Induction of Finite-State Covering Grammars for Text Normalization

Richard Sproat (Google, New York)

joint work with
Ke Wu, Hao Zhang, Kyle Gorman,
Felix Stahlberg, Xiaochang Peng, Brian Roark

CLASP
U Gothenburg
May 16, 2018

Research at Google

Induction of Finite-State Covering Grammars for Text Normalization

Richard Sproat (Google, New York)

joint work with
Ke Wu, Hao Zhang, Kyle Gorman,
Felix Stahlberg, Xiaochang Peng, Brian Roark

CLASP
U Gothenburg
May 16, 2018

Research at Google

Outline

- What is text normalization?
- What is the “state of the art”?
- A suite of neural solutions - and challenges
  - Finite-state covering grammars
- Implications and future directions

What is text normalization?

Definition: Transforming text so that the information in it is presented in some canonical form for a downstream application

Corollary: What counts as normalization depends upon the application

Linguistic standardization: Converting non-standard ways of writing things into a standard written form:

Input	Normalized form
cooooollllll	cool
cu l8r	see you later
udaman	you are the man (?)

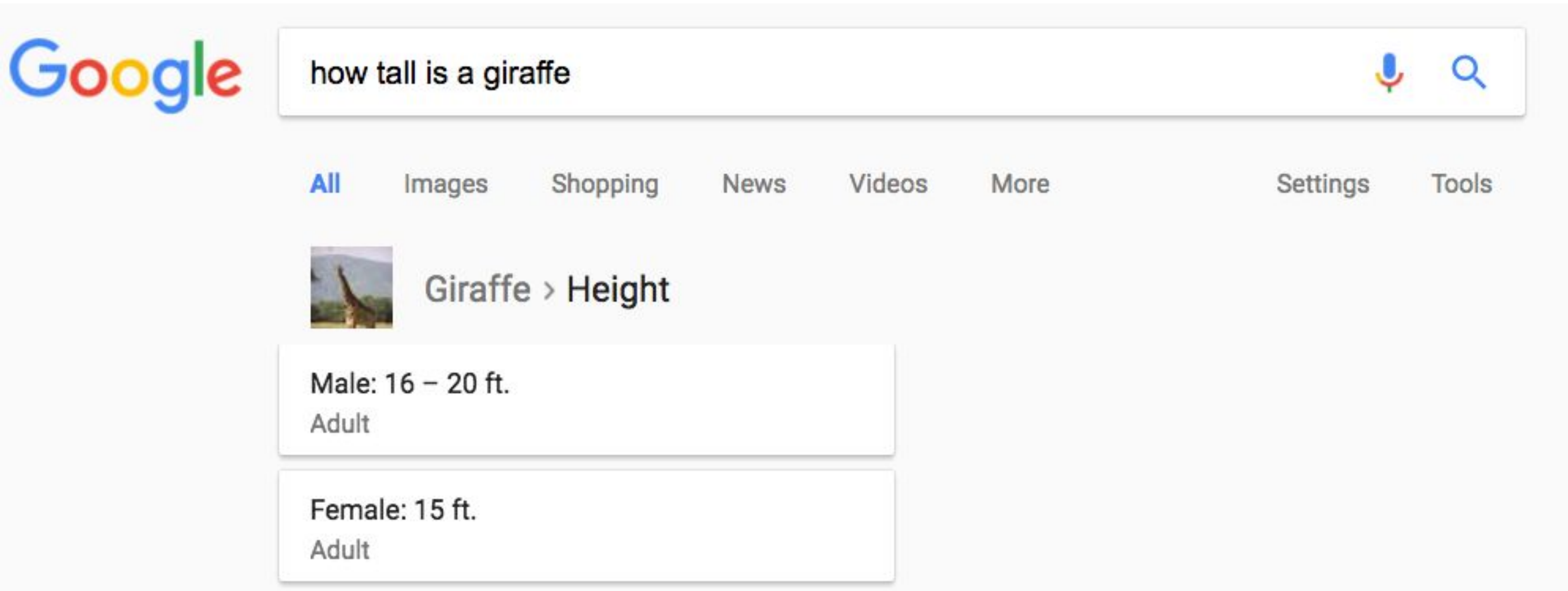
Information extraction/retrieval: Converting written representations of entities (e.g. dates) into a canonical format:

Input	**Normalized form**
November 11	11/11
the 11th of Nov.	11/11
November the eleventh	11/11

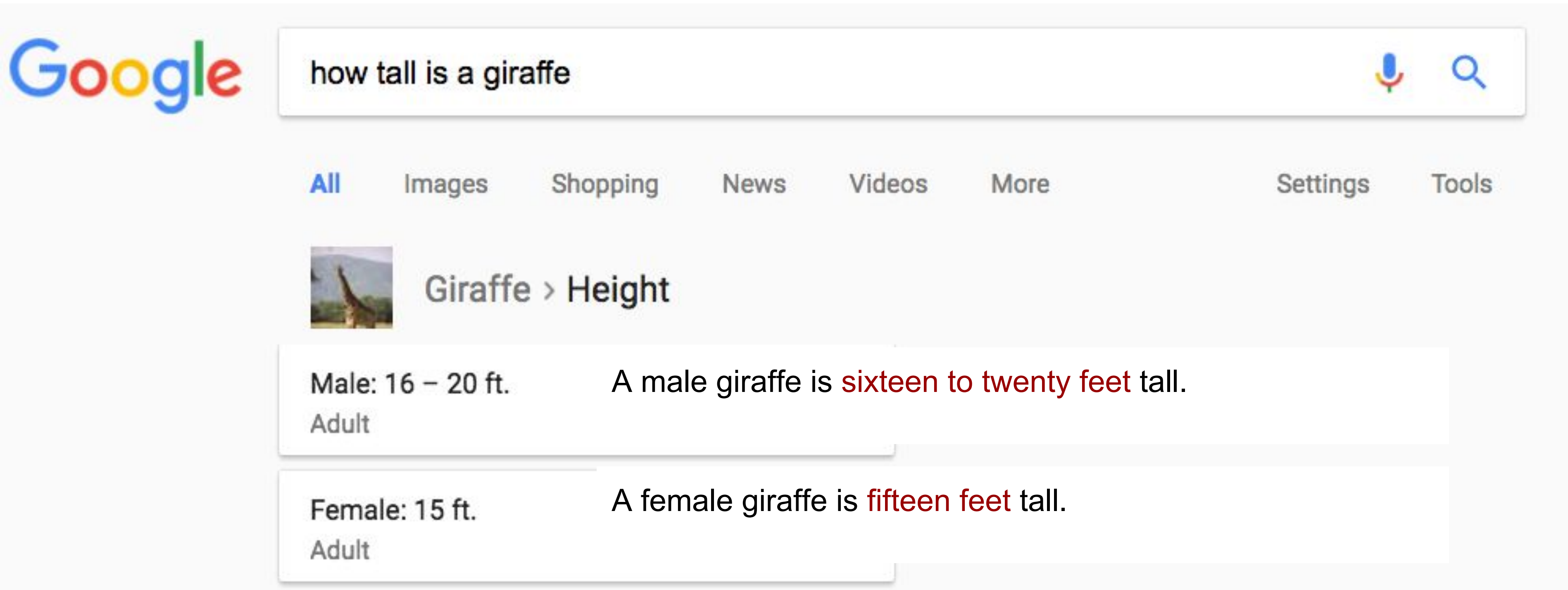
Speech applications: Converting “non-standard words” (NSWs) into a lexical representation of how people would *say* them:

Input	**Normalized form**
11/11	November the eleventh
2.5 cm	two point five centimeters
₹500 note	five hundred rupee note

Text normalization and text generation



Text normalization and text generation

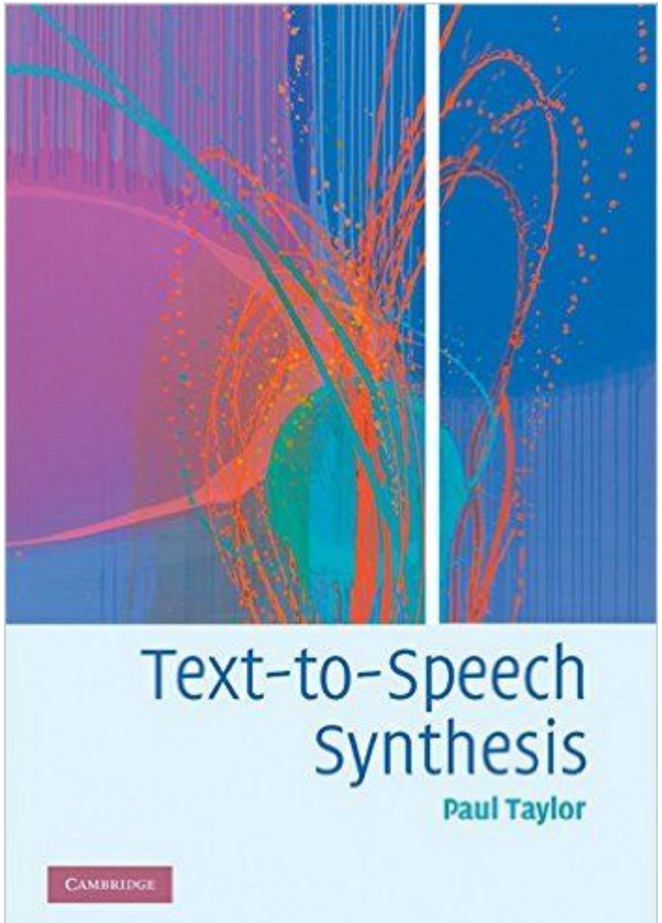


Somebody has to produce that red text - whether it’s done as part of generation or passed to TTS to expand.

But isn’t this trivial?

- The problem is that there are a great many classes of cases
- And languages with heavy inflectional morphology present a particular challenge

A bit of terminology



Paul Taylor, in his textbook on *Text-to-Speech Synthesis* (2009) refers to things like **6 ft, 150 lb**, or **3:30** as instances of *semiotic classes*

Taxonomy of semiotic classes

	EXPN	abbreviation	adv, N.Y, mph, gov't
alpha	LSEQ	letter sequence	CIA, D.C, CDs
	ASWD	read as word	CAT, proper names
	MSPL	misspelling	geogaphy
	NUM	number (cardinal)	12, 45, 1/2, 0·6
	NORD	number (ordinal)	May 7, 3rd, Bill Gates III
	NTEL	telephone (or part of)	212 555-4523
	NDIG	number as digits	Room 101
N	NIDE	identifier	747, 386, I5, pc110, 3A
U	NADDR	number as street address	5000 Pennsylvania, 4523 Forbes
M	NZIP	zip code or PO Box	91020
B	NTIME	a (compound) time	3·20, 11:45
E	NDATE	a (compound) date	2/2/99, 14/03/87 (or US) 03/14/87
R	NYER	year(s)	1998, 80s, 1900s, 2003
S	MONEY	money (US or other)	$3·45, HK$300, Y20,000, $200K
	BMONEY	money tr/m/billions	$3·45 billion
	PRCT	percentage	75%, 3·4%
	SPLT	mixed or "split"	WS99, x220, 2-car
			(see also SLNT and PUNC examples)
	SLNT	not spoken,	word boundary or emphasis character:
M		word boundary	M.bath, KENT*RLTY, _really_
I	PUNC	not spoken,	non-standard punctuation: "***" in
S		phrase boundary	$99,9K***Whites, "..." in DECIDE...Year
C	FNSP	funny spelling	slloooooww, sh*t
	URL	url, pathname or email	http://apj.co.uk, /usr/local, phj@tpt.com
	NONE	should be ignored	ascii art, formatting junk

From Sproat, R. et al (2001), "Normalization of non-standard words." *Computer Speech and Language.*

Taxonomy of semiotic classes

	EXPN	abbreviation	*adv, N.Y, mph, gov't*
alpha	LSEQ	letter sequence	*CIA, D.C, CDs*
	ASWD	read as word	*CAT*, proper names
	MSPL	misspelling	*geogaphy*
	NUM	number (cardinal)	*12, 45, 1/2, 0·6*
	NORD	number (ordinal)	*May 7, 3rd, Bill Gates III*
	NTEL	telephone (or part of)	*212 555-4523*
	NDIG	number as digits	*Room 101*
N	NIDE	identifier	*747, 386, I5, pc110, 3A*
U	NADDR	number as street address	*5000 Pennsylvania, 4523 Forbes*
M	NZIP	zip code or PO Box	*91020*
B	NTIME	a (compound) time	*3·20, 11:45*
E	NDATE	a (compound) date	*2/2/99, 14/03/87* (or US) *03/14/87*
R	NYER	year(s)	*1998, 80s, 1900s, 2003*
S	MONEY	money (US or other)	*$3·45, HK$300, Y20,000, $200K*
	BMONEY	money tr/m/billions	*$3·45 billion*
	PRCT	percentage	*75%, 3·4%*
	SPLT	mixed or "split"	*WS99, x220, 2-car* (see also SLNT and PUNC examples)
	SLNT	not spoken, word boundary	word boundary or emphasis character: *M.bath, KENT*RLTY, _really_*
M I S C	PUNC	not spoken, phrase boundary	non-standard punctuation: "***" in *$99,9K***Whites*, "..." in *DECIDE...Year*
	FNSP	funny spelling	*slloooooww, sh*t*
	URL	url, pathname or email	*http://apj.co.uk, /usr/local, phj@tpt.com*
	NONE	should be ignored	ascii art, formatting junk

Some other cases:

- Seasons/episodes S02E02
- Ratings: 4.5/5, **** (four stars)
- Chess notation: Nc6, Rxc6
- Vision: 20/20
- ...

Sproat & van Esch, 2017, "An Expanded Taxonomy of Semiotic Classes for Text Normalization", *Interspeech*

Sometimes verbalization rules can be very specific

3:03
세시 삼분
se si sam bun [native vs Sino-Korean]
three hour three minute

Sometimes verbalization rules can be very specific

3:03
세시 삼분
se si sam bun [native vs Sino-Korean]
three hour three minute

조폭마누라 3
jopok manura 3
My Wife is a Gangster 3

Image from https://en.wikipedia.org/wiki/File:My_Wife_is_a_Gangster_movie_poster.jpg. We believe this constitutes fair use.

Sometimes verbalization rules can be very specific

3:03
세시 삼분
se si sam bun [native vs Sino-Korean]
three hour three minute

조폭마누라 3 3 → 쓰리
jopok manura 3 *seuri* [English]
My Wife is a Gangster 3 three

Image from https://en.wikipedia.org/wiki/File:My_Wife_is_a_Gangster_movie_poster.jpg. We believe this constitutes fair use.

Statement of problem: text norm for speech applications

"Speak" text like the left column as in the right column:

```
A          a
baby       baby
giraffe    giraffe
is         is
6ft        six feet
tall       tall
and        and
weighs     weighs
150lb      one hundred fifty pounds
.          sil
```

Statement of problem: text norm for speech applications

“Speak” text like the left column as in the right column:

A	a
baby	baby
giraffe	giraffe
is	is
6ft	**six feet**
tall	tall
and	and
weighs	weighs
150lb	**one hundred and fifty pounds**
.	sil

On	on
11/11/2016	**november eleventh twenty sixteen**
£1	**one pound**
was	was
worth	worth
$1.26	**one dollar and twenty six cents**
.	**sil**

Statement of problem: text norm for speech applications

“Speak” text like the left column as in the right column:

A	a
baby	baby
giraffe	giraffe
is	is
6ft	**six feet**
tall	tall
and	and
weighs	weighs
150lb	**one hundred and fifty pounds**
.	sil

On	on
11/11/2016	**november eleventh twenty sixteen**
£1	**one pound**
was	was
worth	worth
$1.26	**one dollar and twenty six cents**
.	**sil**

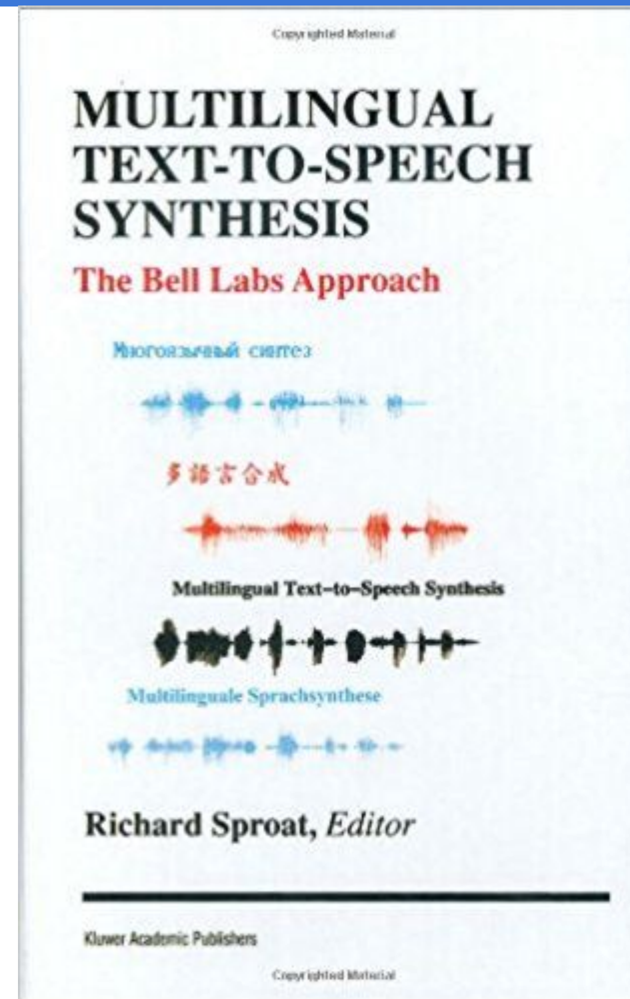
Between 7% and 9% of tokens in Wikipedia require some normalization.

The state of the art for TTS text norm … since the mid 1990's

- Carefully crafted hand-built rules compiled into (weighted) finite-state transducers**
  - The approach we used 20 years ago at Bell Labs is still used today!

**Peter Ebden and Richard Sproat. 2015. "The Kestrel TTS text normalization system", *Natural Language Engineering*, 21(3).

Also open-sourced as Sparrowhawk: https://github.com/google/sparrowhawk



Example of rules, written in Thrax*: Hindi phone numbers

```
parsed_number = (d.DIGIT util.ins_space)* d.DIGIT;

extension = m.extension ("" : " sil विस्तार sil ") parsed_number m.rec_sep;

country_code = m.country_code parsed_number m.rec_sep;

number_part = m.number_part parsed_number m.rec_sep;

number_parts = (number_part (("" : " sil ") number_part)*);

phone_number = Optimize[
   (country_code ("" : " sil "))?
   number_parts
   extension?
];
```

*http://openfst.org/twiki/bin/view/GRM/Thrax

How many rules are there?

Language	# lines of Thrax code
English	9,840
Russian	13,278
Icelandic	2,281
Hindi	4,527
Bangla	4,097
Finnish	9,145
Hungarian	3,220
Filipino (Tagalog)	4,546
Thai	7,085
Khmer	2,582

What about machine learning in text normalization?

- Some previous ML work at Google:
  - Abbreviation expansion (Roark & Sproat, 2014, ACL)
  - Letter sequence classification (Sproat & Hall, 2014, Interspeech)
  - Sentence-boundary detection
  - Homograph disambiguation (Gorman et al, forthcoming)

- *But none of these treat the whole problem*

Could we learn everything from data?

A	a
baby	baby
giraffe	giraffe
is	is
6ft	**six feet**
tall	tall
and	and
weighs	weighs
150lb	**one hundred fifty pounds**
.	sil

- Great simplicity: just need input text, and how it is spoken
- Similar to what (neural) Machine Translation does

Side note

- ASR/MT/TTS voices have had trainable systems for years
  - The point of moving to neural models is not so much simplicity as possible performance gains
- Text normalization has never been fully trainable. A neural approach allows for:
  - fully trainable system
  - ease of adaptation to new domains
  - possible performance gains once we get better data

Caveats

- NMT can rely on lots of *found* data: people translate text for a reason
    - No motivation to produce lots of verbalized text
    - (If you are thinking: what about aligned text and speech? I have a lot to say about that point…)
- ∴ we must create our own data, and we need approaches that work with the amount of data that can be reasonably hand-curated.

How much data is “reasonable”?

- 5-10 million tokens is not unreasonable to hand-curate.
- Seems like a lot … but actually we are well on the way to getting it.
- *But what I report on here depends on normalizations produced by our current TTS text normalization system, Kestrel.*

More caveats

- Neural methods work quite well overall
  - But they are prone "silly errors", like reading

  `2mA` as `two million liters`

- One approach is to constrain decoding with (finite-state) constraints

Outline of remainder of talk

- Datasets
- Baseline attention RNN model + results
- Improvements on the baseline
    - Multitask models w/ tokenization and classification
- Constraints and weak covering grammars:
- Future directions

Data

Data from English and Russian Wikipedia run through Kestrel

	Total # tokens	Training	Test
English	990M	10.5M	100K
Russian	260M	11.1M	100K

The data are open source:
https://github.com/rwsproat/text-normalization-data.

We ran a Kaggle competition based on the data (more on that below)

Data format

```
A         <self>
baby      <self>
giraffe   <self>
is        <self>
6ft       six feet
tall      <self>
and       <self>
Weighs    <self>
150lb     one hundred fifty pounds
.         sil
------------------------------------
NSA       n_letter s_letter a_letter
Williams  у_trans и_trans л_trans ь_trans я_trans м_trans с_trans
```

Data format

```
В                 <self>
1950 году         тысяча девятьсот пятидесятом году
окончил           <self>
школу             <self>
профсоюзного      <self>
движения          <self>
в                 <self>
Москве            <self>
.                 sil
```

How data is presented to RNN

- Seq-to-seq model for each token in context
- Output vocabulary fairly limited: 1-2K words

```
I live at <norm> 123 </norm> King Ave .   <= Input:  chars
            one twenty three              <= Output: words
```

How data is presented to RNN

- Seq-to-seq model for each token in context
- Output vocabulary fairly limited: 1-2K words

```
I live at 123 <norm> King </norm> Ave .   <= Input:  chars
                     <self>               <= Output: words
```

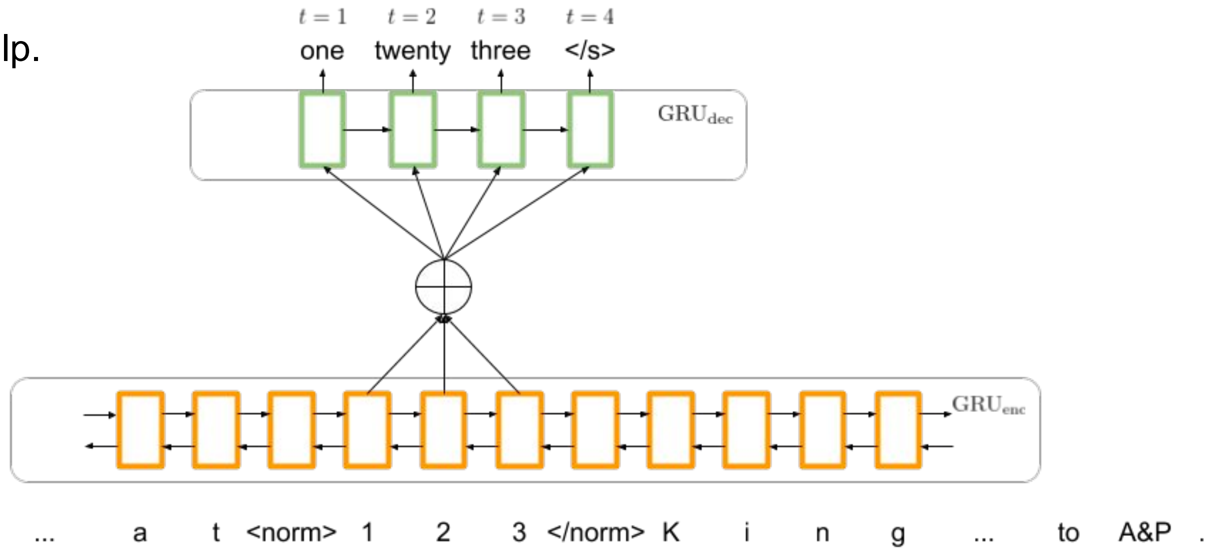
How data is presented to RNN

- Seq-to-seq model for each token in context
- Output vocabulary fairly limited: 1-2K words

```
I live at 123 King <norm> Ave </norm> .   <= Input:  chars
                          avenue          <= Output: words
```

Baseline system

- seq2seq with attention (Bahdanau et al., 2014)
- **Embedding size**: 256.
- **BiRNN**:
  - GRU, 1 layer, 256 units × 2.
- **Decoder RNN**:
  - GRU, 1 layer, 256 units.
- Larger models don't seem to help.



Baseline results (100K test examples)

	English		Russian	
ALL	92416	0.996	93184	0.994
PLAIN	68029	0.997	60747	0.999
PUNCT	17726	1.000	20263	1.000
DATE	2808	0.974	1495	0.977
TRANS	–	–	4103	0.942
LETTERS	1404	0.974	1839	0.991
CARDINAL	1067	0.991	2387	0.954
VERBATIM	894	0.977	1298	1.000
MEASURE	142	0.958	409	0.927
ORDINAL	103	0.971	427	0.981
DECIMAL	89	1.000	60	0.917
ELECTRONIC	21	0.952	2	1.000
DIGIT	37	0.703	16	1.000
MONEY	36	0.972	19	0.895
FRACTION	13	0.846	23	0.739
TIME	8	0.625	8	0.750
ADDRESS	3	1.000	–	–

Silly errors: Complex examples the RNN gets *right*

221.049 km² →

two hundred twenty one point o four nine square kilometers

24 March 1951 →

twenty fourth of march nineteen fifty one

$42,100 →

forty two thousand one hundred dollars.

Typical “silly” errors

Input	Correct	Prediction
2 mA	two milliamperes	two million liters
11/10/2008	the tenth of november two thousand eight	the tenth of october two thousand eight
1/2 cc	half a c c	one minute c c
18:00:00Z	eighteen hours zero minutes and zero seconds z	eighteen hundred cubic minutes
55th	fifty fifth	five fifth
750 вольт	семисот пятидесяти вольт	семьсот пятьдесят гектаров
750 volts	seven hundred fifty volts	seven hundred fifty hectares
70 градусами.	семьюдесятью градусами	семьюдесятью граммов
70 degrees	seventy degrees	seventy grams
16 ГБ	шестнадцати гигабайтов	шестнадцати герц
16 GB	sixteen gigabytes	sixteen hertz

Neural MT has the same issues

Input:	I come from Tunisia.
Reference:	チュニジアの出身です。
	Chunisia no shusshindesu.
	(I’m from Tunisia.)
System:	ノルウェーの出身です。
	Noruue- no shusshindesu.
	(I’m from Norway.)

Philip Arthur, Graham Neubig, Satoshi Nakamura. 2016. Incorporating Discrete Translation Lexicons into Neural Machine Translation. In *EMNLP*.

“The use of continuous representations is a major advantage, allowing NMT to share statistical power between similar words (e.g. “dog” and “cat”) or contexts (e.g. “this is” and “that is”). However, this property also has a drawback in that NMT systems often mistranslate into words that seem natural in the context, but do not reflect the content of the source sentence.”

Sinhala silly errors

Sem. class	*Inp. tok.*	Correct	*Output from the RNN*		
MEASURE	54 g	ග්‍රෑම් පනස් හතර	සැතපුම් පනස් හතර	In output 54g became 54 miles	g - ග්‍රෑම් miles - සැතපුම්
MEASURE	8ms	මිලිතත්පර අට	ගිගාබයිට් අට	In output 8ms became 8GB	ms - මිලිතත්පර GB - ගිගාබයිට්

The problem with silly errors

- Some mistakes are really bad:

  `2mA → two million liters`

- Some less so:

  `$2.50 → two dollar fifty cent`

- Guide the system away from the bad ones using grammatical constraints implemented as finite-state transducers (FSTs)

Silly errors and covering grammars

- The best way we have to counter silly errors is *overgenerating covering grammars* which constrain the decoding for some classes.
  - Crucially this depends on having a *symbolic* output
  - … which is why “end-to-end” TTS like Tacotron* or Char2Wav** will never work
- Two issues:
  - How to *learn* covering grammars
  - How to *use* covering grammars

*Wang et al. 2017. [“Tacotron: Towards end-to-end speech synthesis.”]
**Sotelo et al. 2017. [“Char2Wav: End-to-end speech synthesis.”]

Covering grammar constraints

- Guiding principle:
  - We don’t mind grammars … what we mind is spending massive resources developing grammars
- Key differences from Kestrel’s grammars:
  - Provides a set of *possible* verbalizations, rather than *the* verbalization for a given context
  - Are much easier to write
    - indeed many of them can be learned from small amounts of data

Starting point: learning numbers

- E.g.: read *123* as *one hundred twenty three*
- >70 languages with hand-built grammars
- If we know the meaning of number words:
  - *twenty* → 20 (i.e. 2 * 10^1)
  - *hundred* → 100 (i.e. 10^2)
- ...plus examples of complex number names:
  - *one hundred twenty eight* → 128
- ...then we should be able to infer a grammar

Number expression exotica (1/2)

Large powers of ten that are not powers of 1e3 (Khmer):

ប្រាំបួន	សែន	ប្រាំពិល	ម៉ឺន	ប្រាំ	ពាន់
9	1e5	7	1e4	5	1e3

`(+ (* 9 1e5) (* 7 1e4) (* 5 1e3)) =` **`975,000`**

Weak vigesimalism (French):

quatre-	*vingt-*	*dix-*	*sept*
40	20	10	7

`(+ (* 4 20) 10 7) =` **`97`**

Number expression exotica (2/2)

Creative use of *zero* (Mandarin):

萬　零　五　十

1e4 0　5　10

`(+ 1e4 0 (* 5 10)) =` **`10,050`**

Halving (Welsh):

hanner　*cant*

.5　100

`(* .5 100) =` **`50`**

Universal grammar of numbers

Fortunately, there are limits to the variation. Following Hurford (1975) we view number expressions as simple arithmetic expressions with operators (and parentheses) elided.

The most common operations are addition and multiplication:

- *dix-sept* ‘17’ (lit. ‘ten seven’): addition
- *quatre-vingt* ‘80’ (lit. ‘four twenty’): multiplication

Cues to elided structure

Within a language, there may be systematic cues for recovering the elided arithmetic structure. E.g.:

- In English and French, an expression *X Y* is usually a product if *X* < *Y* and a sum otherwise
- In Malagasy, *amby* ‘rest’ separates two addends; otherwise, it’s a multiplicand

Universal factorization covering grammar

We first build an FST A^{-1} that evaluates arithmetic expressions; e.g., with `(+ (* 4 20) 10 7)` it produces `97`. Then for a digit sequence *d*, define:

$$\Gamma(d) = \pi_o(d \circ A)$$

So Γ(`97`) might produce:

```
(+ 90 7)
(+ 80 10 7)
...
```

Verbalization grammar

We then make an FST M that deletes arithmetic markup, and define (for a lexical map L and a particular verbalization l):

$$\Delta(l) = \pi_i(M \circ L \circ l)$$

So $\Delta($`4 20 10 7`$)$ might produce:

```
(+ 4 20 10 7)
(+ 4 20 (* 10 7))
...
```

Extracting syntactic rules

Then, given a digit sequence/number expression pair (*d*, *l*), the intersection of Γ(*d*) and Δ(*l*) contains the correct factorization of *d*. In most cases this will contain exactly one path. We can use this to extract syntactic rules for number expressions:

```
S  →  (7 | 90 | * | +)
*  →  (7 | 90) 1000
+  →  90 7
```

Putting it all together

We compile the language-specific grammar into a *pushdown transducer*, henceforth *G*. Then our final model is given by:

$$\mathrm{N}(d) = \pi_o(d \circ A \circ M \circ G \circ L)$$

A : Language-universal factorization
M : Language-universal markup deletion
G : Language-specific factorization
L : Language-specific verbalization

Two types of ambiguity (1/2)

1. Expressions which contain multiplication by 1 (as in *one hundred*) or addition with 0 (as in Mandarin) are inherently ambiguous as the 1 or 0 can attach in nearly any location: We simply stipulate that +0 has the highest possible attachment and that *1 has the lowest possible attachment.

Two types of ambiguity (2/2)

2. Numbers that contain "verbal palindromes" like *two hundred two* may have multiple equivalent parses:

```
(* (+ 2 100) 2)    (* 2 (+ 100 2))
(+ (* 2 100) 2)    (+ 2 (* 100 2))
```

While only one of these is "correct", we can only know this by reference to the overall grammar. So we ignore these examples.

Inducing number name grammars

- $\mathscr{A}$: Language-independent FST that maps between digit sequences to possible arithmetic factorizations (sums of products of bases)
    - Derived from knowledge of how languages may factorize numbers
- $\mathscr{L}$: Language-dependent FST that maps from factorizations to words

Inducing language-particular number name grammars

- Given a set of training pairs ...

$\mathcal{I}$	$\mathcal{O}$
`22`	`twenty two`
`302`	`three hundred two`

- … grammar can be extracted from:

$$\pi_{output}[\mathcal{I} \circ \mathcal{A}] \cap \pi_{input}[\mathcal{L} \circ \mathcal{O}]$$

Inducing number name grammars

$\mathcal{I}$ **97**

$\mathcal{O}$ **quatre vingt dix sept**

Inducing number name grammars

$\mathcal{I}$

97

$\mathcal{A}$

`(+ 90 7), (+ 80 10 7), (+ (* 4 20) 10 7) …`

$\mathcal{O}$

`quatre vingt dix sept`

Inducing number name grammars

$\mathcal{I}$

97

$\mathcal{A}$

`(+ 90 7), (+ 80 10 7), (+ (* 4 20) 10 7) …`

`4 20 10 7`

$\mathcal{L}$

$\mathcal{O}$

`quatre vingt dix sept`

Inducing number name grammars

$\mathcal{I}$

97

$\mathcal{A}$

(+ 90 7), (+ 80 10 7), (+ (* 4 20) 10 7) …

(+ (* 4 20) 10 7), (+ 4 20 10 7), (+ 4 20 (* 10 7)) …

4 20 10 7

$\mathcal{L}$

$\mathcal{O}$

quatre vingt dix sept

Inducing number name grammars

$\mathcal{I}$

97

$\mathcal{A}$

(+ 90 7), (+ 80 10 7), **(+ (* 4 20) 10 7)** …

$\cap$

(+ (* 4 20) 10 7), (+ 4 20 10 7), (+ 4 20 (* 10 7)) …

4 20 10 7

$\mathcal{L}$

$\mathcal{O}$

quatre vingt dix sept

Inducing number name grammars

- We extract syntactic rules from the intersection, which usually contains just one analysis:

```
S  →  (7 | 10 | 4 | 20 | * | +)
*  →  4 20
+  →  * 10 7
```

- Resulting grammar $\mathcal{G}$ is combined as follows:

$$\mathcal{A} \circ \mathcal{G} \circ \mathcal{L}$$

Two remaining complexities (1/2)

We may have seen *thirteen thousand* and *fourteen million* but never *fourteen thousand* or *thirteen million*; in such a case, *G* will be deficient. To better generalize, we introduce “pre-terminals” over numerals:

```
...
teen          → (11 | 12 | 13 | ... 19)
power_of_ten → (1000 | 10000 | ... )
...
```

Two remaining complexities (2/2)

To resolve ambiguities in *L* (where needed) we compose N with a language model trained on verbalizations. This knows that in Russian we say `две тысячи` and not `два тысяч`, etc.

Training this model does not require any parallel text.

Benefits

- Learns with about 300 examples
  - Nothing like that is possible with an RNN 😁
- Currently using it to develop number name grammars for 200 languages (about 40 done so far)

Kyle Gorman and Richard Sproat, 2016, "Minimally supervised models for number name normalization," *Transactions of the Association for Computational Linguistics* 4: 507-519.

Results

Locale	Training size	Num. acc.	Morph. acc.	Overlap
eng_us	9,000	1.000	1.000	0%
	300	1.000	1.000	< 1%
kat_ge	9,000	1.000	1.000	0%
	300	1.000	1.000	< 1%
khm_kh	9,000	1.000	1.000	0%
	300	1.000	1.000	< 1%
rus_ru	28,000	1.000	1.000	56%
	9,000	1.000	0.998	0%
	300	1.000	0.998	< 1%

What happens with an RNN?

Training size	LSTM Acc.	Attention Acc.	Overlap
28,000	0.999	1.000	56%
9,000	0.994	1.000	0%
300	< 0.001	< 0.001	$< 1\%$

Covering grammars for general semiotic classes

```
Jan. 4, 1999
date|month:1|day:4|year:1999|
january the fourth nineteen ninety nine
```

Covering grammars for general semiotic classes

```
Jan. 4, 1999
date|month:1|day:4|year:1999|
january the fourth nineteen ninety nine
```

← Assume we have a tokenizer that maps to this representation

Covering grammars for general semiotic classes

```
date|month:1|day:4|year:1999|
january the fourth nineteen ninety nine
```

C: Cardinal numbers

Y: Year readings

O: Ordinal numbers

M: Markup (“date|”, “day:”, “year:” …)

L: Lexicon of month names (“month:1” = “january” …)

E: costly Levenshtein edit distance

Thrax grammar fragment

```
import 'en_year.grm' as y;
import 'number.grm' as n;

export CARDINAL = Optimize[RmWeight[n.CARDINAL_NUMBER_NAME]];

export MONTHS = Optimize[StringFile[
  'en_months.tsv']];

export ORDINAL = Optimize[RmWeight[
   n.ORDINAL_NUMBER_NAME_WITHOUT_OVERT_MARKING]];

export YEAR = y.YEAR;
```

Definition of components

- Define $T[\mathit{class}] = \varepsilon:<\mathit{class}>\ \mathit{class}\ \varepsilon:</\mathit{class}>$
- Define $D = \mathit{tags}:\varepsilon$
- Define $\mathit{Map} = (T[C] \cup T[Y] \cup T[O] \cup T[M] \cup T[L] \cup T[E])^*$
- For input i and output o:
  - Define $P = \mathit{ShortestPath}[[i \circ \mathit{Map}] \circ \pi_{\mathit{input}}[D \circ o]]$

ε	<markup>
date|	ε
ε	</markup>
ε	<month>
month:1	January
ε	</month>
ε	<markup>
|day:	ε
ε	</markup>
ε	<edit>
ε	the
ε	</edit>
ε	<ordinal>
4	fourth
ε	</ordinal>
ε	<markup>
|year:	ε
ε	</markup>
ε	<year>
1999	nineteen ninety nine
ε	</year>
ε	<markup>
|	ε
ε	</markup>

ε	<markup>
date|	ε
ε	</markup>
ε	<month>
month:1	January
ε	</month>
ε	<markup>
|day:	ε
ε	</markup>
ε	<edit>
ε	the
ε	</edit>
ε	<ordinal>
4	fourth
ε	</ordinal>
ε	<markup>
|year:	ε
ε	</markup>
ε	<year>
1999	nineteen ninety nine
ε	</year>
ε	<markup>
|	ε
ε	</markup>

- Replace tagged regions with their class in the path.

ε	<markup>
date|	ε
ε	</markup>
ε	<month>
MONTH	
ε	</month>
ε	<markup>
|day:	ε
ε	</markup>
ε	<edit>
ε	the
ε	</edit>
ε	<ordinal>
ORDINAL	
ε	</ordinal>
ε	<markup>
|year:	ε
ε	</markup>
ε	<year>
YEAR	
ε	</year>
ε	<markup>
|	ε
ε	</markup>

- Replace tagged regions with their class in the path.
- Remove markup

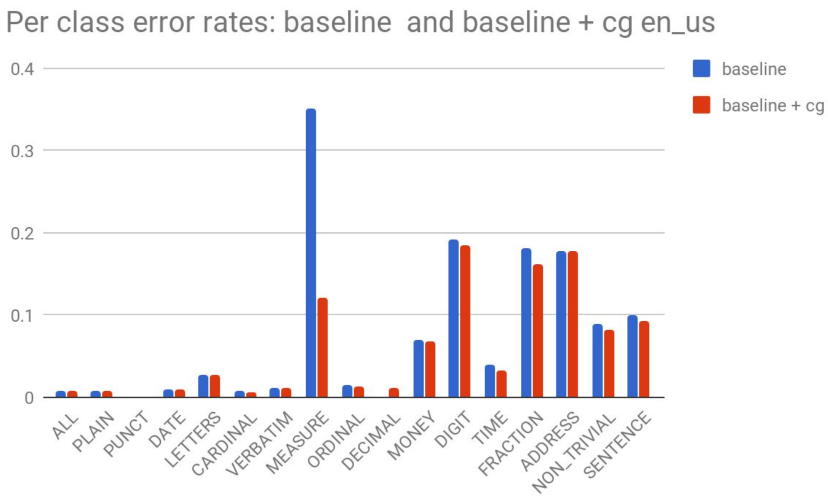
date|	ε
MONTH	
|day:	ε
ε	the
ORDINAL	
|year:	ε
YEAR	
|	ε

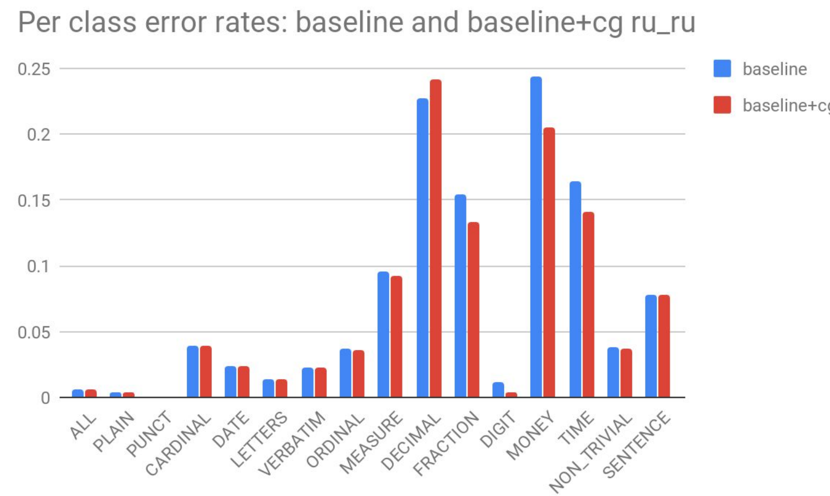
- Replace tagged regions with their class in the path.
- Remove markup

date|	ε
MONTH	
|day:	ε
ε	the
ORDINAL	
|year:	ε
YEAR	
|	ε

- Replace tagged regions with their class in the path.
- Remove markup
- Compute the union of all such paths (possibly dropping paths that do not occur a minimum # of times)
- FstReplace the classes like MONTH, ORDINAL, with the corresponding FSTs that compute the map
- The result will be the covering grammar verbalizer

Error reduction: eval on Kaggle* data, baseline system





*See below

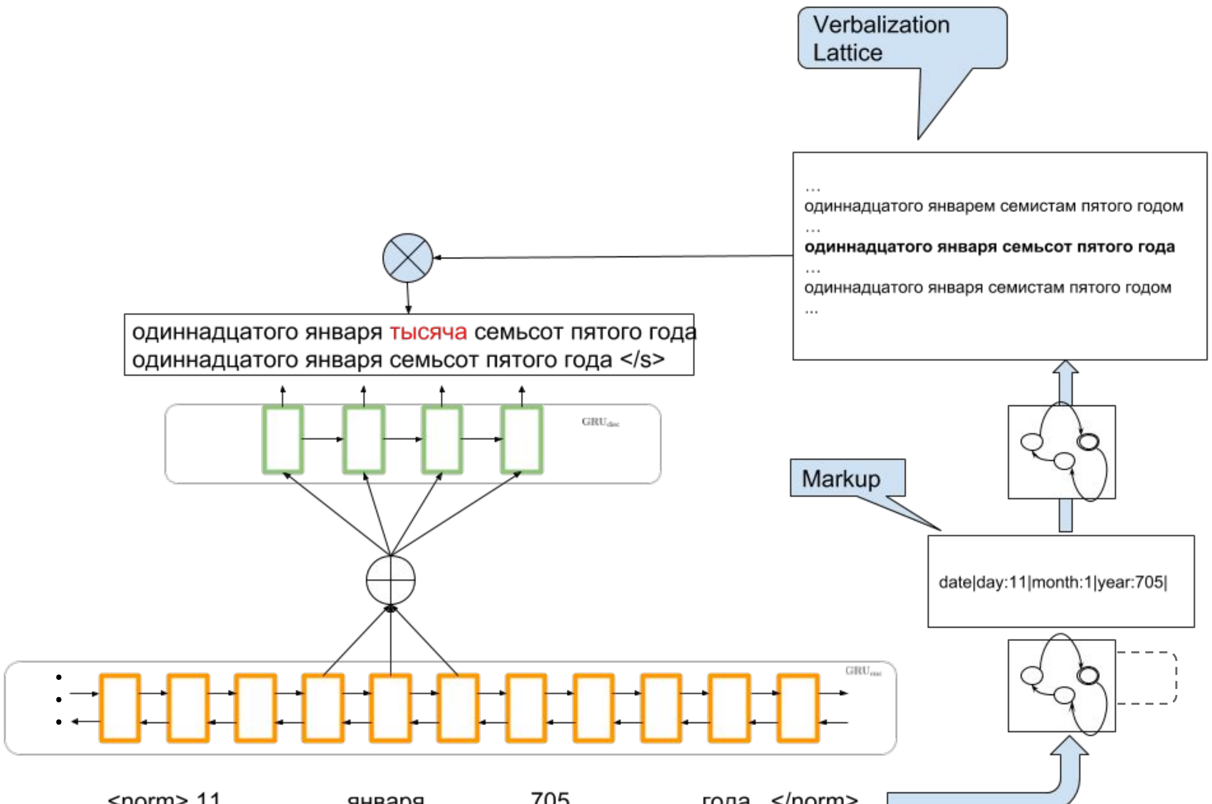
Some details on Russian

Most corrected errors (97) were silly errors:

- 14-05-2013
  - четырнадцатым мая две тысячи тринадцатого года
  - (четырнадцатым марта две тысячи тринадцатого года)
  - *fourteenth of May (March) of the two thousand thirteenth year*
- 11 апреля 678 года
  - одиннадцатое апреля шестьсот семьдесят восьмого года
  - (одиннадцатое апреля тысяча шестьсот семьдесят восьмого года)
  - *eleventh of April of the (one thousand) six hundred seventy eighth year*
- 100 mm
  - сто миллиметров
  - (сто километров)
  - *one hundred millimeters (kilometers)*

Hard constraints

- Basic idea: Constrain decoding to a smaller subset using on-the-fly intersection.



Two flavors of constraining

- Train without constraint; decode with constraint.
- Mask then softmax (i.e. locally normalize among allowed words) is wrong.
  - Distorts ranking of paths.
  - Consider: "a b c" vs "a B c" when,
    - P(a b c </s>) = P(a b c) × P(</s> | a b c) = 0.4 × 0.9
    - P(a B c </s>) = P(a B c) × P(</s> | a B c) = 0.4 × 0.5
    - i.e. P(a b c </s>) > P(a B c </s>)
  - Suppose the constraint only allows "a b c" or "a B c". Mask then softmax gives,
    - Q(a b c) = P(a b c) / (P(a b c) + P(a B c)) = 0.5 = Q(a B c)
    - Q(</s> | a b c) = Q(</s> | a B c) = 1
    - => Q(a b c </s>) = Q(a B c </s>)!
- Softmax then mask is the right thing to do.

Two flavors of constraining

- Train with constraint; decode with constraint.
- We can only mask then softmax.
  - Because global normalization in training is infeasible.
- Saves output layer parameters (16.7% reduction in ALL error rate)

Implementation details

- Hide details of neural modeling under an Acceptor interface.
    - Acceptor: deterministic weighted (non-finite) automaton.
    - start(), next(), logits(), **gather()**
- Build training/decoding logic on top of generic Acceptor interface.
    - Easily adapted for any sequence problems that can be expressed as an Acceptor.
        - Taggers
        - Shift-reduce parsers
- Add constraint as on-the-fly intersection.

```
acceptor = ...
hyps = beam_search_decode(acceptor)
loss = lm_loss(acceptor, refs)
```

```
acceptor = ...
constraint = ...
acceptor = ConstrainedAcceptor(acceptor, constraint)
hyps = beam_search_decode(acceptor)
loss = lm_loss(acceptor, refs)
```

Implications and future directions

- Neural models work well overall
- … but there are still significant challenges in the form of “silly errors”
    - Best solution (thus far) is to provide finite-state constraints (which can be learned in many cases)
    - This solution depends on the fact that we are dealing with *symbolic* output:
        - “End-to-end” TTS proposals like *Tacotron* or *char2wav* have no solution to this problem

Implications and future directions

- Inducing FS constraints remains a challenge
  - Even more important for low-resource languages
- One topic I haven’t specifically addressed:
  - Reordering: $1.50 → one dollar fifty (cents)
  - These can be handled to some extent with *pushdown transducers* but these are limited (e.g. ISO dates: *2000-05-06→May sixth two thousand*)
  - We are currently investigating a neural version of ITG’s for this purpose