Spell Once, Summon Anywhere: A Two-Level Open-Vocabulary Language Model

AAAI 2019 Technical Track

Sabrina J. Mielke *and* Jason Eisner

sjmielke@jhu.edu, jason@cs.jhu.edu
Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA

Language modeling: a generative story of text

$p(\text{the cat chased the})$

Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the})$$

Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the})$$

Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat})$$

Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$
①	`the`
②	`cat`
③	`chased`

Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding
①	`the`	
②	`cat`	
③	`chased`	

Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding $e(w)$
①	`the`	$[\ 0.2, \cdots, 0.0]$
②	`cat`	$[\ 0.4, \cdots, 0.5]$
③	`chased`	$[-0.1, \cdots, 0.2]$

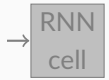
Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding $e(w)$
①	`the`	$[\ 0.2, \cdots, 0.0]$
②	`cat`	$[\ 0.4, \cdots, 0.5]$
③	`chased`	$[-0.1, \cdots, 0.2]$

Text generation with an RNN



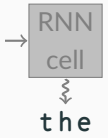
Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding $e(w)$
①	`the`	$[\ 0.2, \cdots, 0.0]$
②	`cat`	$[\ 0.4, \cdots, 0.5]$
③	`chased`	$[-0.1, \cdots, 0.2]$

Text generation with an RNN



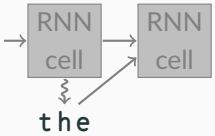
Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding $e(w)$
①	`the`	$[\ 0.2, \cdots, 0.0]$
②	`cat`	$[\ 0.4, \cdots, 0.5]$
③	`chased`	$[-0.1, \cdots, 0.2]$

Text generation with an RNN



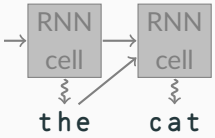
Language modeling: a generative story of text

$$p(\text{the cat chased the}) \;=\; p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding $e(w)$
①	`the`	$[\ 0.2, \cdots, 0.0]$
②	`cat`	$[\ 0.4, \cdots, 0.5]$
③	`chased`	$[-0.1, \cdots, 0.2]$

Text generation with an RNN



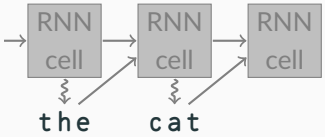
Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding $e(w)$
①	`the`	$[\ 0.2, \cdots, 0.0]$
②	`cat`	$[\ 0.4, \cdots, 0.5]$
③	`chased`	$[-0.1, \cdots, 0.2]$

Text generation with an RNN



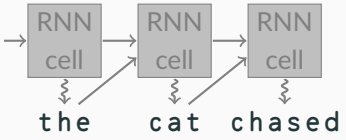
Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding $e(w)$
①	`the`	$[\ 0.2, \cdots, 0.0]$
②	`cat`	$[\ 0.4, \cdots, 0.5]$
③	`chased`	$[-0.1, \cdots, 0.2]$

Text generation with an RNN



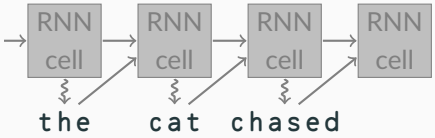
Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding $e(w)$
①	`the`	$[\ 0.2, \cdots, 0.0]$
②	`cat`	$[\ 0.4, \cdots, 0.5]$
③	`chased`	$[-0.1, \cdots, 0.2]$

Text generation with an RNN



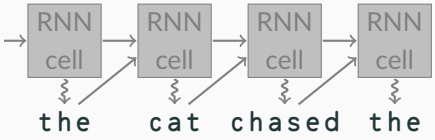
Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding $e(w)$
①	`the`	$[\ 0.2, \cdots, 0.0]$
②	`cat`	$[\ 0.4, \cdots, 0.5]$
③	`chased`	$[-0.1, \cdots, 0.2]$

Text generation with an RNN



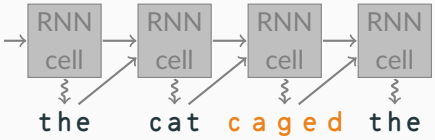
Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding $e(w)$
①	`the`	$[\ 0.2, \cdots, 0.0]$
②	`cat`	$[\ 0.4, \cdots, 0.5]$
③	`chased`	$[-0.1, \cdots, 0.2]$

Text generation with an RNN



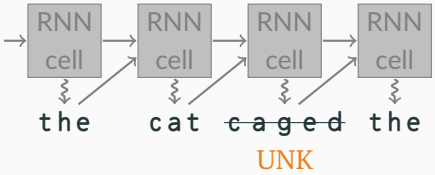
Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding $e(w)$
①	`the`	$[\ 0.2, \cdots, 0.0]$
②	`cat`	$[\ 0.4, \cdots, 0.5]$
③	`chased`	$[-0.1, \cdots, 0.2]$
④	UNK	$[\ 0.3, \cdots, 0.1]$

Text generation with an RNN



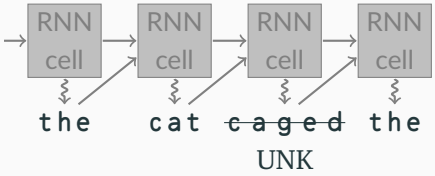
Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding $e(w)$
①	`the`	$[\ 0.2, \cdots, 0.0]$
②	`cat`	$[\ 0.4, \cdots, 0.5]$
③	`chased`	$[-0.1, \cdots, 0.2]$
④	UNK	$[\ 0.3, \cdots, 0.1]$

Text generation with an RNN



...but what is the word?

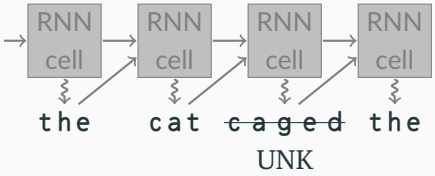
Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding $e(w)$
①	`the`	$[\ 0.2, \cdots, 0.0]$
②	`cat`	$[\ 0.4, \cdots, 0.5]$
③	`chased`	$[-0.1, \cdots, 0.2]$
④	UNK	$[\ 0.3, \cdots, 0.1]$

Text generation with an RNN



...but what is the word?

Pure character-level model as the solution?

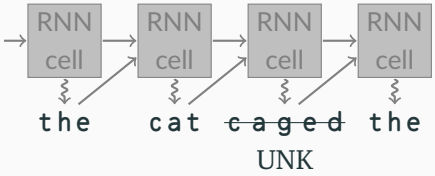
Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding $e(w)$
①	`the`	$[\ 0.2, \cdots, 0.0]$
②	`cat`	$[\ 0.4, \cdots, 0.5]$
③	`chased`	$[-0.1, \cdots, 0.2]$
④	UNK	$[\ 0.3, \cdots, 0.1]$

Text generation with an RNN



...but what is the word?

Pure character-level model as the solution?

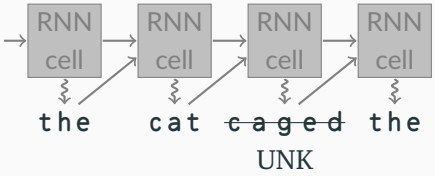
Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding $e(w)$
①	`the`	$[\ 0.2, \cdots, 0.0]$
②	`cat`	$[\ 0.4, \cdots, 0.5]$
③	`chased`	$[-0.1, \cdots, 0.2]$
④	UNK	$[\ 0.3, \cdots, 0.1]$

Text generation with an RNN



...but what is the word?

Pure character-level model as the solution?

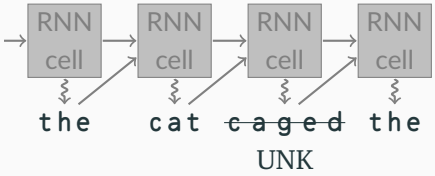
Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding $e(w)$
①	`the`	$[\ 0.2, \cdots, 0.0]$
②	`cat`	$[\ 0.4, \cdots, 0.5]$
③	`chased`	$[-0.1, \cdots, 0.2]$
④	UNK	$[\ 0.3, \cdots, 0.1]$

Text generation with an RNN



...but what is the word?

Pure character-level model as the solution?

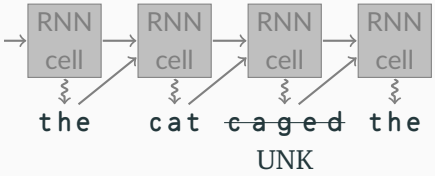
Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding $e(w)$
①	`the`	$[\ 0.2, \cdots, 0.0]$
②	`cat`	$[\ 0.4, \cdots, 0.5]$
③	`chased`	$[-0.1, \cdots, 0.2]$
④	UNK	$[\ 0.3, \cdots, 0.1]$

Text generation with an RNN



...but what is the word?

Pure character-level model as the solution?

Ugh, spelling `the` *again*...

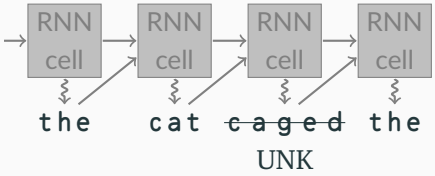
Language modeling: a generative story of text

$$p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased})$$

Lexicon / vocabulary

type w	spelling $\sigma(w)$	embedding $e(w)$
①	`the`	$[\ 0.2, \cdots, 0.0]$
②	`cat`	$[\ 0.4, \cdots, 0.5]$
③	`chased`	$[-0.1, \cdots, 0.2]$
④	UNK	$[\ 0.3, \cdots, 0.1]$

Text generation with an RNN



...but what is the word?

Pure character-level model as the solution?

Ugh, spelling `the` *again*...

...can't we memorize it?

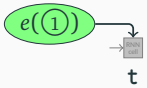
Our model: Spell once, summon anywhere

Our model: Spell once, summon anywhere

Known words only have to be spelled out once,

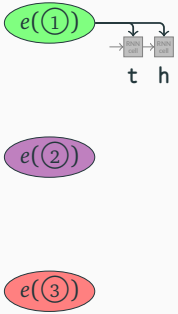
Our model: Spell once, summon anywhere

Known words only have to be spelled out once,



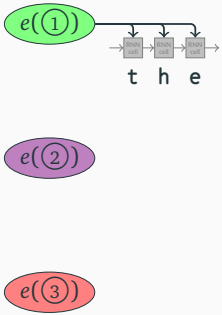
Our model: Spell once, summon anywhere

Known words only have to be spelled out once,



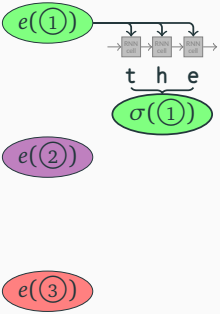
Our model: Spell once, summon anywhere

Known words only have to be spelled out once,



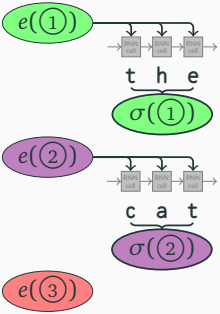
Our model: Spell once, summon anywhere

Known words only have to be spelled out once,



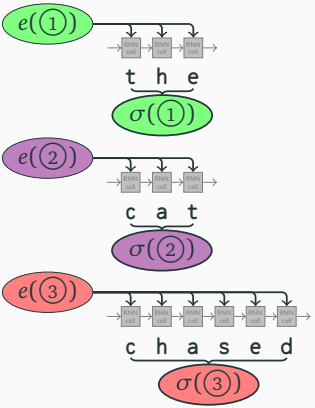
Our model: Spell once, summon anywhere

Known words only have to be spelled out once,



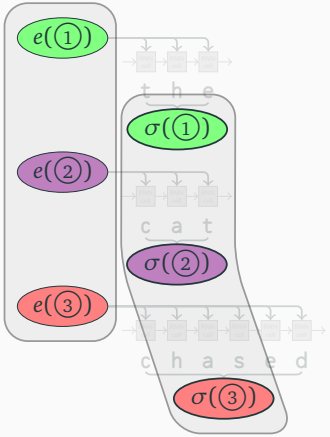
Our model: Spell once, summon anywhere

Known words only have to be spelled out once,



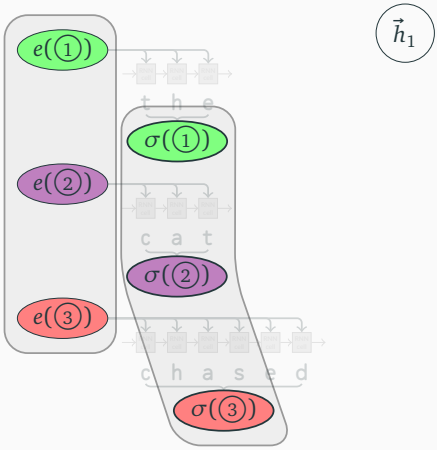
Our model: Spell once, summon anywhere

Known words only have to be spelled out once,



Our model: Spell once, summon anywhere

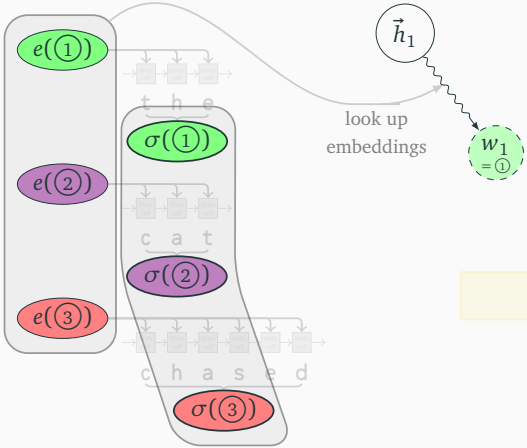
Known words only have to be spelled out once, and can then be summoned anywhere:



$\vec{h}_1$

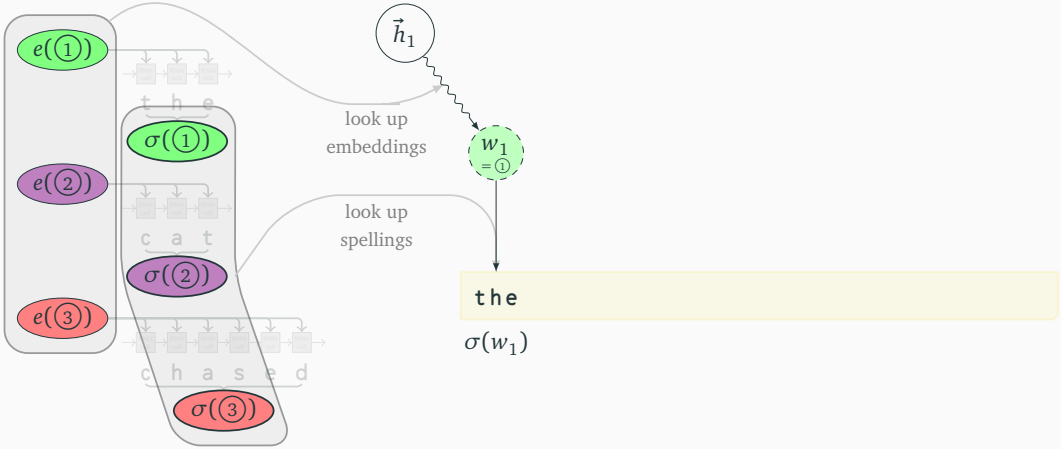
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:



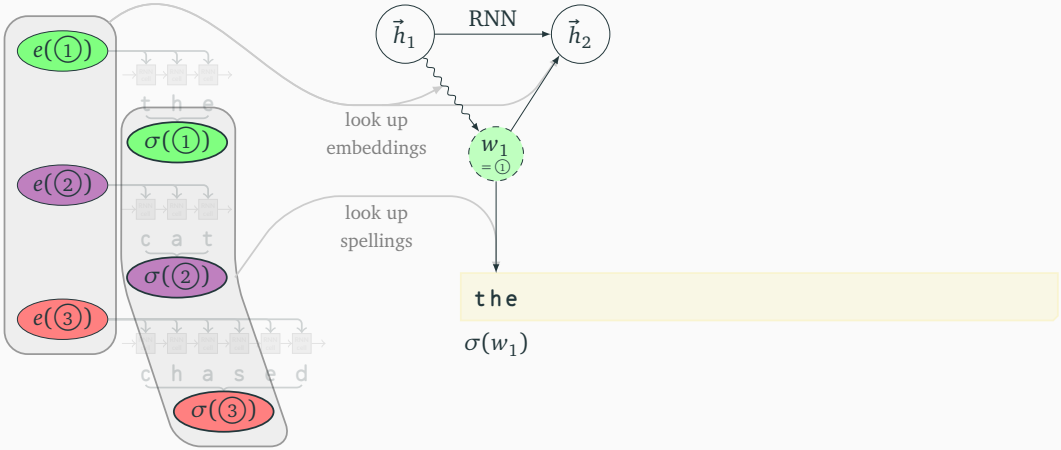
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:



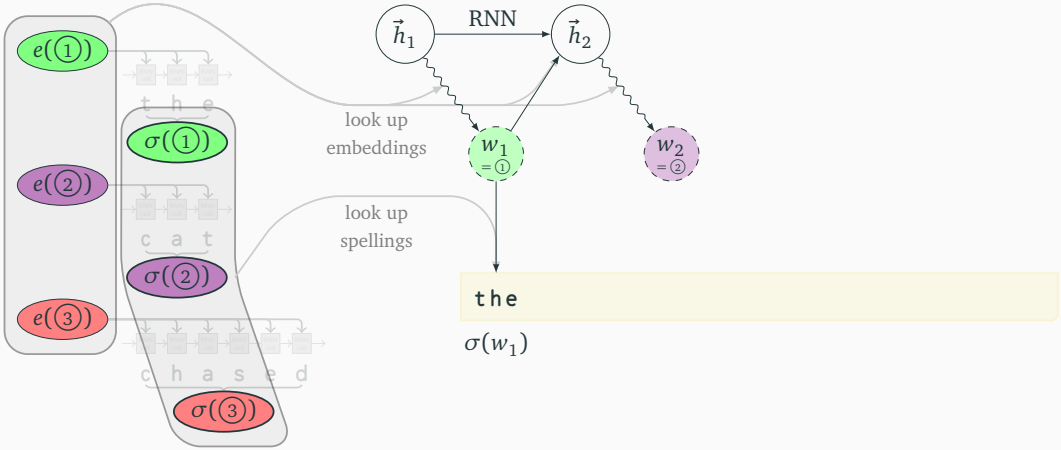
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:



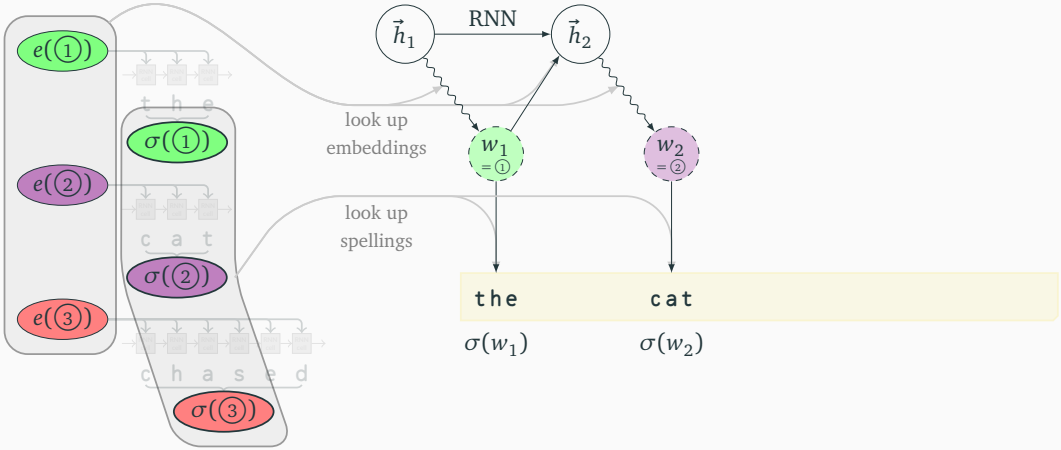
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:



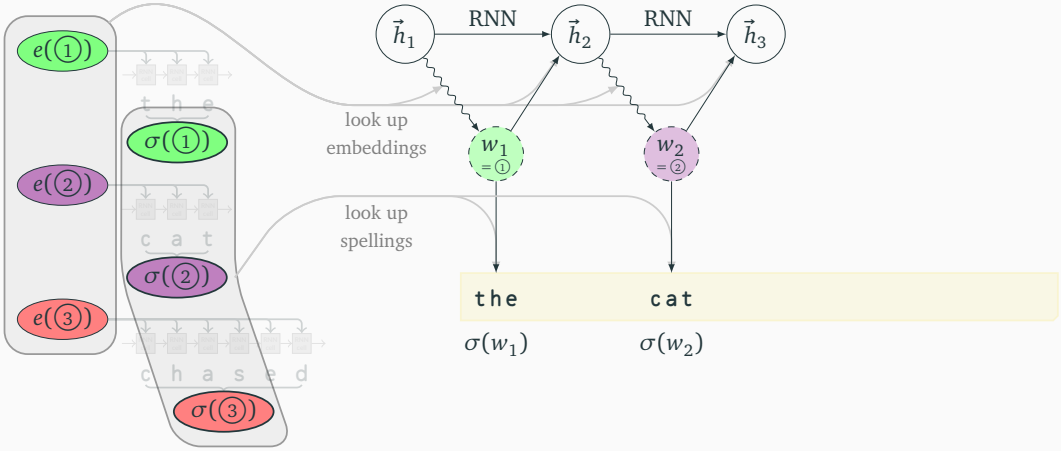
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:



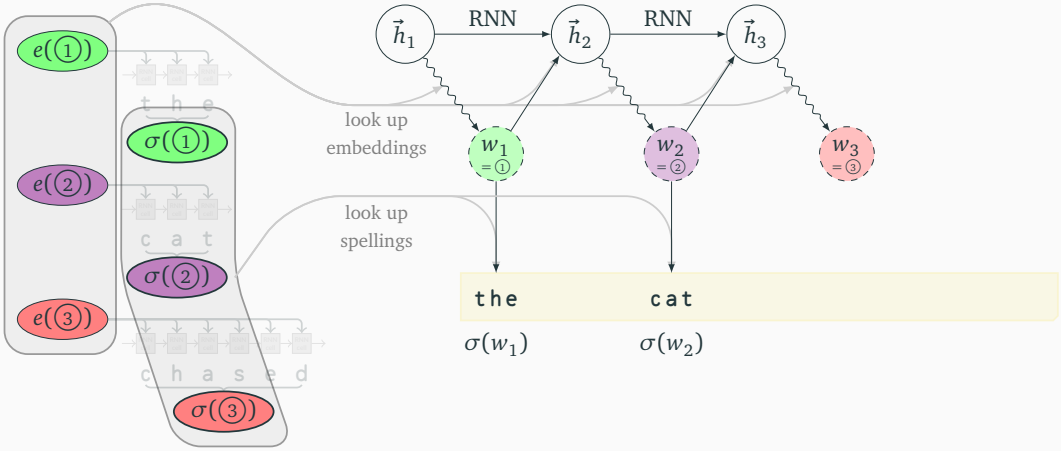
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:



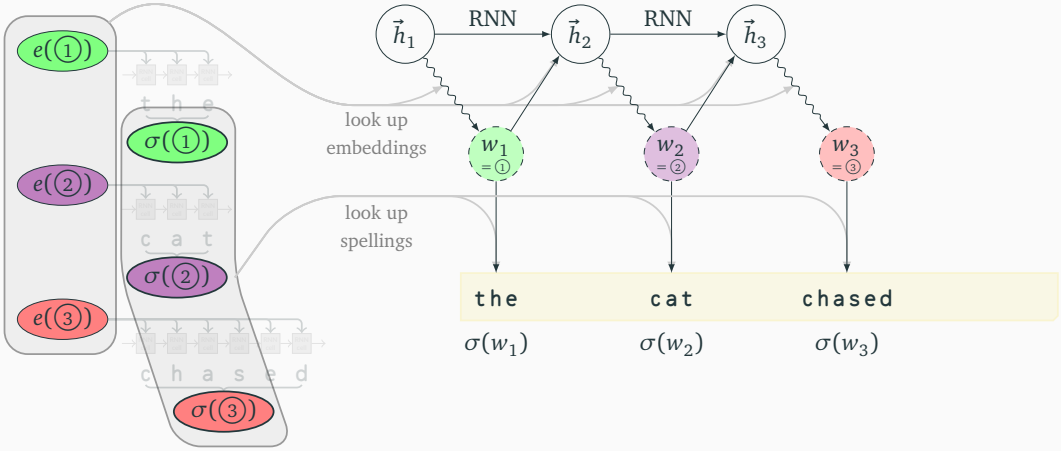
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:



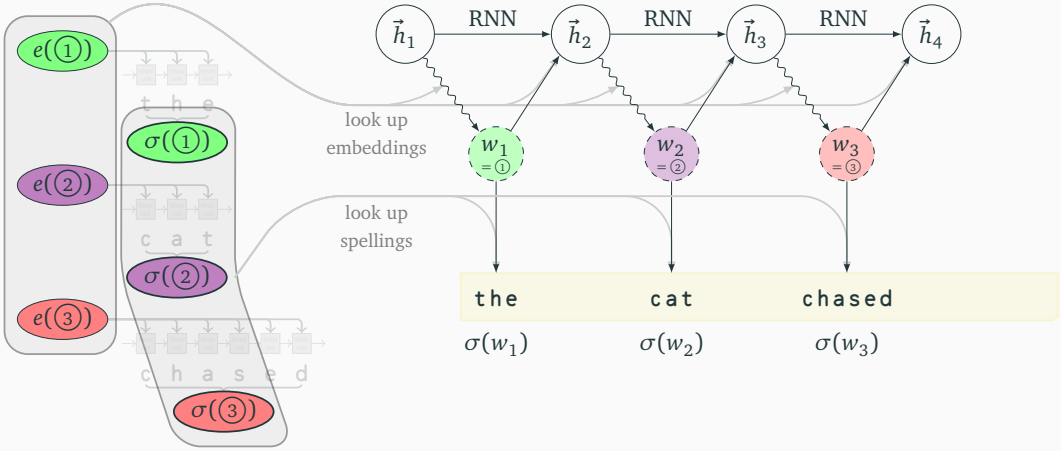
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:



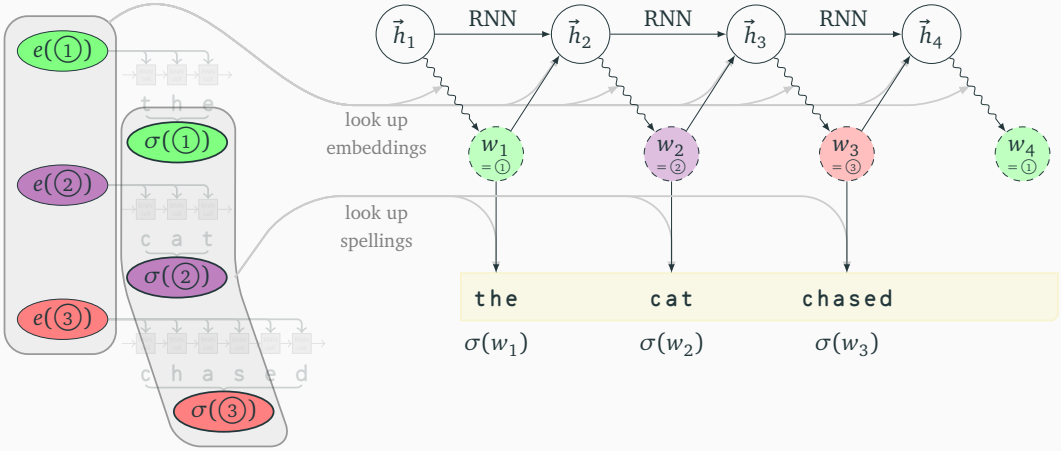
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:



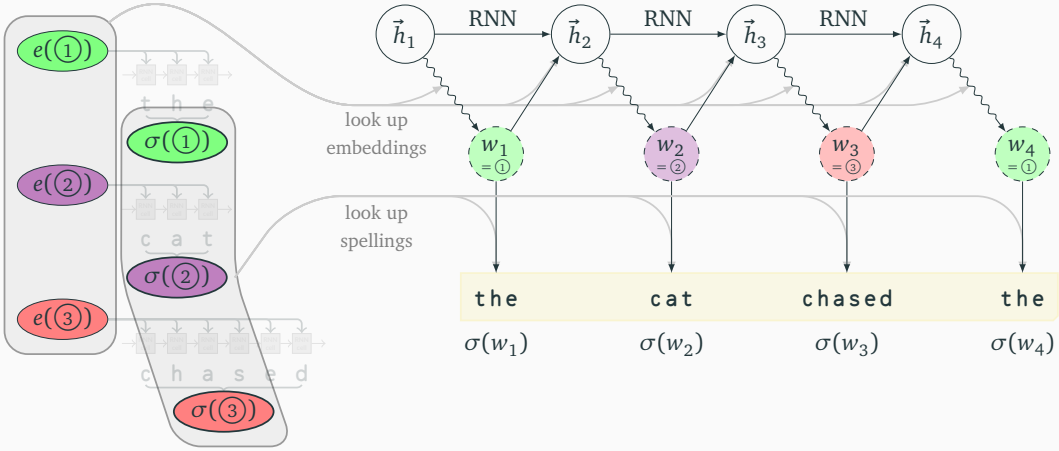
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:



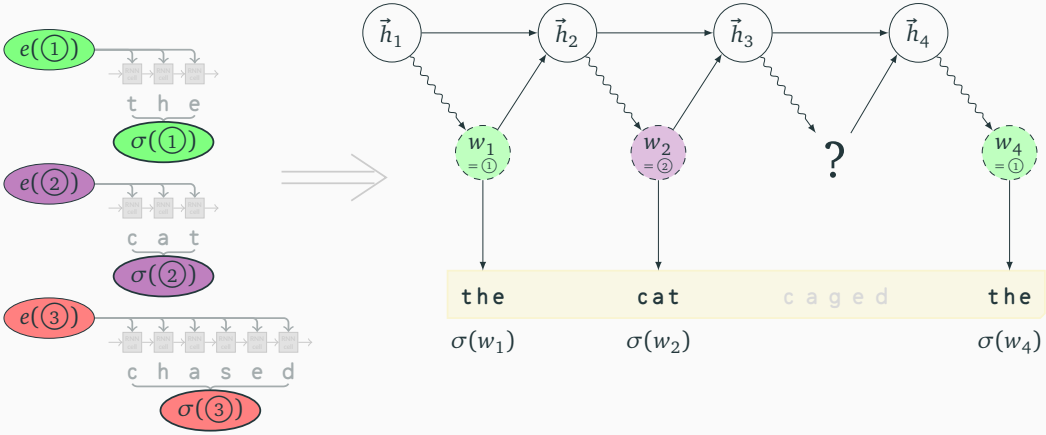
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:



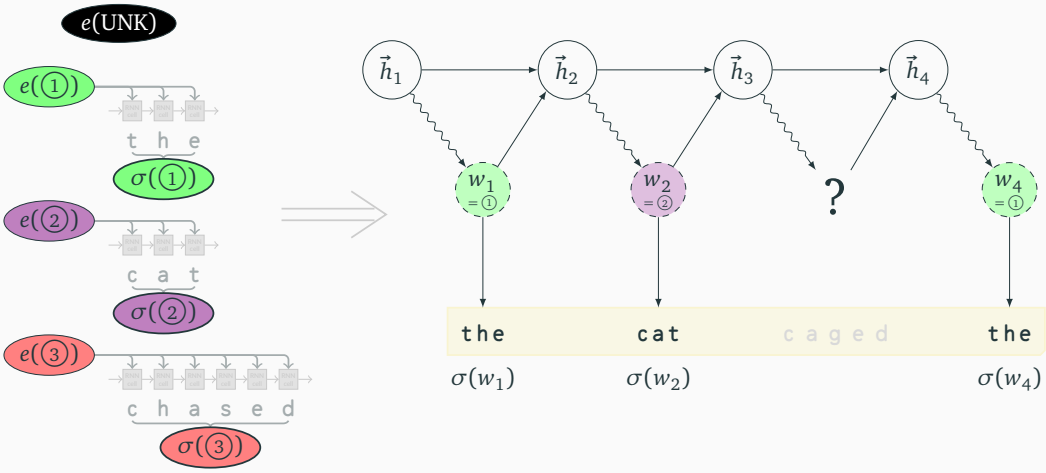
Our model: Spell once, summon anywhere – the open-vocabulary case

Known words only have to be spelled out once, and can then be summoned anywhere.



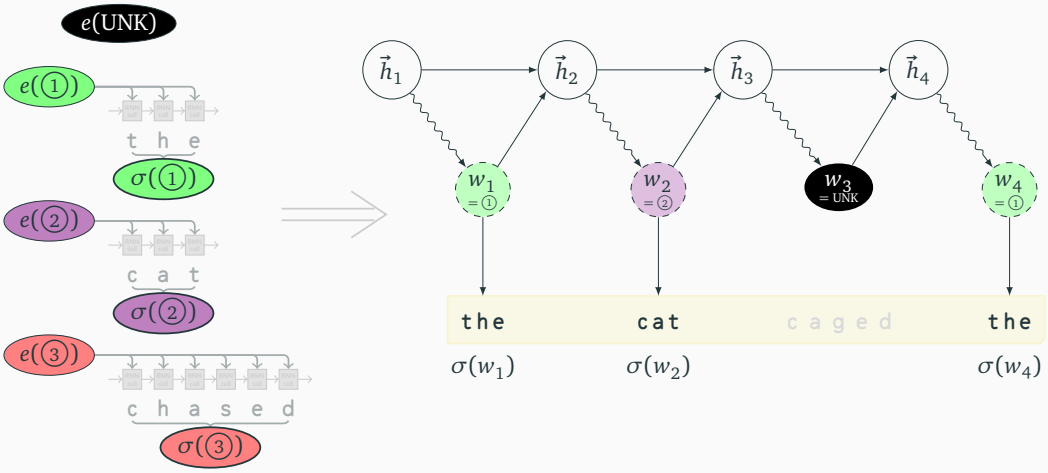
Our model: Spell once, summon anywhere – the open-vocabulary case

Known words only have to be spelled out once, and can then be summoned anywhere.



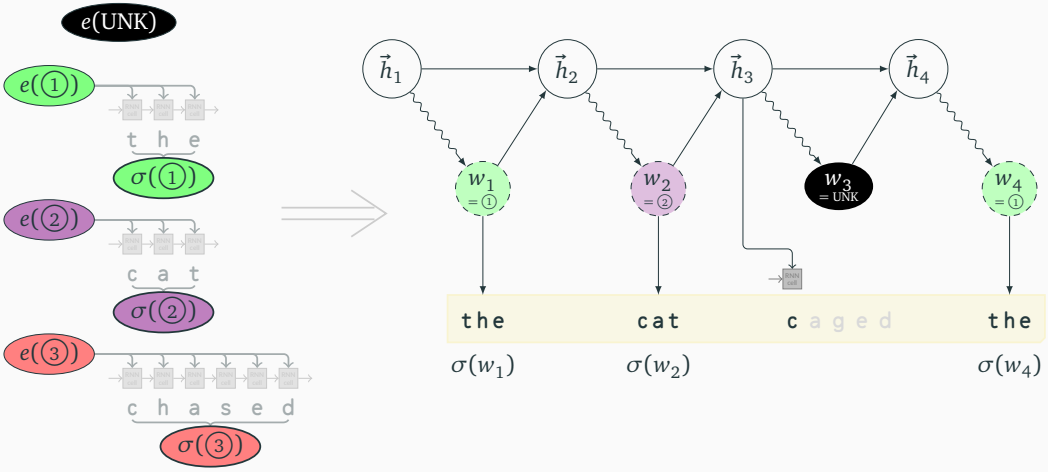
Our model: Spell once, summon anywhere – the open-vocabulary case

Known words only have to be spelled out once, and can then be summoned anywhere.



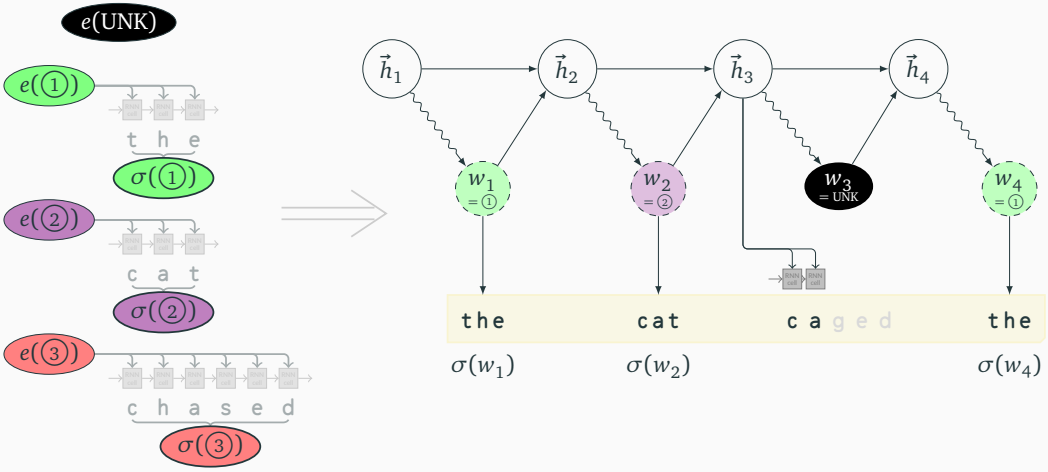
Our model: Spell once, summon anywhere – the open-vocabulary case

Known words only have to be spelled out once, and can then be summoned anywhere.
Unknown words are spelled out "on-demand" using the same character-level model.



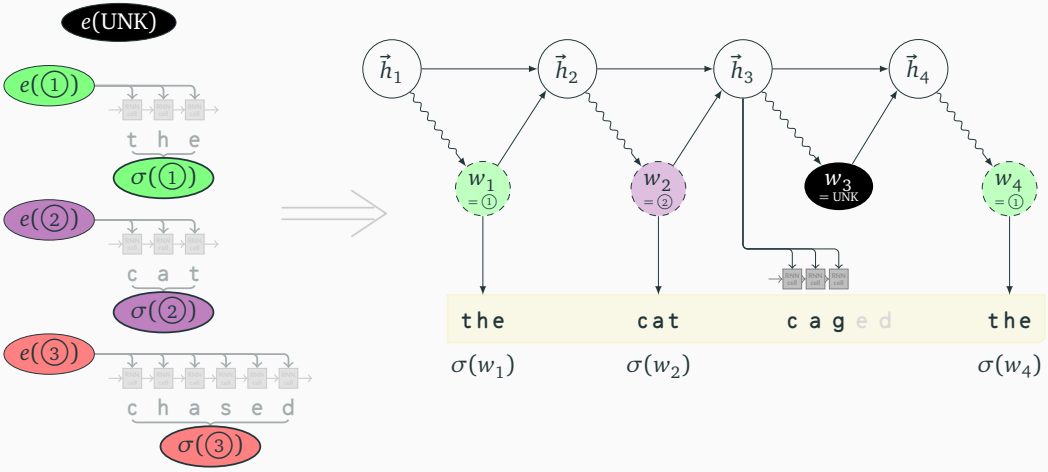
Our model: Spell once, summon anywhere – the open-vocabulary case

Known words only have to be spelled out once, and can then be summoned anywhere.
Unknown words are spelled out "on-demand" using the same character-level model.



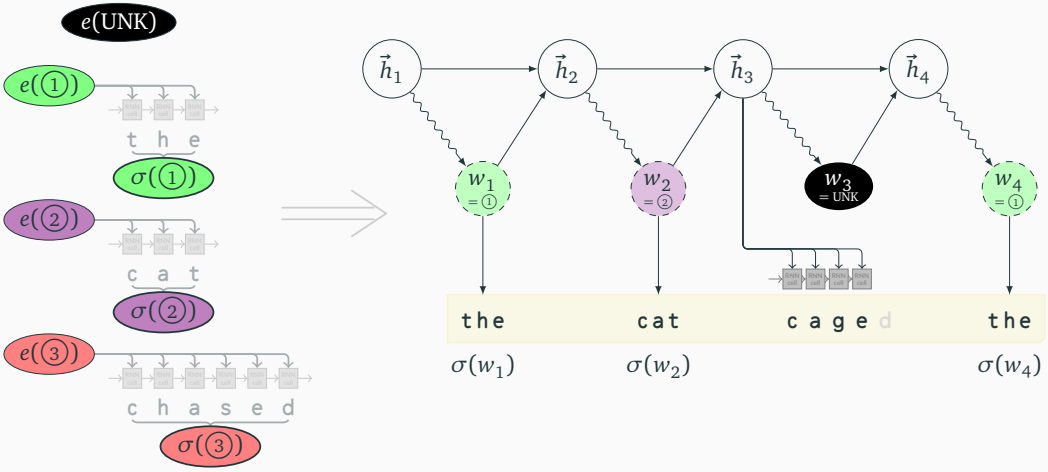
Our model: Spell once, summon anywhere – the open-vocabulary case

Known words only have to be spelled out once, and can then be summoned anywhere.
Unknown words are spelled out "on-demand" using the same character-level model.



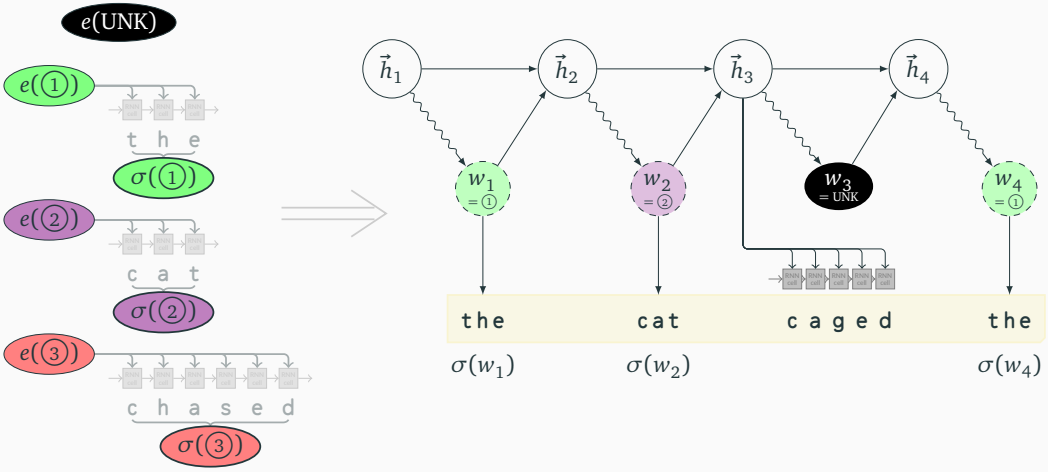
Our model: Spell once, summon anywhere – the open-vocabulary case

Known words only have to be spelled out once, and can then be summoned anywhere.
Unknown words are spelled out "on-demand" using the same character-level model.



Our model: Spell once, summon anywhere – the open-vocabulary case

Known words only have to be spelled out once, and can then be summoned anywhere.
Unknown words are spelled out "on-demand" using the same character-level model.



Samples from the model

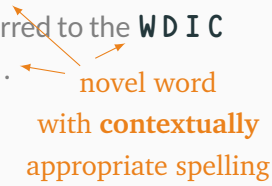
Sampled text from our model:

Following the death of Edward McCartney in `1060` , the new definition was transferred to the `WDIC` of `Fullett` .

Samples from the model

Sampled text from our model:

Following the death of Edward McCartney in `1060`, the new definition was transferred to the `WDIC` of `Fullett`.



Samples from the model

Sampled text from our model:

Following the death of Edward McCartney in `1060`, the new definition was transferred to the `WDIC` of `Fullett`.

novel word with **contextually** appropriate spelling

known spelling $\rightsquigarrow$ *novel spelling sampled from its embedding*

Samples from the model

Sampled text from our model:

Following the death of Edward McCartney in `1060`, the new definition was transferred to the `WDIC` of `Fullett`.

novel word with **contextually** appropriate spelling

known spelling	⇝	*novel spelling sampled from its embedding*
`grounded`	⇝	`stipped`

Samples from the model

Sampled text from our model:

Following the death of Edward McCartney in `1060`, the new definition was transferred to the `WDIC` of `Fullett`.

novel word with **contextually** appropriate spelling

known spelling	⇝	*novel spelling sampled from its embedding*
`grounded`	⇝	`stipped`
`differ`	⇝	`coronate`

Samples from the model

Sampled text from our model:

Following the death of Edward Mc-Cartney in `1060`, the new definition was transferred to the `WDIC` of `Fullett`.

novel word with **contextually** appropriate spelling

known spelling	⇝	*novel spelling sampled from its embedding*
`grounded`	⇝	`stipped`
`differ`	⇝	`coronate`
`Clive`	⇝	`Dickey`

Samples from the model

Sampled text from our model:

Following the death of Edward Mc-Cartney in `1060`, the new definition was transferred to the `WDIC` of `Fullett`.

novel word with **contextually** appropriate spelling

known spelling	⇝	*novel spelling sampled from its embedding*
`grounded`	⇝	`stipped`
`differ`	⇝	`coronate`
`Clive`	⇝	`Dickey`
`Southport`	⇝	`Strigger`

Samples from the model

Sampled text from our model:

Following the death of Edward McCartney in `1060`, the new definition was transferred to the `WDIC` of `Fullett`.

novel word with **contextually** appropriate spelling

known spelling	⇝	*novel spelling sampled from its embedding*
`grounded`	⇝	`stipped`
`differ`	⇝	`coronate`
`Clive`	⇝	`Dickey`
`Southport`	⇝	`Strigger`
`Carl`	⇝	`Wuly`

Samples from the model

Sampled text from our model:

Following the death of Edward McCartney in `1060`, the new definition was transferred to the `WDIC` of `Fullett`.

novel word with **contextually** appropriate spelling

known spelling	⇝	*novel spelling sampled from its embedding*
`grounded`	⇝	`stipped`
`differ`	⇝	`coronate`
`Clive`	⇝	`Dickey`
`Southport`	⇝	`Strigger`
`Carl`	⇝	`Wuly`
`Chants`	⇝	`Tranquels`

Samples from the model

Sampled text from our model:

Following the death of Edward McCartney in `1060`, the new definition was transferred to the `WDIC` of `Fullett`.

novel word with **contextually** appropriate spelling

known spelling	⇝	*novel spelling sampled from its embedding*
`grounded`	⇝	`stipped`
`differ`	⇝	`coronate`
`Clive`	⇝	`Dickey`
`Southport`	⇝	`Strigger`
`Carl`	⇝	`Wuly`
`Chants`	⇝	`Tranquels`
`valuables`	⇝	`migrations`

Samples from the model

Sampled text from our model:

Following the death of Edward McCartney in `1060` , the new definition was transferred to the `WDIC` of `Fullett` .

novel word with **contextually** appropriate spelling

known spelling	⇝	*novel spelling sampled from its embedding*
`grounded`	⇝	`stipped`
`differ`	⇝	`coronate`
`Clive`	⇝	`Dickey`
`Southport`	⇝	`Strigger`
`Carl`	⇝	`Wuly`
`Chants`	⇝	`Tranquels`
`valuables`	⇝	`migrations`

So why is this a good way of modeling language?

Linguistic notions: duality of patterning

The meaningful elements in any language—"words" in everyday parlance [...]—[...] are represented by [a] small stock of distinguishable ~~sounds~~ characters which are in themselves wholly meaningless. – Hockett, 1960

Linguistic notions: duality of patterning

The meaningful elements in any language—"words" in everyday parlance [...]—[...] are represented by [a] small stock of distinguishable ~~sounds~~ which are in themselves wholly meaningless. – Hockett, 1960

characters

"**Meaningless**" character composition should be separate from "**meaningful**" word composition!

Linguistic notions: duality of patterning

The meaningful elements in any language—"words" in everyday parlance [...]—[...] are represented by [a] small stock of distinguishable ~~sounds~~ characters which are in themselves wholly meaningless. – Hockett, 1960

"**Meaningless**" character composition should be separate from "**meaningful**" word composition!

We should need a word's spelling only to *define* it – not to later *use* it.

Duality of patterning $\mapsto$ conditional independence!

So? Why does this linguistics blurb matter?

Duality of patterning ↦ conditional independence!

So? Why does this linguistics blurb matter?

- Irregular words have uncommon spellings
  ...yet we use them like regular words!

children

Duality of patterning ↦ conditional independence!

So? Why does this linguistics blurb matter?

- Irregular words have uncommon spellings
  ...yet we use them like regular words!
- Function words have uncommon spellings
  ...yet we use them all the time without feeling weird!

children

the, of

Duality of patterning ↦ conditional independence!

So? Why does this linguistics blurb matter?

- Irregular words have uncommon spellings
  ...yet we use them like regular words!
- Function words have uncommon spellings
  ...yet we use them all the time without feeling weird!

children

the, of

Recall: We should need a word's spelling only to *define* it – not to later *use* it.

Duality of patterning ↦ conditional independence!

So? Why does this linguistics blurb matter?

- Irregular words have uncommon spellings
  ...yet we use them like regular words!
- Function words have uncommon spellings
  ...yet we use them all the time without feeling weird!

`children`

`the, of`

Recall: We should need a word's spelling only to *define* it – not to later *use* it.

↖ i.e. character-level models do it wrong!

Duality of patterning ↦ conditional independence!

So? Why does this linguistics blurb matter?

- Irregular words have uncommon spellings
  ...yet we use them like regular words!
- Function words have uncommon spellings
  ...yet we use them all the time without feeling weird!

`children`

`the, of`

Recall: We should need a word's spelling only to *define* it – not to later *use* it.

↖ i.e. character-level models do it wrong!

...and they're slow as hell...

Duality of patterning ↦ conditional independence!

So? Why does this linguistics blurb matter?

- Irregular words have uncommon spellings
  ...yet we use them like regular words!
- Function words have uncommon spellings
  ...yet we use them all the time without feeling weird!

`children`

`the, of`

Recall: We should need a word's spelling only to *define* it – not to later *use* it.

↖ i.e. character-level models do it wrong!

...and they're slow as hell...

 usage ⊥ spelling | embedding 

The arbitrariness of the sign ↦ allowing for idiosyncracy

How should a word's embedding and its spelling be connected?

The arbitrariness of the sign ↦ allowing for idiosyncracy

How should a word's embedding and its spelling be connected?

> *The connection between the signifier and the signified is arbitrary.*
>
> – de Saussure, 1916, translated

The arbitrariness of the sign ↦ allowing for idiosyncracy

How should a word's embedding and its spelling be connected?

The connection between the signifier and the signified is arbitrary.

spelling ↗ ↖ *meaning*

– de Saussure, 1916, translated

The arbitrariness of the sign ↦ allowing for idiosyncracy

How should a word's embedding and its spelling be connected?

The connection between the signifier and the signified is arbitrary.

spelling ↗ ↖ *meaning*

– de Saussure, 1916, translated

Meaning is not **fully** predictable from spellings.

The arbitrariness of the sign ↦ allowing for idiosyncracy

How should a word's embedding and its spelling be connected?

The connection between the signifier and the signified is arbitrary.

spelling ↗ ↖ *meaning*

– de Saussure, 1916, translated

Meaning is not **fully** predictable from spellings.

Example: neither `silly` nor `folly` is an adverb, even though they both end in `-ly`!

The arbitrariness of the sign ↦ allowing for idiosyncracy

How should a word's embedding and its spelling be connected?

The connection between the signifier and the signified is arbitrary.

spelling ↗ ↖ *meaning*

– de Saussure, 1916, translated

Meaning is not **fully** predictable from spellings.

Example: neither `silly` nor `folly` is an adverb, even though they both end in `-ly`!

↖ "construction" models like e(caged) := CNN(`c a g e d`) ignore this!

The arbitrariness of the sign ↦ allowing for idiosyncracy

How should a word's embedding and its spelling be connected?

The connection between the signifier and the signified is arbitrary.

spelling ↗ ↖ *meaning*

– de Saussure, 1916, translated

Meaning is not **fully** predictable from spellings.

Example: neither `silly` nor `folly` is an adverb, even though they both end in `-ly`!

↖ "construction" models like e(caged) := CNN(`c a g e d`) ignore this!

⇒ Allow any pairing a priori, but use spellings as prior / regularization!

The arbitrariness of the sign ↦ allowing for idiosyncracy

How should a word's embedding and its spelling be connected?

The connection between the signifier and the signified is arbitrary.

spelling ↗ ↖ *meaning*

– de Saussure, 1916, translated

Meaning is not **fully** predictable from spellings.

Example: neither `silly` nor `folly` is an adverb, even though they both end in `-ly`!

↖ "construction" models like $e(\text{caged}) := \text{CNN}(\texttt{c a g e d})$ ignore this!

⇒ Allow any pairing a priori, but use spellings as prior / regularization!

Outliers (`children`, `the`, ...) may have *idiosyncratic* embeddings!

The arbitrariness of the sign ↦ allowing for idiosyncracy

How should a word's embedding and its spelling be connected?

The connection between the signifier and the signified is arbitrary.

spelling ↗ ↖ *meaning*

– de Saussure, 1916, translated

Meaning is not **fully** predictable from spellings.

Example: neither `silly` nor `folly` is an adverb, even though they both end in `-ly`!

↖ "construction" models like $e(\text{caged}) := \text{CNN}(\texttt{c a g e d})$ ignore this!

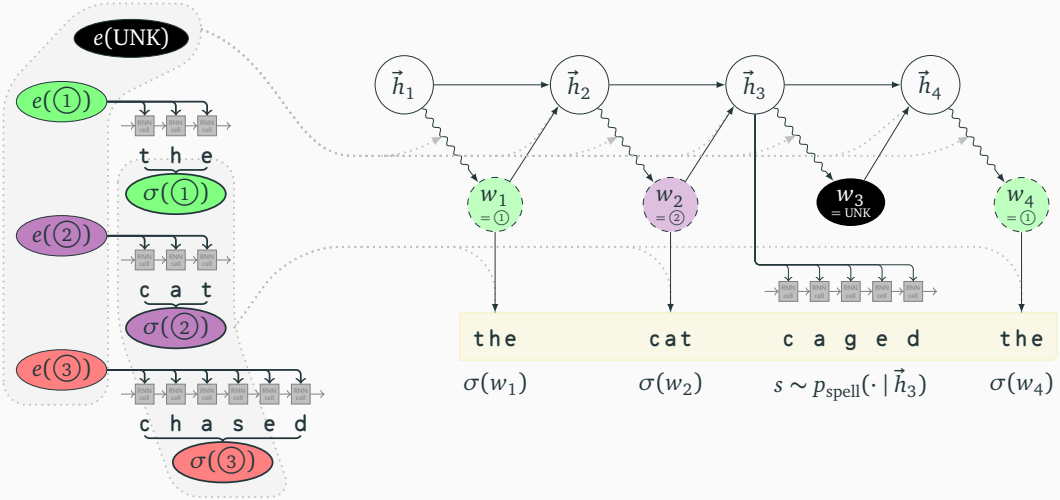
⇒ Allow any pairing a priori, but use spellings as prior / regularization!

Outliers (`children`, `the`, ...) may have *idiosyncratic* embeddings!

 regularize embeddings, don't construct them 

Recap: how does our model implement these ideas?

Embeddings and spellings are connected on the type level, ensuring conditional independence of usage and spelling while assigning positive probability to any pairing!



How do we evaluate open-vocabulary language models?

1. Report likelihood p(held-out text) as perplexity? (↓ lower is better)

How do we evaluate open-vocabulary language models?

1. Report likelihood p(held-out text) as ~~perplexity~~ bits per character (↓ lower is better)

How do we evaluate open-vocabulary language models?

1. Report likelihood p(held-out text) as ~~perplexity~~ bits per character (↓ lower is better)
2. ⚠ no UNKing allowed!

How do we evaluate open-vocabulary language models?

1. Report likelihood p(held-out text) as ~~perplexity~~ bits per character (↓ lower is better)
2. ⚠ no UNKing allowed!*

How do we evaluate open-vocabulary language models?

1. Report likelihood p(held-out text) as ~~perplexity~~ bits per character (↓ lower is better)
2. ⚠ no UNKing allowed!*

---

* Yes, we call some words "UNK" *temporarily*, but we still *generate them fully*!

How do we evaluate open-vocabulary language models?

1. Report likelihood p(held-out text) as ~~perplexity~~ bits per character (↓ lower is better)
2. ⚠ no UNKing allowed!*
   → we must predict every character of the text, regardless of vocabulary size

---

* Yes, we call some words "UNK" *temporarily*, but we still *generate them fully*!

How do we evaluate open-vocabulary language models?

1. Report likelihood p(held-out text) as ~~perplexity~~ bits per character (↓ lower is better)
2. ⚠ no UNKing allowed!*
   → we must predict every character of the text, regardless of vocabulary size

---

* Yes, we call some words "UNK" *temporarily*, but we still *generate them fully*!

⇒ A tunable "vocabulary size" hyperparameter decides what is temporary-UNK.

Results

WikiText-2 (Merity et al., 2017) 2.5 million tokenized words from the English Wikipedia	← 1.8	1.4 ‖	*test*

Results

WikiText-2 (Merity et al., 2017) 2.5 million tokenized words from the English Wikipedia	← 1.8	1.4 ‖	*test*
			1.775

Results

	← 1.8 — 1.4 ‖	*test*
WikiText-2 (Merity et al., 2017) 2.5 million tokenized words from the English Wikipedia		
		1.775
HCLM + cache *previous SOTA* (Kawakami et al., 2017)		1.500

Results

WikiText-2 (Merity et al., 2017) 2.5 million tokenized words from the English Wikipedia	← 1.8 — 1.4 ‖	*test*
		1.775
HCLM + cache *previous SOTA* (Kawakami et al., 2017)		1.500
BPE: 		1.468

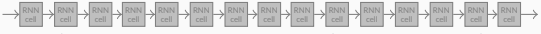
Results

WikiText-2 (Merity et al., 2017) 2.5 million tokenized words from the English Wikipedia	← 1.8 — 1.4 ‖	*test*
		1.775
HCLM + cache *previous SOTA* (Kawakami et al., 2017)		1.500
BPE: 		1.468

Results

WikiText-2 (Merity et al., 2017) 2.5 million tokenized words from the English Wikipedia	← 1.8 — 1.4 ‖	*test*
		1.775
HCLM + cache *previous SOTA* (Kawakami et al., 2017)		1.500
BPE: 		1.468
our full model: Spell Once, Summon Anywhere		**1.455**

Results

WikiText-2 (Merity et al., 2017) 2.5 million tokenized words from the English Wikipedia	on *dev* data novel words	 rare words	 frequent words	*test* all
	**3.89**	2.08	1.38	1.775
HCLM + cache *previous SOTA* (Kawakami et al., 2017)	–	–	–	1.500
BPE: 	4.01	1.70	**1.08**	1.468
our full model: Spell Once, Summon Anywhere	4.00	**1.64**	1.10	**1.455**

Results

WikiText-2 (Merity et al., 2017) 2.5 million tokenized words from the English Wikipedia	on *dev* data novel words	 rare words	 frequent words	*test* all
	**3.89**	2.08	1.38	1.775
HCLM + cache *previous SOTA* (Kawakami et al., 2017)	–	–	–	1.500
BPE: 	4.01	1.70	**1.08**	1.468
our full model: Spell Once, Summon Anywhere	4.00	**1.64**	1.10	**1.455**

...and plenty more baselines, ablations, datasets, and questions answered in the paper!

Conclusion

1. think about language before you model:
   -  usage $\perp$ spelling | embedding 
   -  regularize embeddings, don't construct them 

Conclusion

1. think about language before you model:
   - usage $\perp$ spelling | embedding
   - regularize embeddings, don't construct them
2. simple and criminally underused baselines can beat fancy but bad models
   - model strings by segments?

Conclusion

1. think about language before you model:
   - usage $\perp$ spelling | embedding
   - regularize embeddings, don't construct them
2. simple and criminally underused baselines can beat fancy but bad models
   - model strings by segments?
3. open-vocabulary language modeling is an exciting task!

Spell Once, Summon Anywhere: A Two-Level Open-Vocabulary Language Model

AAAI 2019 Technical Track

Sabrina J. Mielke *and* Jason Eisner

sjmielke@jhu.edu, jason@cs.jhu.edu
Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA