

# Recurrent Neural Language Modeling

Using Transfer Learning to Perform Radiological Sentence Completion

---

Anna Lena Popkes

January 20, 2018

Master Thesis  
University of Bonn

# Outline

Motivation and Main Idea

Foundations

Regularization Techniques

Master Thesis

Preprocessing

Architecture and Training

Experiments

Conclusion and Future Work

## **Motivation and Main Idea**

# Motivation and Main Idea

- Motivation: radiologists spend a lot of time writing medical reports
- The language of different reports is often similar for similar diagnoses
- Idea: construct language model that suggests next most likely word(s) while writing
  - ⇒ saves time and effort
  - ⇒ would help standardize reports

# Foundations

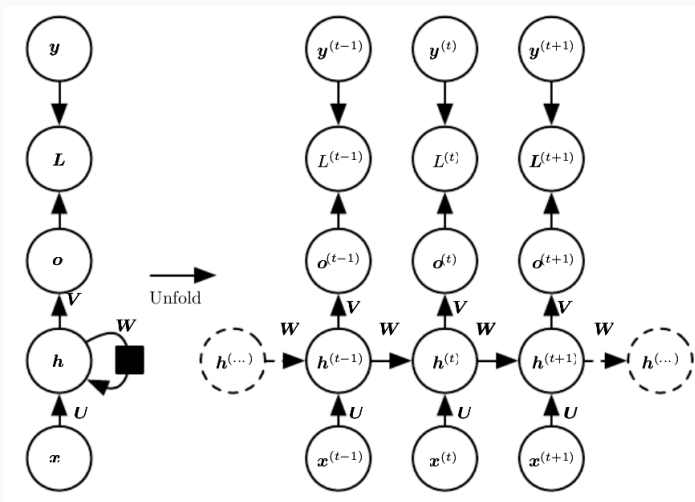
# Statistical Language Modeling

- Goal: learn the probability of a sequence of words
- $P(w_1, \dots, w_m) = \prod_{i=1}^m P(w_i \mid w_1, \dots, w_{i-1})$
- Problem: difficult to compute for large number of words
- Solution: window of  $n$  previous words
- $\prod_{i=1}^m P(w_i \mid w_1, \dots, w_{i-1}) \approx \prod_{i=1}^m P(w_i \mid w_{i-n}, \dots, w_{i-1})$
- Different types of language models exist
- Best performance: recurrent neural language models

# Recurrent Neural Network

- Recurrent neural network: special type of neural network for processing sequential data
- Contains recurrent (cyclic) connections
- Recurrent connections create a kind of "memory"

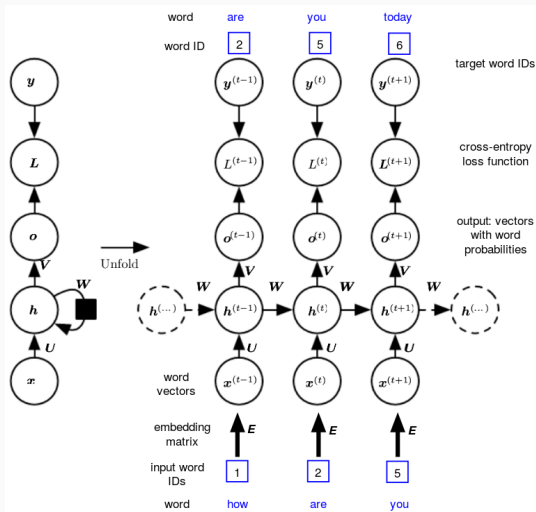
# Recurrent Neural Network



**Figure 1:** Computational graph of a basic RNN mapping an input sequence  $\mathbf{x}$  to an output sequence  $\mathbf{o}$  taken from Goodfellow et al. [2]



# Recurrent Neural Language Model



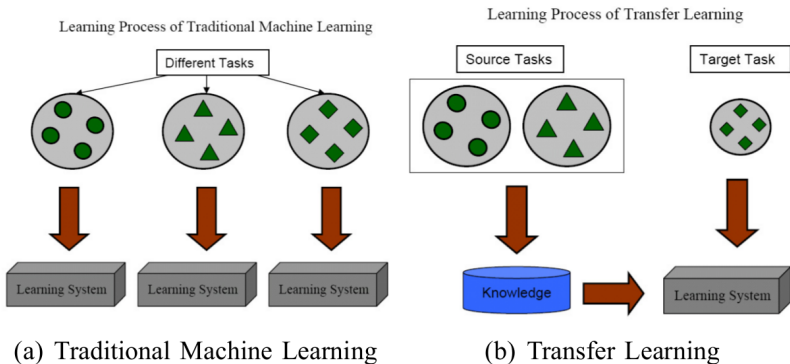
**Figure 2:** Basic recurrent language model. Adapted from Goodfellow et al. [2]

# Recurrent Neural Language Model

- Input sequence: vector of word IDs
- Words are represented as compact and dense  $d$ -dimensional vectors of real numbers (word embeddings)
- Hidden units: long short-term memory (LSTM) units
- LSTMs improve ability to learn long-term dependencies
- Output units with softmax activation function
- Output: vector with as many entries as words in vocabulary
- Contains for each word the probability that it will be the next word in the sequence
- Target sequence: vector of word IDs

# Transfer Learning

- Idea: extract knowledge from source task(s) to improve performance on target task
- Many different forms of transfer learning exist
- Example: transfer pre-trained weights



# Regularization Techniques

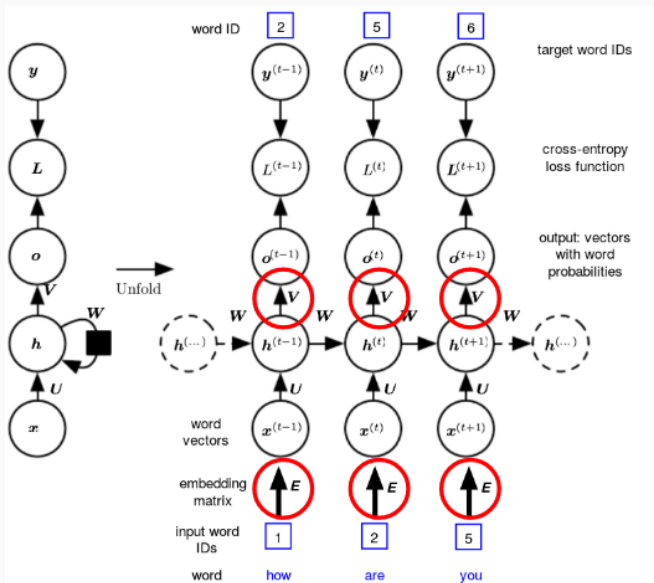
# Regularization Techniques

- Weight tying
- Variational dropout
- Embedding dropout

# Weight tying

- Idea: share weights between embedding and softmax layer
- Reduces total number of model parameters
- Weight tying was reported to improve LSTM language models [3, 7, 4, 5]
- Given word embedding matrix  $E \in \mathbb{R}^{d_x \times |V|}$  and output projection matrix  $V \in \mathbb{R}^{|V| \times d_h}$  set  $V = E^T$

# Weight Tying

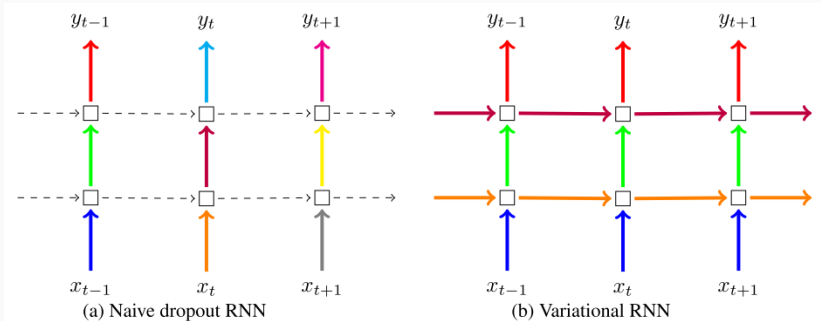


## Variational Dropout (VD)

- Proposed by Gal & Ghahramani in 2016 [1]
- Standard dropout: new binary dropout mask at each time step, no recurrent dropout
- Example: input  $x_0$  to LSTM at time step  $t = 0$  receives different dropout mask than  $x_1$  of same LSTM at  $t = 1$
- Variational dropout: sample binary dropout mask only once, repeatedly use the same mask for all time steps
- Within mini-batch, each example uses unique dropout mask



# Variational Dropout vs. Standard Dropout



**Figure 5:** Comparison of standard dropout (left) and variational dropout (right). Taken from Gal & Ghahramani [1]

# Embedding Dropout

- Also proposed by Gal & Ghahramani [1]
- Embedding matrix is typically optimised during training  
⇒ can lead to overfitting
- Solution: perform dropout on embedding matrix
- Set random rows of embedding matrix to zero
- Corresponds to randomly dropping specific words throughout the input sequence
- Example: "*the dog and the cat*" might become "- dog and - cat" but never "- dog and the cat"
- Encourages model to be independent of certain words in input

# Master Thesis

- Recurrent Neural Network with LSTM units
- Two different German data sets: radiological reports and motortalk forum post
- Tested effects:
  - Transfer Learning
  - Pre-trained embeddings
  - Further inputs

# Pipeline

1. Initial experiments with Penn Treebank Dataset
2. Two rounds of random search for both datasets
3. Select best model architecture (baseline model)
4. Train transfer model for each dataset
5. Train model with pre-trained embeddings for each dataset
6. Analyze medical word embeddings
7. Test sentence completion
8. Test further ways to incorporate knowledge  
Example: augment training examples with additional input vector

# Preprocessing

# Preprocessing

- Strings containing numbers mapped to special *number* string
- 1.920.000 training words, 480.000 validation words and 600.000 test words
- Final vocabulary: all words with count  $\geq 10$
- Vocabulary sizes: 88575 (motortalk) and 21344 (radiology)

# **Architecture and Training**



# Model Architecture and Training

- LSTM architecture with three hidden layers [4, 5]
- Training with stochastic gradient descent (SGD)
- SGD was found to outperform newer algorithms like momentum SGD, Adam, Adagrad and RMSProp [5]
- Evaluation with average per-word perplexity on test set
- Early stopping to prevent overfitting
- Two rounds of random search

# Experiments

## Random Search - Round 1

- Goal: fix model architecture
- Tested parameters: number of hidden neurons, embedding size, decay rate, epoch to start decaying
- Parameter ranges chosen using recent publications [10, 5, 4]
- Weight tying in all models
- Standard input and output dropout of 0.3
- Best architecture selected for round 2

## Random Search - Round 2

- Goal: optimize best model architecture
- Vary dropout and gradient clipping rates
- Parameter ranges chosen using recent publications [10, 5, 4]
- Some models were tested with variational dropout
- Best model selected as *baseline* model

# Transfer Learning

- Final weights of *baseline*-models were used to initialize new architectures (*motor-transfer*, *radio-transfer*)
- *transfer*-models trained solely on
  1. data from the BMW-forums
  2. radiological reports from patients transferred from oncological department
  3. radiological reports for which the radiological examination was given by an X-ray of the patient's thorax while the patient was lying down (abbreviation: KTHL)
- As a comparison, models with randomly initialized parameters were trained

## Transfer Learning Results - Motortalk

model	# epochs	valid perplexity	test perplexity
motor-baseline	41	224.92	218.91
motor-baseline+	38	218.40	212.30
motor-BMW	44	222.70	253.91
motor-BMW+	43	217.83	252.85
motor-transfer	45	183.48	210.40

**Table 1:** Comparison of test and validation perplexity of the motortalk baseline model, transfer model and BMW model

## Transfer Learning Results - Radiology

model	# epochs	valid perplex.	test perplex.
radio-baseline	41	6.16	6.72
radio-baseline+	26	6.17	6.73
radio-oncology	21	23.20	37.87
radio-oncology+	22	23.74	38.26
radio-transfer-oncology	21	23.24	39.96
radio-kthl	23	5.79	6.91
radio-kthl+	38	5.77	6.95
radio-transfer-kthl	26	5.73	6.88

**Table 2:** Comparison of test and validation perplexity of the radiological baseline model, transfer, word2vec model and oncology model

## Pre-trained embeddings

- Train another model whose embedding matrix is initialized with pre-trained embeddings
- CBOW model and Skip-gram model to pre-train embeddings



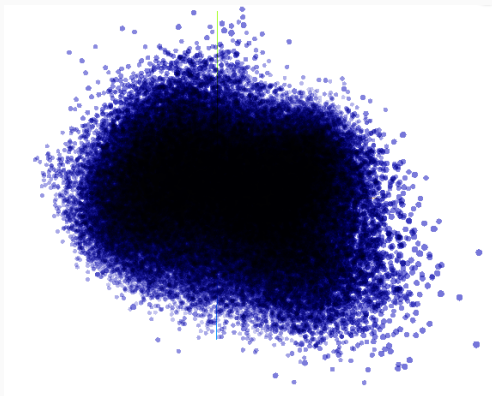
## Pre-Trained Embeddings Results - Radiology

model	# epochs	valid perplex.	test perplex.
radio-baseline	41	6.16	6.72
radio-CBOW	28	6.24	6.85
radio-skipgram	26	6.13	6.68

**Table 3:** Comparison of test and validation perplexity of the radiological word2vec models

# Medical word embeddings

- CBOW model trained on radiological dataset
- Visualization using Principal Component Analysis (PCA)



**Figure 6:** Visualization of medical word embeddings

# Medical word embeddings

- Learned embeddings encode linguistic regularities and patterns that can be represented as linear translations [6]
- Example:  $\text{vec}(\text{heart}) + \text{vec}(\text{enlarged}) \approx \text{vec}(\text{cardiomegaly})$
- Nearest neighbor words reveal semantic similarity between neighboring word vectors
- Example: 5 nearest neighbours of "Blutung" (english: bleeding) are:
  1. "Einblutung"
  2. "Nachblutung"
  3. "Blutungen"
  4. "Blutungskomponente"
  5. "Einnachblutung".

# Medical word embeddings

- Also medical relationships are encoded in the embeddings
- Example: comparing nearest neighbors of "Demenz" (English: dementia) and "Alzheimer" (English: Alzheimer's)

demenz	alzheimer
depression	pick
psychose	parkinson
schizophrenie	addison
alzheimerdemenz	behcet
pnp (polyneuropathie)	menire

**Table 4:** The five nearest neighbors of the words "Demenz" (dementia) and "Alzheimer" (Alzheimer's) in the embedding space learnt by a CBOW model.

- Radiology-transfer-oncology:

Es zeigt sich eine deutliche Regredienz der bekannten multiplen Metastasen. Die größte Metastase im rechten Os ilium ist aktuell nicht mehr nachweisbar. Beurteilung *number* im Verlauf unveränderte Darstellung der bekannten multiplen Lebermetastasen. *number* kein Nachweis neu aufgetretener Metastasen. *number* unverändert geringe Menge freier Flüssigkeit im kleinen Becken. *number* unverändert kein Nachweis pathologisch vergrößerter Lymphknoten.

## **Conclusion and Future Work**

# Conclusion

- Basic recurrent language model performs well but can be enhanced by features like embedding dropout
- Increasing the learning rate at the end of training can be beneficial
- Transfer learning can be beneficial  
⇒ The dataset plays an important role!
- Recurrent language models could be used to perform sentence completion

## Future Work

- Evaluate recurrent language model in praxis
- Change learning rate schedule
- Test different ways to incorporate context
- Correct spelling mistakes, normalize synonyms and Latin spelling variations
- Ensemble learning



*Thank you for your attention!*

# References



Yarin Gal and Zoubin Ghahramani. “A theoretically grounded application of dropout in recurrent neural networks”. In: *Advances in neural information processing systems*. 2016, pp. 1019–1027.



Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.



Hakan Inan, Khashayar Khosravi, and Richard Socher. “Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling”. In: *arXiv preprint arXiv:1611.01462* (2016).



Gábor Melis, Chris Dyer, and Phil Blunsom. “On the state of the art of evaluation in neural language models”. In: *arXiv preprint arXiv:1707.05589* (2017).



Stephen Merity, Nitish Shirish Keskar, and Richard Socher. “Regularizing and optimizing LSTM language models”. In: *arXiv preprint arXiv:1708.02182* (2017).



Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: (2013), pp. 1–12. ISSN: 15324435. DOI: 10.1162/153244303322533223. arXiv: 1301.3781. URL: <http://arxiv.org/abs/1301.3781>.



Ofir Press and Lior Wolf. “Using the output embedding to improve language models”. In: *arXiv preprint arXiv:1608.05859* (2016).



David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. “Learning representations by back-propagating errors”. In: *Cognitive modeling* 5.3 (1988), p. 1.



Zhilin Yang et al. “Breaking the Softmax Bottleneck: A High-Rank RNN Language Model”. In: *arXiv preprint arXiv:1711.03953* (2017).



Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. “Recurrent neural network regularization”. In: *arXiv preprint arXiv:1409.2329* (2014).

## **Appendix**

# N-Gram Models

- N-gram = sequence of  $n$  words
- Example: "three guys standing" (trigram)
- Trigram model:  $P(w_1, \dots, w_m) \approx \prod_{i=1}^m P(w_i \mid w_{i-1}, w_{i-2})$
- N-gram model:

$$P(w_1, \dots, w_m) \approx \prod_{i=1}^m P(w_i \mid w_{i-1}, \dots, w_{i-N+1})$$

- N-gram probabilities can be computed with *maximum likelihood estimation* (MLE):

$$P(w_i \mid w_{i-1}, \dots, w_{i-N+1}) = \frac{\text{count}(w_{i-N+1}, \dots, w_{i-1}, w_i)}{\text{count}(w_{i-N+1}, \dots, w_{i-1})}$$

## N-Gram Models - Example Trigram Model

- Context  $h$ : "the green"
- Goal: predict probability of next word  $w$  being "tree"
- Count how many times "the green" is followed by "tree" and normalize
- $P(w_i \mid w_{i-1}, w_{i-2}) = P(\text{tree} \mid \text{the green}) = \frac{\text{count}(\text{the green tree})}{\text{count}(\text{the green})}$

# N-Gram Models - Problems

- Many word sequences tend to occur only rarely or not at all
- Example: what is  $P(\text{tree} \mid \text{the green})$ ?
- Training corpus might not contain any instance of the sequence
  - $\Rightarrow \text{count}(\text{the green tree})$  would be zero
  - $\Rightarrow P(\text{tree} \mid \text{the green})$  would be zero
  - $\Rightarrow$  underestimates  $P(\text{tree} \mid \text{the green})$
- Solution: smoothing



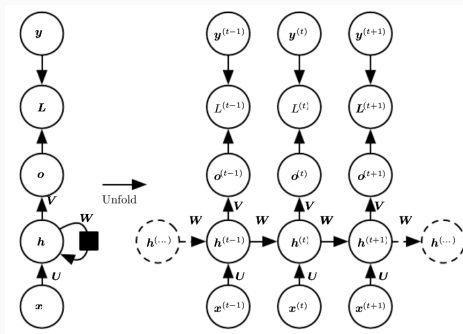
# Word Embeddings

- Idea: represent words as compact and dense  $d$ -dimensional vectors of real numbers
- Based of the *distributional hypothesis*: words occurring in a similar context tend to have similar meanings
- Also called word embeddings
- Most popular technique: word2vec

# Recurrent Neural Networks

- Special type of neural network for processing sequential data
- Contains recurrent (cyclic) connections
- Basic RNN hidden units:  $\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}, \boldsymbol{\theta})$
- Recurrent connections create a kind of "memory"
- Trained with stochastic gradient descent and backpropagation through time

# Recurrent Neural Networks



**Figure 7:** Computational graph of a basic RNN mapping an input sequence  $\mathbf{x}$  to an output sequence  $\mathbf{o}$

Activations of hidden units:

$$\mathbf{h}^{(t)} = \sigma_h(\mathbf{b} + \mathbf{W} \cdot \mathbf{h}^{(t-1)} + \mathbf{U} \cdot \mathbf{x}^{(t)})$$

Output:

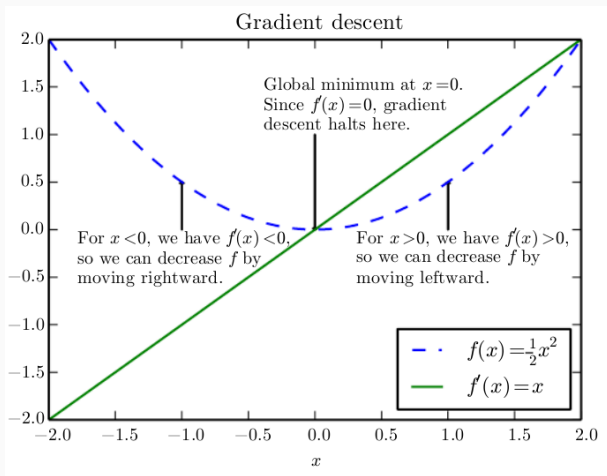
# Recurrent Neural Networks

- Loss function: sum of the individual losses
- Typical loss function for RNNs: *cross entropy loss*
- Cross entropy between probability distribution  $\hat{\mathbf{y}}$  predicted by network, and true distribution  $\mathbf{y}$ :  
$$H_{\mathbf{y}}(\hat{\mathbf{y}}) = - \sum_i y_i \log(\hat{y}_i)$$
- Training with backpropagation through time (BPTT) and stochastic gradient descent (SGD)
- Major challenge for RNNs: learning long-term dependencies
- Possible solution: use *gated* recurrent units, such as Long Short-Term Memory unit

# Gradient-Based Training

- Gradient descent: optimization algorithm to minimize/maximize  $f(x)$
- Procedure: start with initial value  $x_0$
- Make small steps into direction of negative gradient
- $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla f(\mathbf{w}^{(t)})$

# Gradient-Based Training



**Figure 8:** Illustration of gradient descent

# Gradient-Based Training

- Stochastic gradient descent: gradient is computed using mini-batch of training examples
- Computing  $\nabla f(\mathbf{w}^{(t)})$  requires partial derivatives of  $f$  with respect to each parameter
- Done with the *backpropagation* algorithm [8]
- Backpropagation Through Time: extension of backpropagation
- Main idea: *unfold* recurrent architecture and apply backpropagation to the unfolded computational graph

# Evaluating Language Models

- Performance can be evaluated in two ways - extrinsically and intrinsically
- Both strategies complement each other



# Extrinsic Evaluation

- Refers to embedding the language model in an application and measuring how much the application improves
- Only possibility to determine which modifications will be useful with respect to the desired application
- Extrinsic evaluation in word-based language modeling: run word prediction system with different language models and compare predictions
- Disadvantage: high computational complexity

# Intrinsic Evaluation

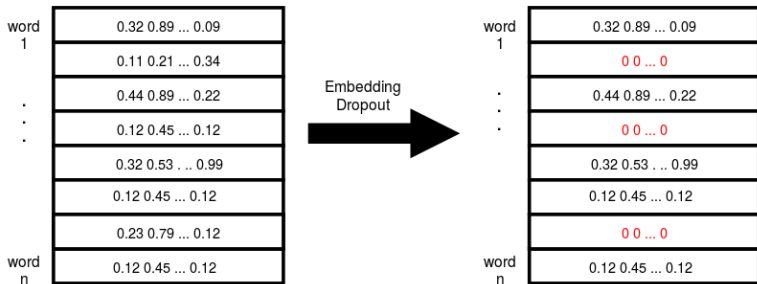
- Typical intrinsic evaluation metric: perplexity
- Perplexity of language model on test set  $\mathbf{w} = \{w_1, \dots, w_m\}$  is the inverse probability of  $\mathbf{w}$ , normalized by the number of words:

$$\begin{aligned} PP(\mathbf{w}) &= P(w_1, \dots, w_m)^{-\frac{1}{m}} \\ &= \sqrt[m]{\frac{1}{P(w_1, \dots, w_m)}} \\ &= \sqrt[m]{\prod_{i=1}^m \frac{1}{P(w_i \mid w_1, \dots, w_{i-1})}} \end{aligned} \tag{1}$$

# Intrinsic Evaluation

- Perplexity is low if conditional probability of word sequence is high  
⇒ Minimizing perplexity of test set is equivalent to maximizing probability of the test set according to the language model
- Perplexity has many favourable properties
- For example, when given a set of test data, it can be evaluated easily
- Also, it is closely related to the cross entropy between the model and a test dataset

# Embedding Dropout



**Figure 9:** Illustration of embedding dropout which randomly selects rows of the embedding matrix and sets them to zero

## **Initial Experiments**

## Initial Experiments - Dataset

- Initial experiments with preprocessed version of Penn Treebank (PTB) data set
- Still widely used in experiments (e.g. [5, 3, 9])
- Reproduction of results of Zaremba et al. [10]
- Test effects of: VD, weight tying, embedding dropout
- Additional weight tying or embedding dropout improves results if model is not regularized too much
- VD sometimes causes extreme perplexity jumps

## Initial Experiments - Model

- Reproduction of results of Zaremba et al. [10]
- 2-layer LSTM architecture
- Three different network sizes: large (1500 units), medium (650 units), small (200 units)

## Initial Experiments - Observations

- Additional embedding dropout improves results if model is not regularized too much
- Example:

Emb. D	Input D	Rec. D	Out. D	TW	Valid	Test
0.3	0.3		0.3		99.33	96.14
	0.3		0.3		106.03	102.95
0.1	0.1		0.1	✓	108.05	103.76
	0.1		0.1	✓	95.7	91.79

**Table 5:** Effect of embedding dropout. Comparison of the word-level perplexities on the validation and test set for small models tested on the PTB dataset



## Initial Experiments - Observations

- Weight tying improves results if model is not regularized too much
- Example:

Input D	Rec. D	Out D	TW	VD	Valid	Test
0.2	0.2	0.2		✓	99.09	94.83
0.2	0.2	0.2	✓	✓	101.06	97.03
0.1	0.1	0.1		✓	101.24	97.98
0.1	0.1	0.1	✓	✓	98.30	94.28

**Table 6:** Effect of weight tying. Comparison of the word-level perplexities on the validation and test set for small models tested on the PTB dataset

## Initial Experiments - Observations

- Variational dropout (VD) not useful per se
- Comparing small model standard dropout  $\Leftrightarrow$  small model VD: VD only improves performance if overall regularization not too high
- Medium model: VD sometimes causes extreme perplexity jumps
- Vanishes when using weight tying

Input D	Rec. D	Out. D	TW	VD	Valid	Test
0.5	0.3	0.5		✓	686.95	639.65
0.5	0.3	0.5	✓	✓	95.29	91.51

**Table 7:** Effect of variational dropout. Comparison of the word-level perplexities on the validation and test set for medium models tested on the PTB dataset

## Initial Experiments - Results

Network	Model	Valid	Test	Valid	Test
				(Zaremba et. al)	
Small	LSTM-SD	99.33	96.14	-	-
	LSTM-SD+TW+ED	95.7	91.79	-	-
Medium	LSTM-SD	86.11	82.88	86.2	82.7
	LSTM-VD+TW+ED	82.0	78.64	-	-

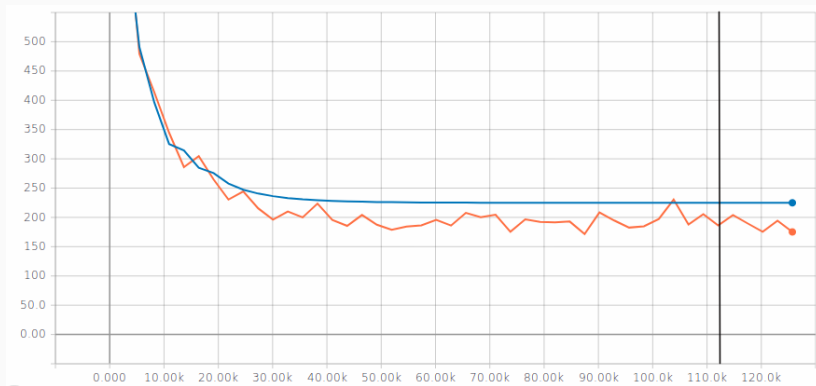
**Table 8:** Comparison of the word-level perplexities on the validation and test set for different models tested on the PTB dataset. VD stands for variational dropout, TW for weight tying, ED for embedding dropout.

## Sentence Completion - Motortalk

- *motor-baseline* (best word)

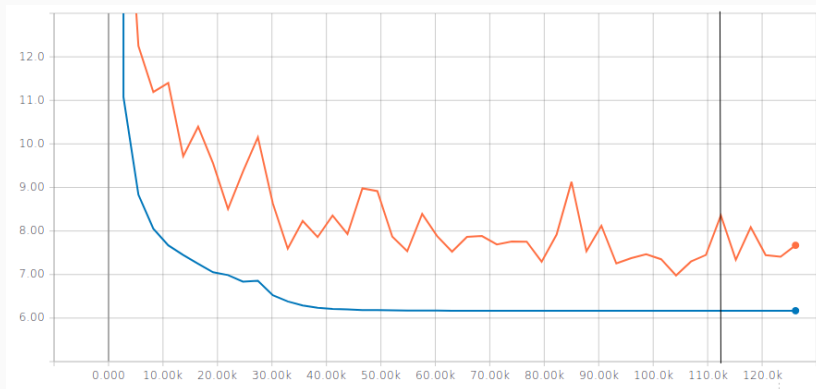
Möglicherweise nicht. Ich habe mir vor kurzem einen *number* gekauft und habe mir die *number* Zoll Felgen gekauft. Ich habe mir die *number* Felgen gekauft und die sind auch nicht zugelassen.

## Random Search - Training curve Motortalk



**Figure 10:** Development of training (orange) and validation (blue) perplexity of the best model (*motor-baseline*) trained on the motortalk corpus. x-axis: total number of training steps/batches (2733 batches per epoch). Early stopping point marked with black line.

# Random Search - Training curve Radiology



**Figure 11:** Development of training (orange) and validation (blue) perplexity of the best model (*radiology-baseline*) trained on the radiological corpus. x-axis: total number of training steps/batches (2733 batches per epoch). Early stopping point marked with black line.

