Transliterated mobile keyboard input via weighted finite-state transducers

Lars Hellsten, Brian Roark, Prasoon Goyal*, Cyril Allauzen,
Françoise Beaufays, Tom Ouyang, Michael Riley & David Rybach
Google and *NYU

FSMNLP conference, 4 Sept 2017

Transliteration Modeling for Keyboard

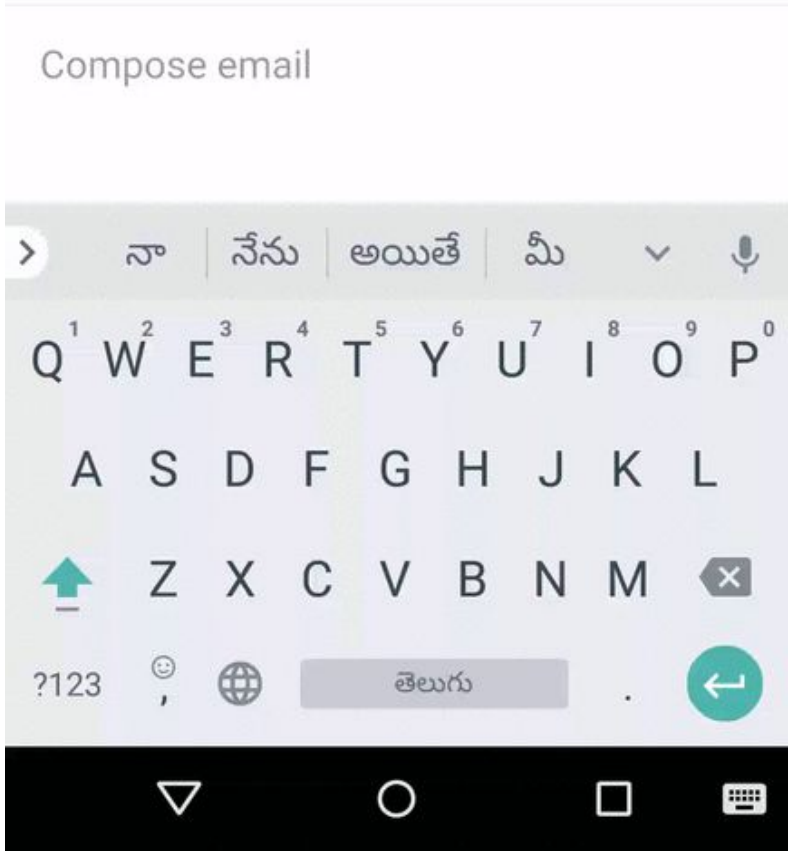
Problem: Many writing systems are difficult to represent compactly on a soft keyboard of the sort used on mobile devices

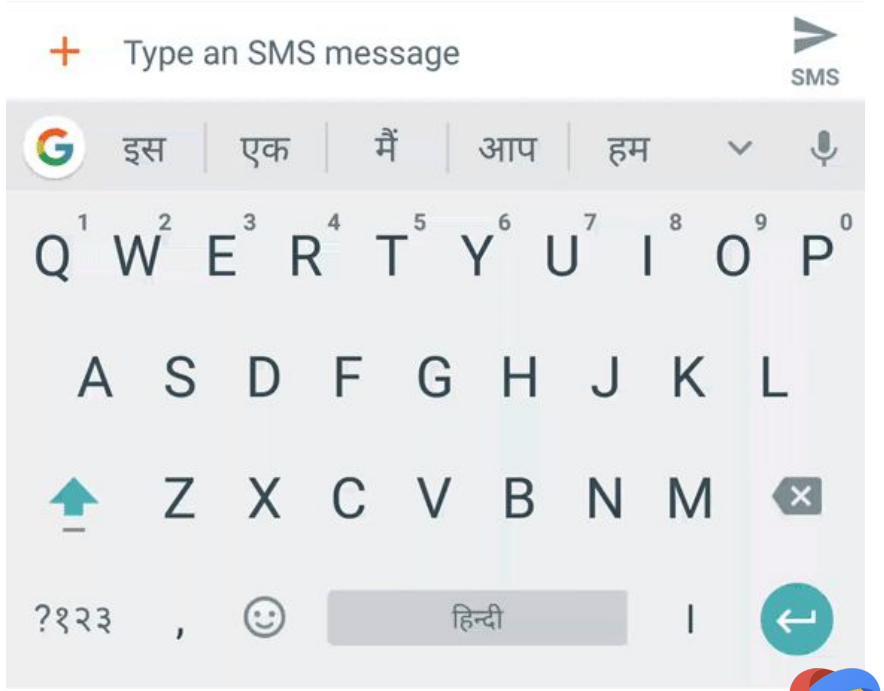
- E.g., Chinese or Japanese with potentially thousands of characters
- Brahmic scripts (e.g., Devanagari) in South Asia have fewer symbols than these, but have complex multi-symbol ligatures and vowel diacritics

Common to use a *romanization* system, type on QWERTY keyboard

- Pinyin in Chinese is a long-standing standard system
- No standard system for many languages

Gesturing and tapping in Telugu and Hindi





Overview

Like speech/OCR, KB requires inference over large search space

- Combining models of the input space (e.g., acoustic model) with an output model (language model)
- Weighted finite-state transducers permit off-line optimization

Transliteration represented as another weighted transduction

- Pair language models (aka joint multi-grams) form compact WFSTs.

Several complications require specialized optimizations

- On-device decoding relatively resource impoverished, models small and fast.
- Local determinization and weight pushing achieve large speedups.

Outline

Background and preliminaries

- Indic scripts
- Weighted finite-state transducers and keyboard decoding
- Romanization and transliteration

Methods

- Pair language models (aka joint multi-grams)
- WFST optimization of resulting transducers

Experimental results

- Experiments on development set, validating WFST optimizations
- Hindi and Tamil results versus existing Indic IME system

Currently available systems and future directions

Indic Scripts

Most South Asian scripts descended from Brahmi script

- Including Devanagari, Bengali, Tamil, Gujarati and others;
- Excluding Perso-Arabic (e.g., for Urdu); and a few others.

Consider the word ‘sanskrit’ in several of Brahmic scripts:

Gujarati:	સંસ્કૃત	Hindi:	संस्कृत
Bengali:	সংস্কৃত	Tamil:	சமஸ்கிருத

Brahmic (or Indic) scripts look different but share similarities

- Organized around orthographic syllable (*akṣara*);
- diacritics are used to signal non-default syllables; and
- ligatures signal combinations of consonants

Indic Scripts

Example: ‘sanskrit’ in Hindi is संस्कृत

Three orthographic ‘syllables’ (*akṣara*): सं (san) स्कृ (skr) त (t)

In Unicode, diacritics and ligatures encoded as symbols:

I. स ं
II. स ् क ृ
III. त

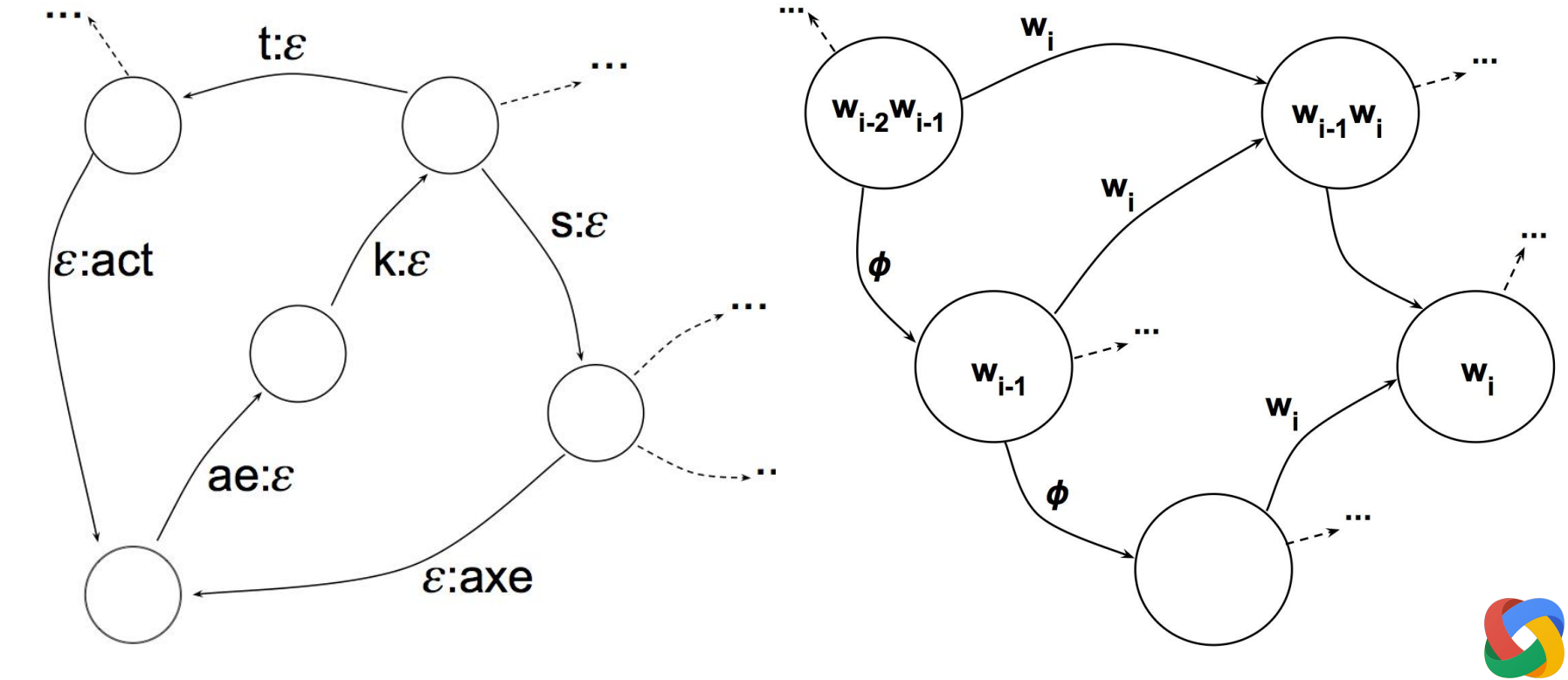
Can align romanized string (sanskrit or sanskrt):

s:स a:ϵ n:ं s:स ϵ:् k:क r:ृ i:ϵ t:त

Alignment forms the basis for the transliteration model

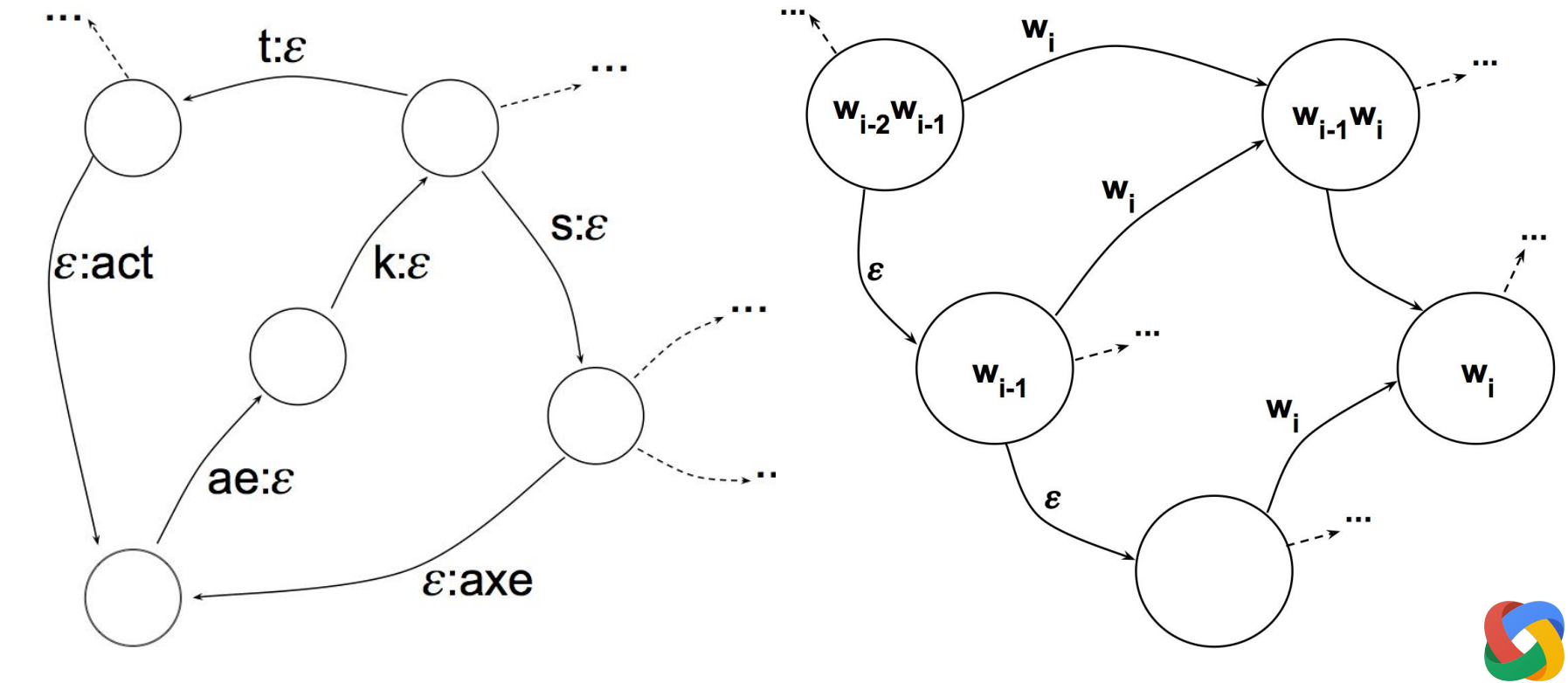
WFSTs for speech recognition

Pronunciation transducer: (composed with) LM automaton:



WFSTs for speech recognition (epsilon backoff)

Pronunciation transducer: (composed with) LM automaton:



WFSTs for speech recognition

Compose multiple component models (e.g., lexicon L, language model G) and search over resulting transducer

- Perform off-line optimization of resulting transducer (L ○ G)
- Can use transducer (C) to introduce phone context sensitivity (C ○ L ○ G), e.g., α _ β
- Non-deterministic on both sides (many words with same pronunciation and/or multiple pronunciations per word)
- Special ‘tricks’ to enable structure sharing via ‘determinization’
- Weight pushing provides ‘lookahead’ cost, improves search

Variants of this approach are widely used in speech recognition

- See Mohri, Pereira and Riley (2002)

Keyboard Models

Keyboard has trivial ‘pronunciation’ lexicon in contrast to speech:

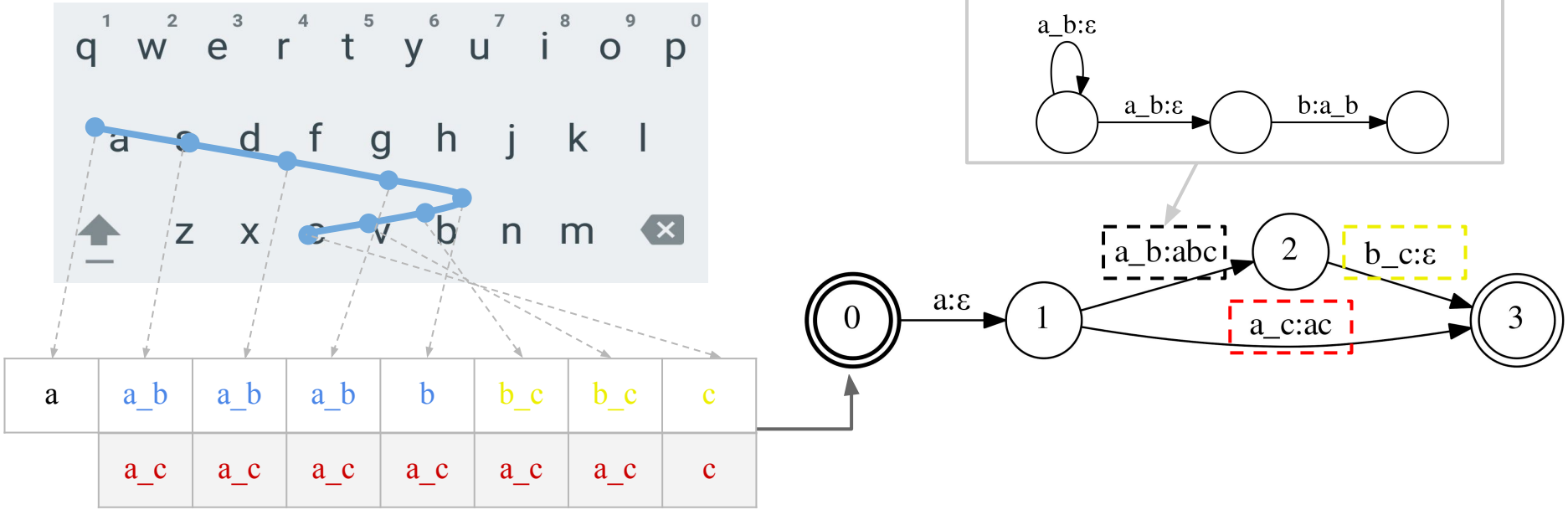
Lexicon:	word	‘pronunciation’
Speech	though	dh ow
Keyboard	though	t h o u g h

Can build in context sensitivity by looking at preceding keys

Input model is a spatial model over keyboard (vs. acoustic model)

Resulting decoder has many similarities to speech decoder

FST Keyboard Decoder



Transducers

C: bi-key → letter
L: letters → words
G: LM (grammar)

⟹ Decoder graph: $(C \circ L) \circ G$

Transliteration Models

Transliteration brings back non-trivial lexicon as in speech:

Lexicon:	word	‘pronunciation’
Speech	though	dh ow
Keyboard	though	t h o u g h
Transliteration	तैयार	t a y y a r

Can use pronunciation modeling methods for transliteration.

Typically less consensus around canonical transliteration in Indic languages than pronunciation (even in English)

3% absolute WER increase when restricting to 10-best transliterations per word.

Training translit model T from given labeled data (lexicon)

Use existing grapheme to phoneme (g2p) tools for aligning lexicon at the symbol level:

Lexicon entry: टस t e s u

EM yields alignment: ट:t ɛ:e स:s ɛ:u

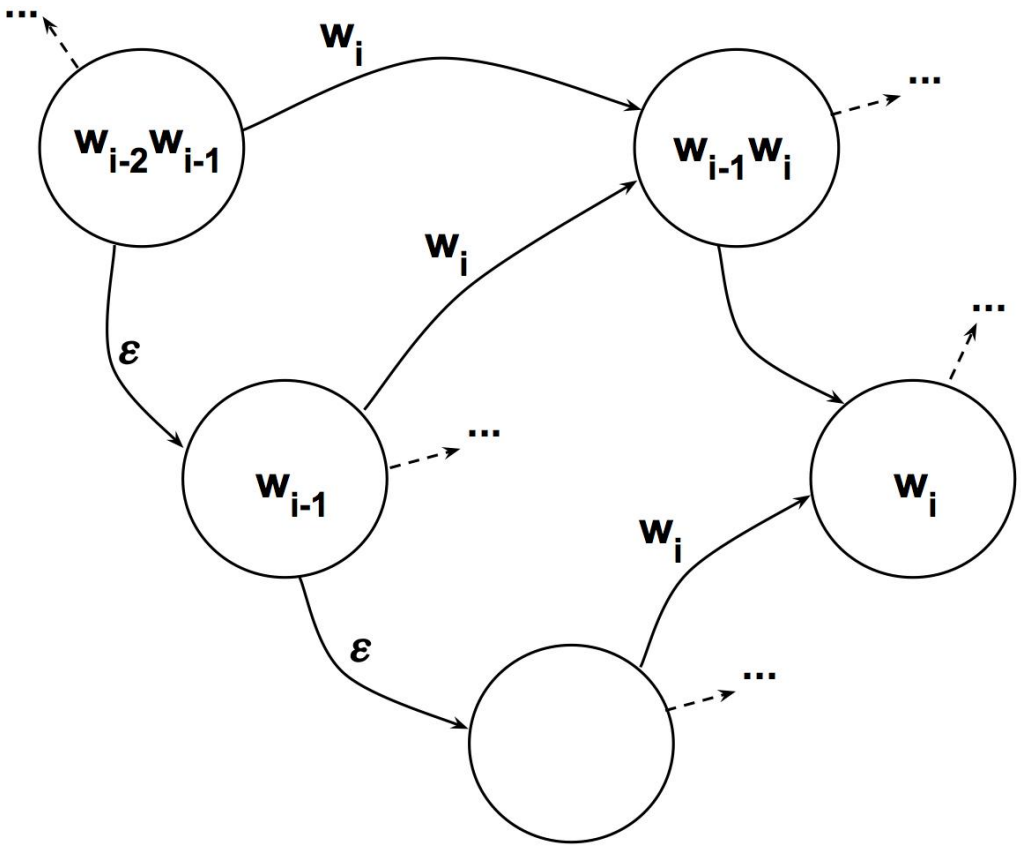
Can train a pruned, smoothed pair LM from alignments:

P(स:s | ट:t ɛ:e)

This can be converted from an LM automata to transducer.

Some issues around eliminating epsilon cycles and other opts.

LM automaton format



Training translit model T from given labeled data (lexicon)

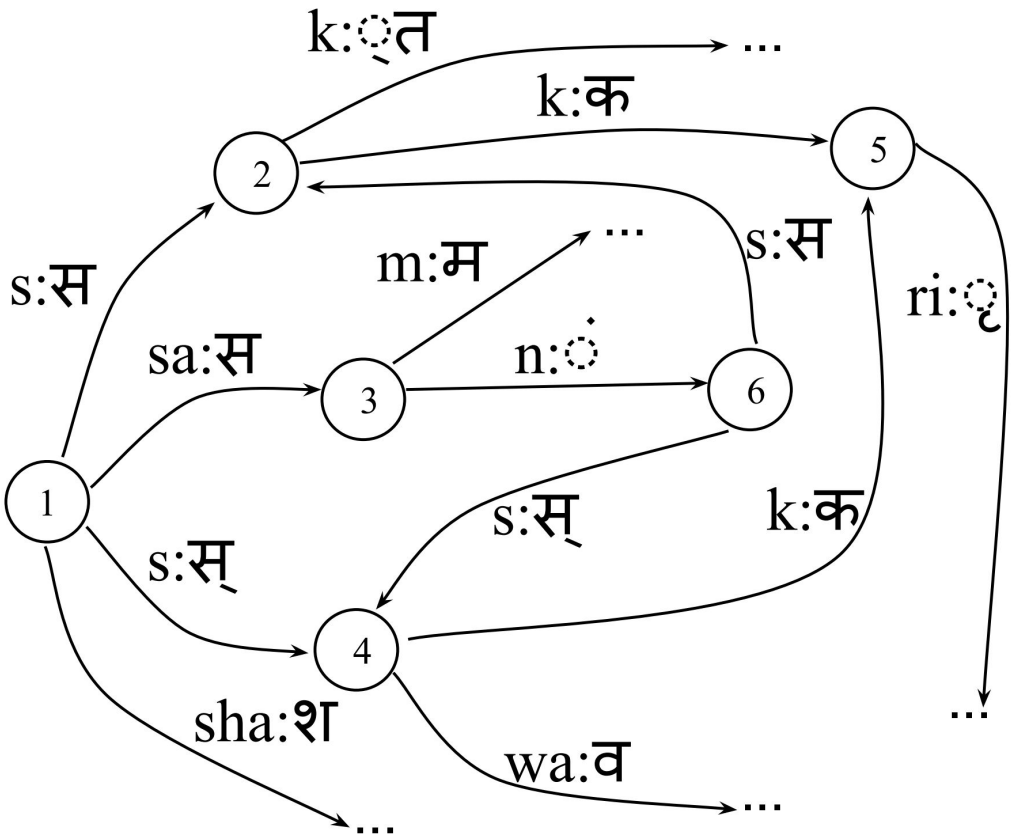
Remove epsilon cycles

- No unigram insertions or deletions (avoids backoff epsilon cycle)
- Restrict to only one insertion or deletion in sequence

Combine input and/or output symbols in second round of EM

- Similar to approach used in g2p for phonetisaurus system.
- Also has the benefit of extending dependencies in model.

Transliteration LM automaton format

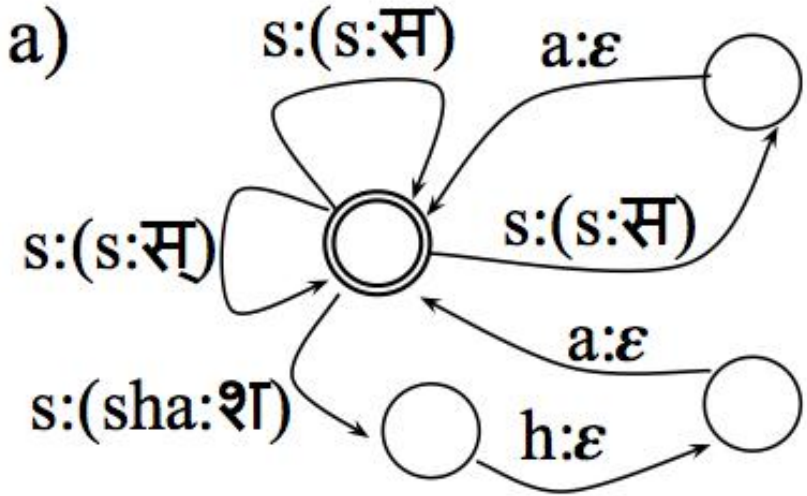


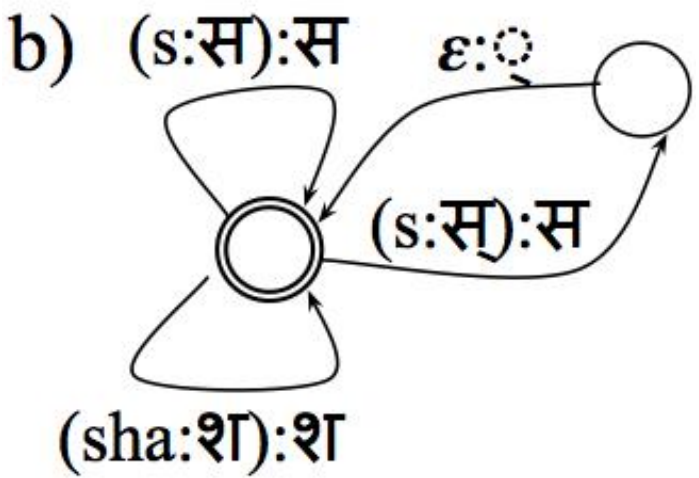
Simple conversion to UTF8 codepoint transducer

Convert to UTF8 codepoints on input and output of transducer:

a) Transducer *I* maps from input UTF8 codepoints to output pair LM symbols
b) Transducer *O* maps from input pair LM symbols to output UTF8 codepoints

Compose pair LM *P* with *I* and *O* to yield transducer: $I \circ P \circ O$

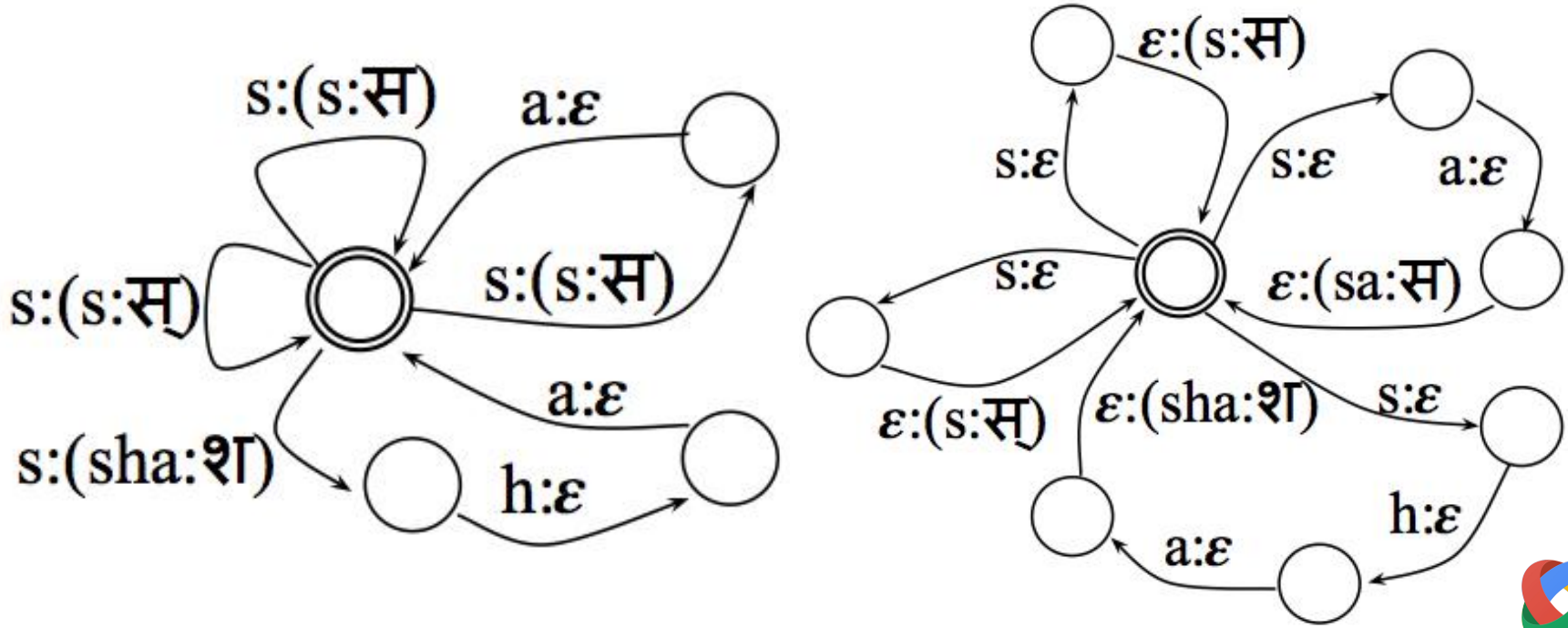




Optimizations to UTF-8 to pair LM transducer (I)

Variant of algorithm in Mohri, Pereira and Riley (2002)

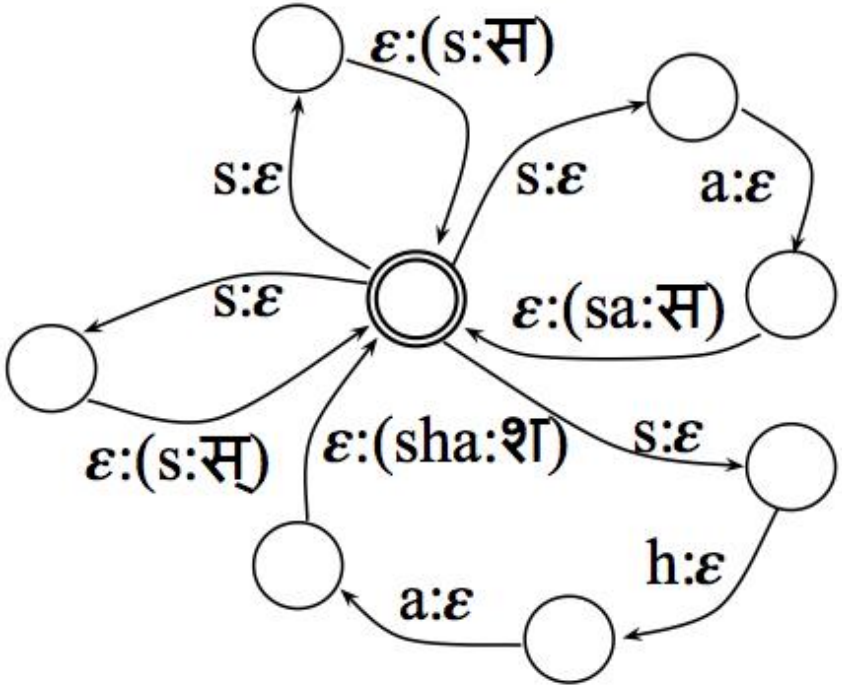
- Add full symbol to output of final epsilon transition back to start state

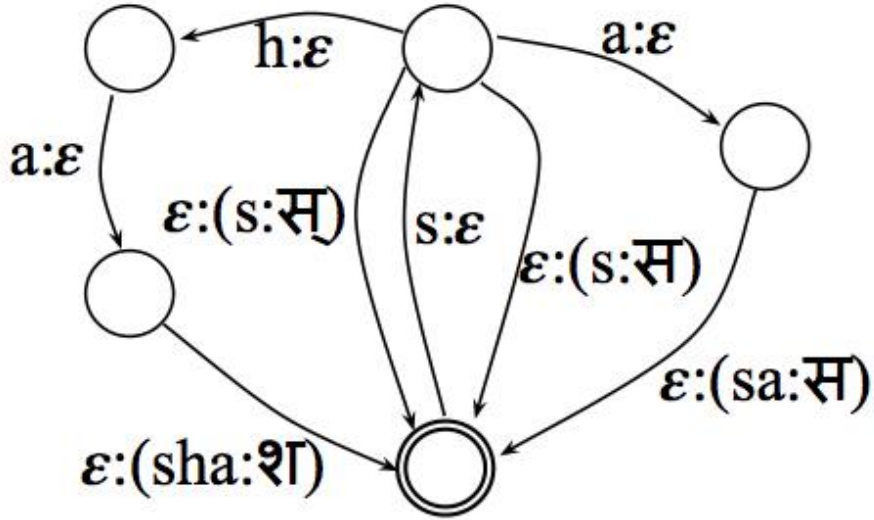


Optimizations to UTF-8 to pair LM transducer (II)

Variant of algorithm in Mohri, Pereira and Riley (2002)

- Then determinize over the encoded (pair) symbol.

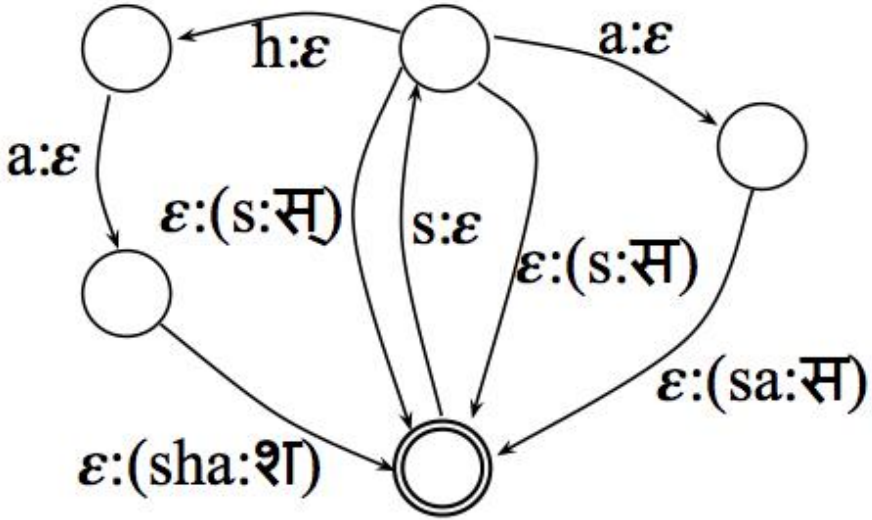


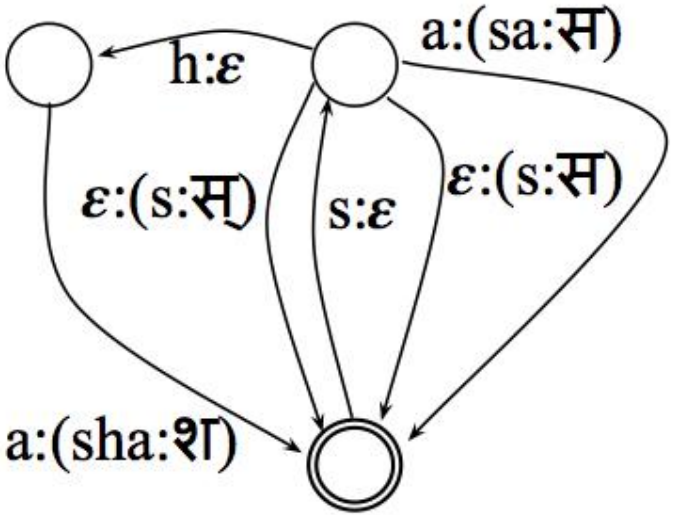


Optimizations to UTF-8 to pair LM transducer (III)

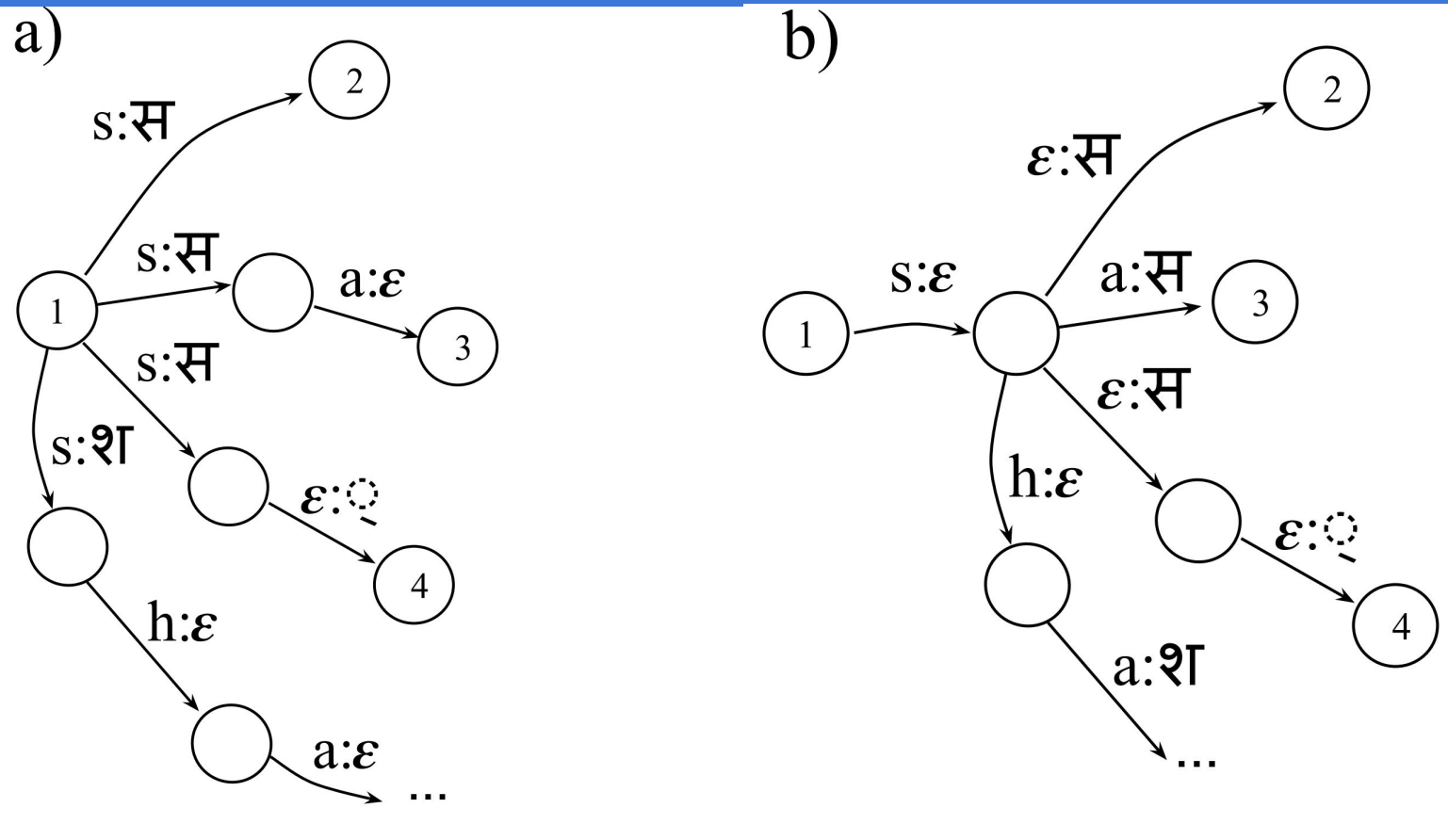
Variant of algorithm in Mohri, Pereira and Riley (2002)

- Finally combine sequential input/output epsilons to save a state.





Resulting transducer from a) simple; and b) optimized



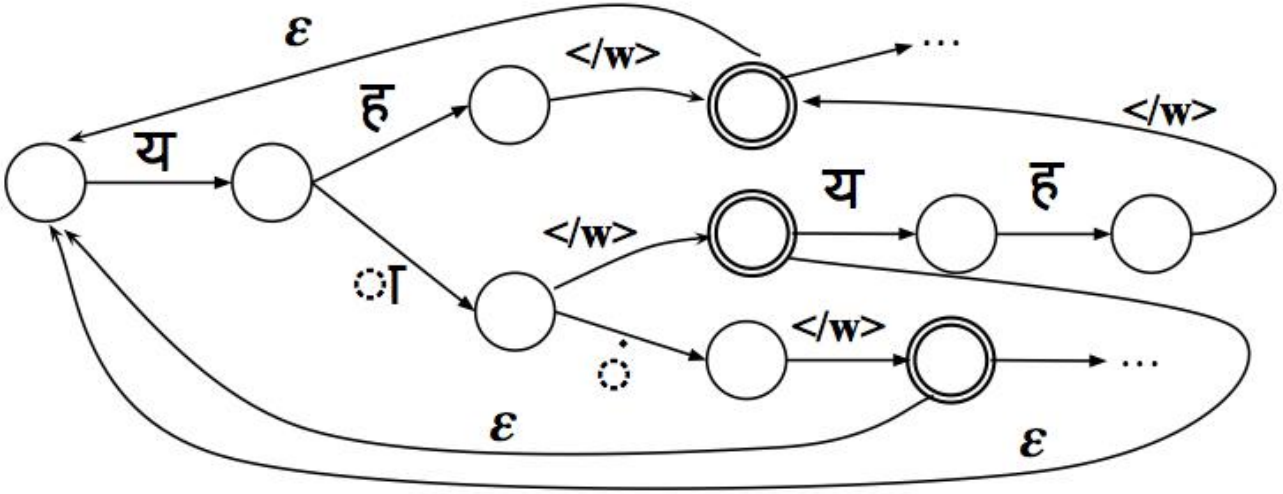
Compact representation of L ○ G

Since symbols spell out word, omit words to save space

Include end-of-word symbol to trigger word reconstruction

Build trie of letters of words leaving each state in the n-gram model

Encode in a variation of LOUDS compact format



From joint to conditional transliteration model

Pair language model trained T is a joint model of input/output strings

- Thus target string (e.g., Devanagari) assigned probability by both T, G

Convert to a conditional model by deriving normalization factor

- Calculated by marginalizing over each word's set of transliterations
- Efficient to derive by: (1) composing with lexicon; (2) projecting to output (words); (3) removing epsilon; and (4) determinizing over the log semiring

Normalization cost (appropriately pushed) is included in L ○ G

- Special procedure required for next word prediction and word completion

Experimental validation

Experiments in Hindi and Tamil

- Validation and test sets from native speakers, copying prompted (spoken) utterances at a natural pace without correcting errors

Compared against the Google Indic Keyboard

- Existing Hindi and Tamil keyboard (downloaded 50M-100M times)
- Uses HMM decoder with syllable-based transliteration model

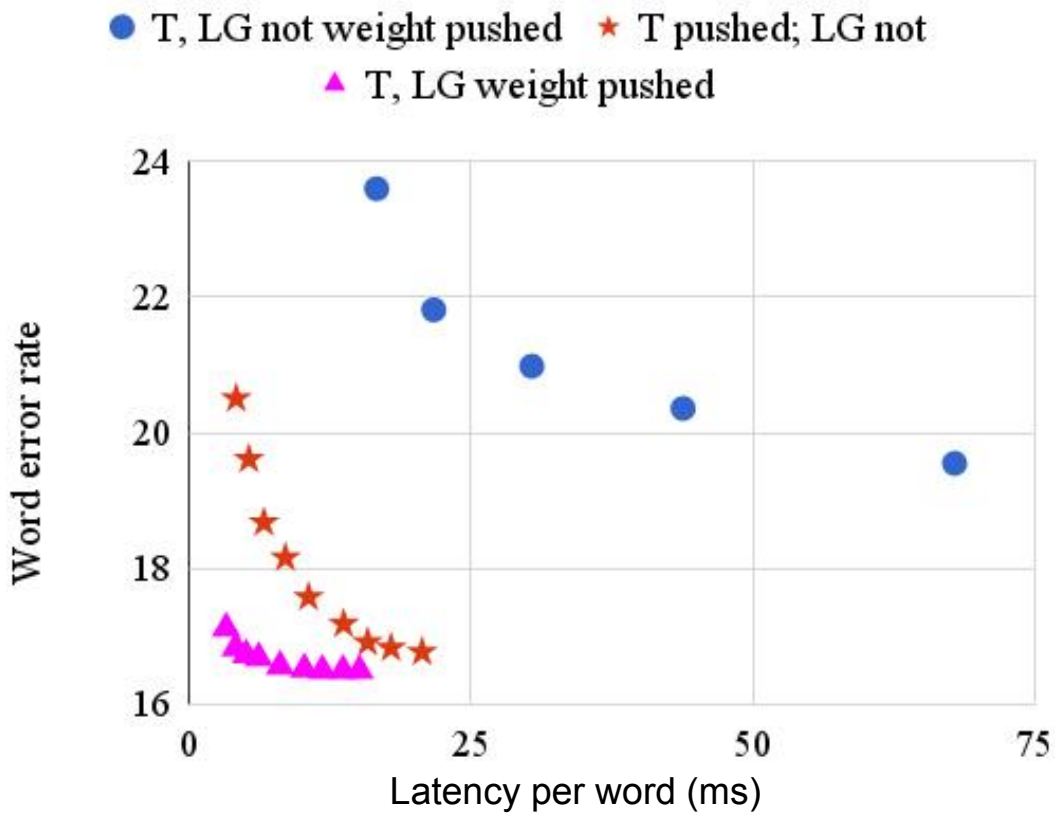
Language model was identical for both

- Vocabulary size approximately 150k with 750k n-grams in LM

Impact of LOUDS encoding of L ○ G

Language	L ○ G size (MB)		
	standard	compact	LOUDS
Hindi	85.3	26.3	6.72
Tamil	126.5	37.1	9.80

Impact of transducer optimizations (beam sweep on dev)



Blind testset comparison with baseline (Google Indic Keyboard)

System	Hindi		Tamil	
	WER	LPC (ms)	WER	LPC (ms)
Baseline	19.5	3.0	26.2	2.1
WFST-based	16.4	0.5	22.6	0.2

Launched transliteration keyboards

Launched 22 languages (some with multiple scripts) in 2017

Assamese†, Bengali, Bodo*†, Dogri*•, Gujarati, Hindi*, Kannada, Kashmiri*•, Konkani*, Maithili*, Malayalam, Manipuri, Marathi, Nepali, Odia, Punjabi‡•, Sanskrit†, Santali□, Sindhi*•, Tamil, Telugu, Urdu•

Script names (if not same as language):

†Bengali, *Devanagari, ‡Gurmukhi, □Ol Chiki, •Perso-Arabic

Summary and Future directions

On-device keyboard decoding requires (1) small models (≤ 10MB); (2) low latency (≤ 20ms); and (3) low working memory use

WFST optimizations make mobile transliteration keyboard possible

- Encoding L ○ G as an automaton and using LOUDS achieves (1)
- Determinization and weight pushing achieve (2) and (3)

Some future directions

- Making use of a joint transliteration and language model
- Incorporating neural models