Fast and Non-Approximative Language Model Prefixqueries for Word Prediction using Top-k Joining Techniques

Bachelor Thesis

Lukas Schmelzeisen

lukas@uni-koblenz.de

Institute for Web Science and Technologies,
University of Koblenz-Landau

August 6, 2015

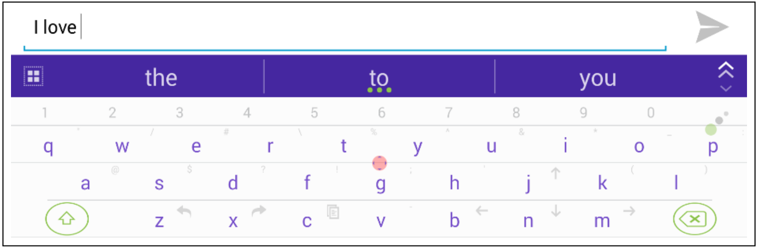
Outline

1. Next Word Prediction
2. Language Models
3. Top-k Joining Techniques

Section 1

Next Word Prediction

What is Next Word Prediction?



Guess the *next intended* word

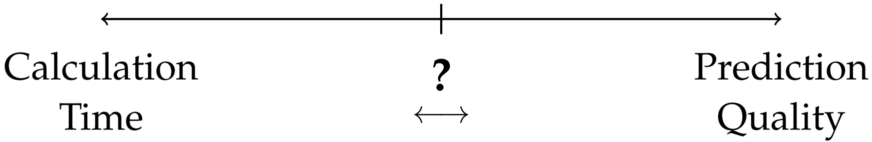
from words already entered

Benefits of good Word Prediction

- Faster typing
- Spelling / Grammar
- ...
- Metric for Language Models?

Thesis Goal

Word prediction usually optimized for speed **or** quality



Thesis Goal

Word prediction usually optimized for speed **or** quality

Calculation Time ⟵ **?** ⟷ ⟶ Prediction Quality

This thesis optimizes for speed while maintaining best quality!

Next Word Prediction

$$\mathrm{NWP}(h) = \underset{w \in \Sigma}{\arg\max}\ \mathrm{P}(w \mid h)$$

Next Word Prediction

history of entered words

$$\mathrm{NWP}(h) = \arg\max_{w \in \Sigma} \mathrm{P}(w \mid h)$$

Next Word Prediction

history of entered words

$$\mathrm{NWP}(h) = \arg\max_{w \in \Sigma} \mathrm{P}(w \mid h)$$

for each word in vocabulary

Section 2

Language Models

How to find probabilities?

$$P(w \mid h)\ ?$$

How to find probabilities?

$$P(w \mid h) \text{ ?}$$

Solution: Statistical Language Models

- analyze text corpora
- to estimate sequence probabilities

How to find probabilities?

$$P\left(w \mid h\right)\ ?$$

Solution: Statistical Language Models

- analyze text corpora
- to estimate sequence probabilities

We will employ two state-of-the-art Language Models:

- Modified Kneser Ney Smoothing
  (Chen and Goodman 1999)
- Generalized Language Model
  (Pickhardt et al. 2014)

Modified Kneser Ney Smoothing

$$\mathsf{P}_{\mathrm{MKN}}(w_n \mid w_1^{n-1}) = \frac{\mathsf{c}^{\mathsf{d}}(w_1^n) + \gamma(w_1^{n-1})\, \hat{\mathsf{P}}_{\mathrm{MKN}}(w_n \mid w_2^{n-1})}{\mathsf{c}(w_1^{n-1}\,\square\,)}$$

Modified Kneser Ney Smoothing

$$P_{\mathrm{MKN}}(w_n \mid w_1^{n-1}) = \frac{c^d(w_1^n) + \gamma(w_1^{n-1})\,\hat{P}_{\mathrm{MKN}}(w_n \mid w_2^{n-1})}{c(w_1^{n-1}\,\square)}$$

Generalized Language Model

$$P_{\mathrm{GLM}}(w_n \mid w_1^{n-1}) = \frac{c^d(w_1^n) + \frac{\gamma(w_1^{n-1})}{\#_\partial w_1^{n-1}} \sum_{j=1}^{\#_\partial w_1^{n-1}} \hat{P}_{\mathrm{GLM}}(w_n \mid \partial_j w_1^{n-1})}{c(w_1^{n-1} \square)}$$

Generalized Language Model

$$P_{\mathrm{GLM}}(w_n \mid w_1^{n-1}) = \frac{c^d(w_1^n) + \frac{\gamma(w_1^{n-1})}{\#_\partial w_1^{n-1}} \sum_{j=1}^{\#_\partial w_1^{n-1}} \hat{P}_{\mathrm{GLM}}(w_n \mid \partial_j w_1^{n-1})}{c(w_1^{n-1} \square)}$$

Example: Probability of "you" after "I love"

$$\mathsf{P}_{\text{MKN}}(\text{you} \,|\, \text{I love}) = \frac{\mathsf{c}^{\mathsf{d}}(\text{I love you}) + \gamma(\text{I love})\, \hat{\mathsf{P}}_{\text{MKN}}(\text{you} \,|\, \text{love})}{\mathsf{c}(\text{I love} \,\square\,)}$$

$$\hat{\mathsf{P}}_{\text{MKN}}(\text{you} \,|\, \text{love}) = \frac{\mathsf{N}^{\mathsf{d}}_{1+}(\,\bullet\,\text{love you}) + \gamma(\text{love})\, \hat{\mathsf{P}}_{\text{MKN}}(\text{you})}{\mathsf{N}_{1+}(\,\bullet\,\text{love}\,\bullet\,)}$$

$$\hat{\mathsf{P}}_{\text{MKN}}(\text{you}) = \frac{\mathsf{N}_{1+}(\,\bullet\,\text{you})}{\mathsf{N}_{1+}(\,\bullet\,\bullet\,)}$$

Example: Probability of "you" after "I love"

$$P_{\mathrm{MKN}}(\text{you} \mid \text{I love}) = \frac{c^{d}(\text{I love you}) + \gamma(\text{I love})\,\hat{P}_{\mathrm{MKN}}(\text{you} \mid \text{love})}{c(\text{I love} \,\square\,)}$$

$$\hat{P}_{\mathrm{MKN}}(\text{you} \mid \text{love}) = \frac{N^{d}_{1+}(\bullet\,\text{love you}) + \gamma(\text{love})\,\hat{P}_{\mathrm{MKN}}(\text{you})}{N_{1+}(\bullet\,\text{love}\,\bullet)}$$

$$\hat{P}_{\mathrm{MKN}}(\text{you}) = \frac{N_{1+}(\bullet\,\text{you})}{N_{1+}(\bullet\,\bullet)}$$

Idea: express as Weighted Sums

$$P(w \mid h) = \sum_{i=1}^{N} \lambda_i^h \, \alpha_i^h(w)$$

Idea: express as Weighted Sums

depends on argument w

$$P\left(w \mid h\right) = \sum_{i=1}^{N} \lambda_i^h \, \alpha_i^h(w)$$

Idea: express as Weighted Sums

depends on argument w

independent of argument w

$$P\left(w \mid h\right) = \sum_{i=1}^{N} \lambda_i^h \, \alpha_i^h(w)$$

Idea: express as Weighted Sums

depends on argument w

independent of argument w

$$P(w \mid h) = \sum_{i=1}^{N} \lambda_i^h \, \alpha_i^h(w)$$

all words in vocabulary

Idea: express as Weighted Sums

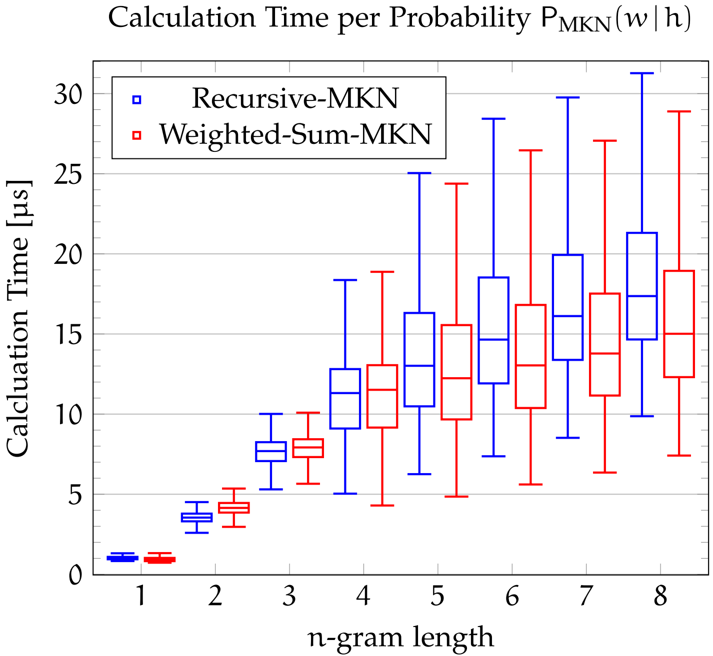
depends on argument w

independent of argument w

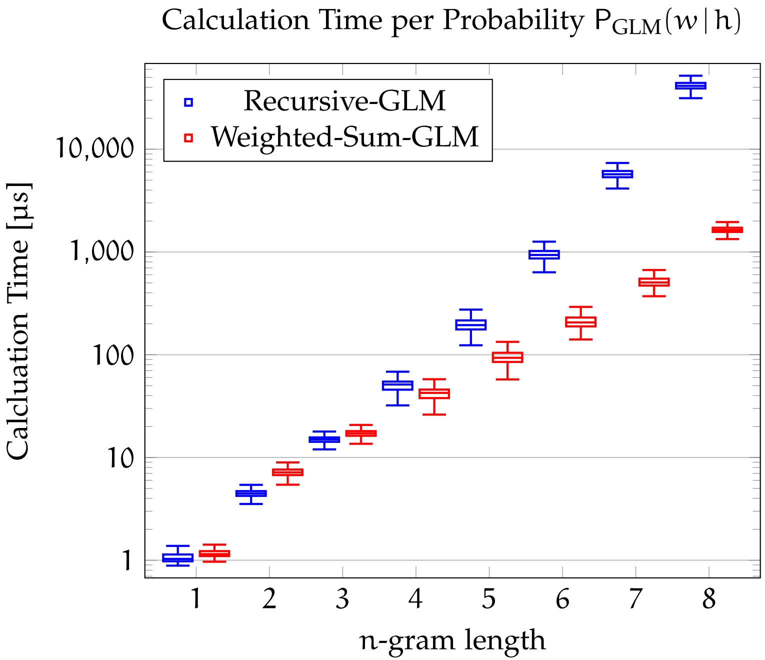
$$P\,(w \,|\, h) = \sum_{i=1}^{N} \lambda_i^h \, \alpha_i^h(w)$$

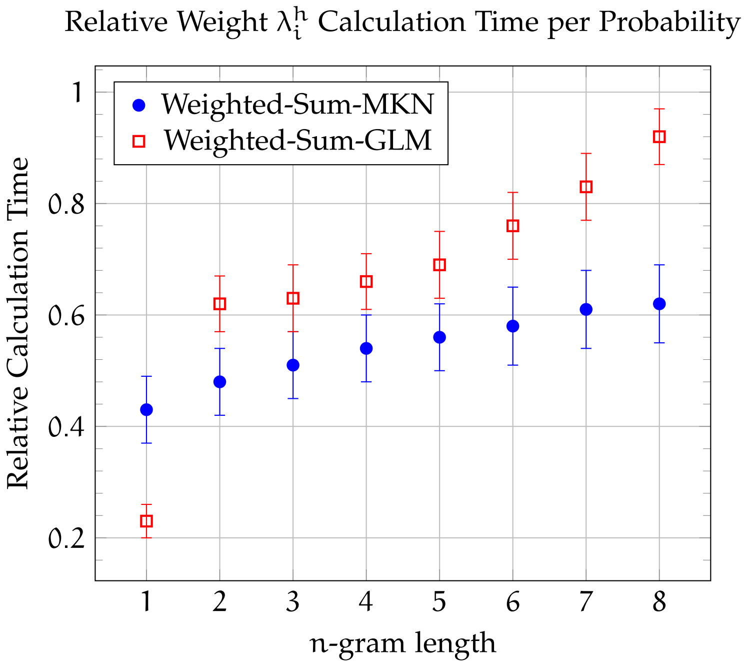
user given, thus fixed

all words in vocabulary



Calculation Time per Probability $P_{MKN}(w \mid h)$
Recursive-MKN
Weighted-Sum-MKN
Calcluation Time [µs]
0
5
10
15
20
25
30
1
2
3
4
5
6
7
8
n-gram length


Calculation Time per Probability $P_{GLM}(w|h)$
Recursive-GLM
Weighted-Sum-GLM
Calcluation Time [μs]
10,000
1,000
100
10
1
1
2
3
4
5
6
7
8
n-gram length



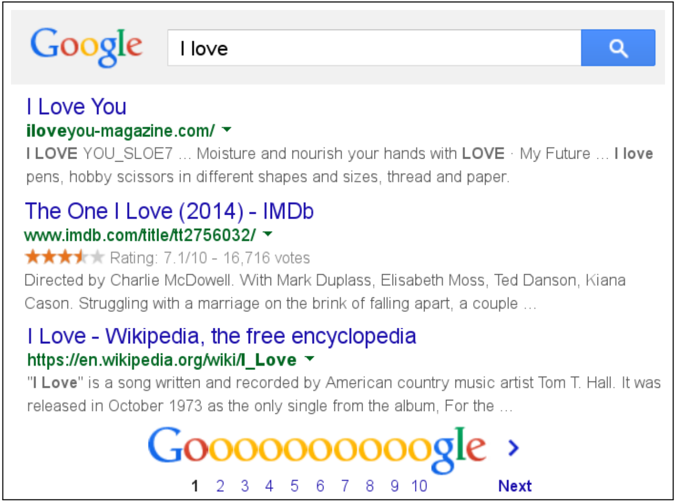
Relative Weight λ_i^h Calculation Time per Probability
Weighted-Sum-MKN
Weighted-Sum-GLM
Relative Calculation Time
n-gram length
1
0.8
0.6
0.4
0.2
1
2
3
4
5
6
7
8

Section 3

Top-k Joining Techniques

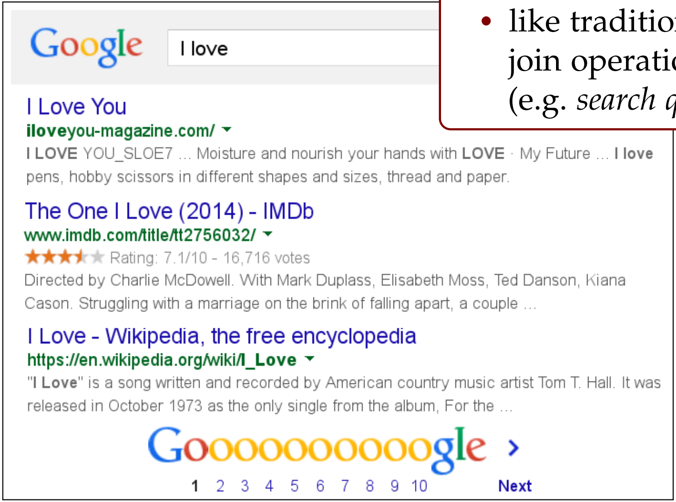
What are Top-k Join Queries?

What are Top-k Join Queries?

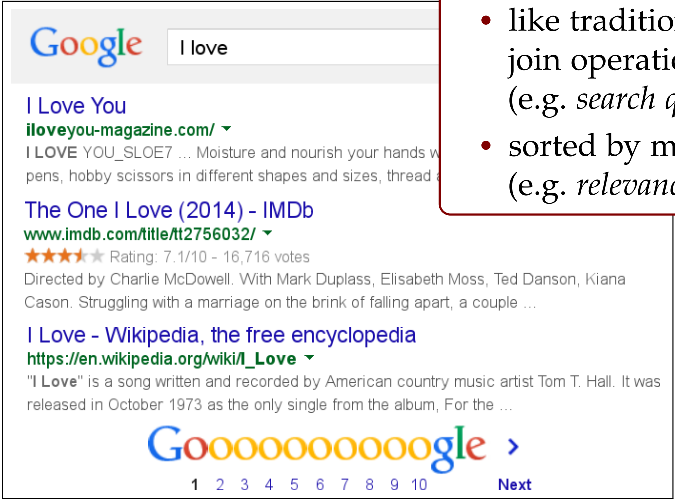


What are Top-k Join Queries?

- like traditional join operations (e.g. *search query*)

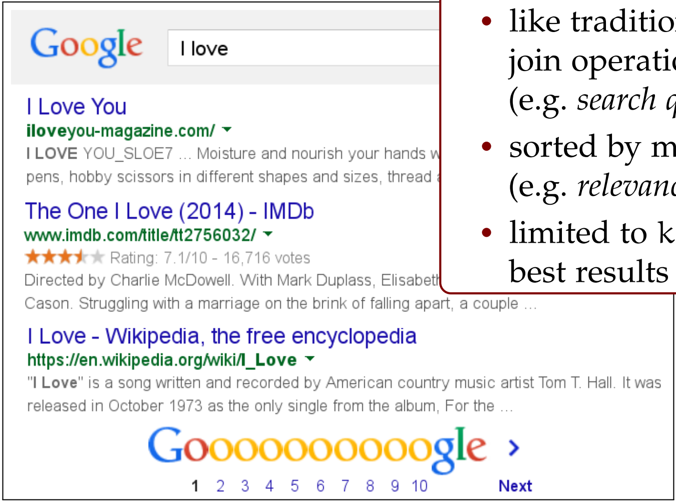


What are Top-k Join Queries?



- like traditional join operations (e.g. *search query*)
- sorted by metric (e.g. *relevance*)

What are Top-k Join Queries?



- like traditional join operations (e.g. *search query*)
- sorted by metric (e.g. *relevance*)
- limited to $k = 3$ best results

Calculating probabilities for all words is slow!

Calculating probabilities for all words is slow!

Observation: Words with high probabilities have high occurrence counts (probabilities are monotone)

Calculating probabilities for all words is slow!

Observation: Words with high probabilities have high occurrence counts (probabilities are monotone)

Idea: Keep sorted lists of occurrence counts

ϵ		love		I love	
the	50	you	30	you	25
a	40	the	20	a	5
you	35	a	10	the	3

Calculating probabilities for all words is slow!

Observation: Words with high probabilities have high occurrence counts (probabilities are monotone)

Idea: Keep sorted lists of occurrence counts

ϵ		love		I love	
the	50	you	30	you	25
a	40	the	20	a	5
you	35	a	10	the	3

We apply two Top-k Joining Techniques to NWP:

- Threshold Algorithm (Fagin et al. 2001)
- No Random Access Algorithm (Fagin et al. 2001)

Comparison

Threshold Algorithm	No Random Access Algorithm
– Requires Sorted and Random Access	+ Only Sorted Access necessary
+ Faster	
+ Less Memory	

Data structure

Data structure

- Sorted lists are unique for each history h

ϵ		love		I love	
the	50	you	30	you	25
a	40	the	20	a	5
you	35	a	10	the	3

Data structure

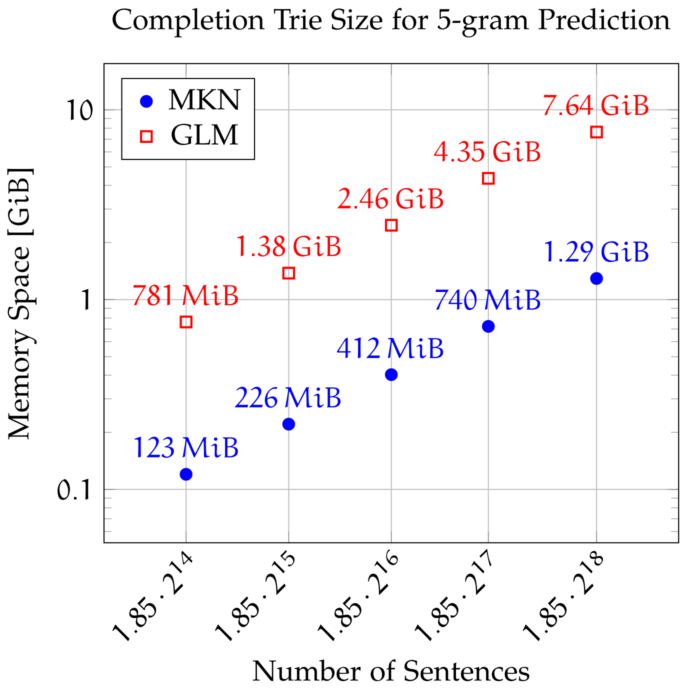
- Sorted lists are unique for each history h

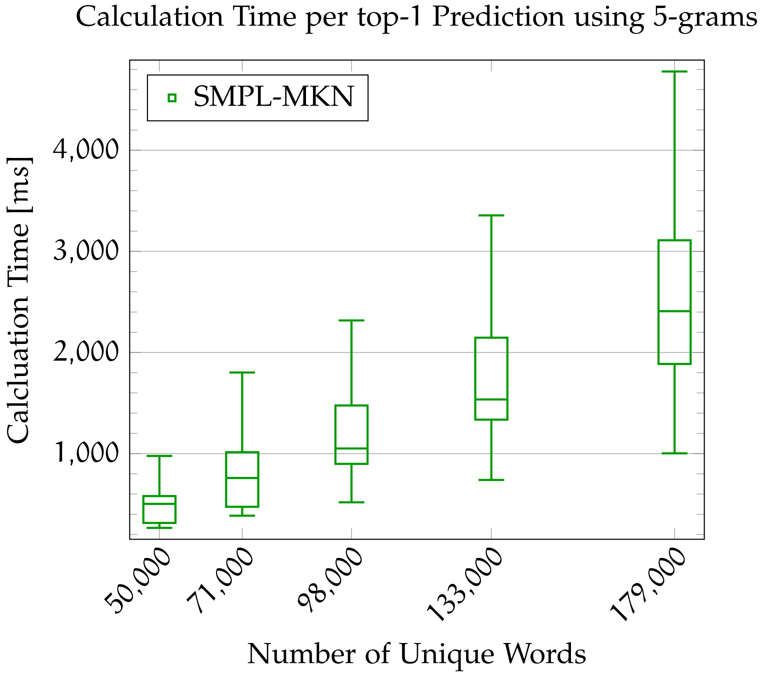
ϵ		love		I love	
the	50	you	30	you	25
a	40	the	20	a	5
you	35	a	10	the	3

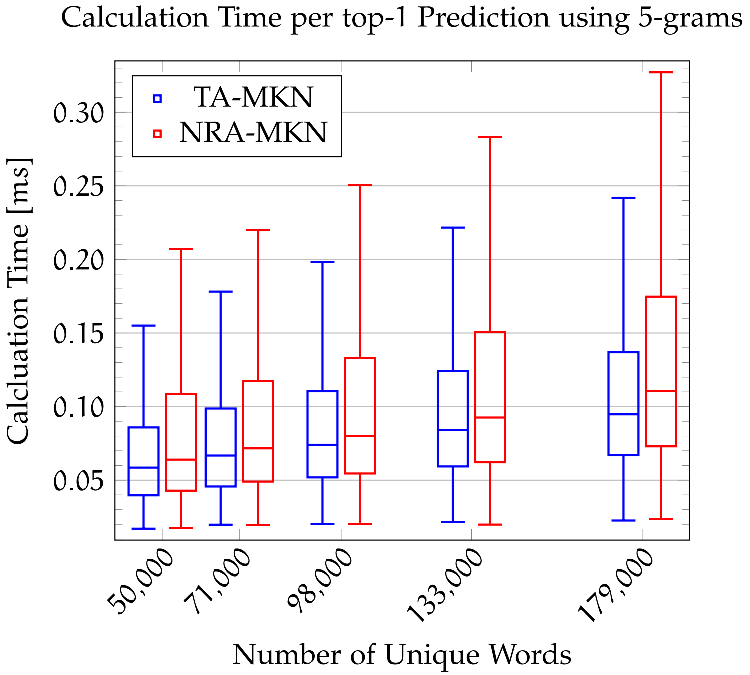
- Optimized data structure: *Completion Trie* (Hsu and Ottaviano 2013)
  - Optimized for prefix-retrieval
  - High data compression

Prediction Quality

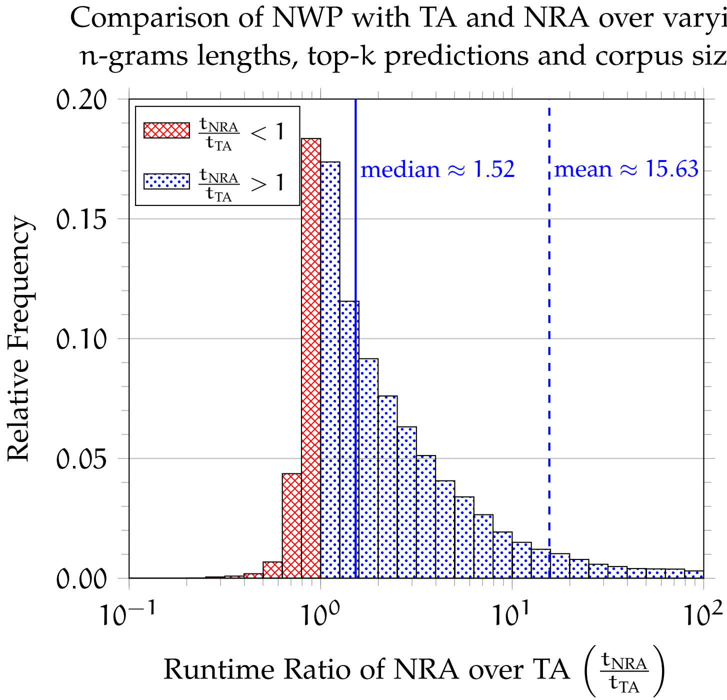
	Keystroke Savings	
n-gram length	MKN	GLM
2	0.44	0.44
3	0.50	0.50
4	0.51	0.51
5	0.51	0.52


Completion Trie Size for 5-gram Prediction
MKN
GLM
Memory Space [GiB]
10
1
0.1
781 MiB
1.38 GiB
2.46 GiB
4.35 GiB
7.64 GiB
123 MiB
226 MiB
412 MiB
740 MiB
1.29 GiB
1.85 · 2^14
1.85 · 2^15
1.85 · 2^16
1.85 · 2^17
1.85 · 2^18
Number of Sentences


Calculation Time per top-1 Prediction using 5-grams
SMPL-MKN
Calcluation Time [ms]
1,000
2,000
3,000
4,000
50,000
71,000
98,000
133,000
179,000
Number of Unique Words



Calculation Time per top-1 Prediction using 5-grams
Calcluation Time [ms]
0.05
0.10
0.15
0.20
0.25
0.30
TA-MKN
NRA-MKN
50,000
71,000
98,000
133,000
179,000
Number of Unique Words


Comparison of NWP with TA and NRA over varying n-grams lengths, top-k predictions and corpus sizes
$\frac{t_{NRA}}{t_{TA}} < 1$
$\frac{t_{NRA}}{t_{TA}} > 1$
median ≈ 1.52
mean ≈ 15.63
Relative Frequency
0.20
0.15
0.10
0.05
0.00
10^{-1}
10^{0}
10^{1}
10^{2}
Runtime Ratio of NRA over TA $\left(\frac{t_{NRA}}{t_{TA}}\right)$

Section 4

Conclusion

Conclusion

- Weighted Sum representation for probabilities
  - Calculation equally fast for MKN, double as fast for GLM
  - Speeds up word prediction by 50% (MKN) to 70% (GLM)
- Top-k Joining
  - Speeds up word prediction by multiple orders of magnitude
  - Threshold Algorithm preferable to No Random Access Algorithm

Thanks for your Attention!

http://141.26.208.6/

Section 5

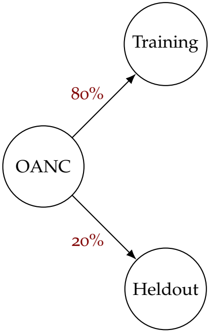
Experimental Setup

Evaluation Corpus

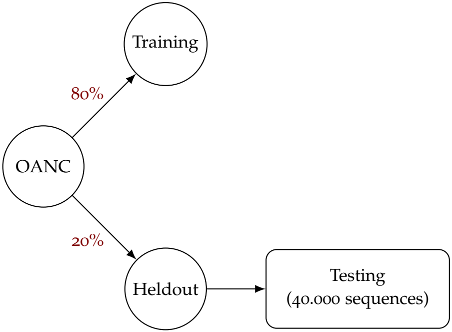
Open American National Corpus (Ide and Suderman 2007)

- open collection of American English
- historical and contemporary
- written text of all genres
- around 600.000 sentences / 14.000.000 words

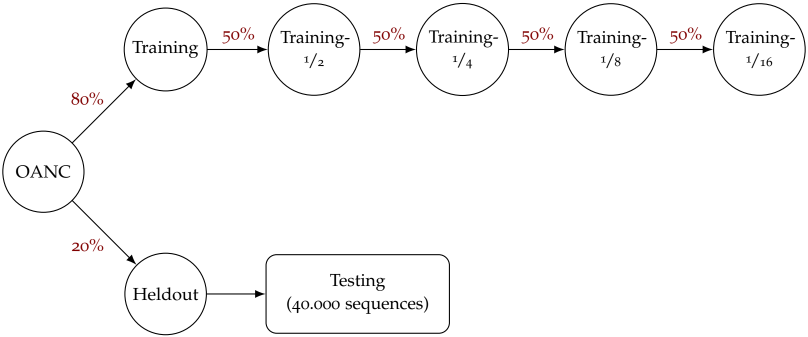
Experimental Setup



Experimental Setup



Experimental Setup



Section 6

Top-k Join Examples

Top-k joins combined with Weighted Sums

Top-k joins combined with Weighted Sums

$$P\left(w \,|\, h\right) = \sum_{i=1}^{N} \lambda_i^h \, \alpha_i^h(w)$$

Top-k joins combined with Weighted Sums

$$P(w \mid h) = \sum_{i=1}^{N} \lambda_i^h \, \alpha_i^h(w)$$

- Weights λ_i^h can be precomputed

Top-k joins combined with Weighted Sums

$$P\,(w\,|\,h) = \sum_{i=1}^{N} \lambda_i^h \, \alpha_i^h(w)$$

- Weights λ_i^h can be precomputed
- Sorted lists can store terms $\alpha_i^h(w)$

ϵ		love		I love	
the	50	you	30	you	25
a	40	the	20	a	5
you	35	a	10	the	3

Top-k joins combined with Weighted Sums

$$P(w \mid h) = \sum_{i=1}^{N} \lambda_i^h \, \alpha_i^h(w)$$

- Weights λ_i^h can be precomputed
- Sorted lists can store terms $\alpha_i^h(w)$

ϵ		love		I love	
the	50	you	30	you	25
a	40	the	20	a	5
you	35	a	10	the	3

- In examples we will assume $\lambda_1^h, \ldots, \lambda_N^h = 1$
  (not possible in practice!)

Top-k joins combined with Weighted Sums

$$P(w \mid h) = \sum_{i=1}^{N} \lambda_i^h \, \alpha_i^h(w)$$

- Weights λ_i^h can be precomputed
- Sorted lists can store terms $\alpha_i^h(w)$

ϵ		love		I love	
the	50	you	30	you	25
a	40	the	20	a	5
you	35	a	10	the	3

- In examples we will assume $\lambda_1^h, \ldots, \lambda_N^h = 1$ (not possible in practice!)
- $P(\text{you} \mid \text{I love}) = 35 + 30 + 25 = 95$

Threshold Algorithm

$$\arg\max_{w \in \Sigma} P_{\mathrm{MKN}}(w \mid \mathrm{I\ love})$$

Threshold Algorithm

$$\arg\max_{w \in \Sigma} \mathrm{P}_{\mathrm{MKN}}(w \mid \mathrm{I\ love})$$

ϵ	love	I love

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)

$$\arg\max_{w \in \Sigma} \mathrm{P}_{\mathrm{MKN}}(w \mid \text{I love})$$

ϵ	love	I love

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists

$$\arg\max_{w \in \Sigma} P_{\mathrm{MKN}}(w \mid \text{I love})$$

ϵ	love	I love

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability

$$\arg\max_{w \in \Sigma} \mathrm{P}_{\mathrm{MKN}}(w \mid \text{I love})$$

ϵ	love	I love

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w \in \Sigma} P_{MKN}(w \mid \text{I love})$$

ϵ	love	I love

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w \in \Sigma} \mathrm{P}_{\mathrm{MKN}}(w \mid \mathrm{I\ love})$$

ϵ	love	I love

$$\mathrm{P}_{\mathrm{threshold}} = \infty + \infty + \infty \ = \infty$$

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w \in \Sigma} P_{\mathrm{MKN}}(w \mid \mathrm{I\ love})$$

ϵ	love	I love
$\boxed{\text{the} \quad 50}$		

$$P_{\mathrm{threshold}} = \infty + \infty + \infty \;= \infty$$

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w \in \Sigma} \mathrm{P}_{\mathrm{MKN}}(w \mid \mathrm{I\ love})$$

ϵ	love	I love
$\boxed{\text{the} \quad 50}$		
	the 20	
		the 3

$$\mathrm{P}_{\text{threshold}} = \infty + \infty + \infty = \infty$$

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w \in \Sigma} \mathsf{P}_{\mathrm{MKN}}(w \mid \mathrm{I\ love})$$

ϵ	love	I love
$\boxed{\text{the} \quad 50}$		
	the 20	
		the 3

$$\mathsf{P}_{\text{threshold}} = \infty + \infty + \infty = \infty$$

$$\mathsf{P}(\text{the} \mid \text{I love}) = 50 + 20 + 3 = 73$$

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w \in \Sigma} \mathrm{P}_{\mathrm{MKN}}(w \mid \mathrm{I\ love})$$

ϵ	love	I love
$\boxed{\text{the} \quad 50}$		
	the 20	
		the 3

$$\mathrm{P}_{\mathrm{threshold}} = 50 + \infty + \infty = \infty$$

$$\mathrm{P}\left(\mathrm{the} \mid \mathrm{I\ love}\right) = 50 + 20 + 3 = 73$$

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w \in \Sigma} \mathrm{P}_{\mathrm{MKN}}(w \mid \text{I love})$$

ϵ	love	I love
$\boxed{\text{the} \quad 50}$	$\boxed{\text{you} \quad 30}$	
	the 20	
		the 3

$$\mathrm{P}_{\text{threshold}} = 50 + \infty + \infty = \infty$$
$$\mathrm{P}(\text{the} \mid \text{I love}) = 50 + 20 + 3 = 73$$

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w \in \Sigma} P_{\text{MKN}}(w \mid \text{I love})$$

ϵ		love		I love	
$\boxed{\text{the} \quad 50}$		$\boxed{\text{you} \quad 30}$		you	25
		the	20		
you	35			the	3

$$P_{\text{threshold}} = 50 + \infty + \infty = \infty$$
$$P(\text{the} \mid \text{I love}) = 50 + 20 + 3 = 73$$

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w \in \Sigma} \mathrm{P}_{\mathrm{MKN}}(w \mid \text{I love})$$

ϵ		love		I love	
$\boxed{\text{the} \quad 50}$		$\boxed{\text{you} \quad 30}$		you	25
		the	20		
you	35			the	3

$$\mathrm{P}_{\text{threshold}} = 50 + \infty + \infty = \infty$$

$$\mathrm{P}(\text{you} \mid \text{I love}) = 35 + 30 + 25 = 90$$

$$\mathrm{P}(\text{the} \mid \text{I love}) = 50 + 20 + 3 = 73$$

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w \in \Sigma} P_{\text{MKN}}(w \mid \text{I love})$$

ϵ		love		I love	
$\boxed{\text{the}\quad 50}$		$\boxed{\text{you}\quad 30}$		you	25
		the	20		
you	35			the	3

$$P_{\text{threshold}} = 50 + 30 + \infty = \infty$$
$$P(\text{you} \mid \text{I love}) = 35 + 30 + 25 = 90$$
$$P(\text{the} \mid \text{I love}) = 50 + 20 + 3 = 73$$

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w \in \Sigma} P_{\text{MKN}}(w \mid \text{I love})$$

ϵ		love		I love	
$\boxed{\text{the}}$	$\boxed{50}$	$\boxed{\text{you}}$	$\boxed{30}$	$\boxed{\text{you}}$	$\boxed{25}$
		the	20		
you	35			the	3

$$\begin{aligned} P_{\text{threshold}} &= 50 + 30 + \infty &&= \infty \\ P(\text{you} \mid \text{I love}) &= 35 + 30 + 25 &&= 90 \\ P(\text{the} \mid \text{I love}) &= 50 + 20 + 3 &&= 73 \end{aligned}$$

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w \in \Sigma} \mathrm{P}_{\mathrm{MKN}}(w \mid \mathrm{I\ love})$$

ϵ		love		I love	
$\boxed{\text{the} \quad 50}$		$\boxed{\text{you} \quad 30}$		$\boxed{\text{you} \quad 25}$	
		the	20		
you	35			the	3

$$\mathrm{P}_{\mathrm{threshold}} = 50 + 30 + 25 = 105$$
$$\mathrm{P}(\mathrm{you} \mid \mathrm{I\ love}) = 35 + 30 + 25 = 90$$
$$\mathrm{P}(\mathrm{the} \mid \mathrm{I\ love}) = 50 + 20 + 3 = 73$$

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w\in\Sigma} P_{\mathrm{MKN}}(w \mid \text{I love})$$

ϵ		love		I love	
the	50	$\boxed{\text{you}}$	$\boxed{30}$	$\boxed{\text{you}}$	$\boxed{25}$
$\boxed{\text{a}}$	$\boxed{40}$	the	20		
you	35			the	3

$$\begin{aligned} P_{\text{threshold}} &= 50 + 30 + 25 &&= 105 \\ P(\text{you} \mid \text{I love}) &= 35 + 30 + 25 &&= 90 \\ P(\text{the} \mid \text{I love}) &= 50 + 20 + 3 &&= 73 \end{aligned}$$

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w \in \Sigma} P_{\mathrm{MKN}}(w \mid \text{I love})$$

ϵ		love		I love	
the	50	$\boxed{\text{you}}$	$\boxed{30}$	$\boxed{\text{you}}$	$\boxed{25}$
$\boxed{\text{a}}$	$\boxed{40}$	the	20	a	5
you	35	a	10	the	3

$$P_{\text{threshold}} = 50 + 30 + 25 = 105$$
$$P(\text{you} \mid \text{I love}) = 35 + 30 + 25 = 90$$
$$P(\text{the} \mid \text{I love}) = 50 + 20 + 3 = 73$$

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w \in \Sigma} P_{\text{MKN}}(w \mid \text{I love})$$

ϵ		love		I love	
the	50	$\boxed{\text{you}}$	$\boxed{30}$	$\boxed{\text{you}}$	$\boxed{25}$
$\boxed{\text{a}}$	$\boxed{40}$	the	20	a	5
you	35	a	10	the	3

$$
\begin{aligned}
P_{\text{threshold}} &= 50 + 30 + 25 &&= 105 \\
P(\text{you} \mid \text{I love}) &= 35 + 30 + 25 &&= 90 \\
P(\text{the} \mid \text{I love}) &= 50 + 20 + 3 &&= 73 \\
P(\text{a} \mid \text{I love}) &= 40 + 10 + 5 &&= 55
\end{aligned}
$$

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w \in \Sigma} P_{\mathrm{MKN}}(w \mid \mathrm{I\ love})$$

ϵ		love		I love	
the	50	$\boxed{\text{you}}$	$\boxed{30}$	$\boxed{\text{you}}$	$\boxed{25}$
$\boxed{\text{a}}$	$\boxed{40}$	the	20	a	5
you	35	a	10	the	3

$$\begin{aligned} P_{\text{threshold}} &= 40 + 30 + 25 &&= 95 \\ P(\text{you} \mid \text{I love}) &= 35 + 30 + 25 &&= 90 \\ P(\text{the} \mid \text{I love}) &= 50 + 20 + 3 &&= 73 \\ P(\text{a} \mid \text{I love}) &= 40 + 10 + 5 &&= 55 \end{aligned}$$

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w\in\Sigma} P_{\text{MKN}}(w \mid \text{I love})$$

ϵ		love		I love	
the	50	you	30	$\boxed{\text{you}}$	$\boxed{25}$
$\boxed{\text{a}}$	$\boxed{40}$	$\boxed{\text{the}}$	$\boxed{20}$	a	5
you	35	a	10	the	3

$$\begin{aligned} P_{\text{threshold}} &= 40 + 30 + 25 &= 95 \\ P(\text{you} \mid \text{I love}) &= 35 + 30 + 25 &= 90 \\ P(\text{the} \mid \text{I love}) &= 50 + 20 + 3 &= 73 \\ P(\text{a} \mid \text{I love}) &= 40 + 10 + 5 &= 55 \end{aligned}$$

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w \in \Sigma} P_{\text{MKN}}(w \mid \text{I love})$$

ϵ		love		I love	
the	50	you	30	$\boxed{\text{you} \quad 25}$	
$\boxed{\text{a} \quad 40}$		$\boxed{\text{the} \quad 20}$		a	5
you	35	a	10	the	3

$$\begin{aligned} P(\text{you} \mid \text{I love}) &= 35 + 30 + 25 &&= 90 \\ P_{\text{threshold}} &= 40 + 20 + 25 &&= 85 \\ P(\text{the} \mid \text{I love}) &= 50 + 20 + 3 &&= 73 \\ P(\text{a} \mid \text{I love}) &= 40 + 10 + 5 &&= 55 \end{aligned}$$

Threshold Algorithm

1. *Sorted Access* to lists in any order (e.g. round robin)
2. For new words, *Random Access* to to all other lists
3. Compute *threshold* of highest possible unseen probability
4. Terminate when k probabilities greater threshold have been found

$$\arg\max_{w \in \Sigma} P_{\text{MKN}}(w \mid \text{I love})$$

ϵ		love		I love	
the	50	you	30	$\boxed{\text{you}}$	$\boxed{25}$
$\boxed{\text{a}}$	$\boxed{40}$	$\boxed{\text{the}}$	$\boxed{20}$	a	5
you	35	a	10	the	3

$$\begin{aligned} P(\text{you} \mid \text{I love}) &= 35 + 30 + 25 &&= 90 \\ P_{\text{threshold}} &= 40 + 20 + 25 &&= 85 \\ P(\text{the} \mid \text{I love}) &= 50 + 20 + 3 &&= 73 \\ P(\text{a} \mid \text{I love}) &= 40 + 10 + 5 &&= 55 \end{aligned}$$

No Random Access Algorithm

No Random Access Algorithm

1. *Sorted Access* to lists in any order

No Random Access Algorithm

1. *Sorted Access* to lists in any order
2. Keep track of all seen counts

No Random Access Algorithm

1. *Sorted Access* to lists in any order
2. Keep track of all seen counts
3. Compute *upper* and *lower bounds* for each probability

No Random Access Algorithm

1. *Sorted Access* to lists in any order
2. Keep track of all seen counts
3. Compute *upper* and *lower bounds* for each probability
4. Terminate when k lower bounds greater than all other upper bounds have been found

Section 7

References

References I

Chen, Stanley F. and Goodman, Joshua T. (1999). "An empirical study of smoothing techniques for language modeling." In: *Computer Speech & Language* 13.4, pp. 359–393 (cit. on pp. 12 sqq.).

Fagin, Ronald, Lotem, Amnon, and Naor, Moni (2001). "Optimal Aggregation Algorithms for Middleware." In: *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS '01. Santa Barbara, California, USA: ACM, pp. 102–113 (cit. on pp. 35–38).

References II

Hsu, Bo-June (Paul) and Ottaviano, Giuseppe (2013). "Space-efficient Data Structures for Top-k Completion." In: *Proceedings of the 22Nd International Conference on World Wide Web*. WWW '13. Rio de Janeiro, Brazil: International World Wide Web Conferences Steering Committee, pp. 583–594 (cit. on pp. 40 sqq.).

Ide, Nancy and Suderman, Keith (2007). *The Open American National Corpus (OANC)*. URL: http://www.anc.org/data/oanc/ (visited on July 5, 2015) (cit. on p. 52).

References III

Pickhardt, René, Gottron, Thomas, Körner, Martin, Wagner, Paul Georg, Speicher, Till, and Staab, Steffen (2014). "A Generalized Language Model as the Combination of Skipped n-grams and Modified Kneser Ney Smoothing." In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Vol. abs/1404.3377. Baltimore, Maryland: Association for Computational Linguistics, pp. 1145–1154 (cit. on pp. 12 sqq.).