Neural Text Generation from Structured Data with Application to the Biography Domain

Remi Lebret ´
EPFL, Switzerland

David Grangier
Facebook AI Research

Michael Auli
Facebook AI Research

(EMNLP)

http://aclweb.org/anthology/D/D16/D16-1128.pdf

Presenter : Abhinav Kohar (aa18)
March 29, 2018

Outline

- Task
- Approach / Model
- Evaluation
- Conclusion

Task: Biography Generation (Concept-to-text Generation)

- Input (Fact table/Infobox) Output (Biography)

Frederick Parker-Rhodes	
Born	21 November 1914 Newington, Yorkshire
Died	2 March 1987 (aged 72)
Residence	UK
Nationality	British
Fields	Mycology, Plant Pathology, Mathematics, Linguistics, Computer Science
Known for	Contributions to computational linguistics, combinatorial physics, bit-string physics, plant pathology, and mycology
Author abbrev. (botany)	Park.-Rhodes

Frederick Parker-Rhodes
(21 March 1914 - 21 November 1987)
was an English linguist,
plant pathologist, computer scientist,
mathematician, mystic, and mycologist.

Task: Biography Generation (Concept-to-text Generation)

- Input (Fact table / Infobox)

Frederick Parker-Rhodes

Born	21 November 1914 Newington, Yorkshire
Died	2 March 1987 (aged 72)
Residence	UK
Nationality	British
Fields	Mycology, Plant Pathology, Mathematics, Linguistics, Computer Science
Known for	Contributions to computational linguistics, combinatorial physics, bit-string physics, plant pathology, and mycology
Author abbrev. (botany)	Park.-Rhodes

Output (Biography)

Frederick Parker-Rhodes
(21 March 1914 - 21 November 1987)
was an English linguist,
plant pathologist, computer scientist,
mathematician, mystic, and mycologist.

- **Characteristics of the work**:
- Using word and field embeddings along with NLM
- Scale to large # of words and fields (350 words -> 400k words)
- Flexibility (does not restrict relations between field and generated text)

Table conditioned language model

- Local and global conditioning
- Copy actions

Table conditioned language model

N-gram Language Model

$$P(s) \approx \prod_{t=1}^{T} P(w_t | c_t)$$

$$\begin{aligned} s &= w_1, \ldots, w_T \\ c_t &= w_{t-(n-1)}, \ldots, w_{t-1} \end{aligned}$$

Table conditioned language model

N-gram Language Model

$$P(s) \approx \prod_{t=1}^{T} P(w_t | c_t) \,.$$

$$s = w_1, \ldots, w_T$$
$$c_t = w_{t-(n-1)}, \ldots, w_{t-1}$$

Using Table Information

Table Conditioned Language Model

$$P(s|z, g_f, g_w) = \prod_{t=1}^{T} P(w_t | c_t, \boxed{z_{c_t}, g_f, g_w}) \,.$$

Table Information

$$P(s|z, g_f, g_w) = \prod_{t=1}^{T} P(w_t|c_t, z_{c_t}, g_f, g_w).$$

Input

Table (g_f, g_w)

name	John Doe
birthdate	18 April 1352
birthplace	Oxford UK
occupation	placeholder
spouse	Jane Doe
children	Johnnie Doe

input text (c_t, z_{c_t})

	John	Doe	(	18	April	1352	)	is	a
c_t	13944	unk	17	37	92	25	18	12	4
z_{c_t}	(name,1,2)	(name,2,1) (spouse,2,1) (children,2,1)	∅	(birthd.,1,3)	(birthd.,2,2)	(birthd.,3,1)	∅	∅	∅

output candidates $(w \in \mathcal{W} \cup \mathcal{Q})$

	the	…	april	…	placeholder	…	john	…	doe
w	1	…	92	…	5302	…	13944	…	unk
z_w	∅		(birthd.,2,2)		(occupation,1,1)		(name,1,2)		(name,2,1) (spouse,2,1) (children,2,1)

$$P(s|z, g_f, g_w) = \prod_{t=1}^{T} P(w_t|c_t, z_{c_t}, g_f, g_w).$$

Input

Table (g_f, g_w)

name	John Doe
birthdate	18 April 1352
birthplace	Oxford UK
occupation	placeholder
spouse	Jane Doe
children	Johnnie Doe

Generated Texts Information

input text (c_t, z_{c_t})

	John	Doe	(	18	April	1352	)	is	a
c_t	13944	unk	17	37	92	25	18	12	4
z_{c_t}	(name,1,2)	(name,2,1) (spouse,2,1) (children,2,1)	∅	(birthd.,1,3)	(birthd.,2,2)	(birthd.,3,1)	∅	∅	∅

output candidates $(w \in \mathcal{W} \cup \mathcal{Q})$

	the	…	april	…	placeholder	…	john	…	doe
w	1	…	92	…	5302	…	13944	…	unk
z_w	∅		(birthd.,2,2)		(occupation,1,1)		(name,1,2)		(name,2,1) (spouse,2,1) (children,2,1)

$$P(s|z, g_f, g_w) = \prod_{t=1}^{T} P(w_t|c_t, z_{c_t}, g_f, g_w).$$

Input

Table (g_f, g_w)

name	John Doe
birthdate	18 April 1352
birthplace	Oxford UK
occupation	placeholder
spouse	Jane Doe
children	Johnnie Doe

Generated Texts Information

input text (c_t, z_{c_t})

	John	Doe	(	18	April	1352	)	is	a
c_t	13944	unk	17	37	92	25	18	12	4
z_{c_t}	(name,1,2)	(name,2,1) (spouse,2,1) (children,2,1)	$\emptyset$	(birthd.,1,3)	(birthd.,2,2)	(birthd.,3,1)	$\emptyset$	$\emptyset$	$\emptyset$

output candidates $(w \in \mathcal{W} \cup \mathcal{Q})$

	the	…	april	…	placeholder	…	john	…	doe
w	1	…	92	…	5302	…	13944	…	unk
z_w	$\emptyset$		(birthd.,2,2)		(occupation,1,1)		(name,1,2)		(name,2,1) (spouse,2,1) (children,2,1)

(Output) Vocabulary Information

$$P(s|z, g_f, g_w) = \prod_{t=1}^{T} P(w_t|\boxed{c_t}, z_{c_t}, g_f, g_w).$$

Context words:
already generated tokens

input text (c_t, z_{c_t})

	John	Doe	(	18	April	1352	)	is	a
c_t	13944	unk	17	37	92	25	18	12	4
z_{c_t}	(name,1,2)	(name,2,1) (spouse,2,1) (children,2,1)	∅	(birthd.,1,3)	(birthd.,2,2)	(birthd.,3,1)	∅	∅	∅

Table (g_f, g_w)

name	John Doe
birthdate	18 April 1352
birthplace	Oxford UK
occupation	placeholder
spouse	Jane Doe
children	Johnnie Doe

output candidates $(w \in \mathcal{W} \cup \mathcal{Q})$

	the	…	april	…	placeholder	…	john	…	doe
w	1	…	92	…	5302	…	13944	…	unk
z_w	∅		(birthd.,2,2)		(occupation,1,1)		(name,1,2)		(name,2,1) (spouse,2,1) (children,2,1)

$$P(s|z, g_f, g_w) = \prod_{t=1}^{T} P(w_t|c_t, \boxed{z_{c_t}}, g_f, g_w).$$

field and position of ct :

(name, start, end)

Table (g_f, g_w)

name	John Doe
birthdate	18 April 1352
birthplace	Oxford UK
occupation	placeholder
spouse	Jane Doe
children	Johnnie Doe

input text (c_t, z_{c_t})

	John	Doe	(	18	April	1352	)	is	a
c_t	13944	unk	17	37	92	25	18	12	4
z_{c_t}	(name,1,2)	(name,2,1) (spouse,2,1) (children,2,1)	$\emptyset$	(birthd.,1,3)	(birthd.,2,2)	(birthd.,3,1)	$\emptyset$	$\emptyset$	$\emptyset$

output candidates $(w \in \mathcal{W} \cup \mathcal{Q})$

	the	…	april	…	placeholder	…	john	…	doe
w	1	…	92	…	5302	…	13944	…	unk
z_w	$\emptyset$		(birthd.,2,2)		(occupation,1,1)		(name,1,2)		(name,2,1) (spouse,2,1) (children,2,1)

Motivation Z_{ct} -Allows model to encode field specific regularity

eg: Number of date field is followed by month , Last token of name field followed by “(” or “was born”

$$P(s|z, g_f, g_w) = \prod_{t=1}^{T} P(w_t|c_t, z_{c_t}, \boxed{g_f}, \boxed{g_w}).$$

field name of the table

words of the Table

input text (c_t, z_{c_t})

	John	Doe	(	18	April	1352	)	is	a
c_t	13944	unk	17	37	92	25	18	12	4
z_{c_t}	(name,1,2)	(name,2,1) (spouse,2,1) (children,2,1)	∅	(birthd.,1,3)	(birthd.,2,2)	(birthd.,3,1)	∅	∅	∅

Table (g_f, g_w)

name	John Doe
birthdate	18 April 1352
birthplace	Oxford UK
occupation	placeholder
spouse	Jane Doe
children	Johnnie Doe

output candidates $(w \in \mathcal{W} \cup \mathcal{Q})$

	the	…	april	…	placeholder	…	john	…	doe
w	1	…	92	…	5302	…	13944	…	unk
z_w	∅		(birthd.,2,2)		(occupation,1,1)		(name,1,2)		(name,2,1) (spouse,2,1) (children,2,1)

Why G_f, G_w: fields impacts structure of generation eg: politician/athlete

Actual token helps distinguish eg: hockey player/basketball player

$$P(s|z, g_f, g_w) = \prod_{t=1}^{T} P(w_t|\boxed{c_t}, \boxed{z_{c_t}}, \boxed{g_f}, \boxed{g_w}).$$

Local Conditioning

input text (c_t, z_{c_t})

	John	Doe	(	18	April	1352	)	is	a
c_t	13944	unk	17	37	92	25	18	12	4
z_{c_t}	(name,1,2)	(name,2,1) (spouse,2,1) (children,2,1)	∅	(birthd.,1,3)	(birthd.,2,2)	(birthd.,3,1)	∅	∅	∅

Table (g_f, g_w)

name	John Doe
birthdate	18 April 1352
birthplace	Oxford UK
occupation	placeholder
spouse	Jane Doe
children	Johnnie Doe

Global Conditioning

output candidates $(w \in \mathcal{W} \cup \mathcal{Q})$

	the	…	april	…	placeholder	…	john	…	doe
w	1	…	92	…	5302	…	13944	…	unk
z_w			(birthd.,2,2)		(occupation,1,1)		(name,1,2)		(name,2,1) (spouse,2,1) (children,2,1)

Local conditioning : context dependent

Global conditioning : context independent

Copy Actions

Model can copy infobox's actual words to the output

W: Vocabulary words , Q: All tokens in table

Eg: If "Doe" is not in W, Doe will be included in Q as "name_2"

Table (g_f, g_w)

name	John Doe
birthdate	18 April 1352
birthplace	Oxford UK
occupation	placeholder
spouse	Jane Doe
children	Johnnie Doe

input text (c_t, z_{c_t})

	John	Doe	(	18	April	1352	)	is	a
c_t	13944	unk	17	37	92	25	18	12	4
z_{c_t}	(name,1,2)	(name,2,1) (spouse,2,1) (children,2,1)	∅	(birthd.,1,3)	(birthd.,2,2)	(birthd.,3,1)	∅	∅	∅

output candidates $(w \in \mathcal{W} \cup \mathcal{Q})$

	the	…	april	…	placeholder	…	john	…	doe
w	1	…	92	…	5302	…	13944	…	unk
z_w	∅		(birthd.,2,2)		(occupation,1,1)		(name,1,2)		(name,2,1) (spouse,2,1) (children,2,1)

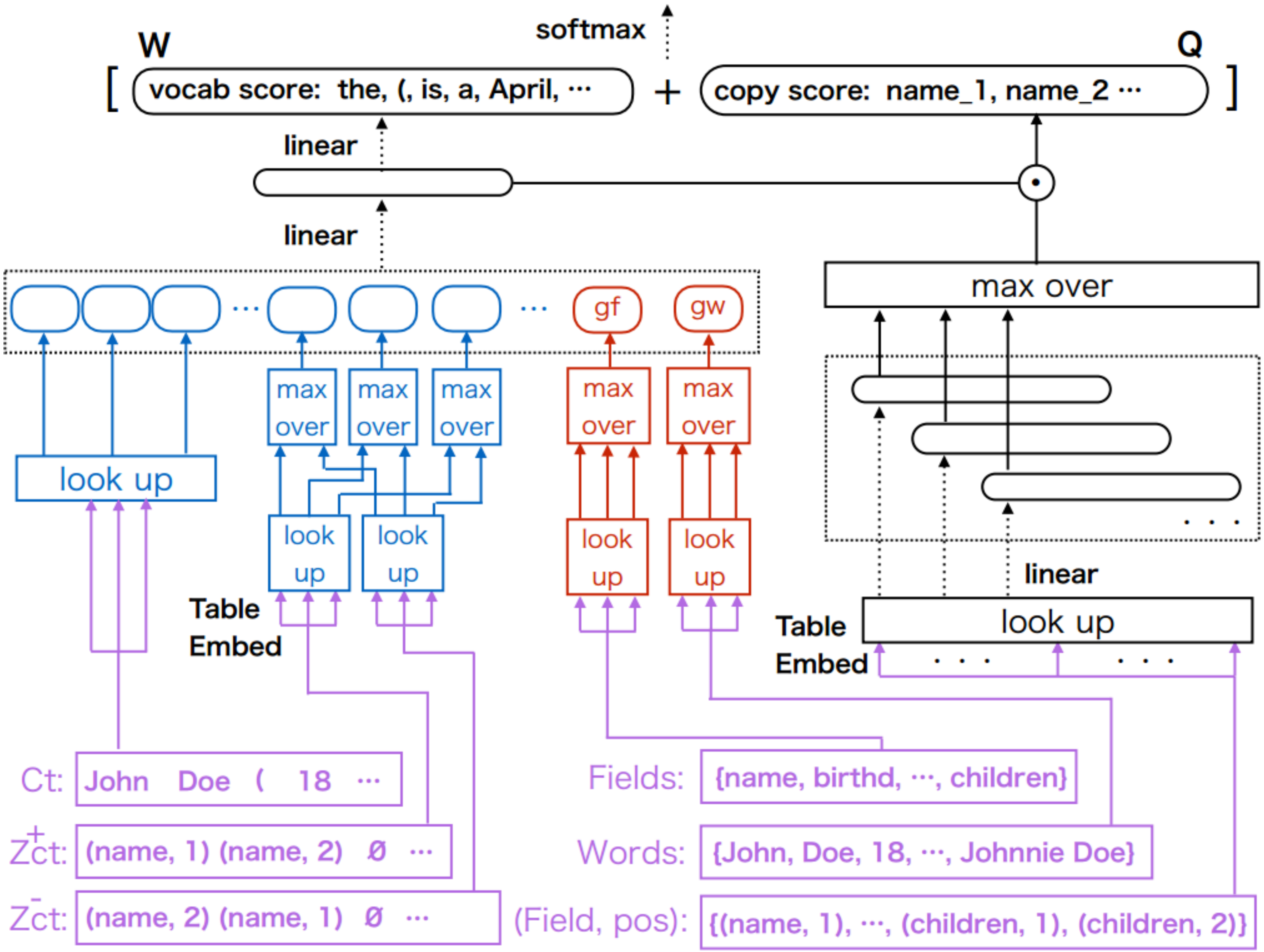
Model

$$P(s|z, g_f, g_w) = \prod_{t=1}^{T} P(w_t|c_t, z_{c_t}, g_f, g_w).$$

?

Neural Language Model

- Table conditioned language model
- Local conditioning
- Global conditioning
- Copy actions



W
softmax
Q
[vocab score: the, (, is, a, April, ··· + copy score: name_1, name_2 ···]
linear
linear
gf
gw
max over
max over
look up
look up
Table Embed
linear
Ct: John Doe (18 ···
Z+ct: (name, 1) (name, 2) Ø ···
Z-ct: (name, 2) (name, 1) Ø ···
Fields: {name, birthd, ···, children}
Words: {John, Doe, 18, ···, Johnnie Doe}
(Field, pos): {(name, 1), ···, (children, 1), (children, 2)}

Table (g_f, g_w)

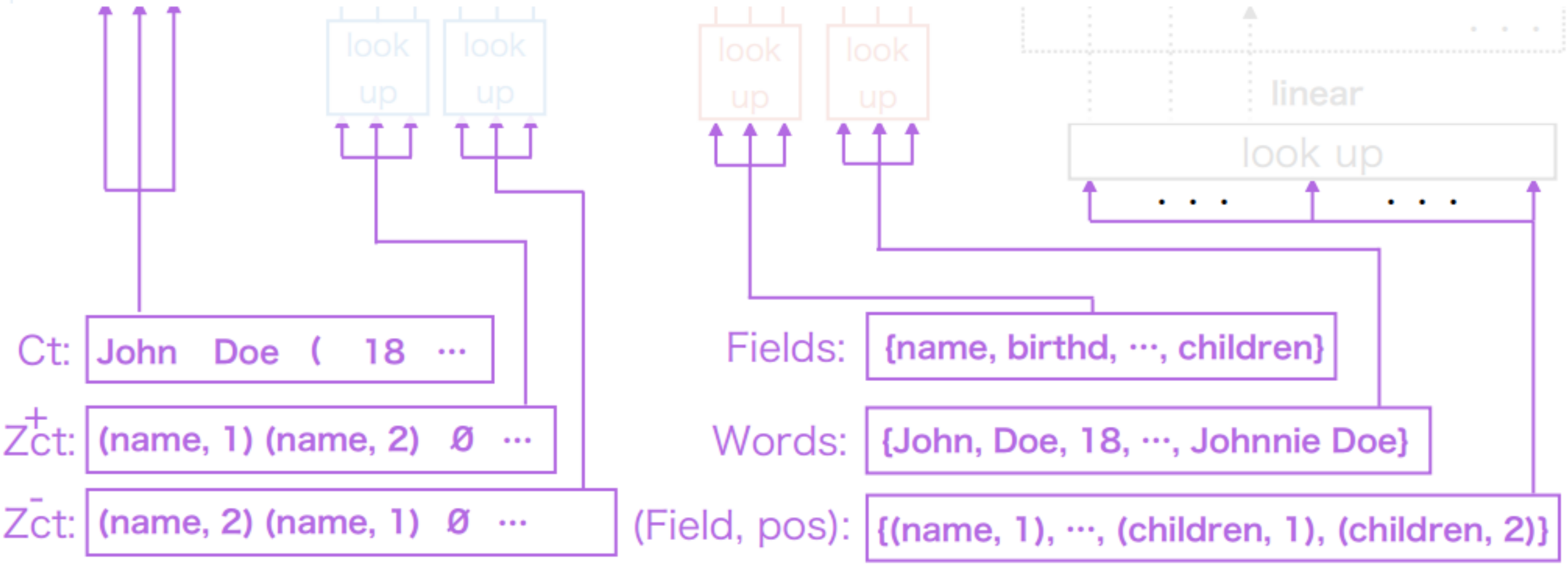
name	John Doe
birthdate	18 April 1352
birthplace	Oxford UK
occupation	placeholder
spouse	Jane Doe
children	Johnnie Doe

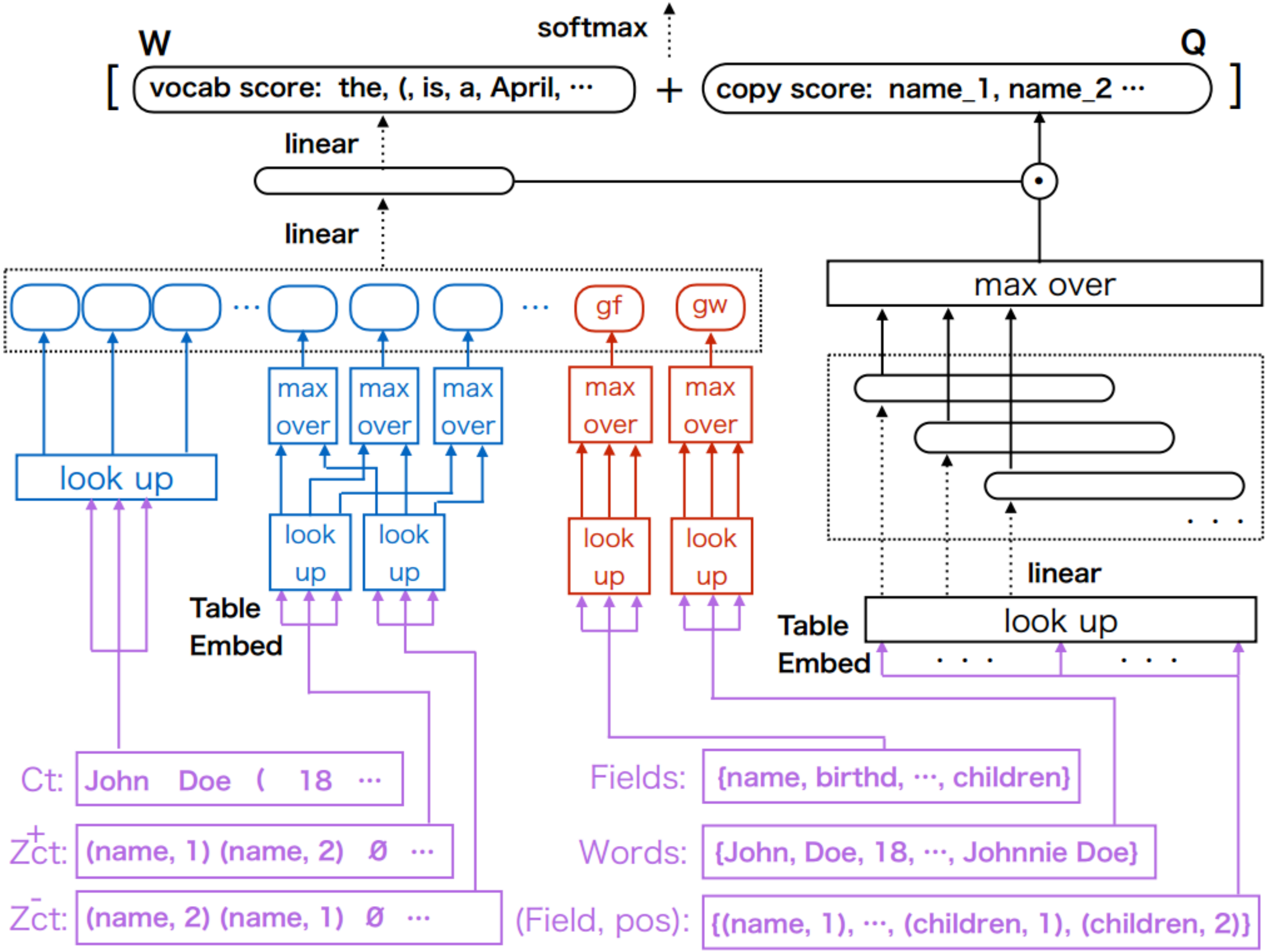
input text (c_t, z_{c_t})

	John	Doe	(	18	April	1352	)	is	a
c_t	13944	unk	17	37	92	25	18	12	4
z_{c_t}	(name,1,2)	(name,2,1) (spouse,2,1) (children,2,1)	∅	(birthd.,1,3)	(birthd.,2,2)	(birthd.,3,1)	∅	∅	∅

output candidates $(w \in \mathcal{W} \cup \mathcal{Q})$

	the	…	april	…	placeholder	…	john	…	doe
w	1	…	92	…	5302	…	13944	…	unk
z_w	∅		(birthd.,2,2)		(occupation,1,1)		(name,1,2)		(name,2,1) (spouse,2,1) (children,2,1)





softmax
W
Q
vocab score: the, (, is, a, April, ···
+
copy score: name_1, name_2 ···
linear
linear
gf
gw
max over
max over
look up
look up
Table Embed
linear
Ct: John Doe (18 ···
Z+ct: (name, 1) (name, 2) Ø ···
Z-ct: (name, 2) (name, 1) Ø ···
Fields: {name, birthd, ···, children}
Words: {John, Doe, 18, ···, Johnnie Doe}
(Field, pos): {(name, 1), ···, (children, 1), (children, 2)}

Training

- The neural language model is trained to minimize the negative log-likelihood of a training sentence *s* with stochastic gradient descent (SGD; LeCun et al. 2012) :

$$L_\theta(s) = -\sum_{t=1}^{T} \log P(w_t | c_t, z_{c_t}, g_f, g_w) \,.$$

Evaluation

- Dataset and baseline
- Result
- Quantitative Analysis

Dataset and Baseline

- Biography Dataset : WIKIBIO
  - 728,321 articles from English Wikipedia
  - Extract first "biography" sentence from each article + article infobox
- Baseline
  - Interpolated **Kneser-Ney (KN) model**
  - Replace word occurring in both table/sent with special tokens
  - Decoder emits words from regular vocab or special tokens (replace special tokens with corresponding words from table)

	Mean	Percentile	
		5%	95%
# tokens per sentence	26.1	13	46
# tokens per table	53.1	20	108
# table tokens per sent.	9.5	3	19
# fields per table	19.7	9	36

Table 2: Dataset statistics

Parameter	Value		
# word types	$\lvert\mathcal{W}\rvert$	=	20,000
# field types	$\lvert\mathcal{F}\rvert$	=	1,740
Max. # tokens in a field	l	=	10
word/field embedding size	d	=	64
global embedding size	g	=	128
# hidden units	nhu	=	256

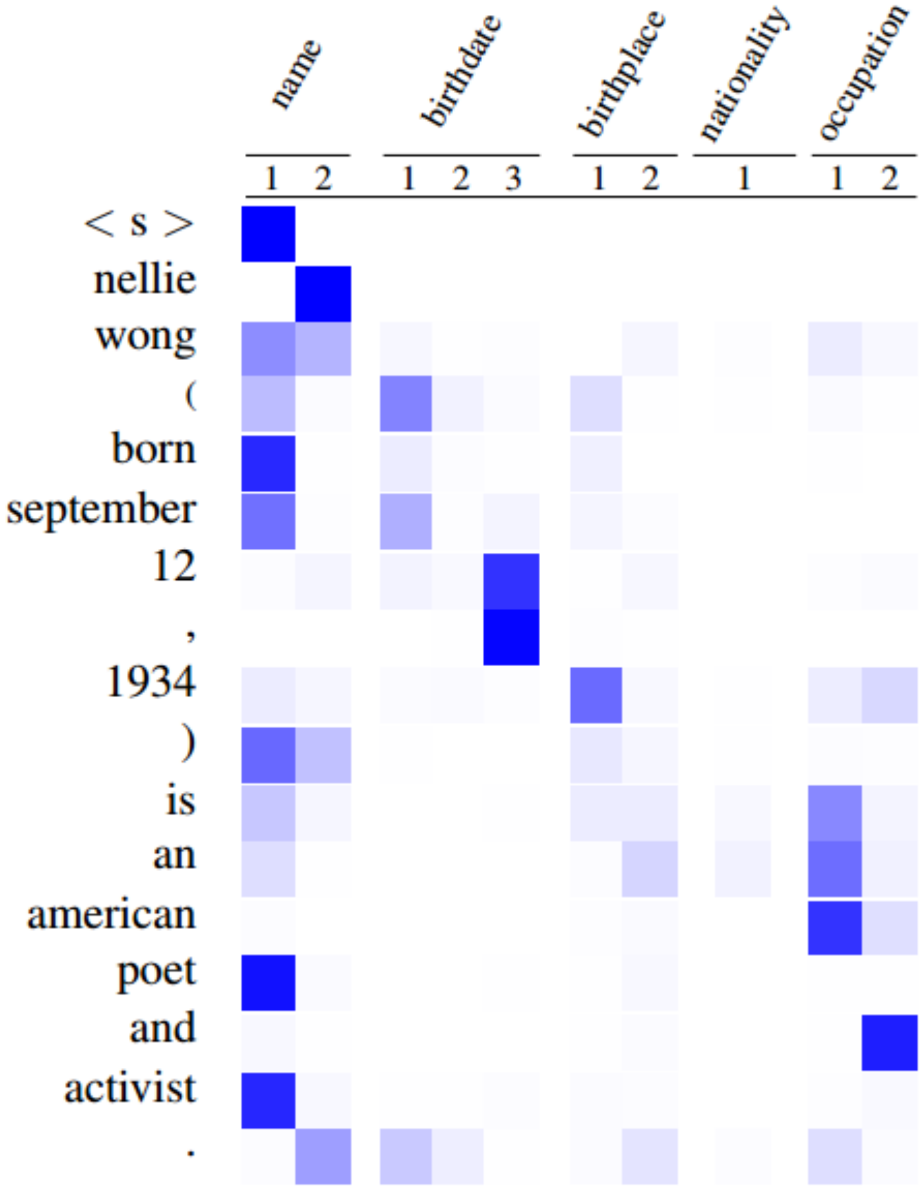
Template KN model

- The introduction section of the table in input (shown earlier):
- “name 1 name 2 (birthdate 1 birthdate 2 birthdate 3 – deathdate 1 deathdate 2 deathdate 3) was an english linguist , fields 3 pathologist , fields 10 scientist , mathematician , mystic and mycologist .”

Experimental results: Metrics

	Model	Perplexity	BLEU	ROUGE	NIST
w/o copy	KN	10.51	2.21	0.38	0.93
	NLM	9.40 ± 0.01	2.41 ± 0.33	0.52 ± 0.08	1.27 ± 0.26
	+ Local (field, start, end)	8.61 ± 0.01	4.17 ± 0.54	1.48 ± 0.23	1.41 ± 0.11
w/ copy	Template KN	$7.46^{\star}$	19.8	10.7	5.19
	Table NLM w/ Local (field, start)	$4.60 \pm 0.01^{\dagger}$	26.0 ± 0.39	19.2 ± 0.23	6.08 ± 0.08
	+ Local (field, start, end)	$4.60 \pm 0.01^{\dagger}$	26.6 ± 0.42	19.7 ± 0.25	6.20 ± 0.09
	+ Global (field)	$\mathbf{4.30 \pm 0.01}^{\dagger}$	33.4 ± 0.18	23.9 ± 0.12	7.52 ± 0.03
	+ Global (field & word)	$4.40 \pm 0.02^{\dagger}$	$\mathbf{34.7 \pm 0.36}$	$\mathbf{25.8 \pm 0.36}$	$\mathbf{7.98 \pm 0.07}$

Experimental results: Attention mechanism



- large bias to continue a field if it has not generated all tokens from table
- Nicely handles transitions between field types

  e.g. birthdate -> occupation

Quantitative analysis

Model	Generated Sentence
Reference	frederick parker-rhodes (21 march 1914 – 21 november 1987) was an english linguist, plant pathologist, computer scientist, mathematician, mystic, and mycologist.
Baseline (Template KN)	frederick parker-rhodes (born november 21 , 1914 – march 2 , 1987) was an english cricketer .
Table NLM +Local (field, start)	frederick parker-rhodes (21 november 1914 – 2 march 1987) was an australian rules footballer who played with carlton in the victorian football league (vfl) during the XXXXs and XXXXs .
+ Global (field)	frederick parker-rhodes (21 november 1914 – 2 march 1987) was an english mycology and plant pathology , mathematics at the university of uk .
+ Global (field, word)	frederick parker-rhodes (21 november 1914 – 2 march 1987) was a british computer scientist , best known for his contributions to computational linguistics .

- Local only **cannot** predict right occupation
- Global (field) helps to understand he was a scientist
- Global (field,word) can infer the **correct occupation**
- Date issue?

- **Conclusion:**
  - Generate fluent descriptions of arbitrary people based on structured data
  - Local and Global conditioning improves model by large margin
  - Model outperforms KN language model by 15 BLEU
  - Order of magnitude more data and bigger vocab
- **Thoughts:**
  - Generation of longer biographies
  - Improving encoding of field values/embeddings
  - Better loss function
  - Better strategy for evaluation of factual accuracy

References:

- http://aclweb.org/anthology/D/D16/D16-1128.pdf
- http://ofir.io/Neural-Language-Modeling-From-Scratch/
- http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/
- https://github.com/odashi/mteval
- http://cs.brown.edu/courses/cs146/assets/files/langmod.pdf
- https://cs.stanford.edu/~angeli/papers/2010-emnlp-generation.pdf

Questions?

Performance : Sentence decoding