optimize`.

Both share the same C++ template implementation in `optimize.h`.

Preprocessing

We first apply ϵ-removal (Mohri, 2002a) if the input WFST is not known to be ϵ-free.

Optimizing acceptors

Not all acceptors are determinizable.

- Acceptors which do not have weights other than $\bar{0}$ and/or $\bar{1}$ along their cycles—as well as acylic and unweighted acceptors—are determinizable over a wide variety of semirings (Mohri, 2009). We then apply determinization and minimization if the acceptor is not known to be deterministic.
- However, it is difficult to determine whether determinization will even terminate for cyclic weighted non-deterministic acceptors (Allauzen and Mohri, 2003). Therefore, we heuristically apply determinization and minimization to such acceptors *viewed as unweighted*. This is guaranteed to halt.

Optimizing transducers

Similarly, not all transducers are determinizable.

- Even transducers without weighted cycles may be non-functional. Therefore, we heuristically apply determinization and minimization to such transducers *viewed as acceptors*. This is guaranteed to halt.
- For cyclic weighted transducers, we heuristically apply determinization and minimization to such transducers *viewed as unweighted acceptors*. This is also guaranteed to halt.

Postprocessing

When an weighted cyclic automaton is heuristically optimized as if it was unweighted, we also apply arc-sum mapping as a post-processing step. This eliminates trivial (i.e., same-state) cases of non-determinism due to identically labeled arcs with different weights leaving the same state, which may be introduced during heuristic determinization-minimization.

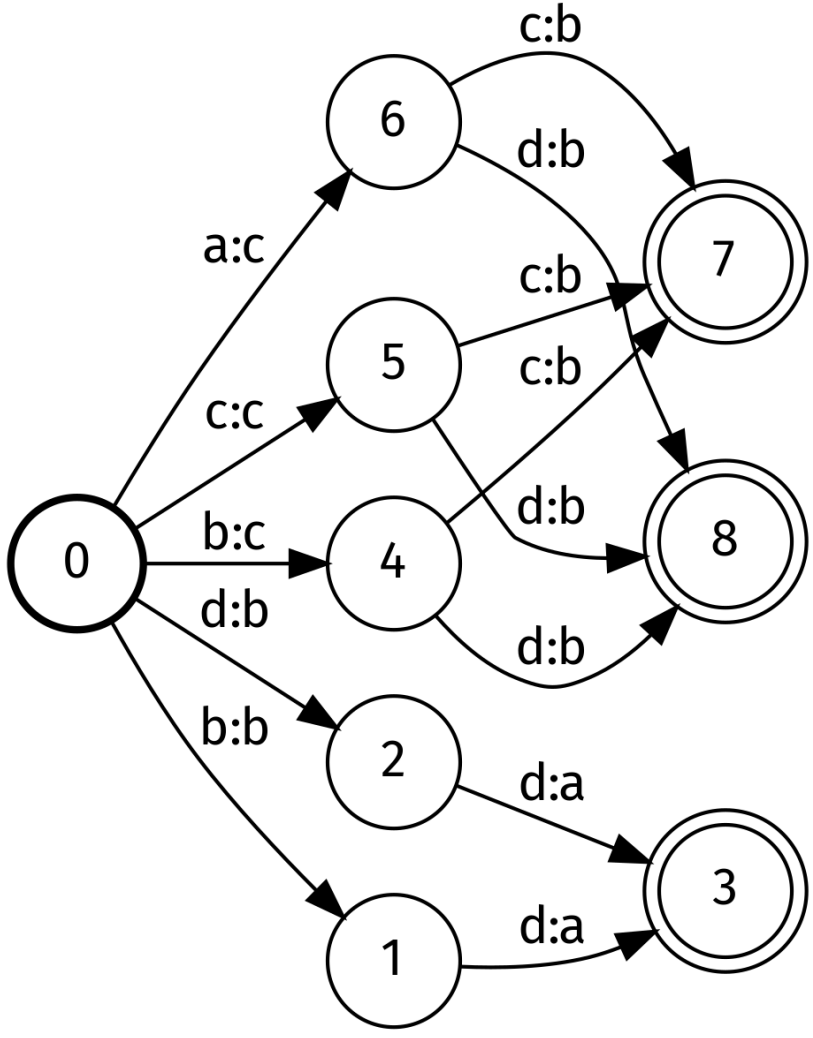


Figure: Finite transducer before optimization.

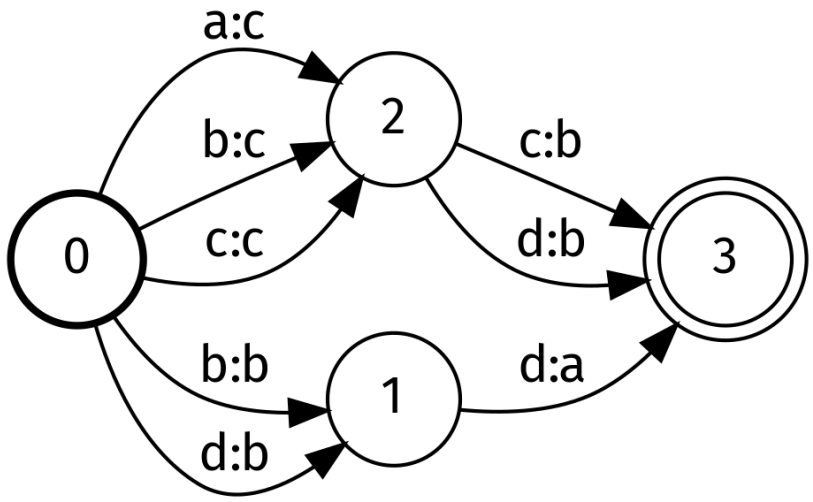


Figure: Equivalent finite transducer after optimization.

Evaluation

- We apply the above algorithm to a sample of 700 speech recognition word lattices derived from Google Voice Search traffic, lattices previously used Mohri and Riley (2015) to evaluate related algorithms.
- Each lattice path represents a single hypothesis transcription from a production-grade automatic speech recognizer.
- These lattices are acyclic and ϵ-free, non-deterministic, and weighted, and thus the algorithm above is guaranteed to produce a deterministic, minimal, ϵ-free acceptor.

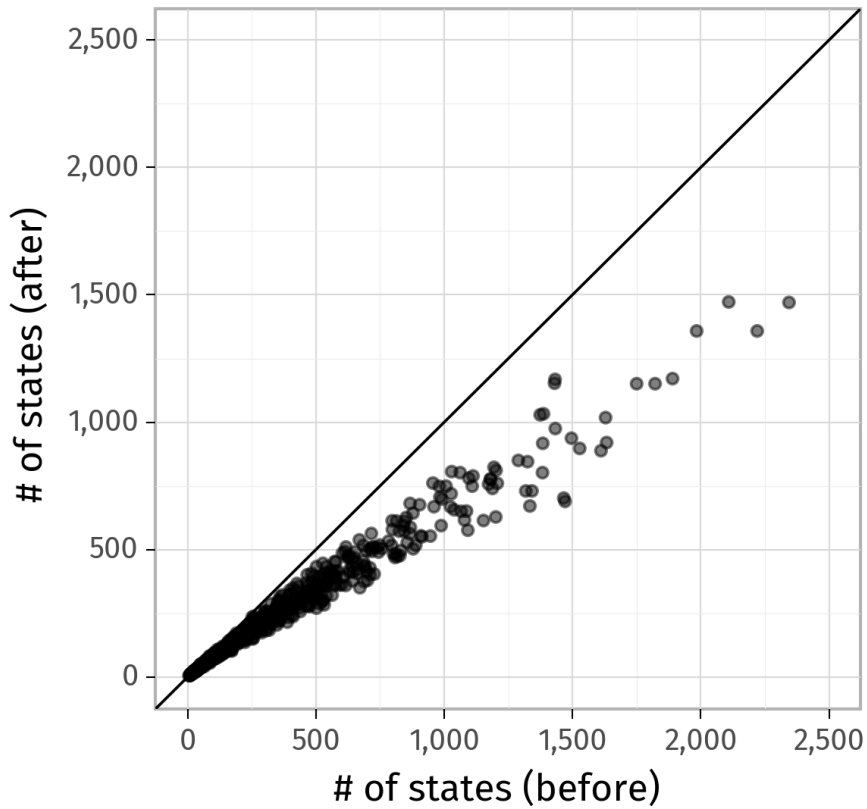


Figure: Word lattice optimization with the proposed algorithm. The *x*-axis shows the number of states before optimization; the *y*-axis shows the number of states after optimization.

Results

- Optimization substantially reduces the number of states, particularly for the larger lattices.
- The post-optimization “after” automaton is never larger than the “before” automaton.

Related work

An earlier version of this algorithm was proposed by Allauzen et al. (2004). The above evaluation is reported by Gorman and Sproat (2021, §4.5).

A* shortest string decoding for non-idempotent semirings

Motivations

Various circumstances force us to build WFST models we cannot decode efficiently or exactly due to restrictions on shortest-path algorithms. We attempt to remedy these restrictions.

Three types of expectation maximization

- In *vanilla EM* (Dempster et al., 1977), we collect counts in semirings isomorphic to Plus-Times.
- In *Viterbi EM* (Brown et al., 1993, 293), we collect counts in semirings isomorphic to Max-Times.
- In *lateen EM* (Spitkovsky et al., 2011), we alternate between vanilla and Viterbi EM according to some training schedule.

Yet there is no way to compute the shortest path in semirings isomorphic to Plus-Times.

Preliminaries

Without loss of generality, we consider single-source ϵ-free acyclic acceptors, using $z[p] = x[p] = y[p]$ to denote the string of a path p.

Shortest distance

Let $P_{q \to r}$ be the set of all paths from q to r where $q, r \in Q$. Then:

- the *forward shortest distance* $\alpha \subseteq Q \times \mathbb{K}$ maps from a state $q \in Q$ to the $\oplus$-sum of the $\otimes$-product of the weights of all paths from s to q:

$$\alpha(q) = \bigoplus_{p \in P_{s \to q}} \bigotimes_{k_i \in k[p]} k_i.$$

- the *backwards shortest distance* $\beta \subseteq Q \times \mathbb{K}$ maps from a state $q \in Q$ to the $\oplus$-sum of $\otimes$-product of the weights of all paths from q to any final state:

$$\beta(q) = \bigoplus_{f \in F} \left(\bigoplus_{p \in P_{q \to f}} \bigotimes_{k_i \in k[p]} k_i \otimes \omega(f) \right).$$

Shortest path

- The *total shortest distance* through an automaton is given by $\beta(s)$.
- The *shortest path* through an automaton is a complete path whose weight is equal to $\beta(s)$.

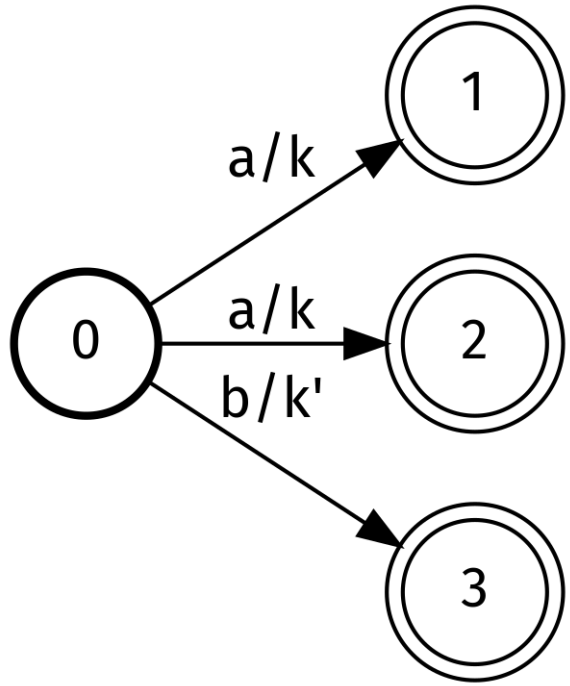


Figure: Automata over non-idempotent semirings need not have a shortest path. Consider the figure above. If $k \oplus k \preceq k < k'$, then the total shortest distance is $k \oplus k$, which need not correspond to any one path.

Shortest string

Let P_z be a set of paths with string $z \in \Sigma^*$, and let the weight of P_z be

$$\sigma(z) = \bigoplus_{p \in P_z} \bar{k}[p].$$

Then a *shortest string* z is one such that $\forall z' \in \Sigma^*, \sigma(z) \preceq \sigma(z')$.

Lemma I

Lemma

In an idempotent semiring, a shortest path's string is also a shortest string.

Proof

Let p be a shortest path. By definition, $\bar{k}[p] \preceq \bar{k}[p']$ for all complete paths p'. It follows that

$$\forall z' \in \Sigma^* : \sigma(z[p]) = \bigoplus_{p \in P_z} \bar{k}[p] \preceq \sigma(z'[p']) = \bigoplus_{p' \in P_z} \bar{k}[p']$$

so $z[p]$ is the shortest string.

Companion semirings

The *companion semiring* of a monotonic negative semiring $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ with a total order $\preceq$ is the semiring $(\mathbb{K}, \widehat{\oplus}, \otimes, \bar{0}, \bar{1})$ where $\widehat{\oplus}$ is the minimum binary operator for $\preceq$:

$$a \mathbin{\widehat{\oplus}} b = \begin{cases} a & \text{if } a \preceq b \\ b & \text{otherwise} \end{cases}$$

For example, the tropical semiring

$$(\mathbb{R} \cup \{-\infty, +\infty\}, \min, +, +\infty, 0)$$

is the companion semiring for the log semiring

$$(\mathbb{R} \cup \{-\infty, +\infty\}, \oplus_{\log}, +, +\infty, 0).$$

Lemma II

Lemma

In a DFA over a monotonic semiring, a shortest string is the string of a shortest path in that DFA viewed over the corresponding companion semiring.

Proof

Determinism implies that for all complete path p', $\bar{k}[p'] = \sigma(z[p'])$. Let z be the shortest string in the DFA and p the unique path admitting the string z. Then

$$\bar{k}[p] = \sigma(z) \preceq \sigma(z[p']) = \bar{k}[p']$$

for any complete path p'. Hence

$$\bar{k}[p] = \overbrace{\bigoplus_{p' \in P_{s \to F}} \bar{k}[p']}.$$

Thus p is a shortest path in the DFA viewed over the companion semiring.

Shortest-first search

Dijkstra's (1959) algorithm is an example of a *shortest-first* search strategy appropriate for idempotent semirings. At every iteration, the algorithm explores the state q which minimizes $\alpha(q)$, the shortest distance from the initial state s to q, until all states have been visited.

A* search

In the variant known as A* search (Hart et al., 1968), search priority is instead a function of $F \subseteq Q \times \mathbb{K}$, known as the *heuristic*, which gives an estimate of the weight of paths from some state to a final state. At every iteration, A* instead explores the state q which minimizes $\alpha(q) \otimes F(q)$.

Dijkstra again

Then, Dijkstra's algorithm is just a special case of A* search using $F = \bar{1}$.

Heuristics

A heuristic is:

- *admissible* if it never overestimates the shortest distance to a state. That is, it is admissible if $\forall q \in Q : F(q) \preceq \beta(q)$.
- *consistent* if it never overestimates the cost of reaching a successor state. That is, it is consistent if $\forall q, r \in Q$ such that $F(q) \preceq k \otimes F(r)$ if $(q, z, k, r) \in \delta$, i.e., if there is a transition from q to r with some label z and weight k.

If F is *admissible* and *consistent*, A* search is guaranteed to find a shortest path (if one exists) after visiting all states such that $F(q) \preceq \beta(s)$ (Hart et al., 1968, 104f.).

Preliminaries

Consider an acyclic, ϵ-free WFSA over a monotonic negative semiring $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ with total order $\preceq$ for which we wish to find the shortest string. The same WFSA can also be viewed as a WFSA over the corresponding companion semiring $(\mathbb{K}, \widehat{\oplus}, \otimes, \bar{0}, \bar{1})$, and we denote by $\widehat{\beta}$ the backward shortest-distance over this companion semiring.

Proof I

Theorem

The backwards shortest distance of an WFSA over a monotonic negative semiring is an admissible heuristic for the A* search over its companion semiring.

Proof

In a monotonic negative semiring, the $\oplus$-sum of any n terms is upper-bounded by each of the n terms and hence by the $\widehat{\oplus}$-sum of these n terms. It follows that

$$\beta(q) = \bigoplus_{p \in P_{q \to F}} \bar{k}[p] \preceq \widehat{\bigoplus_{p \in P_{q \to F}}} \bar{k}[p] = \widehat{\beta}(q)$$

showing that $F = \beta$ is an admissible heuristic for $\widehat{\beta}$.

Proof II

Theorem

The backwards shortest distance of an WFSA over a monotonic negative semiring is a consistent heuristic for the A* search over its companion semiring.

Proof

We again use the property that an $\oplus$-sum of any n terms is upper-bounded by any of these terms. If (q, z, k, r) be a transition in δ

$$\beta(q) = \bigoplus_{p \in P_{q \to F}} \bar{k}[p] = \bigoplus_{(q,z',k',r') \in \delta} k' \otimes \beta(r') \preceq k \otimes \beta(r)$$

showing that $F = \beta$ is a consistent heuristic.

Naïve algorithm

A naïve algorithm suggests itself. Given a non-deterministic WFSA over the monotonic negative semiring $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$:

- apply determinization to obtain an equivalent DFA.
- compute β_d, the DFA's backwards shortest distance.
- perform A* search using β_d as the heuristic.

Exponential blowup

Determinization has an exponential worse-case complexity in time and space and is often prohibitive in practice. Yet determinization—and the computation of elements of β_d—only need to be performed for states **actually visited** during search.

Our algorithm

Let β_n denote backwards shortest distance over a non-deterministic WFSA over the monotonic negative semiring $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$. Then:

- compute β_n over $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$.
- lazily determinize the WFSA (Mohri, 1997), lazily computing β_d from β_n over $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$
- perform A* search using β_d as the heuristic over the companion semiring $(\mathbb{K}, \widehat{\oplus}, \otimes, \bar{0}, \bar{1})$.

Evaluation

We search for the shortest string over a sample of 700, acyclic, ϵ-free non-deterministic WFSA word lattices derived from Google Voice Search traffic. For this, we use the OpenGrm-BaumWelch command-line tool `baumwelchdecode` to implement the above algorithm over the log semiring, with the tropical semiring as the companion semiring.

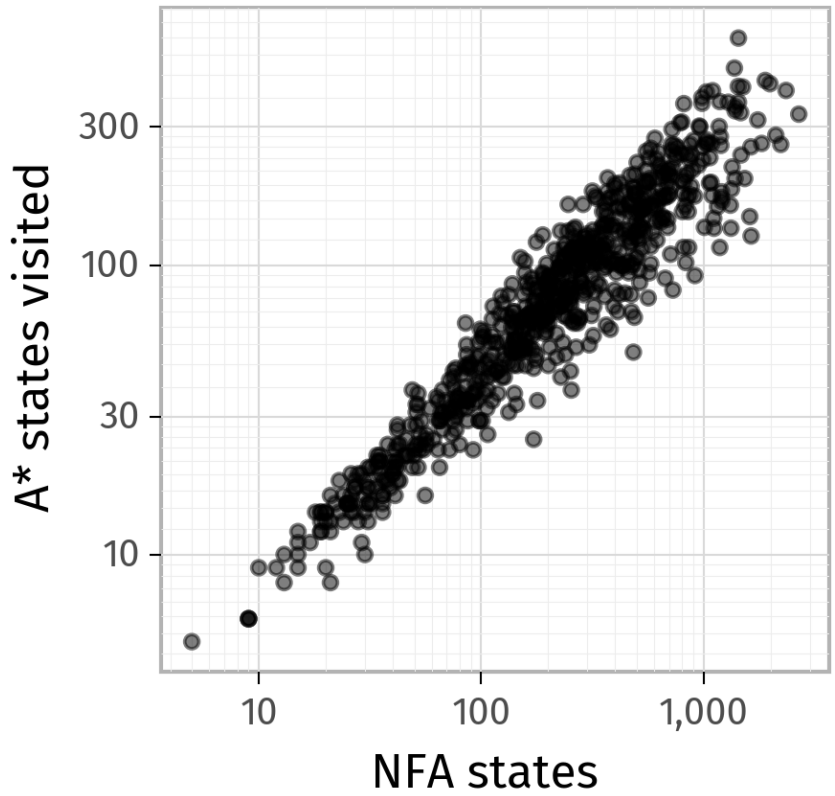


Figure: Word lattice decoding with the proposed algorithm. The *x*-axis shows the number of states in each word lattice NFA; the *y*-axis shows the number of states visited by A* decoding. Both axes are log scale.

Results

- The relationship between the size of the NFA and the number of DFA states visited by the proposed decoding method appears roughly monomial (i.e., log-log linear).
- The size of the full DFA was measured by applying the OpenFst command-line tool `fstdeterminize` to the lattices, which produces an approximately 7x increase in the size of the lattices.
- From this we infer that the proposed heuristic substantially reduces the number of DFA states that are visited.

Applications

Single shortest string over non-idempotent semirings can be used for exact decoding of:

- interpolated (e.g., Jelinek et al., 1983) language models of the form

$$\hat{P}(w \mid h) = \lambda_h \tilde{P}(w \mid h) + (1 - \lambda_h)\hat{P}(w \mid h').$$

- "decipherment" models (e.g., Knight et al., 2006) of the form

$$\hat{P}(p \mid c) \propto P(p)P(c \mid p)$$

  trained with classic expectation maximization.

Related work

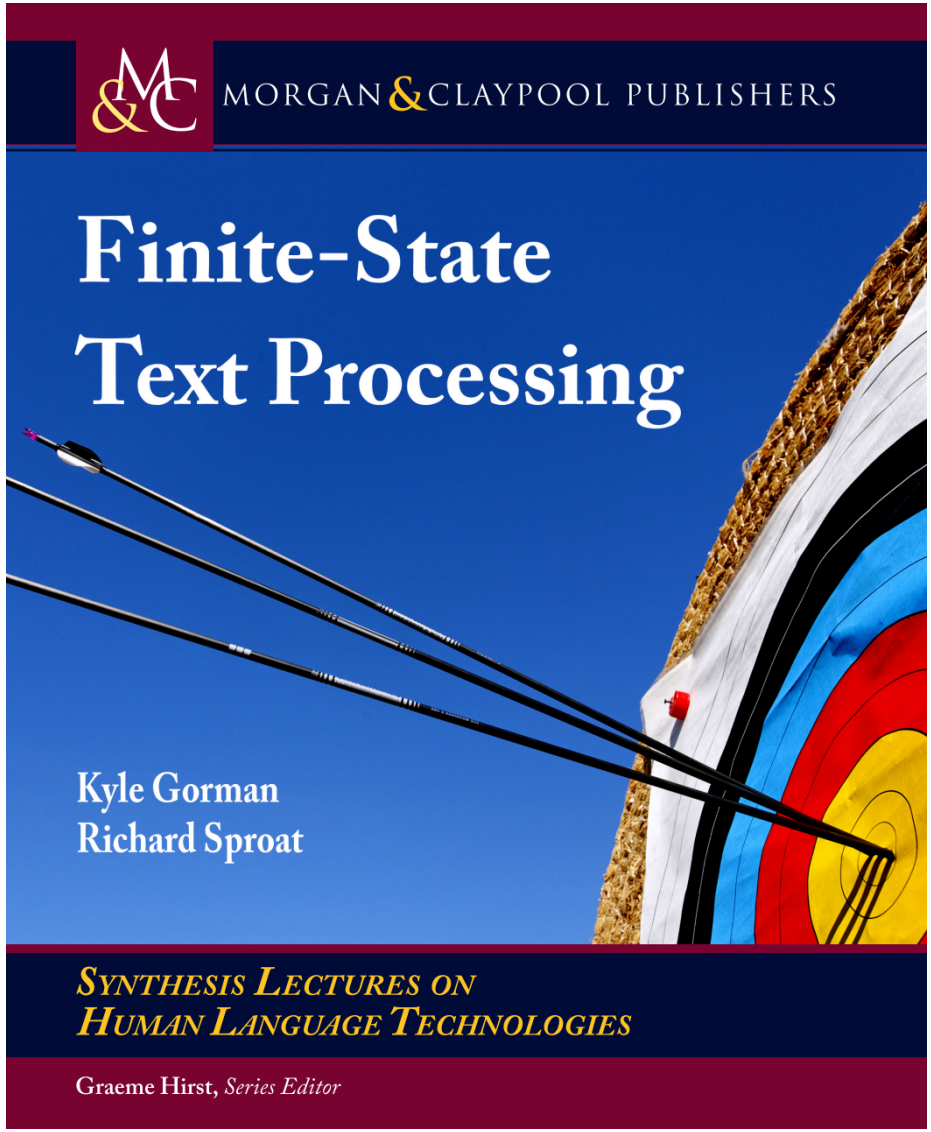
Mohri and Riley (2002) describe a related algorithm for finding the n-best strings over an idempotent finite-state automaton. Much like the algorithm proposed here, they use A* search and on-the-fly determinization; however, they do not consider non-idempotent semirings.

Questions?

Acknowledgments

Thanks to:

- Kevin Knight and Richard Sproat
- Cyril Allauzen, Sasha Gutkin, Brian Roark, Christo Kirov, Jeffrey Sorenson, Lawrence Wolf-Sonkin
- Jillian Chang, Jackson Lee, Michael McAuliffe, Arya McCarthy


MORGAN & CLAYPOOL PUBLISHERS
Finite-State Text Processing
Kyle Gorman
Richard Sproat
SYNTHESIS LECTURES ON HUMAN LANGUAGE TECHNOLOGIES
Graeme Hirst, Series Editor

Further reading

- Gorman and Sproat, 2021: introduces WFST text processing in Python
- Mohri, 2009: reviews major WFST algorithms
- Mohri, 2002b: discusses shortest-distance and shortest-path algorithms

More information

openfst.org

opengrm.org

baumwelch.opengrm.org

ngram.opengrm.org

pynini.opengrm.org

thrax.opengrm.org

Backup slides

Hybridization

- Filtration (Ng et al., 2017; Sproat and Jaitly, 2017; Zhang et al., 2019; Pusateri et al., 2017)
- Data augmentation (Schwartz et al., 2019; Lane and Bird, 2020)
- Distillation (Weiss et al., 2018; Suresh et al., 2021)
- Weighting (Rastogi et al., 2016; Lin et al., 2019)

CUDAfication

- WFST algorithms on CUDA (Argueta and Chiang, 2017, 2018)
- Decoder graphs on CUDA (Chen et al., 2018; Fukunaga et al., 2019; Braun et al., 2020)

References I

C. Allauzen and M. Mohri. Efficient algorithms for testing the twins property. *Journal of Automata, Languages and Combinatorics*, 8(2): 117–144, 2003.

C. Allauzen and M. Riley. Algorithms for weighted finite automata with faillure transitions. In *Proceedings of the 20th International Conference on Implementation and Application of Automata*, pages 46–58, 2018.

C. Allauzen, M. Mohri, M. Riley, and B. Roark. A generalized construction of integrated speech recognition transducers. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 761–764, 2004.

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 12th International Conference on Implementation and Application of Automata*, pages 11–23, 2007.

References II

A. Argueta and D. Chiang. Decoding with finite-state transducers on GPUs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1044–1052, 2017.

A. Argueta and D. Chiang. Composing finite state transducers on GPUs. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2697–2705, 2018.

H. Braun, J. Luitjens, R. Leary, T. Kaldewey, and D. Povey. GPU-accelerated Viterbi exact lattice decoder for batched online and offline speech recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 7874–7878, 2020.

References III

P. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

Z. Chen, J. Luitjens, H. Xu, Y. Wang, D. Povey, and S. Khudanpur. A GPU-based WFST decoder with exact lattice generation. In *Proceedings of INTERSPEECH*, pages 2212–2216, 2018.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

P. Ebden and R. Sproat. The Kestrel TTS text normalization system. *Natural Language Engineering*, 21:1–21, 2014.

References IV

D. Fukunaga, Y. Tanaka, and Y. Kageyama. GPU-based WFST decoding with extra large language model. In *Proceedings of INTERSPEECH*, pages 3815–3819, 2019.

K. Gorman. Pynini: a Python library for weighted finite-state grammar compilation. In *ACL Workshop on Statistical NLP and Weighted Automata*, pages 75–80, 2016.

K. Gorman and R. Sproat. Minimally supervised models for number normalization. *Transactions of the Association for Computational Linguistics*, 4:507–519, 2016.

K. Gorman and R. Sproat. *Finite-State Text Processing*. Morgan & Claypool, 2021.

References V

K. Gorman, G. Mazovetskiy, and V. Nikolaev. Improving homograph disambiguation with supervised machine learning. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, pages 1349–1352, 2018.

K. Gorman, C. Kirov, B. Roark, and R. Sproat. Structured abbreviation expansion in context. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 995–1005, 2021.

K. Hall and R. Sproat. Russian stress prediction using maximum entropy ranking. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 879–883, 2013.

P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimal cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

References VI

M. Jansche. Computer-aided quality assurance of an Icelandic pronunciation dictionary. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 2111–2114, 2014.

F. Jelinek, L. R. Bahl, and R. L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 5:179–190, 1983.

K. Knight, A. Nair, N. Rashod, and K. Yamada. Unsupervised analysis for decipherment problems. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 499–506, 2006.

W. Lane and S. Bird. Bootstrapping techniques for polysynthetic morphological analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6652–6661, 2020.

References VII

C.-C. Lin, H. Zhu, M. R. Gormley, and J. Eisner. Neural finite-state transducers: beyond rational relations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 272–283, 2019.

C. D. Manning. Last words: computational linguistics and deep learning. *Computational Linguistics*, 41(4):701–707, 2015.

M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

M. Mohri. Minimization algorithms for sequential transducers. *Journal of Automata, Languages and Combinatorics*, 234(1–2):177–201, 2000.

M. Mohri. Generic epsilon-removal and input epsilon-normalization algorithms for weighted transducers. *International Journal of Computer Science*, 13(1):129–143, 2002a.

References VIII

M. Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3): 321–350, 2002b.

M. Mohri. Weighted automata algorithms. In M. Droste, W. Kuich, and H. Vogler, editors, *Handbook of weighted automata*, pages 213–254. Springer, New York, 2009.

M. Mohri and M. Riley. An efficient algorithm for the n-best-strings problem. In *7th International Conference on Spoken Language Processing*, pages 1313–1316, 2002.

M. Mohri and M. D. Riley. On the disambiguation of weighted automata. In *Proceedings of the 20th International Conference on Implementation and Application of Automata*, pages 263–278, 2015.

A. H. Ng, K. Gorman, and R. Sproat. Minimally supervised written-to-spoken text normalization. In *ASRU*, pages 665–670, 2017.

References IX

E. Pusateri, B. R. Ambati, E. Brooks, O. Platek, D. McAllaster, and V. Nagesha. A mostly data-driven approach to inverse text normalization. In *Proceedings of INTERSPEECH*, pages 2784–2788, 2017.

K. Rao, F. Peng, H. Sak, and F. Beaufays. Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *ICASSP*, pages 4225–4229, 2015.

P. Rastogi, R. Cotterell, and J. Eisner. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 623–633, 2016.

S. Ritchie, R. Sproat, K. Gorman, D. van Esch, C. Schallhart, B. Nikos, B. Brard, J. F. Mortensen, M. Holt, and E. Mahon. Unified verbalization for speech recognition & synthesis across languages. In *INTERSPEECH*, pages 3530–3534, 2019.

References X

B. Roark and R. Sproat. Hippocratic abbreviation expansion. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369, 2014.

B. Roark, R. Sproat, C. Allauzen, M. Riley, J. Sorensen, and T. Tai. The OpenGrm open-source finite-state grammar software libraries. In *Proceedings of the ACL 2012 System Demonstrations*, pages 61–66, 2012.

L. Schwartz, E. Chen, B. Hunt, and S. L. Schreiner. Bootstrapping a neural morphological analyzer for St. Lawrence Island Yupik from a finite-state transducer. In *Proceedings of the 3rd Workshop on the Use of Computational Methods in the Study of Endangered Languages Volume 1 (Papers)*, pages 87–96, 2019.

References XI

M. Shugrina. Formatting time-aligned ASR transcriptions for readability. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 198–206, 2010.

V. I. Spitkovsky, H. Alshawi, and D. Jurafsky. Lateen EM: unsupervised training with multiple objectives, applied to dependency grammar induction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1269–1280, 2011.

R. Sproat. Multilingual text analysis for text-to-speech synthesis. *Natural Language Engineering*, 2(4):369–380, 1996.

R. Sproat and K. Hall. Applications of maximum entropy rankers to problems in spoken language processing. In *INTERSPEECH*, pages 761–764, 2014.

References XII

R. Sproat and N. Jaitly. An RNN model of text normalization. In *INTERSPEECH*, pages 754–758, 2017.

R. Sproat, A. W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards. Normalization of non-standard words. *Computer Speech & Language*, 15:287–333, 2001.

A. T. Suresh, B. Roark, M. Riley, and V. Schogol. Approximating probabilistic models as weighted finite automata. *Computational Linguistics*, 47(2): 221–254, 2021.

G. Weiss, Y. Goldberg, and E. Yahav. Extracting automata from recurrent neural networks using queries and counterexamples. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5247–5256, 2018.

References XIII

H. Zhang, R. Sproat, A. H. Ng, F. Stahlberg, X. Peng, K. Gorman, and B. Roark. Neural models of text normalization for speech applications. *Computational Linguistics*, 45(2):293–337, 2019.