Word representations for out-of-domain tagging

Erik Schill Daniël de Kok

SFB833 A3, University of Tübingen

Introduction

State-of-the art

- State-of-the art in part-of-speech tagging:
  - LSTM tagger;
  - LSTM-based word representations.

State-of-the art

- State-of-the art in part-of-speech tagging:
  - LSTM tagger;
  - LSTM-based word representations.
- Reported accuracies higher than 97%. E.g. Ling et al., 2015:
  - Catalan: 98.92%
  - English 97.36%
  - German: 98.08%

State-of-the art

- State-of-the art in part-of-speech tagging:
  - LSTM tagger;
  - LSTM-based word representations.
- Reported accuracies higher than 97%. E.g. Ling et al., 2015:
  - Catalan: 98.92%
  - English 97.36%
  - German: 98.08%
- However: typically trained & evaluated **in-domain**

State-of-the art

- State-of-the art in part-of-speech tagging:
  - LSTM tagger;
  - LSTM-based word representations.
- Reported accuracies higher than 97%. E.g. Ling et al., 2015:
  - Catalan: 98.92%
  - English 97.36%
  - German: 98.08%
- However: typically trained & evaluated **in-domain**
- Questions:
  - What effect do the word representations have on in-domain and out-of-domain tagging?
  - (How well do LSTM taggers fare out-of-domain?)

Take-home messages

1. Do not use vanilla word2vec or fastText:
   - use subword units;
   - use an embedding model that takes word order into account.
2. Simpler n-gram models can trump character RNNs in out-of-domain tagging.

Setup

- Use an off-the-shelf state-of-the-art tagger;
- plug in different word representations;
- evaluate **in domain** and **out of domain**.

Tagging model

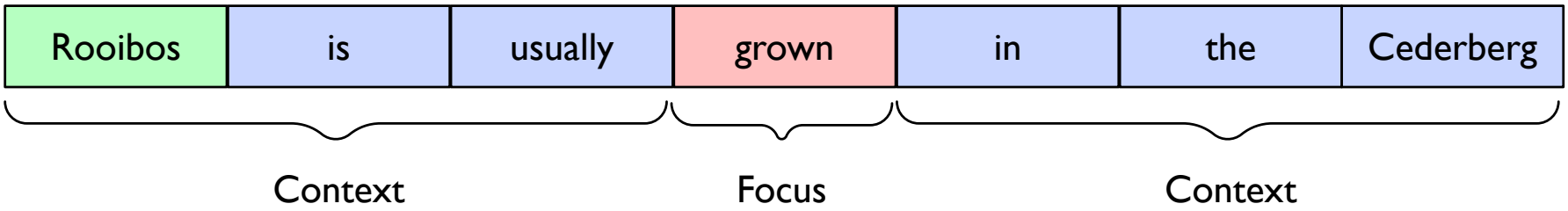
- Standard RNN architecture for sequence labeling:
  1. Bi-directional LSTM over the word representations;
  2. unidirectional LSTM over the outputs states of (1);
  3. softmax layer over the outputs states of (2).

Tagging model

- Standard RNN architecture for sequence labeling:
  1. Bi-directional LSTM over the word representations;
  2. unidirectional LSTM over the outputs states of (1);
  3. softmax layer over the outputs states of (2).
- Explored, but discarded for equal or worse results:
  1. CRF layer for classification;
  2. dillated convolutions.

The contestants

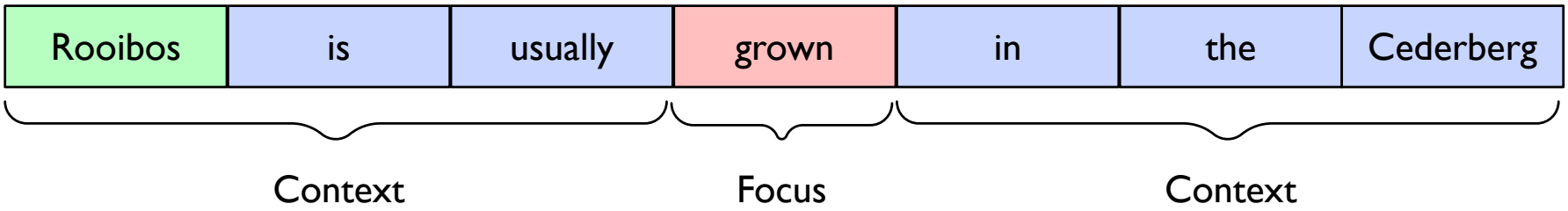
Training framework (Mikolov et al., 2013)



Predict $p(+|w_t, w_c)$:

- w_t: the focus word at position t
- w_c: a context word at position c

Training framework (Mikolov et al., 2013)



Predict $p(+|w_t, w_c)$:

- w_t: the focus word at position t
- w_c: a context word at position c

For example: $p(+|grown, Rooibos)$

Training framework (Mikolov et al., 2013)

Training objective:

- $p(+|w_t, w_c) = 1$ iff w_c is in the context of w_t
- $p(-|w_t, w_n) = 1$ for negative w_n, which is *not* the context of w_t

Training framework (Mikolov et al., 2013)

Training objective:

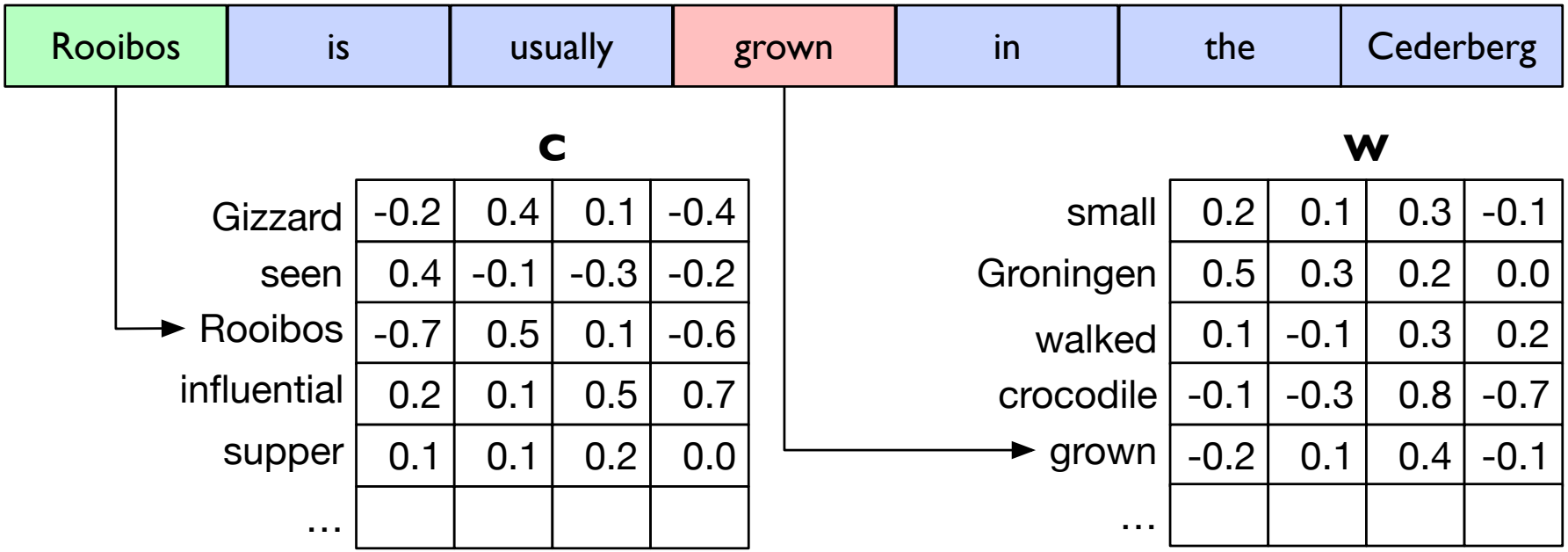
- $p(+|w_t, w_c) = 1$ iff w_c is in the context of w_t
- $p(-|w_t, w_n) = 1$ for negative w_n, which is *not* the context of w_t

Probability function:

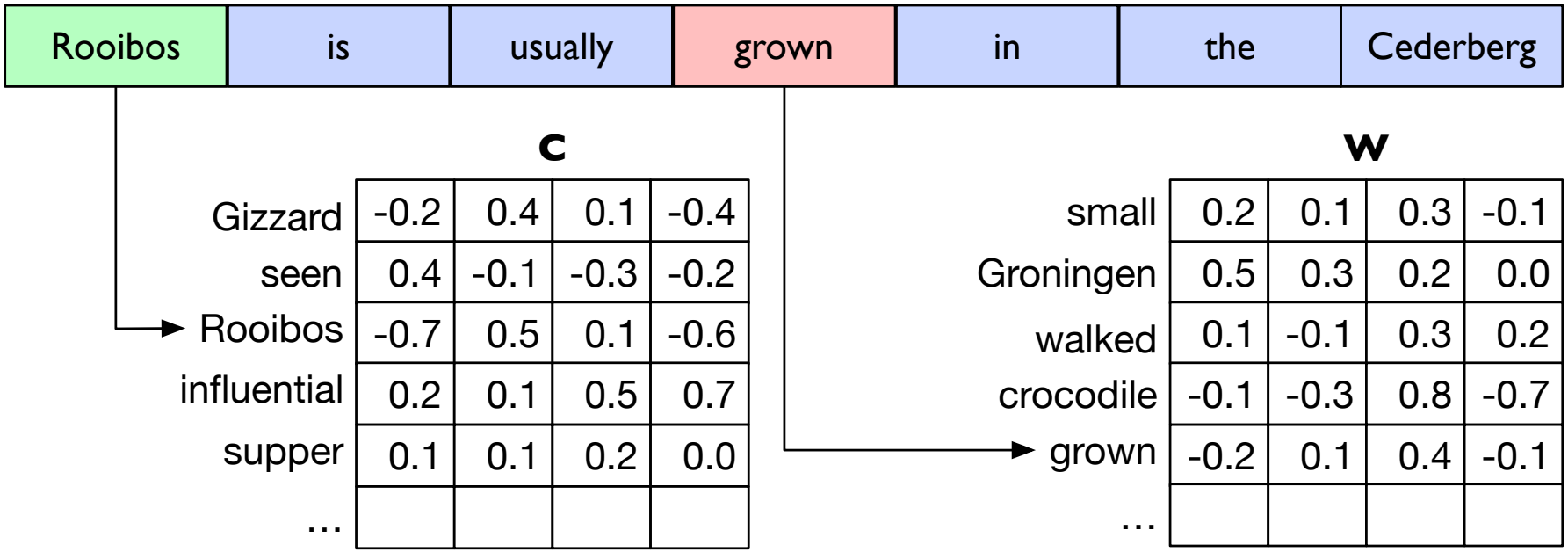
If s is a scoring function for word-context pairs and σ the logistic function, then:

$$p(+|w_t, w_c) = \sigma(s(w_t, w_c))$$
$$p(-|w_t, w_c) = 1 - p(+|w_t, w_c)$$

Skip-gram (Mikolov et al., 2013)

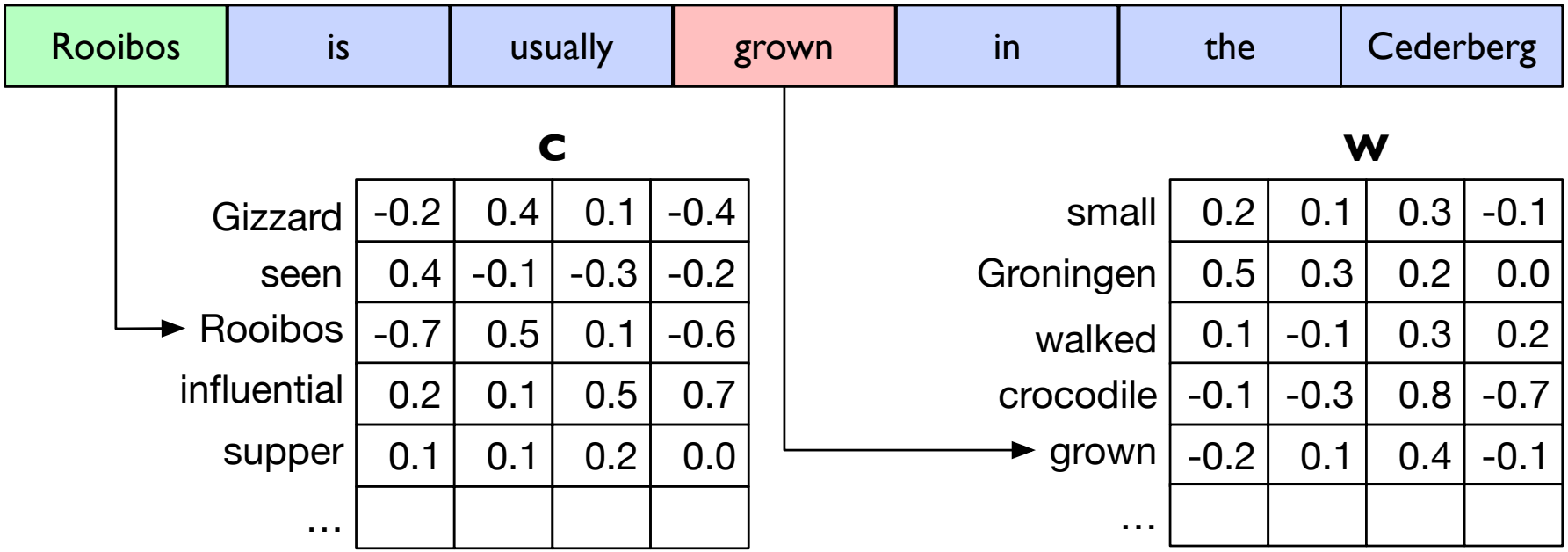


Skip-gram (Mikolov et al., 2013)



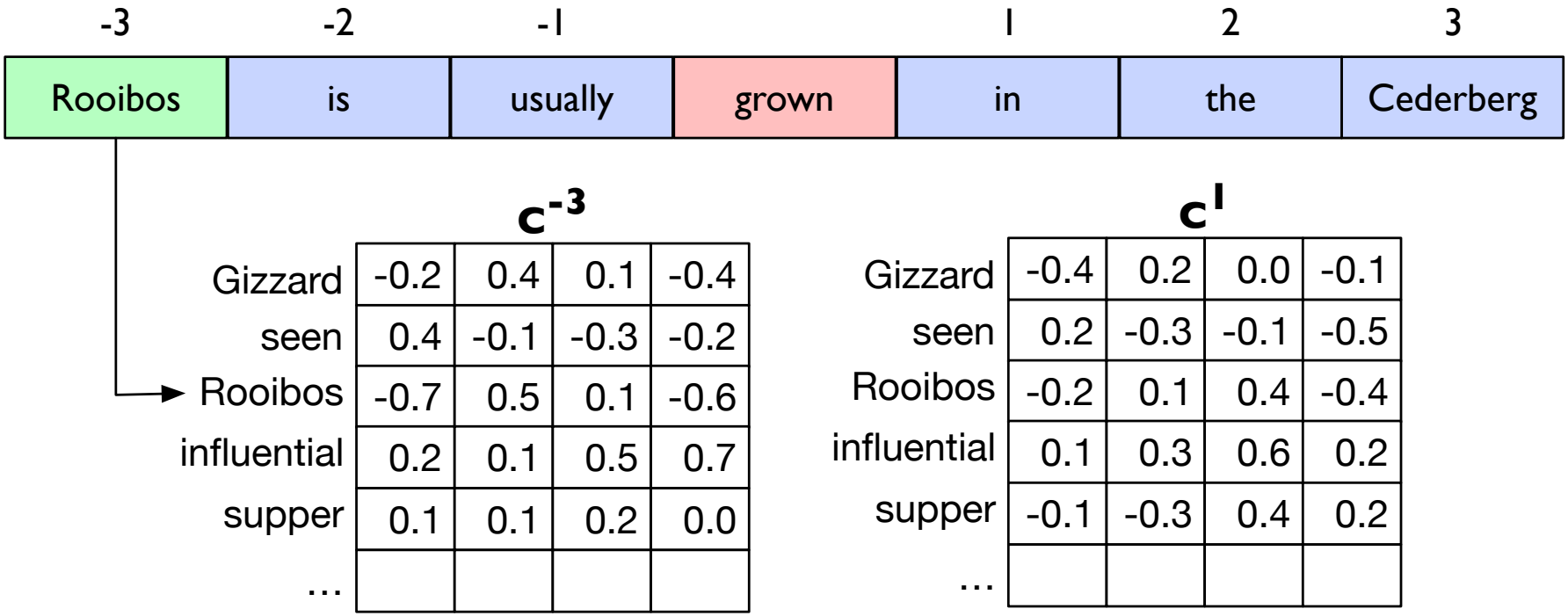
- Word and context embedding matrices: $\mathbf{W}, \mathbf{C} \in \mathbb{R}^{|V| \times d}$

Skip-gram (Mikolov et al., 2013)

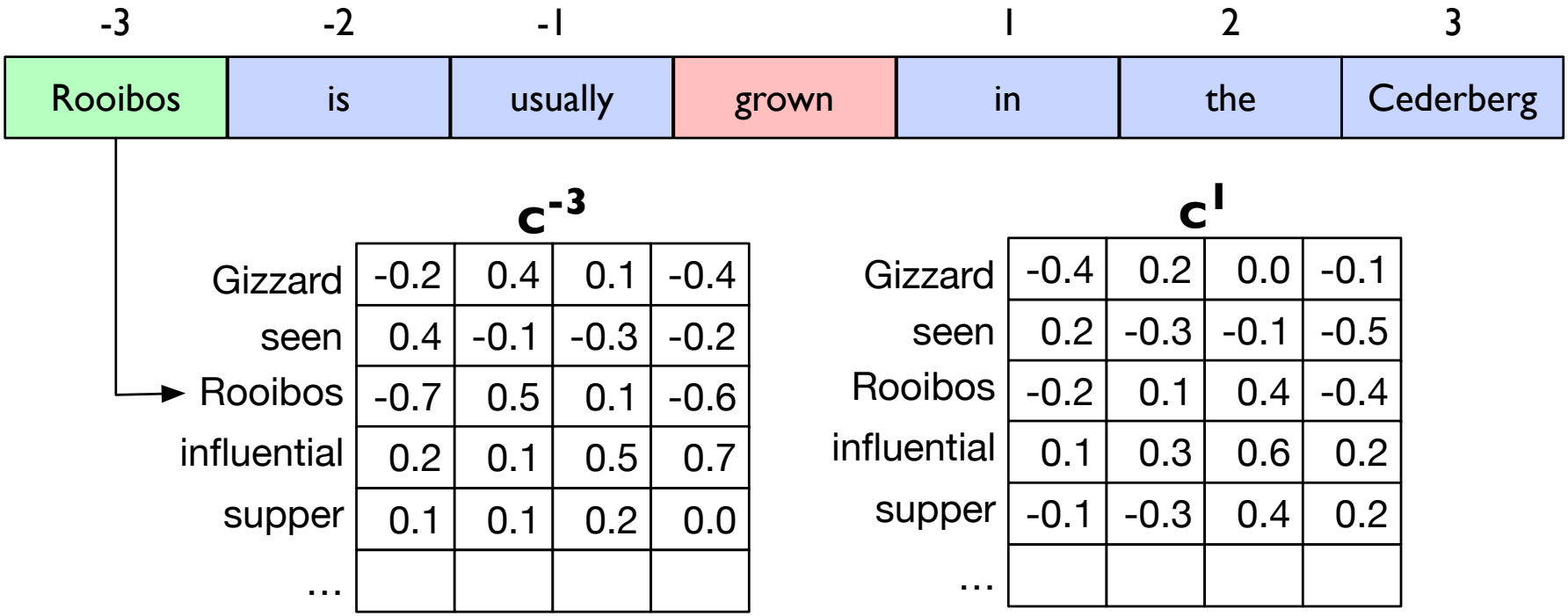


- Word and context embedding matrices: $\mathbf{W}, \mathbf{C} \in \mathbb{R}^{|V| \times d}$
- Scoring function: $s(w_t, w_c) = \mathbf{W}_{w_t} \cdot \mathbf{C}_{w_c}$
- After training: discard **C**. **W** are the word representations.

Structured skip-gram (Wang et al., 2015)

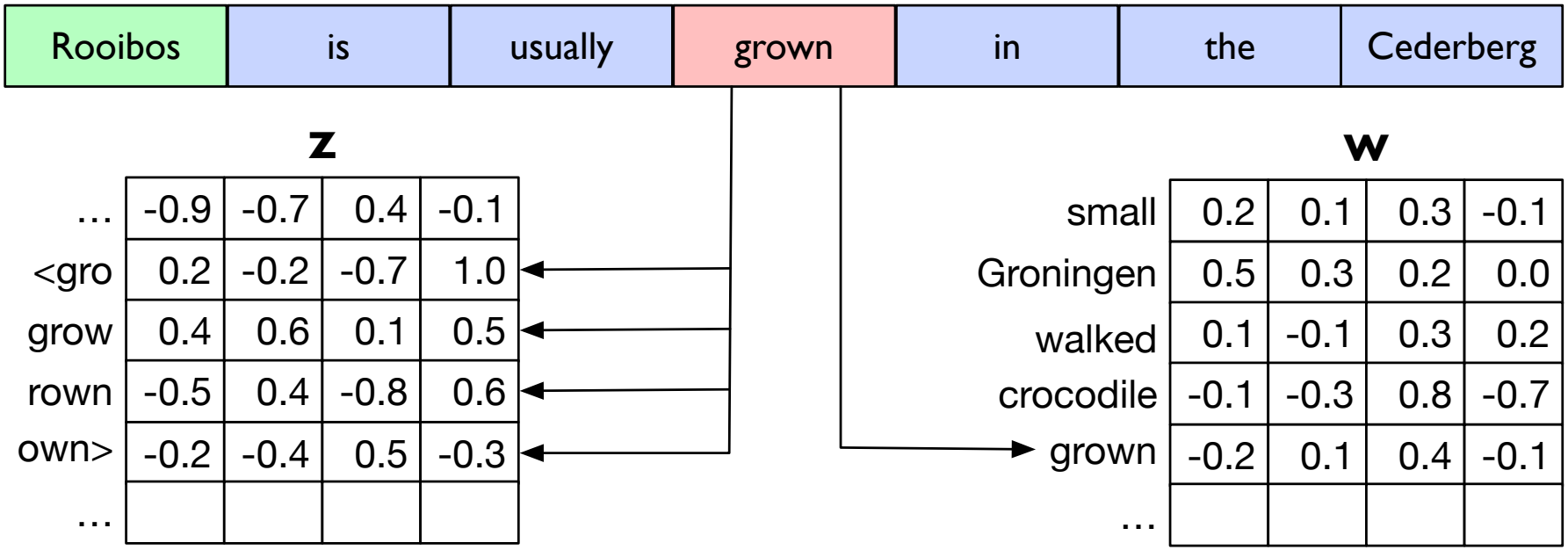


Structured skip-gram (Wang et al., 2015)

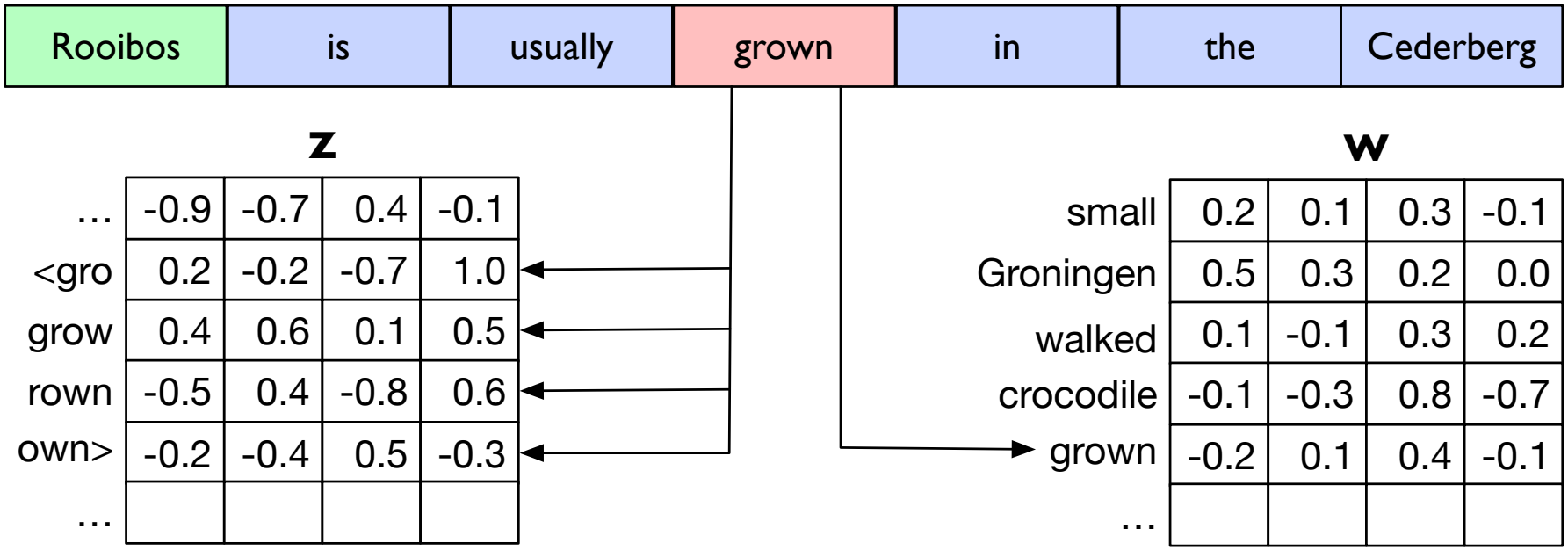


- Scoring function: $s(w_t, w_c) = \mathbf{W}_{w_t} \cdot \mathbf{C}^c_{w_c}$

Subword information (Bojanowski et al., 2016)

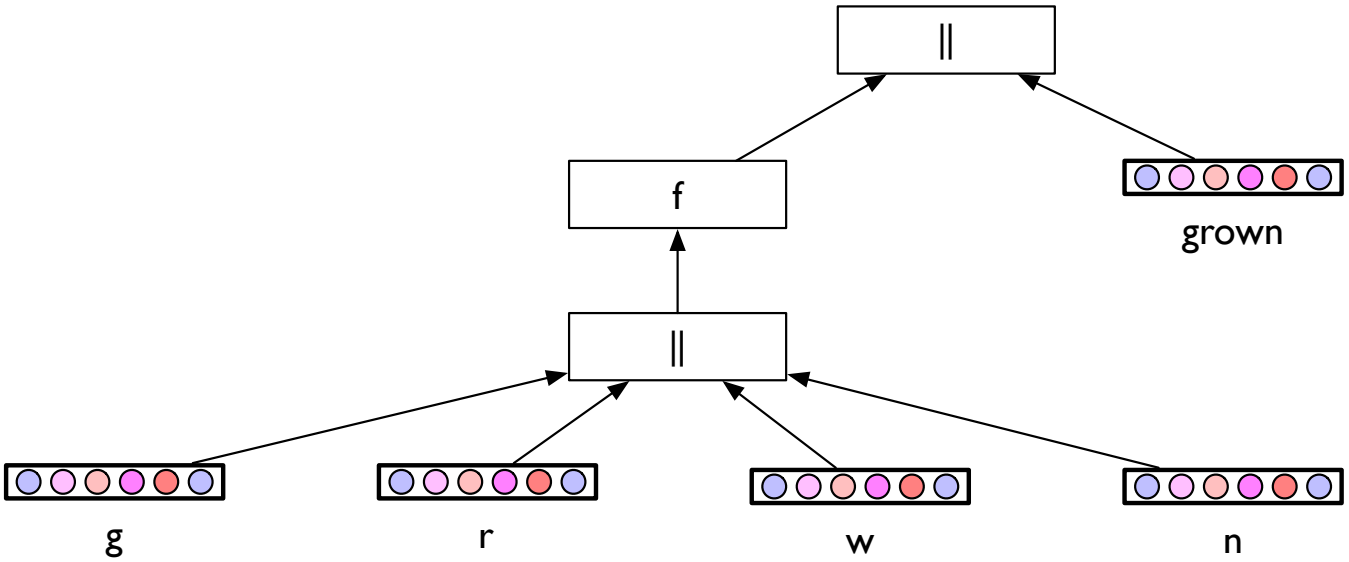


Subword information (Bojanowski et al., 2016)



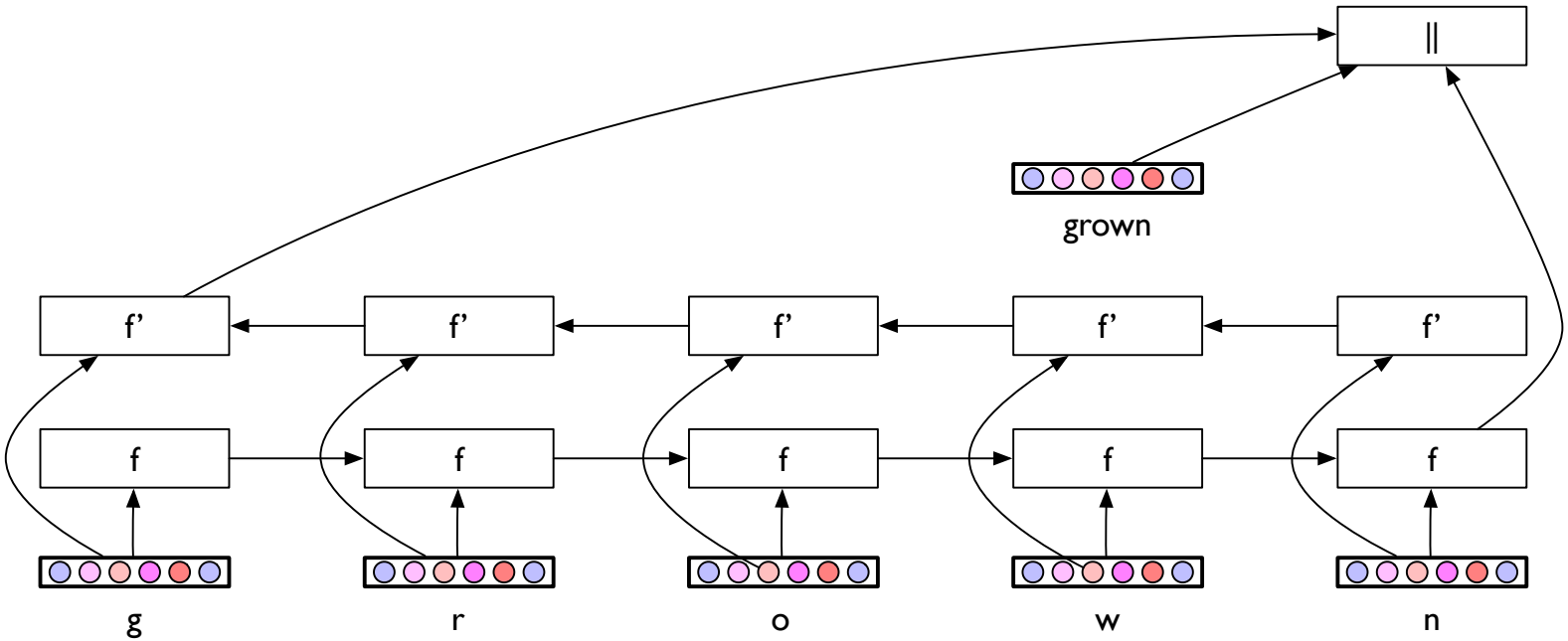
- Scoring function: $s(w_t, w_c) = \mathbf{W}_{w_t} \cdot \mathbf{C}_{w_c} + \sum_{g \in \mathcal{G}_{w_t}} [\mathbf{Z}_g \cdot \mathbf{C}_{w_c}]$

Affix representations (De Kok, 2015)



- Example: prefix and suffix length = 2;
- in our experiments: 8;
- concatenated representation fed to tagger RNN.

BiLSTM word representations (Ling et al., 2015)



Evaluation

In-domain data: TüBa-D/Z

- TüBa-D/Z release 10: 95,595 sentences, 1,787,801 tokens
- Text from the German Taz newspaper.
- Splits:
  - 70% training set
  - 10% development set
  - 20% held-out evaluation set

Out-of-domain data: NoSta-D (Dipper et al., 2013)

Subcorpus	Variety	Sentences	Tokens
Anselm corpus	historical	81	1,085
BeMaTac	spoken[1]	1,791	8,002
Falko	learner	144	5,866
Kafka	literary prose	200	4,870
Unicum	chat	787	3,694

[1] Routing directions.

Embeddings training data

The word/character embeddings are trained on:

Subcorpus	Sentences	Tokens
Taz newspaper	22.8M	393.7M
Wikipedia 201801	39.9M	803.6M
Europarl	2.2M	55.0M

Results

Tagger	TüBa-D/Z	NoSta-D	NoSta-D-A	Tokens/s
HMM trigram	97.29	83.12	86.45	108,416

Results

Tagger	TüBa-D/Z	NoSta-D	NoSta-D-A	Tokens/s
HMM trigram	97.29	83.12	86.45	108,416
skipgram	98.55	85.98	89.32	57,705

Results

Tagger	TüBa-D/Z	NoSta-D	NoSta-D-A	Tokens/s
HMM trigram	97.29	83.12	86.45	108,416
skipgram	98.55	85.98	89.32	57,705
structgram	98.65	86.03	89.29	

Results

Tagger	TüBa-D/Z	NoSta-D	NoSta-D-A	Tokens/s
HMM trigram	97.29	83.12	86.45	108,416
skipgram	98.55	85.98	89.32	57,705
structgram	98.65	86.03	89.29	
skipgram + subword	98.84	85.34	88.87	55,042

Results

Tagger	TüBa-D/Z	NoSta-D	NoSta-D-A	Tokens/s
HMM trigram	97.29	83.12	86.45	108,416
skipgram	98.55	85.98	89.32	57,705
structgram	98.65	86.03	89.29	
skipgram + subword	98.84	85.34	88.87	55,042
structgram + subword	98.98	87.45	90.67	

Results

Tagger	TüBa-D/Z	NoSta-D	NoSta-D-A	Tokens/s
HMM trigram	97.29	83.12	86.45	108,416
skipgram	98.55	85.98	89.32	57,705
structgram	98.65	86.03	89.29	
skipgram + subword	98.84	85.34	88.87	55,042
structgram + subword	98.98	87.45	90.67	
structgram + affix	99.00	86.57	89.76	28,171

Results

Tagger	TüBa-D/Z	NoSta-D	NoSta-D-A	Tokens/s
HMM trigram	97.29	83.12	86.45	108,416
skipgram	98.55	85.98	89.32	57,705
structgram	98.65	86.03	89.29	
skipgram + subword	98.84	85.34	88.87	55,042
structgram + subword	98.98	87.45	90.67	
structgram + affix	99.00	86.57	89.76	28,171
structgram + char RNN	99.03	87.03	90.01	17,284

Subcorpora

Tagger	Anselm	BeMaTac	Falko	Kafka	Unicum
structgram + affix	29.68	90.95	95.72	96.22	74.17
structgram + subword	30.14	**91.86**	**95.94**	95.98	**77.10**
structgram + char RNN	**33.73**	91.13	95.87	**96.26**	74.96

Wrap-up

Wrap-up

Conclusions:

- Recurrent neural network taggers outperform HMMs, also out-of-domain.

Wrap-up

Conclusions:

- Recurrent neural network taggers outperform HMMs, also out-of-domain.
- Pick your word representations wisely:
  1. Use subword units for good unknown word handling.
  2. Use an embedding model that takes word order into account.

Wrap-up

Conclusions:

- Recurrent neural network taggers outperform HMMs, also out-of-domain.
- Pick your word representations wisely:
  1. Use subword units for good unknown word handling.
  2. Use an embedding model that takes word order into account.
- Simpler n-gram models seem to generalize better out-of-domain than character RNNs.

Wrap-up

Conclusions:

- Recurrent neural network taggers outperform HMMs, also out-of-domain.
- Pick your word representations wisely:
  1. Use subword units for good unknown word handling.
  2. Use an embedding model that takes word order into account.
- Simpler n-gram models seem to generalize better out-of-domain than character RNNs.

Future work:

- How deeper representations such as ELMo (Peters et al., 2018) fare?
- Qualitative analysis of the errors made by the best models.

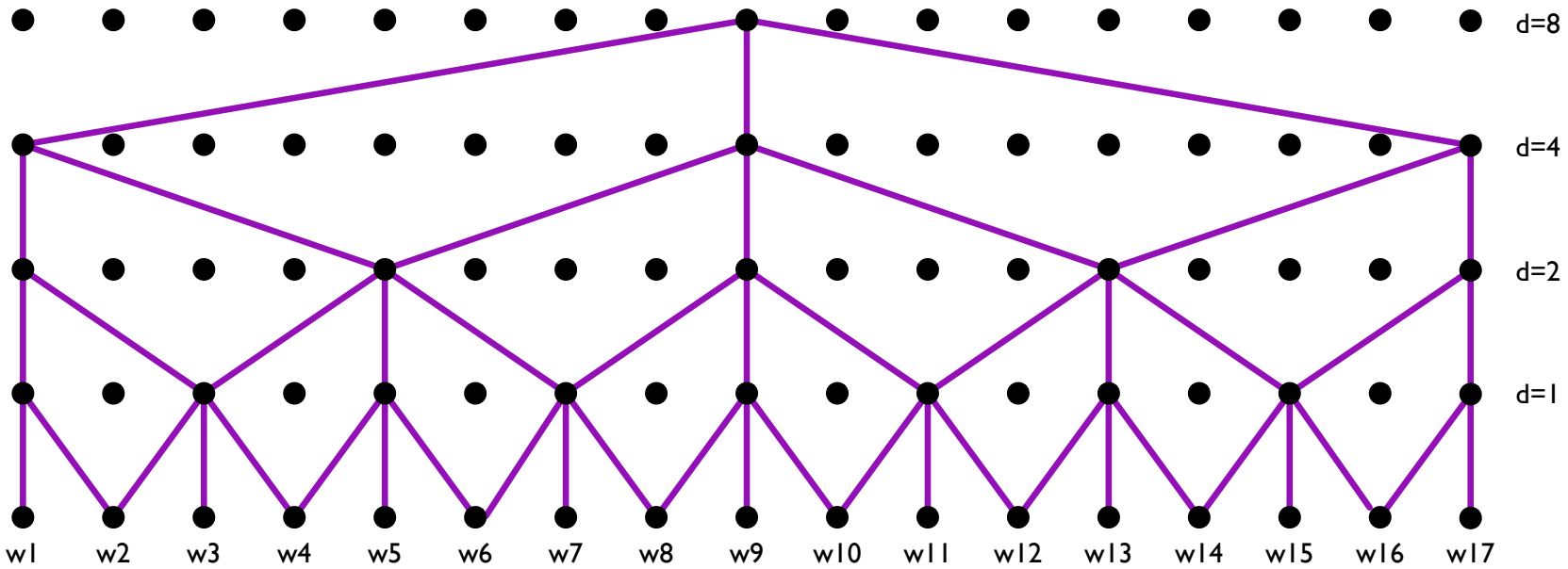
Software

- finalfrontier: https://github.com/danieldk/finalfrontier
  - Models:
    - skipgram
    - structured skipgram
    - soon: depedency
  - Subword representations
  - Experimental Python module
- sticker: https://git.sr.ht/~danieldk/sticker
  - Models
    - Bidirectional RNN
    - Dillated convolutions
  - word2vec or finalfrontier embeddings

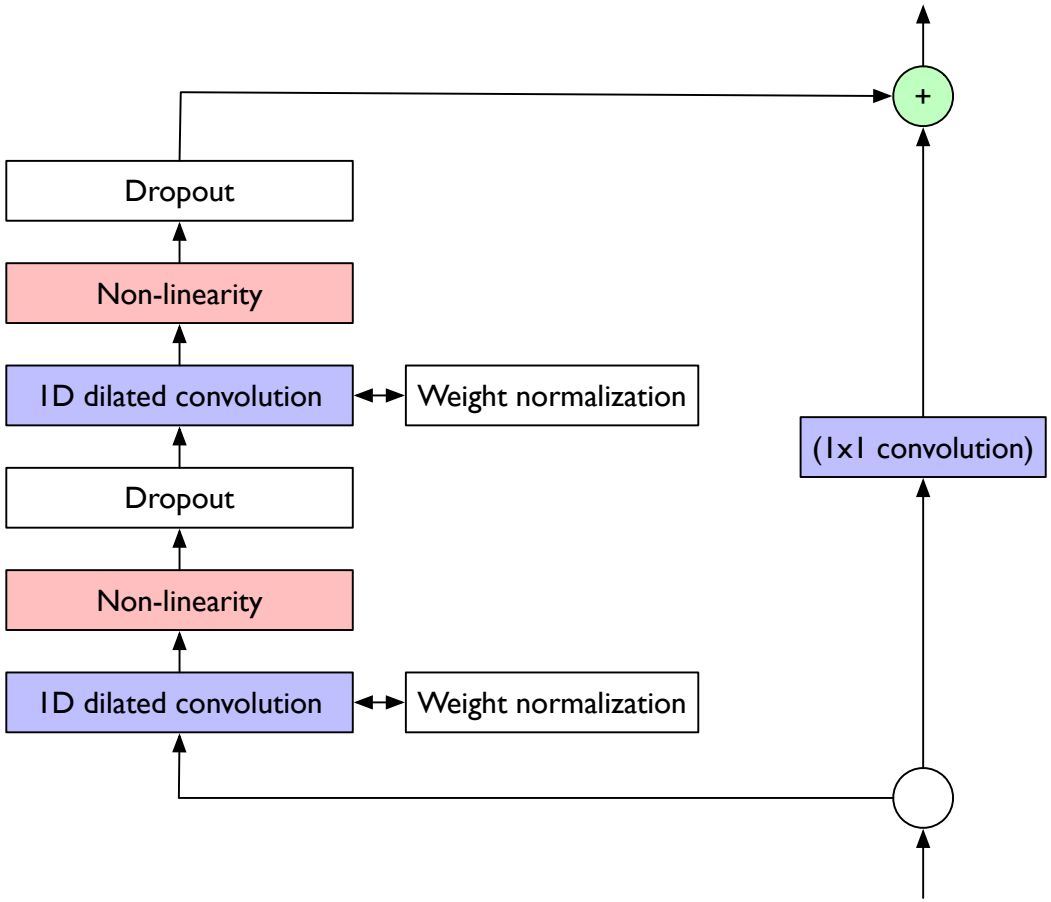
Thank you!

Appendix

Dilated convolutions



Residual units



Results dillated convolutions

Tagger	TüBa-D/Z	NoSta-D	NoSta-D-A
structgram + subword	98.98	87.45	90.67
structgram + char RNN	99.03	87.03	90.01
structgram + subword + conv	98.88	86.80	89.99

What if we throw both char RNNs and subword units at it?

Tagger	TüBa-D/Z	NoSta-D	NoSta-D-A
structgram + subword	98.98	87.45	90.67
structgram + char RNN	99.03	87.03	90.01
structgram + subword + char RNN	99.04	87.15	90.26

What about convolutions over characters?

Tagger	TüBa-D/Z	NoSta-D	NoSta-D-A
structgram + subword	98.98	87.45	90.67
structgram + char RNN	99.03	87.03	90.01
structgram + char convolutions	98.90	86.47	90.76

How many type/tokens are IV?

Corpus	% Types	% Tokens
TüBa-D/Z	84.71	97.51
Anselm corpus	40.38	53.92
BeMaTac	92.17	98.64
Falko	93.40	97.74
Kafka	96.73	98.54
Unicum	61.57	78.05

Hyperparameters

- Tagger:
  - BiDi LSTM layer: 100 units
  - LSTM layer: 100 units
  - Batch size: 512
  - Optimizer: Adam
- Character RNN:
  - BiDi LSTM layer: 50 units

Hyperparameters (2)

- Word embeddings:
  - Dimensions: 300
  - Context size: 10
  - Minimum count: 30
  - Negative samples: 5
- Character embeddings:
  - Dimensions: 50
  - Structured skip-gram
  - Context size: 5
  - Negative samples: 10