Deterministic Letter-to-Sound Transduction in the Taxi/Grimm Corpus Indexing System

Bryan Jurish

moocow@bbaw.de

Berlin-Brandenburgische Akademie der Wissenschaften
Jägerstrasse 22/23 · 10117 Berlin · Germany

22 December, 2006

The Big Picture

The Big Picture

- Grimm Quotation Evidence Corpus
- Taxi Corpus Indexing System

Letter-to-Sound Conversion

- Festival LTS Rulesets
- Failed Attempts

  . . . or: "Why `lextools` *Won't Cut The Butter"*
- LTS Transducer Generation

Results

- Morphological Coverage

Demonstration

- Taxi/Grimm HTTP Query Server

Grimm Corpus

Sources

- *Deutsches Wörterbuch* originally by Jacob and Wilhelm Grimm
- SGML sources provided by the Universität Trier
- Conversion to XML at the BBAW

Quotation Evidence Corpus

- Verse quotations encoded in sources

382,766	verse quotations
6,581,509	tokens
557,271	distinct orthographic types

- Prose quotation extraction work in progress

Grimm Corpus: Problems

Historical Orthography

- Variant orthographical conventions
- Disparate sources
  (incl. Walther von der Vogelweide, *ca.* A.D. 1200)
- Not supported by conventional analysis tools
  (TAGH, moot, etc.)

Examples

- "fröhlich" $\rightsquigarrow$ *frölich, fröhlich, vrœlich, frœlich, fr$\overset{e}{o}$lich, fr$\overset{e}{o}$hlich, vrölich, fröhlig, frölig, . . .*
- "herzenleid" $\rightsquigarrow$ *hertzenleid, herzenleid, herzenleit, hertzenleyd, hertzenleidt, herzenlaid, hertzenlaid, hertzenlaidt, hertzenlaydt, herzenleyd, . . .*

Historical Orthography Workaround

Intuitions

- Pronunciation endures longer than orthography
- Phonetic form is a better indicator of lexical identity than orthographic form

Workaround Idea

- Map each orthographic type to a phonetic form
  - Letter-to-Sound (LTS) Translation Module
  - ideally, as a finite-state transducer (FST)
- Adapt synchronically oriented tools to operate on phonetic forms
  - First step: extend known analyses to phonetically identical types: extensional join

Taxi Indexing System

TAXI: Another XML Index

- Abstract index for XML-structured text corpora
- `MySQL` relational database for backend storage

Build Subsystem

- Converts raw SGML Sources to TAXI XML
- Entity resolution, tokenization, document analysis

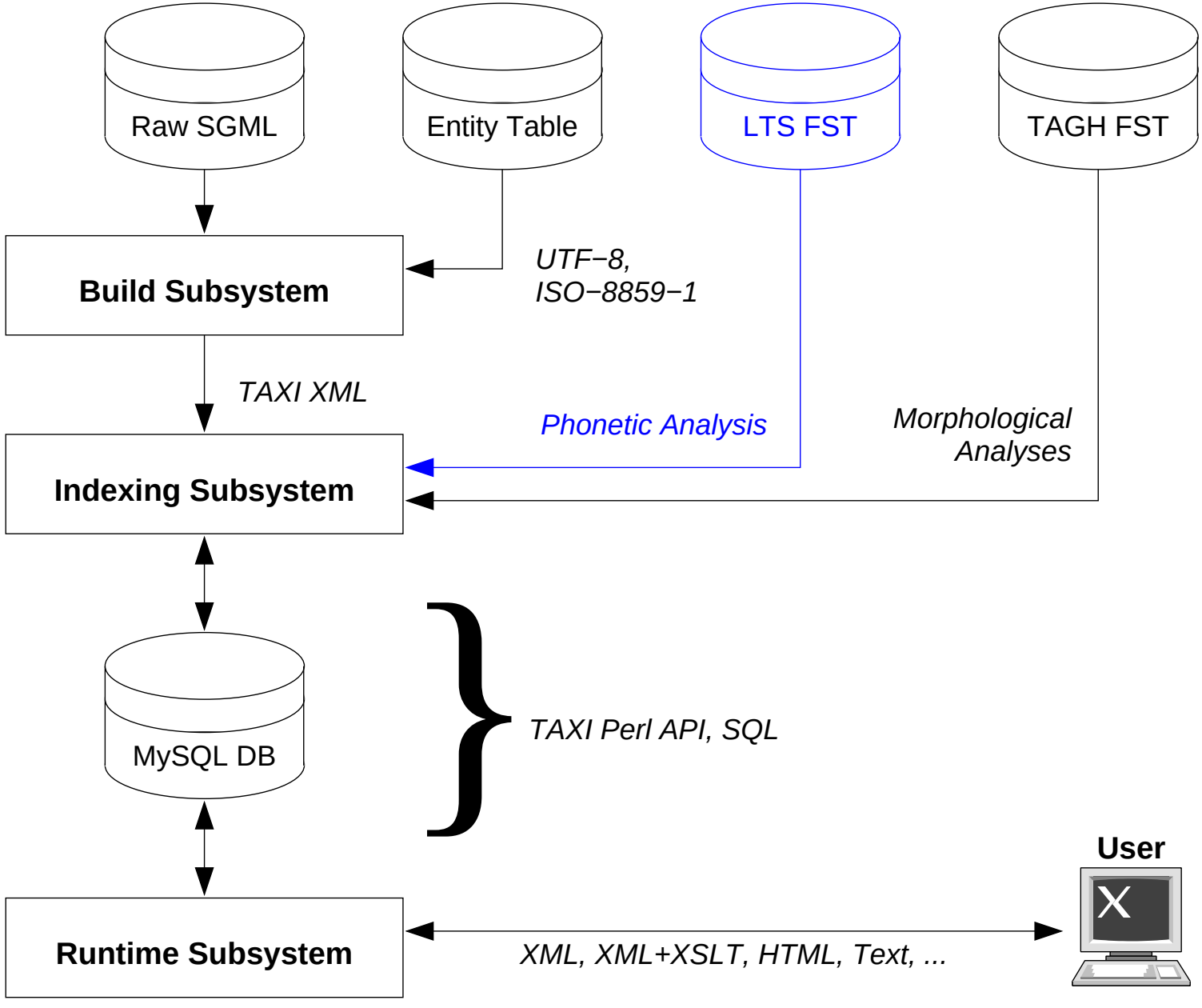
Indexing Subsystem

- Backend `MySQL` database administration
- Performs phonetic & morphological analysis

Runtime Subsystem

- User query handling (parsing, retrieval, formatting)
- HTTP server mode for remote queries

Taxi/Grimm System Architecture



Letter-to-Sound Conversion

The Big Picture

- Grimm Quotation Evidence Corpus
- Taxi Corpus Indexing System

Letter-to-Sound Conversion

- Festival LTS Rulesets
- Failed Attempts

  . . . or: "Why `lextools` *Won't Cut The Butter"*
- LTS Transducer Generation

Results

- Morphological Coverage

Demonstration

- Taxi/Grimm HTTP Query Server

LTS Conversion: Overview

Festival Letter-to-Sound Rulesets

- Festival Text-to-Speech System
- IMS German Festival Package
- Syntax & Semantics of Festival LTS Rules

Failed Attempts

- AT&T Tools: `lexcomplex`, `lexrulecomp`
- Roche & Schabes' Brill Tagger Emulation

LTS Transducer Generation

- Aho-Corasick Pattern Matchers
- Intermediate Alphabets
- Output Filter

Festival & IMS German Festival

Festival TTS System *(Black & Taylor, 1997)*

`http://www.cstr.ed.ac.uk/projects/festival`

- Free modular Text-to-Speech System (X11 license)
- Implemented in SCHEME and C++
- Abstract: language- and model-independent
- Extensible: research- & development-oriented
- SLOW

IMS German Festival Module *(Möhler* et al.*, 2001)*

`http://www.ims.uni-stuttgart.de/phonetik/synthesis`

- German language support module for festival
- Available free of cost for research purposes
- Unmaintained: broken in festival versions $\geq$ 1.4.1
- Includes a working German LTS rule-set

Festival LTS Rules: Syntax

Alphabets

- Σ_g: finite input alphabet (**graphemes**)
- Σ_p: finite output alphabet (**phones**)
- $\Sigma_g \cap \Sigma_p = \emptyset$

Rules

- $Rule \ ::= \ (\alpha[\beta]\gamma \rightarrow \pi) \qquad \in \Sigma_g^* \times \Sigma_g^+ \times \Sigma_g^* \times \Sigma_p^*$
- Rule target length: $|(\alpha[\beta]\gamma \rightarrow \pi)| = |\beta|$

Ruleset

- Strictly ordered finite sequence
- $R = \langle r_1, \ldots, r_{n_R} \rangle$
- $r_i \prec r_j$ iff $i < j$ for $1 \leq i, j \leq n_R$

Festival LTS Rules: Semantics 1

Informal Procedural Semantics

- Input is processed from **left to right**
- The **first matching rule** is applied at each position

Rule Matching

[Notation: $r \sim_i w$]

- The rule $r = (\alpha[\beta]\gamma \rightarrow \pi)$ **matches** an input string $w = w_1 \cdots w_n$ at position i iff:

$$w_{i-|\alpha|} \cdots w_{i+|\beta\gamma|} = \alpha\beta\gamma$$

Rule Applicability

[Notation: $r \approx_i w$]

- The rule r is **applicable** to w at i iff:

$$r \sim_i w \quad \text{and} \quad \forall r' \in R. r' \prec r \implies r' \not\sim_i w$$

Festival LTS Rules: Semantics 2

Configurations

- $\langle \underbrace{w}_{\text{input}}, \underbrace{i}_{\text{position}}, \underbrace{p}_{\text{output}} \rangle \in \Sigma_g^* \times \mathbb{N} \times \Sigma_p^*$

Successive Rule Application

- $\underbrace{\langle w, i, p \rangle}_{\text{source config}} \vdash_R \underbrace{\langle w, i + |\beta|, p\pi \rangle}_{\text{sink config}}$ iff $\underbrace{(\alpha[\beta]\gamma \rightarrow \pi) \approx_i w}_{\text{use first applicable rule}}$

LTS Function

- $LTS : \Sigma_g^* \rightarrow \Sigma_p^* : w \mapsto p$ iff:

$$\underbrace{\langle w, 1, \varepsilon \rangle}_{\text{initial configuration}} \quad \vdash_R^* \quad \underbrace{\langle w, |w| + 1, p \rangle}_{\text{final configuration}}$$

Festival LTS Rules: Algorithm

function LTS

Parameter: ordered rules $\langle R, \prec \rangle$

Paramter: word $w \in \Sigma_g^*$

Returns: phonetic string $p \in \Sigma_p^*$

begin

$i := 1$ /* Initialize: read position */

$p := \varepsilon$ /* Initialize: output buffer */

while $(i < |w|)$ **do**

if $(R \sim_i w = \emptyset)$ **then error;** /* No match found */

$(\alpha[\beta]\gamma \rightarrow \pi) = \min_{\prec} R \sim_i w$ /* First match */

$p := p\pi$ /* Accumulate output */

$i := i + |\beta|$ /* Consume input */

end while

return p

end function

Festival LTS Rules: Example

Rules

	[a]	$C $C	→ *a*
	[a]		→ *a:*
$Vb	[c h]		→ *x*
	[c]		→ *k*
	[e]		→ *ə*
#	[s]	$V	→ *z*
	[s]		→ *s*

Example Ruleset (with symbol classes)

Festival LTS Rules: Example

Rules		Input
[a] $C $C	→ *a*	sache
[a]	→ *a:*	
$Vb [c h]	→ *x*	
[c]	→ *k*	
[e]	→ *ə*	
# [s] $V	→ *z*	
[s]	→ *s*	

Example input word

Festival LTS Rules: Example

Rules		Input
[a] $C $C	→ *a*	#sache#
[a]	→ *a:*	
$Vb [c h]	→ *x*	
[c]	→ *k*	
[e]	→ *ə*	
# [s] $V	→ *z*	
[s]	→ *s*	

BOS & EOS markers

Festival LTS Rules: Example

Rules		Input
[a] $C $C	→ *a*	#[sache#
[a]	→ *a:*	
$Vb [c h]	→ *x*	
[c]	→ *k*	
[e]	→ *ə*	
# [s] $V	→ *z*	
[s]	→ *s*	

Initialize read position

Festival LTS Rules: Example

Rules		Input	:	Output
[a] $C $C	→ *a*	#[sache#	:	ε
[a]	→ *a:*			
$Vb [c h]	→ *x*			
[c]	→ *k*			
[e]	→ *ə*			
# [s] $V	→ *z*			
[s]	→ *s*			

Initialize output buffer

Festival LTS Rules: Example

Rules		Input	:	Output
[a] $C $C	→ *a*	#[sache#	:	ε
[a]	→ *a:*	#[sache#	:	ε
$Vb [c h]	→ *x*			
[c]	→ *k*			
[e]	→ *ə*			
# [s] $V	→ *z*			
[s]	→ *s*			

Find matching rules

Festival LTS Rules: Example

Rules	Input	:	Output
[a] $C $C → *a*	#[sache#	:	ε
[a] → *a:*	#[s]ache#	:	*z*
$Vb [c h] → *x*			
[c] → *k*			
[e] → *ə*			
# [s] $V → ***z***			
[s] → *s*			

Apply first matching rule

Festival LTS Rules: Example

Rules	Input	:	Output
[a] $C $C → a	#[sache#	:	ε
[a] → a:	#[s]ache#	:	z
$Vb [c h] → x	#s[a]che#	:	za
[c] → k			
[e] → ə			
# [s] $V → z			
[s] → s			

Increment, match & apply

Festival LTS Rules: Example

Rules		Input	:	Output
[a] $C $C	→ *a*	#[sache#	:	*ε*
[a]	→ *a:*	#[s]ache#	:	*z*
$Vb [c h]	→ ***x***	#s[a]che#	:	*za*
[c]	→ *k*	#sa[ch]e#	:	*zax*
[e]	→ *ə*			
# [s] $V	→ *z*			
[s]	→ *s*			

Increment, match & apply

Festival LTS Rules: Example

Rules	Input	:	Output
[a] $C $C → *a*	#[sache#	:	*ε*
[a] → *a:*	#[s]ache#	:	*z*
$Vb [c h] → *x*	#s[a]che#	:	*za*
[c] → *k*	#sa[ch]e#	:	*zax*
[**e**] → **ə**	#sach[e]#	:	*zax* **ə**
# [s] $V → *z*			
[s] → *s*			

Increment, match & apply

Festival LTS Rules: Example

Rules		Input	:	Output
[a] $C $C	→ *a*	#[sache#	:	*ε*
[a]	→ *a:*	#[s]ache#	:	*z*
$Vb [c h]	→ *x*	#s[a]che#	:	*za*
[c]	→ *k*	#sa[ch]e#	:	*zax*
[e]	→ *ə*	#sach[e]#	:	*zaxə*
# [s] $V	→ *z*	sache	:	*/zaxə/*
[s]	→ *s*			

Final output

Festival LTS Rules: Analysis

Desideratum

- FST implementation of $LTS(\cdot)$ function

Clearly...

- Festival LTS rule application is deterministic
- Each rule $(\alpha[\beta]\gamma \rightarrow \pi)$ defines a regular transduction $\alpha(\beta : \pi)\gamma$

Problems

- Rule precedence (bleeding)
- Rule-dependent position incrementation (derivability $\not\equiv$ WFST best-path)
- Rule context lookahead / -behind

Failed Attempt: `lexcomplex`

Idea

- Convert each LTS rule r to a regular expression $[\![r]\!]$:
  - $[\![(\alpha[\beta]\gamma \rightarrow \pi)]\!] = \alpha(\beta : \pi)\gamma$
- Compile results with `lexcomplex`:
  - $[\![R]\!]_{lex} = \bigcup_{r \in R} [\![r]\!]$

Problems

- No rule precedence $\Longrightarrow$ non-deterministic FST
  - Workaround: use costs to simulate precedence
- Only one rule is applied by $[\![R]\!]$
  - Kleene-closure $[\![R]\!]^*_{lex}$ also consumes contexts
  - No lookahead / -behind

OUT-OF-CHEESE ERROR: REDO FROM START

Failed Attempt: `lexcomplex -m`

Idea

- Convert each $r \in R$ to regex $[\![r]\!]$ as above
- Compile results with `lexcomplex -m`:
  - $[\![R]\!]_{lex-m} = ((Id(\Sigma) - Prj_1([\![R]\!]_{lex})) \cup [\![R]\!]_{lex})^*$

Problems

- `-m` mode designed for single symbol input
- No rule precedence, non-deterministic FST
- $[\![R]\!]_{lex-m}$ also consumes contexts
  - No lookahead / -behind

OUT-OF-CHEESE ERROR: REDO FROM START

Failed Attempt: `lexcomplex -M`

Idea

- Convert each $r \in R$ to regex $[\![r]\!]$ as above
- Compile results with `lexcomplex -M`:
  - $[\![R]\!]_{lex-M} =$ *undocumented*

Problems

- `-M` mode designed for parallel batch rewrites
- No rule precedence, non-deterministic FST
- $[\![R]\!]_{lex-M}$ consumes contexts
  - No lookahead / -behind

OUT-OF-CHEESE ERROR: REDO FROM START

Failed Attempt: `lexrulecomp`

Idea

- Convert each $r \in R$ to a rewrite rule $[\![r]\!]_{lexrul}$:
  - $[\![(\alpha[\beta]\gamma \rightarrow \pi)]\!]_{lexrul} = \beta \rightarrow \pi \ / \ \alpha \ \_ \ \gamma$
- Compile result with `lexrulecomp`:
  - $[\![R]\!]_{lexrul} =$ *undocumented*
- Left-to-right obligatory mode

Problems

- $[\![R]\!]_{lexrul}$ handles overlapping contexts improperly
  - Bad lookahead / -behind ?
  - Bad position incrementation ?

OUT-OF-CHEESE ERROR: REDO FROM START (AGAIN)

LTS FST: Method

Step 1: Rule Match Detection

- Aho-Corasick Pattern Matchers (ACPMs)
- Distinguish left *vs.* right of current I/O position
- Intermediate alphabets: matched rule subsets
- Composition of (reversed) partial ACPMs models LTS configurations

Step 2: Output Filter

- Composed with match-detection FST
- Selects applicable rules
- Simulates position increment

Aho-Corasick Pattern Matching

Given

- A set $P = \{p_1, \ldots, p_{n_P}\} \subseteq \Sigma^*$ of patterns

Task

- Identify all occurrences of any pattern $p \in P$ in an input string $w \in \Sigma^*$

Aho-Corasick Algorithm (Aho & Corasick, 1975)

- Constructs a pattern-matching FST $\mathcal{AC}_P : \Sigma^* \to \mathfrak{P}(P)^*$
- Linear runtime complexity: $\mathcal{O} = \mathcal{O}(|w|)$
  - Really $\mathcal{O}(|w| + \sum_{i=1}^{n_P} |p_i| + \sum_{j=1}^{|w|} |P \sim_j w|)$

. . . but who's counting?

ACPM Construction Sketch

Input

- Prefix Tree Acceptor (trie) for P

Output

- Finite-state transducer $\mathcal{AC}_P : \Sigma^* \to \mathfrak{P}(P)^*$ such that $\mathcal{AC}_P(w) = \bigodot_{i=1}^{|w|} \{p \in P \mid p = w_{i-|p|}..w_i\}$

Method

- Transition function $\texttt{goto} : Q \times \Sigma \to Q$
- Failure function $\texttt{fail} : Q \to Q$
- Output function $\texttt{out} : Q \to \mathfrak{P}(P)$

Together, `goto`, `fail`, and `out` define a conventional arc-dependent input-deterministic transition function

$$\delta : Q \times \Sigma \to Q \times \mathfrak{P}(P)$$

ACPM Limitations & Workarounds

Delayed Output

- ACPM output occurs at pattern terminus
- **Problem** for LTS rule lookahead / -behind
- **Solution**: construct 2 independent ACPMs

Information Loss

- ACPM outputs only pattern sets, not input symbols
- **Problem** for dual-ACPM construction (need both)
- **Solution**: preserve input in lookbehind ACPM

Fixed Increment

- ACPM outputs a pattern set for each input symbol
- **Problem** for LTS rule-dependent increment
- **Solution**: filter out unneeded outputs

LTS FST: Match Detection

Left-Context Matching (Lookbehind)

$$
\begin{aligned}
M_L &\cong \mathcal{AC}_{\{\alpha \mid (\alpha[\beta]\gamma \to \pi) \in R\}} \\
&: \Sigma_w^* \to (\Sigma_w \times \mathfrak{P}(R))^* \\
&: w \mapsto \bigodot_{i=0}^{|w|} \langle w_i, \{(\alpha[\beta]\gamma \to \pi) \in R \mid w_{i-|\alpha|..i} = \alpha\} \rangle
\end{aligned}
$$

Target & Right-Context Matching (Lookahead)

$$
\begin{aligned}
M_R &\cong \text{reverse}\left(\mathcal{AC}_{\{(\beta\gamma)^{-1} \mid (\alpha[\beta]\gamma \to \pi) \in R\}}\right) \\
&: (\Sigma_w \times \mathfrak{P}(R))^* \to \mathfrak{P}(R)^* \\
&: \langle w_i, S_i \rangle_I \mapsto \\
&\quad \bigodot_{i \in I} \left(S_{i-1} \cap \{(\alpha[\beta]\gamma \to \pi) \in R \mid w_{i..i+|\beta\gamma|} = \beta\gamma\}\right)
\end{aligned}
$$

Match Detection FST (Current Match Subset)

$$
M_{LR} = M_L \circ M_R \ : \ \Sigma_w^* \to \mathfrak{P}(R)^* \ : \ w \mapsto \bigodot_{i=1}^{|w|} R \sim_i w
$$

LTS FST: Output Filter

Notation

- Let $\Sigma_{LR} \subseteq \mathfrak{P}(R)$ be the output alphabet of M_{LR}
- For each $S \in \Sigma_{LR}$, let $(\alpha_S[\beta_S]\gamma_S \rightarrow \pi_S) = \min_{\prec} S$

Output Filter Construction

- Define output filter $M_O : {\Sigma_{LR}}^* \rightarrow {\Sigma_p}^*$ as:

$$M_O = \left(\bigcup_{S \in \Sigma_{LR}} \left[\underbrace{(S : \pi_S)}_{\text{apply}} \underbrace{(\Sigma_{LR} : \varepsilon)^{|\beta_S|-1}}_{\text{increment}} \right] \right)^*$$

LTS Transducer

- Then, the desired LTS FST can be defined as:

$$M_{LTS} = (M_{LR} \circ M_O) = (M_L \circ M_R \circ M_O) : \Sigma_w^* \rightarrow \Sigma_p^*$$

LTS FST: Example

Input	#	s	a	c	h	e	#
$\mathbf{M_L}$ $\longrightarrow$	$\emptyset$	$\left\{\begin{array}{ll} [a]ch & \rightarrow a \\ [a] & \rightarrow a:, \\ [c] & \rightarrow k, \\ [e] & \rightarrow \text{ə}, \\ \#[s]a & \rightarrow z, \\ [s] & \rightarrow s \end{array}\right\}$	$\left\{\begin{array}{ll} [a]ch & \rightarrow a, \\ [a] & \rightarrow a:, \\ [c] & \rightarrow k, \\ [e] & \rightarrow \text{ə}, \\ [s] & \rightarrow s \end{array}\right\}$	$\left\{\begin{array}{ll} [\,a\,]ch & \rightarrow a, \\ [\,a\,] & \rightarrow a:, \\ a[ch] & \rightarrow x, \\ [\,c\,] & \rightarrow k, \\ [\,e\,] & \rightarrow \text{ə}, \\ [\,s\,] & \rightarrow s \end{array}\right\}$	$\emptyset$	$\left\{\begin{array}{ll} [a]ch & \rightarrow a, \\ [a] & \rightarrow a:, \\ [c] & \rightarrow k, \\ [e] & \rightarrow \text{ə}, \\ [s] & \rightarrow s \end{array}\right\}$	$\emptyset$
$\mathbf{M_R}$ $\longleftarrow$	$\emptyset$	$\left\{\begin{array}{ll} \#[s]a & \rightarrow z, \\ [s] & \rightarrow s \end{array}\right\}$	$\left\{\begin{array}{ll} [a]ch & \rightarrow a, \\ [a] & \rightarrow a:, \end{array}\right\}$	$\left\{\begin{array}{ll} a[ch] & \rightarrow x, \\ [\,c\,] & \rightarrow k, \end{array}\right\}$	$\emptyset$	$\left\{\begin{array}{ll} [e] & \rightarrow \text{ə}, \end{array}\right\}$	$\emptyset$
$\mathbf{M_O}$ $\longrightarrow$	ε	z	a	x	ε	ə	ε

Results

The Big Picture

- Grimm Quotation Evidence Corpus
- Taxi Corpus Indexing System

Letter-to-Sound Conversion

- Festival LTS Rulesets
- Failed Attempts

  . . . or: "Why `lextools` *Won't Cut The Butter"*
- LTS Transducer Generation

Results

- Morphological Coverage

Demonstration

- Taxi/Grimm HTTP Query Server

Results: Performance

Letter-to-Sound FST

- Number of LTS Rules : 396
- Number of States : 1,037
- Number of Final States : 292
- Number of Arcs : 131,440
- Compilation Time : *ca.* 5 minutes

Performance

LTS Method	Throughput (tok/s)	Relative
`festival` (TCP)	28.53	−4875.57 %
`festival` (pipe)	1391.45	± 0.00 %
FST (libgfsm)	9124.69	+ 555.77 %

Results: Grimm Corpus

Types (alphabetic, non-foreign)

- Number of Orthographic Types: 323,561
  - Morphologically Analyzed : 137,613 (42.53 %)
- Number of Phonetic Types : 267,933
  - Morphologically Analyzed : 130,796 (48.82 %)

Tokens (alphabetic, non-foreign)

- Number of Tokens : 5,491,978
  - Orthographic form known: 4,600,619 (83.77 %)
  - Phonetic form known : 5,011,177 (91.25 %)
- “Error” Reduction : 410,558 (46.09 %)

Demonstration

The Big Picture

- Grimm Quotation Evidence Corpus
- Taxi Corpus Indexing System

Letter-to-Sound Conversion

- Festival LTS Rulesets
- Failed Attempts

  . . . or: "Why `lextools` *Won't Cut The Butter"*
- LTS Transducer Generation

Results

- Morphological Coverage

Demonstration

- Taxi/Grimm HTTP Server: `kira.bbaw.de:8765`

Thanks for Listening !

/ daŋ·kə ʃøn / !