Neural Models for NLP

Lecture 15

COMS 4705, Spring 2020

Yassine Benajiba

Columbia University

One more thing about embeddings

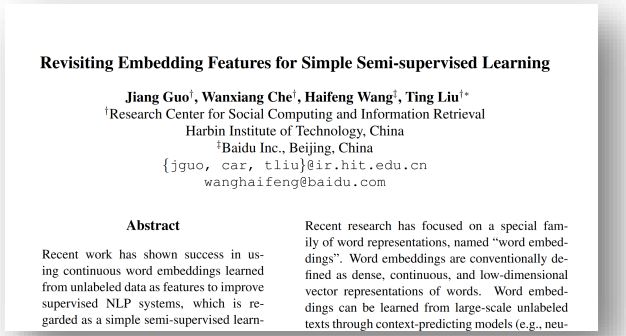
Revisiting Embedding Features for Simple Semi-supervised Learning

Jiang Guo[†], **Wanxiang Che**[†], **Haifeng Wang**[‡], **Ting Liu**[†*]
[†]Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
[‡]Baidu Inc., Beijing, China
{jguo, car, tliu}@ir.hit.edu.cn
wanghaifeng@baidu.com

Abstract

Recent work has shown success in using continuous word embeddings learned from unlabeled data as features to improve supervised NLP systems, which is regarded as a simple semi-supervised learning

Recent research has focused on a special family of word representations, named "word embeddings". Word embeddings are conventionally defined as dense, continuous, and low-dimensional vector representations of words. Word embeddings can be learned from large-scale unlabeled texts through context-predicting models (e.g., neu-



Revisiting Embedding Features for Simple Semi-supervised Learning

Jiang Guo[†], **Wanxiang Che**[†], **Haifeng Wang**[‡], **Ting Liu**[†*]
[†]Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
[‡]Baidu Inc., Beijing, China
{jguo, car, tliu}@ir.hit.edu.cn
wanghaifeng@baidu.com

Abstract

Recent work has shown success in using continuous word embeddings learned from unlabeled data as features to improve supervised NLP systems, which is regarded as a simple semi-supervised learn-

Recent research has focused on a special family of word representations, named "word embeddings". Word embeddings are conventionally defined as dense, continuous, and low-dimensional vector representations of words. Word embeddings can be learned from large-scale unlabeled texts through context-predicting models (e.g., neu-

3.2 Binarization of Embeddings

One fairly natural approach for converting the continuous-valued word embeddings to discrete values is binarization by dimension.

Formally, we aim to convert the continuous-valued embedding matrix $C_{d\times N}$, to another matrix $M_{d\times N}$ which is discrete-valued. There are various conversion functions. Here, we consider a simple one. For the i^{th} dimension of the word embeddings, we divide the corresponding row vector C_i into two halves for positive (C_{i+}) and negative (C_{i-}), respectively. The conversion function is then defined as follows:

$$M_{ij} = \phi(C_{ij}) = \begin{cases} U_+, & if\ C_{ij} \geq mean(C_{i+}) \\ B_-, & if\ C_{ij} \leq mean(C_{i-}) \\ 0, & otherwise \end{cases}$$

where $mean(v)$ is the mean value of vector v, U_+ is a string feature which turns on when the value

Revisiting Embedding Features for Simple Semi-supervised Learning

Jiang Guo[†], **Wanxiang Che**[†], **Haifeng Wang**[‡], **Ting Liu**[†*]
[†]Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
[‡]Baidu Inc., Beijing, China
{jguo, car, tliu}@ir.hit.edu.cn
wanghaifeng@baidu.com

Abstract

Recent work has shown success in using continuous word embeddings learned from unlabeled data as features to improve supervised NLP systems, which is regarded as a simple semi-supervised learn-

Recent research has focused on a special family of word representations, named "word embeddings". Word embeddings are conventionally defined as dense, continuous, and low-dimensional vector representations of words. Word embeddings can be learned from large-scale unlabeled texts through context-predicting models (e.g., neu-

3.2 Binarization of Embeddings

One fairly natural approach for converting the continuous-valued word embeddings to discrete values is binarization by dimension.

Formally, we aim to convert the continuous-valued embedding matrix $C_{d\times N}$, to another matrix $M_{d\times N}$ which is discrete-valued. There are various conversion functions. Here, we consider a simple one. For the i^{th} dimension of the word embeddings, we divide the corresponding row vector C_i into two halves for positive (C_{i+}) and negative (C_{i-}), respectively. The conversion function is then defined as follows:

$$M_{ij} = \phi(C_{ij}) = \begin{cases} U_+, & if\ C_{ij} \geq mean(C_{i+}) \\ B_-, & if\ C_{ij} \leq mean(C_{i-}) \\ 0, & otherwise \end{cases}$$

where $mean(v)$ is the mean value of vector v, U_+ is a string feature which turns on when the value

Setting	F1
Baseline	83.43
+DenseEmb†	86.21
+BinarizedEmb	86.75
+ClusterEmb	86.90
+DistPrototype	**87.44**
+BinarizedEmb+ClusterEmb	87.56
+BinarizedEmb+DistPrototype	87.46
+ClusterEmb+DistPrototype	**88.11**
+Brown	87.49
+Brown+ClusterEmb	88.17
+Brown+DistPrototype	88.04
+Brown+ClusterEmb+DistPrototype	**88.58**
Finkel et al. (2005)	86.86
Krishnan and Manning (2006)	87.24
Ando and Zhang (2005)	89.31
Collobert et al. (2011)	88.67

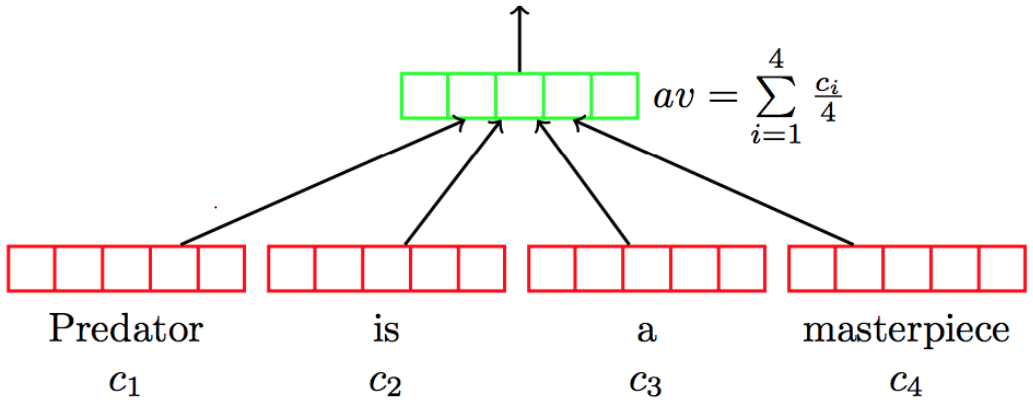
Table 3: The performance of semi-supervised NER on the CoNLL-2003 test data, using various embedding features. † DenseEmb refers to the method used by Turian et al. (2010), i.e., the direct use of the dense and continuous embeddings.

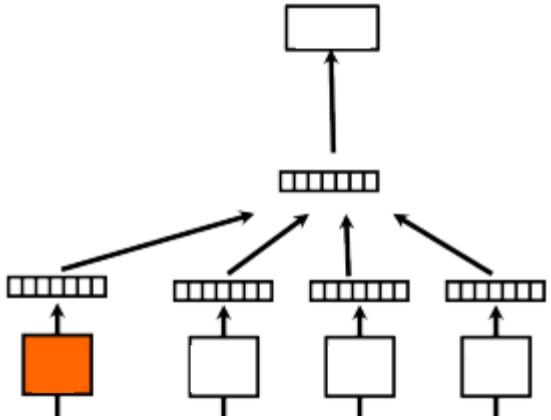
No learning

No learning

If you don’t want to use any additional learning, you could just stop here and take the average of the embeddings as the representation of the sentence. What are the pros and cons of such an approach?

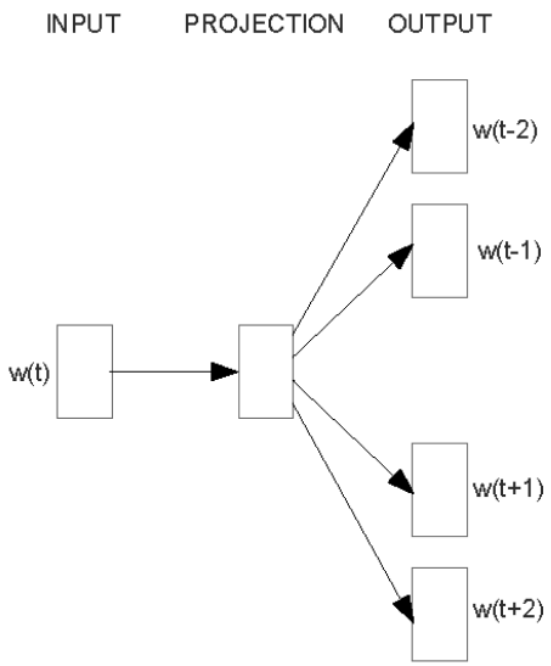
What if you wanted to weight the words without any additional learning, what could you use?



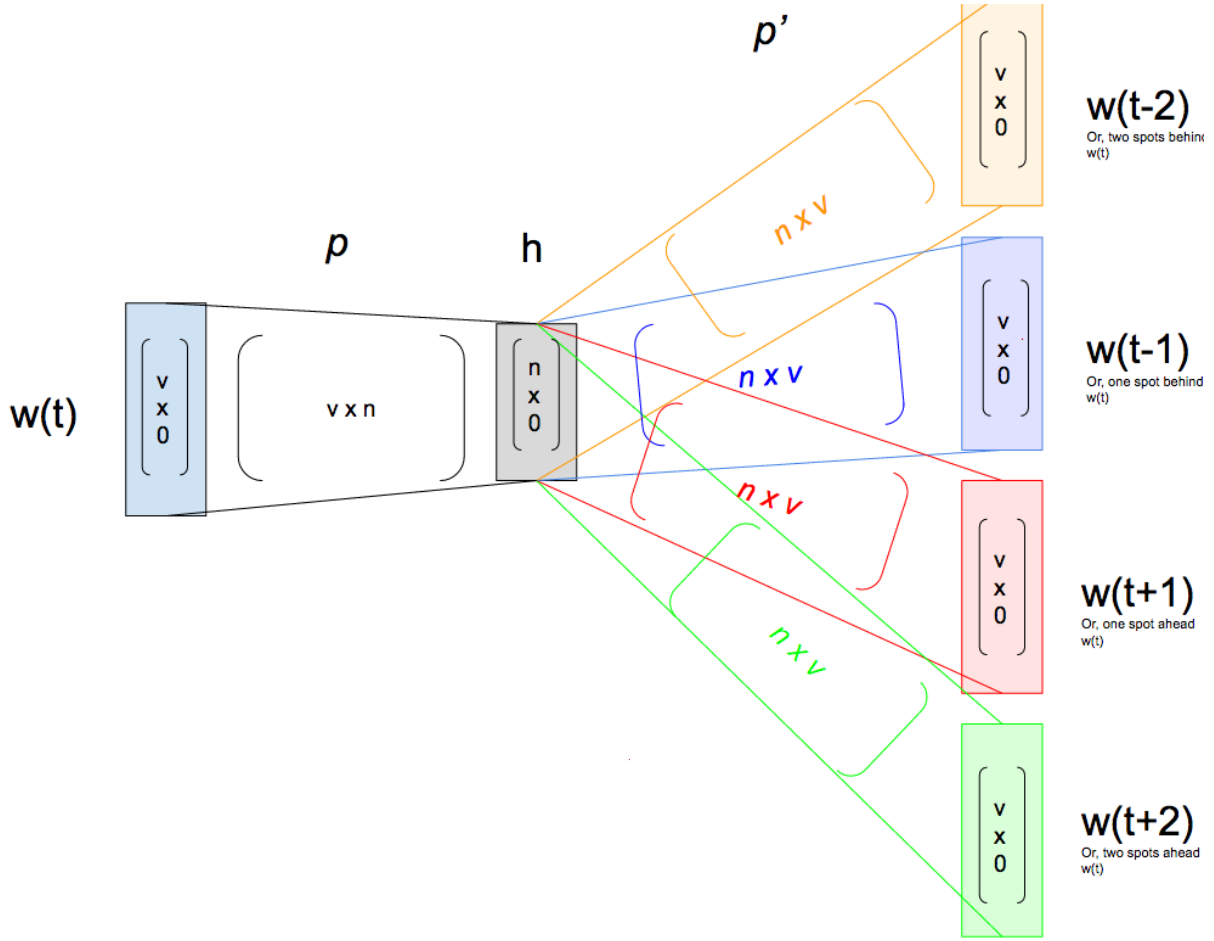


doc2vec

Doc2vec/paragraph2vec



Skip-gram



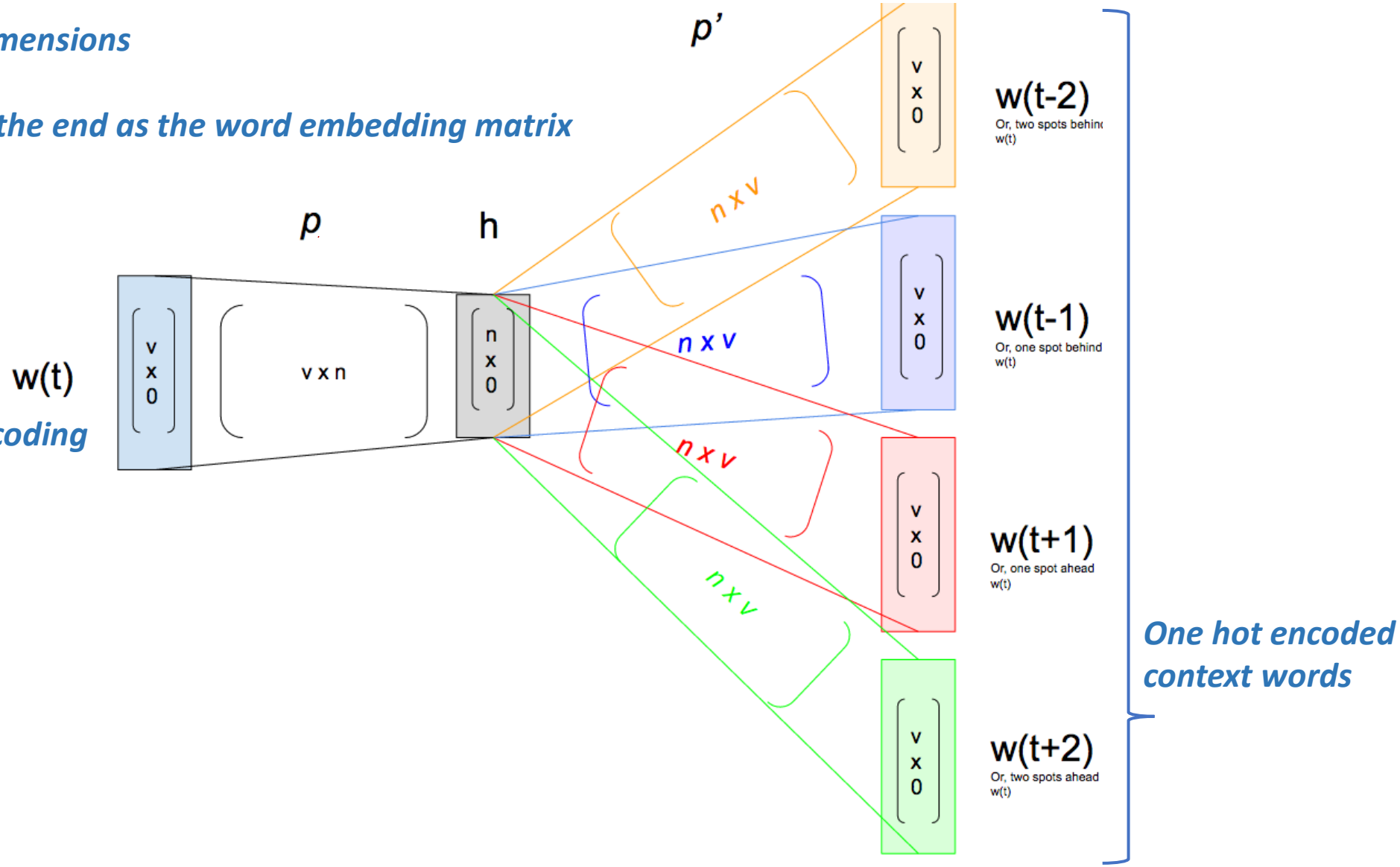
https://medium.com/district-data-labs/forward-propagation-building-a-skip-gram-net-from-the-ground-up-9578814b221

At the word level, Skip gram takes a word as an input and predicts its context.
We can do few changes and get a document representation.

Doc2vec/paragraph2vec

p': Matrix to increase # dimensions
This is what we throw away at the end

p: Matrix to reduce # dimensions
One row per word
This is what we save at the end as the word embedding matrix



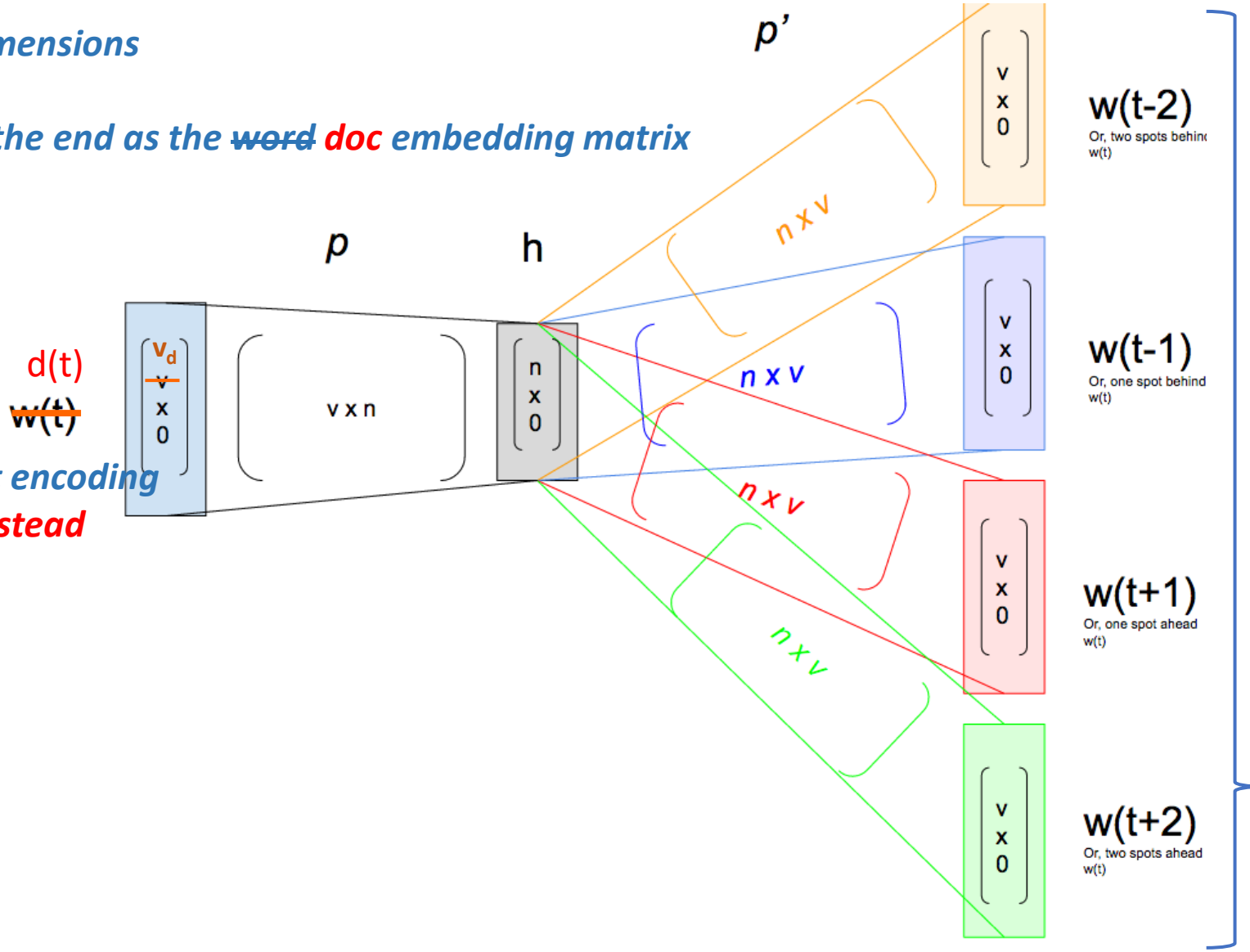
Input word one hot encoding

One hot encoded context words

Doc2vec/paragraph2vec

p': Matrix to increase # dimensions
This is what we throw away at the end

p: Matrix to reduce # dimensions
One row per ~~word~~ doc
This is what we save at the end as the ~~word~~ doc embedding matrix



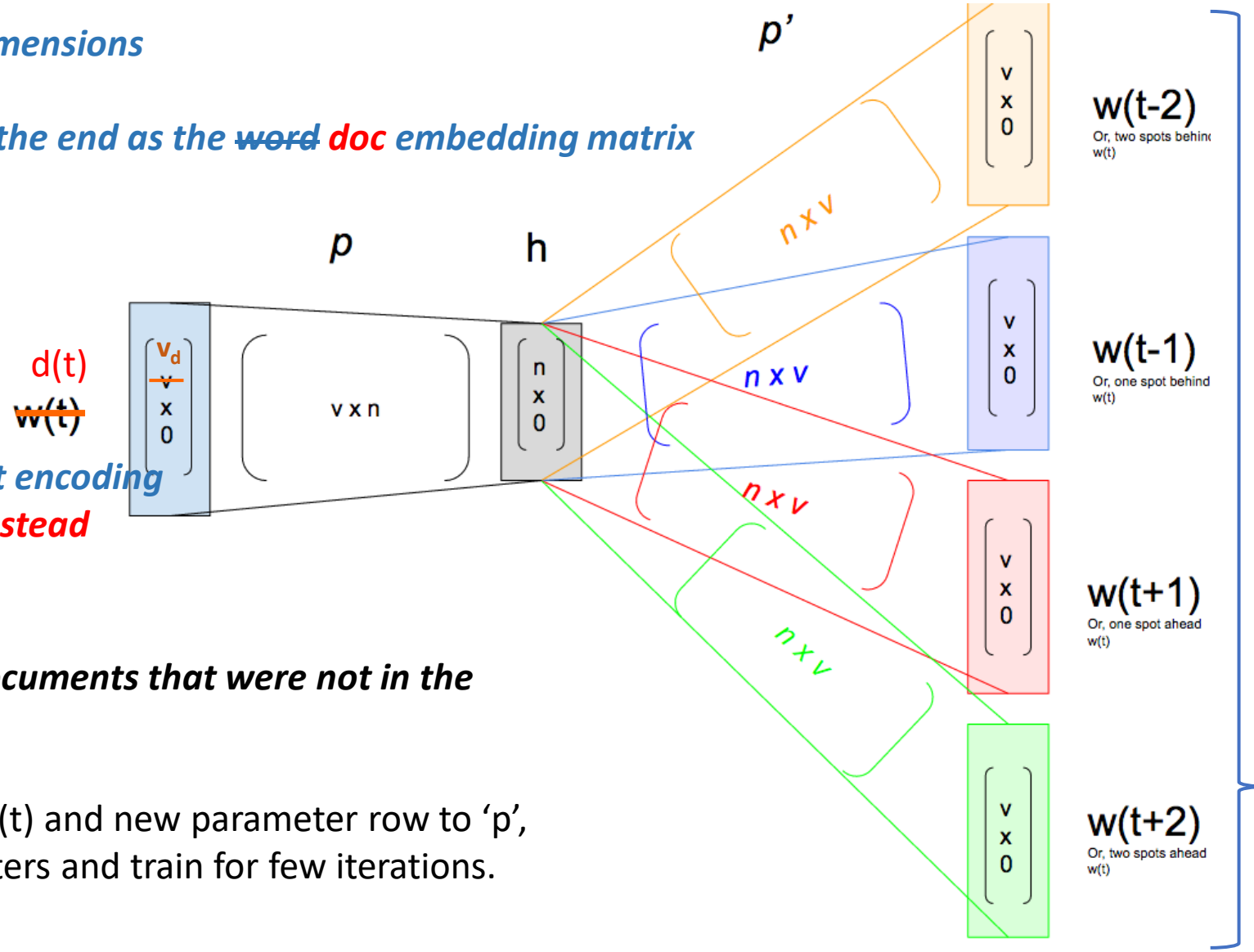
Input ~~word~~ doc one hot encoding
We should call it d(t) instead

One hot encoded ~~context~~ words occurring in the document

Doc2vec/paragraph2vec

p': Matrix to increase # dimensions
This is what we throw away at the end

p: Matrix to reduce # dimensions
One row per ~~word~~ doc
This is what we save at the end as the ~~word~~ doc embedding matrix



Input ~~word~~ doc one hot encoding
We should call it d(t) instead

One hot encoded ~~context~~ words occurring in the document

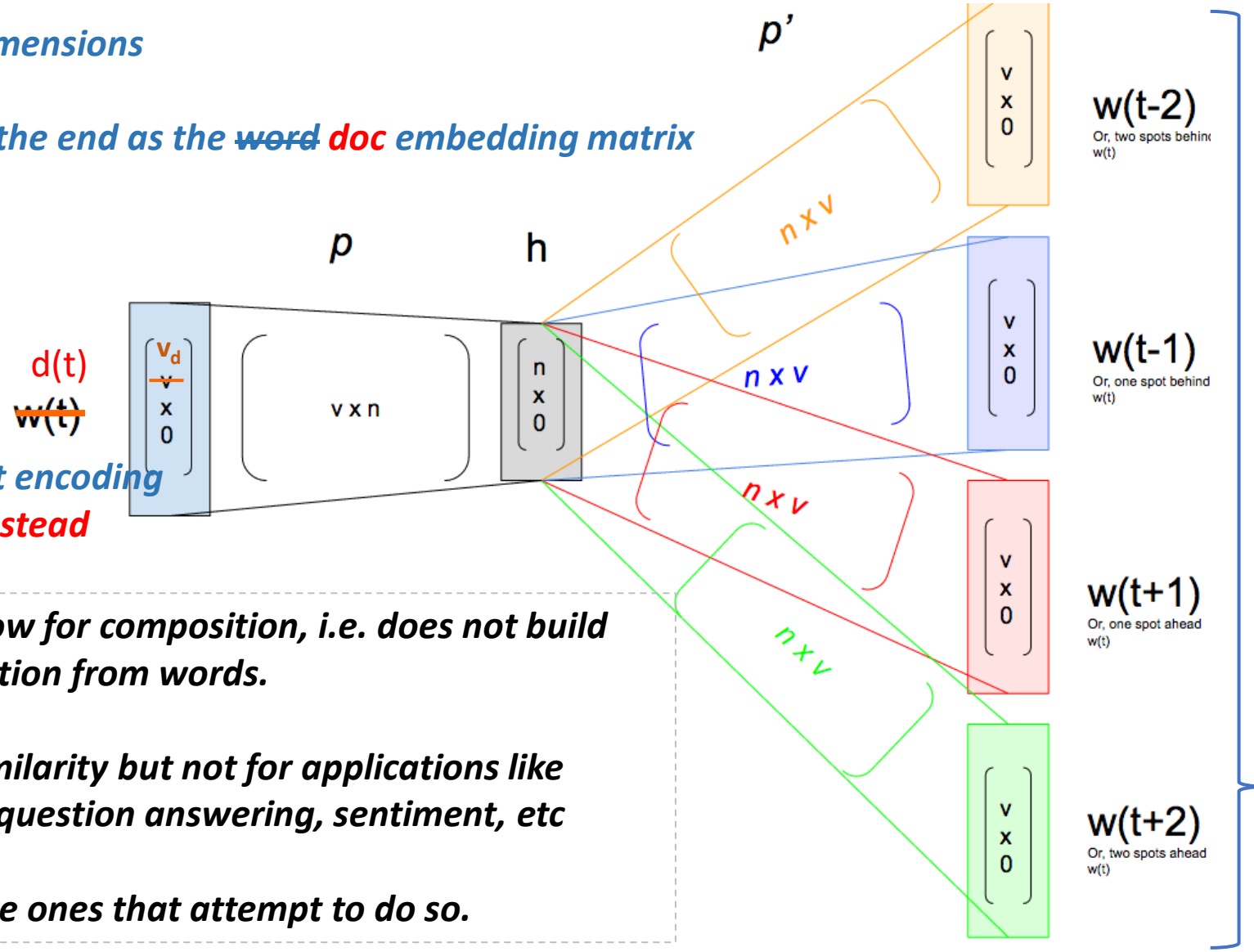
What happens for documents that were not in the training?

Add new column to d(t) and new parameter row to 'p', freeze all old parameters and train for few iterations.

Doc2vec/paragraph2vec

p': Matrix to increase # dimensions
This is what we throw away at the end

p: Matrix to reduce # dimensions
One row per ~~word~~ doc
This is what we save at the end as the ~~word~~ doc embedding matrix



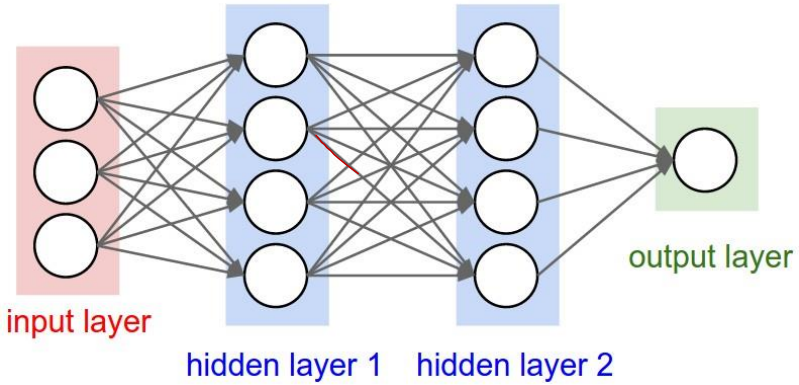
Input ~~word~~ doc one hot encoding
We should call it d(t) instead

One hot encoded ~~context~~ words occurring in the document

Approach doesn't allow for composition, i.e. does not build document representation from words.

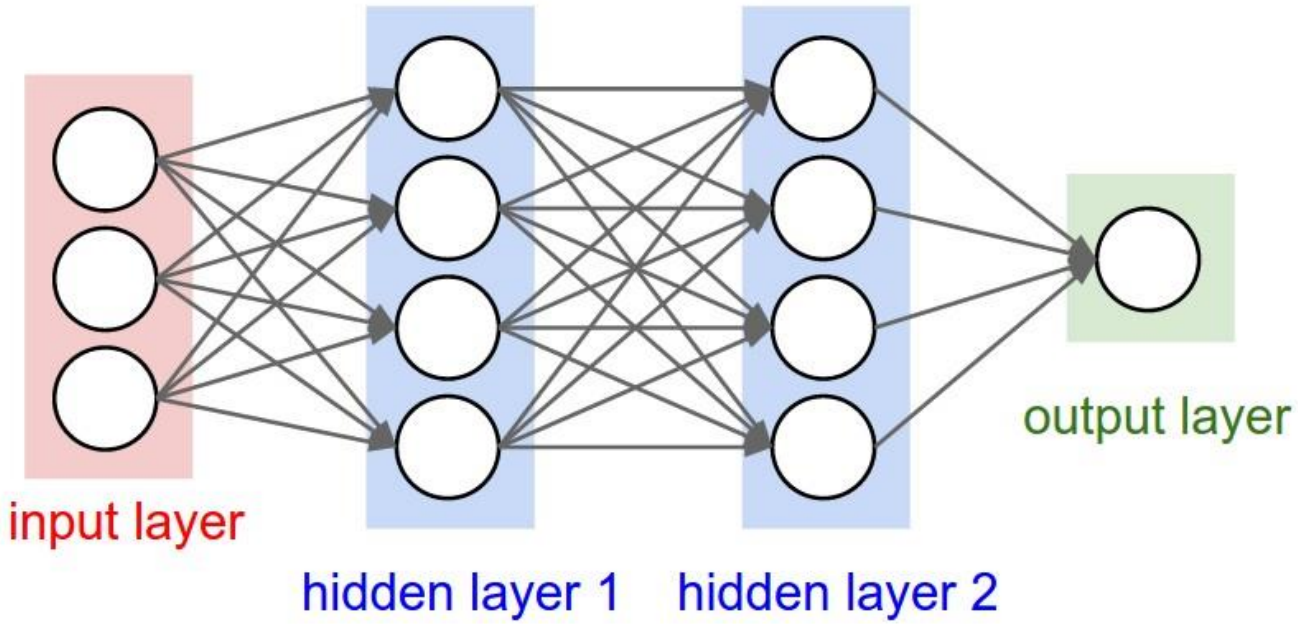
Nice for document similarity but not for applications like machine translation, question answering, sentiment, etc

Let's take a look at the ones that attempt to do so.



dense nets

dense nets



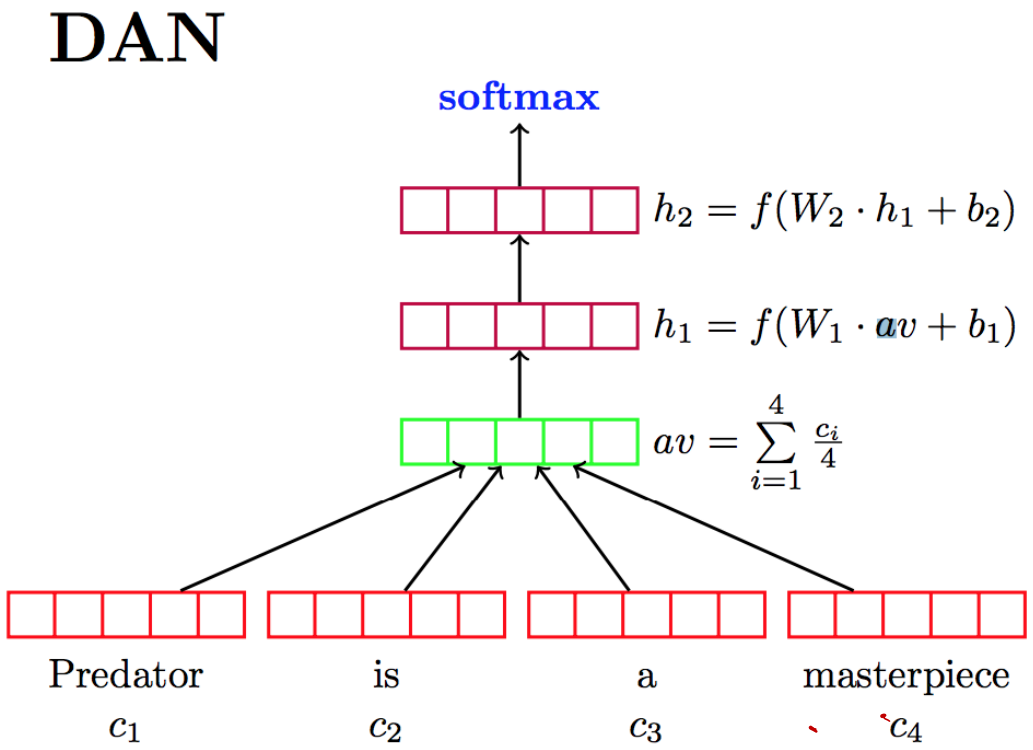
dense nets
Deep Averaging Networks (DANs)

Deep Unordered Composition Rivals Syntactic Methods for Text Classification

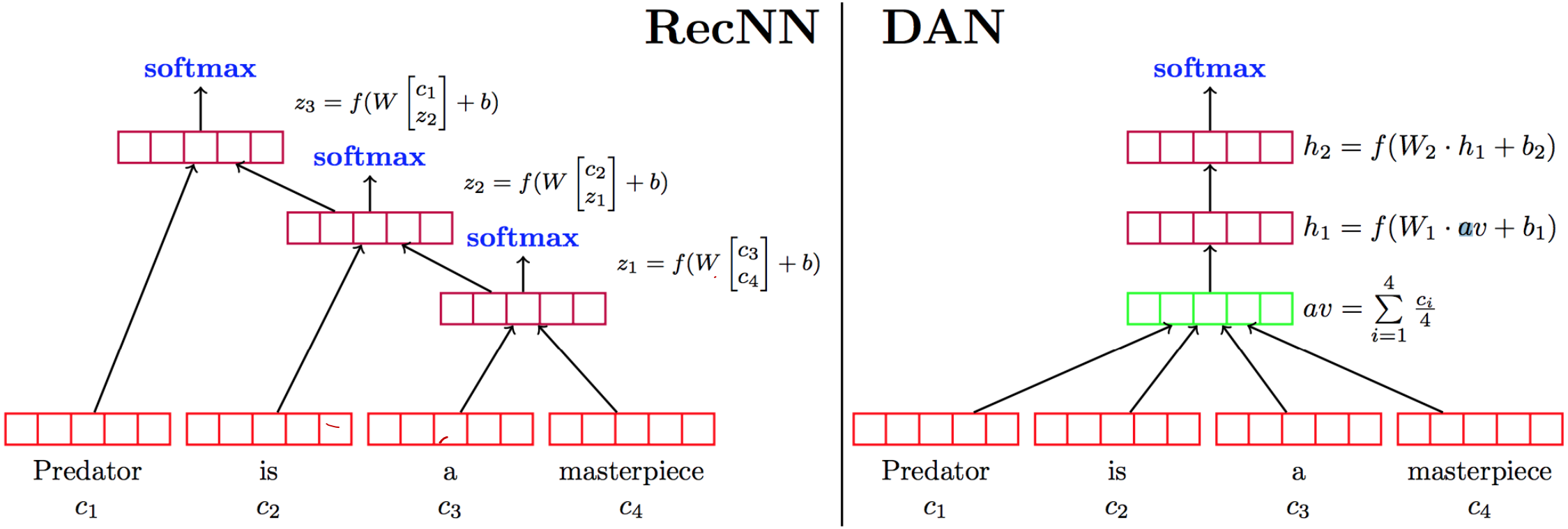
Mohit Iyyer,[1] Varun Manjunatha,[1] Jordan Boyd-Graber,[2] Hal Daumé III[1]
[1]University of Maryland, Department of Computer Science and UMIACS
[2]University of Colorado, Department of Computer Science
{miyyer,varunm,hal}@umiacs.umd.edu, Jordan.Boyd.Graber@colorado.edu

https://cs.umd.edu/~miyyer/pubs/2015_acl_dan.pdf

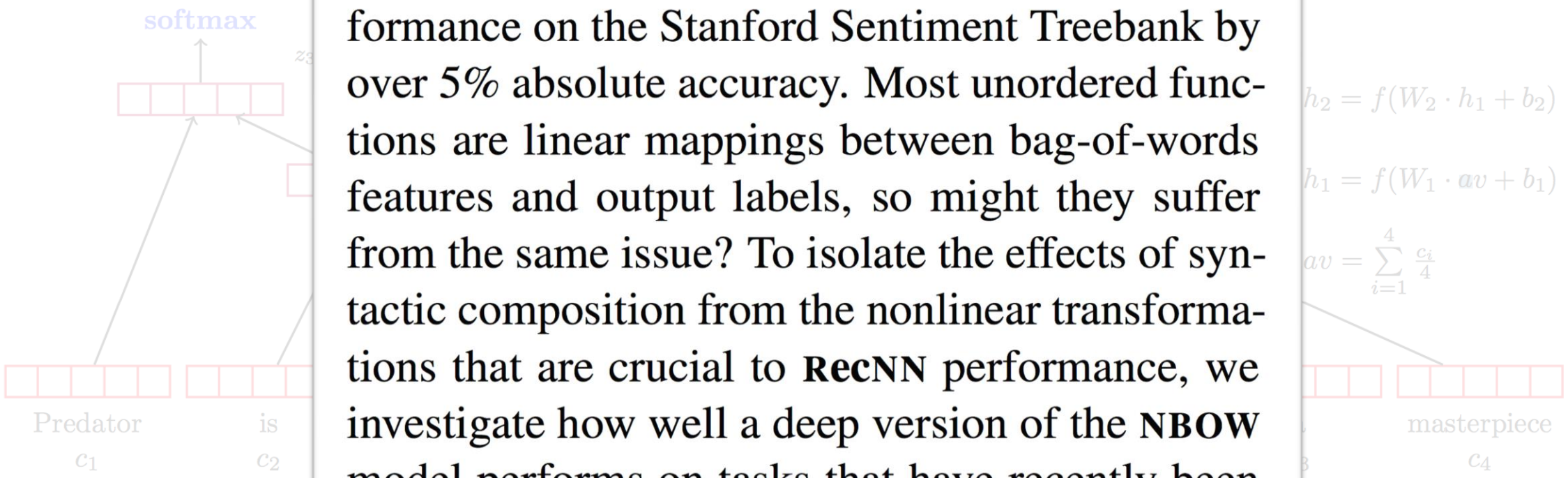
dense nets
Deep Averaging Networks (DANs)



dense nets
Deep Averaging Networks (DANs)



dense nets
Deep Averaging Networks (DANs)



ties? Socher et al. (2013b) report that removing the nonlinearities from their **RecNN** models drops performance on the Stanford Sentiment Treebank by over 5% absolute accuracy. Most unordered functions are linear mappings between bag-of-words features and output labels, so might they suffer from the same issue? To isolate the effects of syntactic composition from the nonlinear transformations that are crucial to **RecNN** performance, we investigate how well a deep version of the **NBOW** model performs on tasks that have recently been dominated by syntactically-aware models.

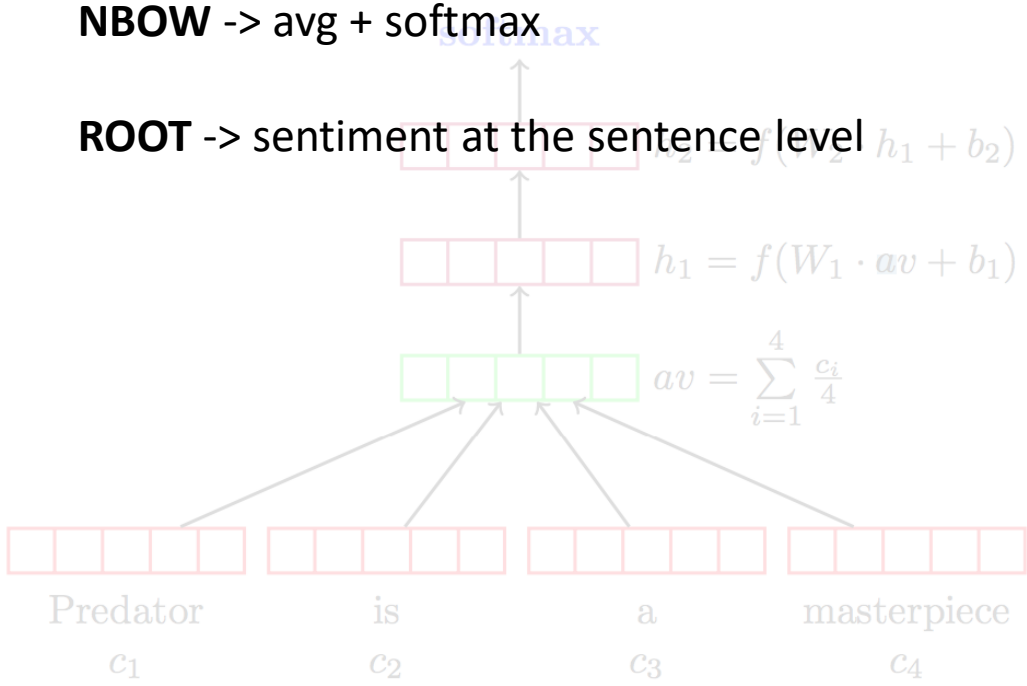
dense nets
Deep Averaging Networks (DANs)

Model	RT	SST fine	SST bin	IMDB	Time (s)
DAN-ROOT	—	46.9	85.7	—	**31**
DAN-RAND	77.3	45.4	83.2	88.8	136
DAN	80.3	47.7	86.3	89.4	136
NBOW-RAND	76.2	42.3	81.4	88.9	91
NBOW	79.0	43.6	83.6	89.0	91
BiNB	—	41.9	83.1	—	—
NBSVM-bi	79.4	—	—	91.2	—
RecNN*	77.7	43.2	82.4	—	—
RecNTN*	—	45.7	85.4	—	—
DRecNN	—	49.8	86.6	—	431
TreeLSTM	—	**50.6**	86.9	—	—
DCNN*	—	48.5	86.9	89.4	—
PVEC*	—	48.7	87.8	**92.6**	—
CNN-MC	**81.1**	47.4	**88.1**	—	2,452
WRRBM*	—	—	—	89.2	—

RAND -> randomly initialized embeddings

NBOW -> avg + softmax

ROOT -> sentiment at the sentence level



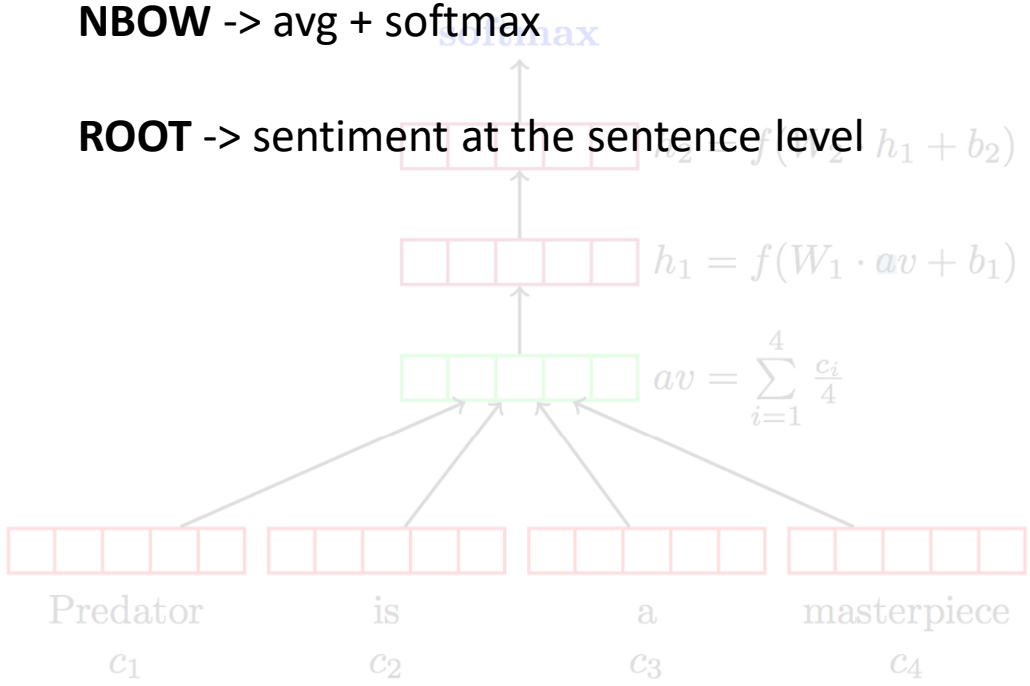
dense nets
Deep Averaging Networks (DANs)

Model	RT	SST fine	SST bin	IMDB	Time (s)
DAN-ROOT	—	46.9	85.7	—	**31**
DAN-RAND	77.3	45.4	83.2	88.8	136
DAN	80.3	47.7	86.3	89.4	136
NBOW-RAND	76.2	42.3	81.4	88.9	91
NBOW	79.0	43.6	83.6	89.0	91
BiNB	—	41.9	83.1	—	—
NBSVM-bi	79.4	—	—	91.2	—
RecNN*	77.7	43.2	82.4	—	—
RecNTN*	—	45.7	85.4	—	—
DRecNN	—	49.8	86.6	—	431
TreeLSTM	—	**50.6**	86.9	—	—
DCNN*	—	48.5	86.9	89.4	—
PVEC*	—	48.7	87.8	**92.6**	—
CNN-MC	**81.1**	47.4	**88.1**	—	2,452
WRRBM*	—	—	—	89.2	—

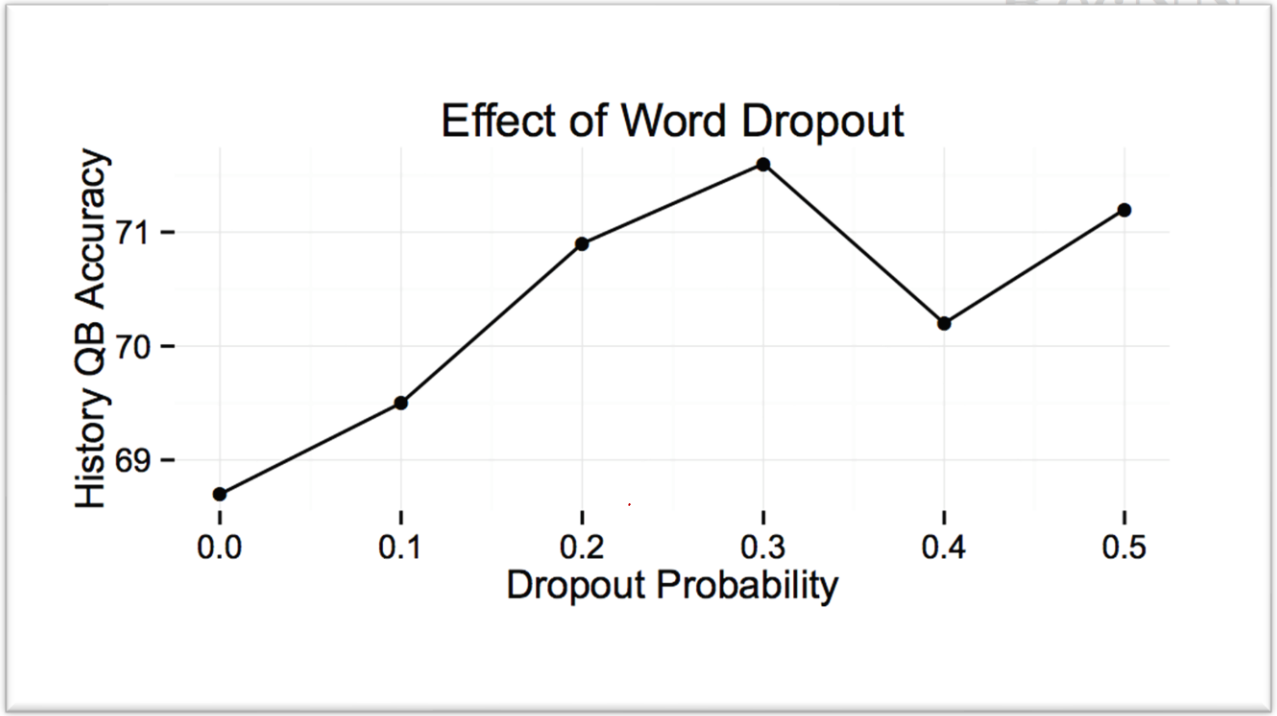
RAND -> randomly initialized embeddings

NBOW -> avg + softmax

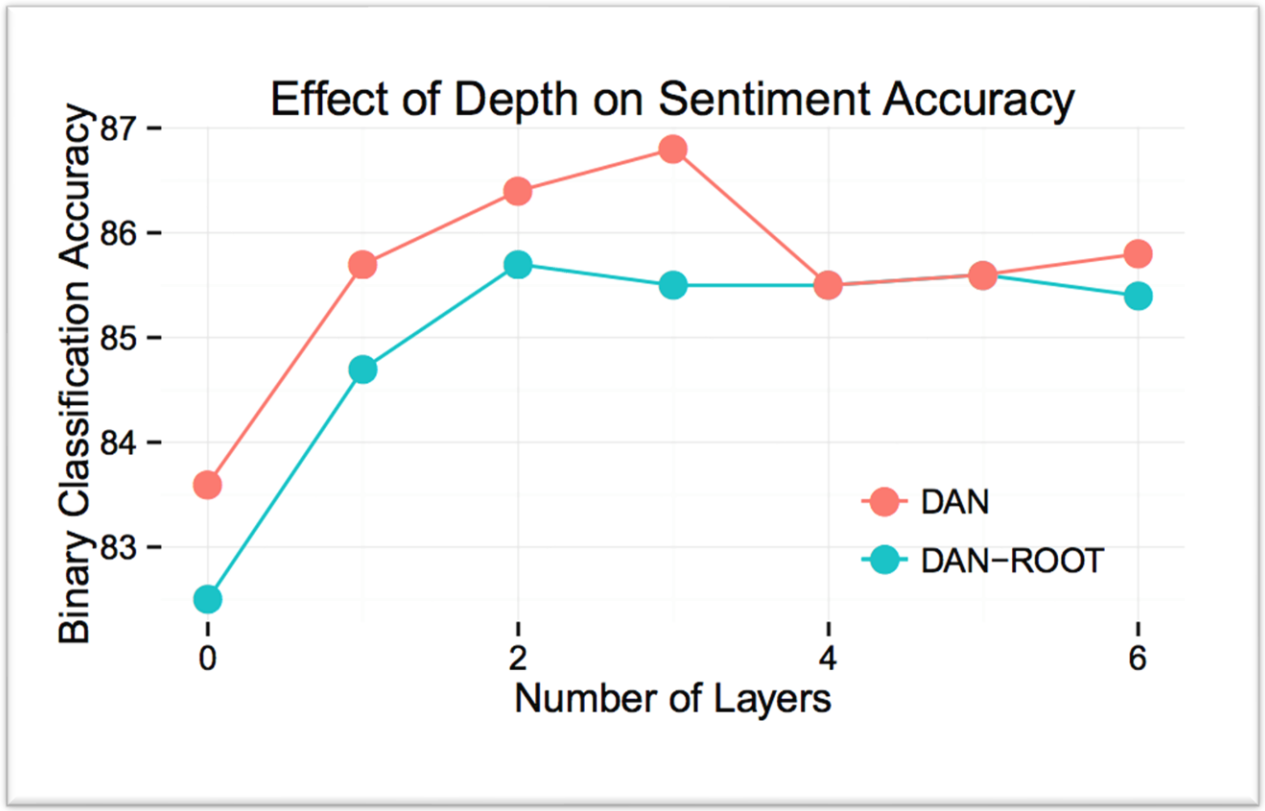
ROOT -> sentiment at the sentence level



dense nets
Deep Averaging Networks (DANs)



dense nets
Deep Averaging Networks (DANs)

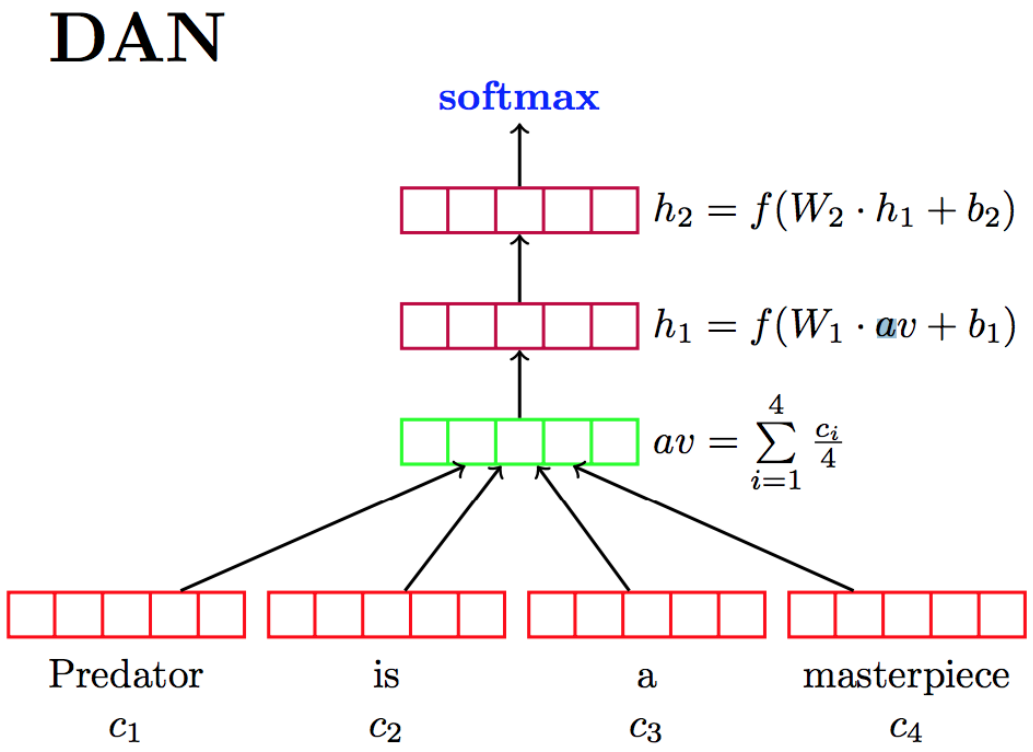


dense nets
Deep Averaging Networks (DANs)

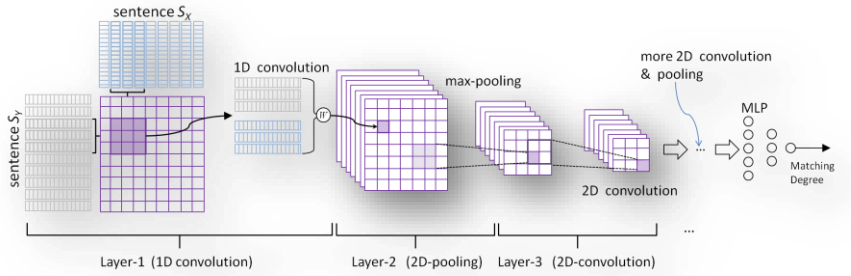
Simple

Fast

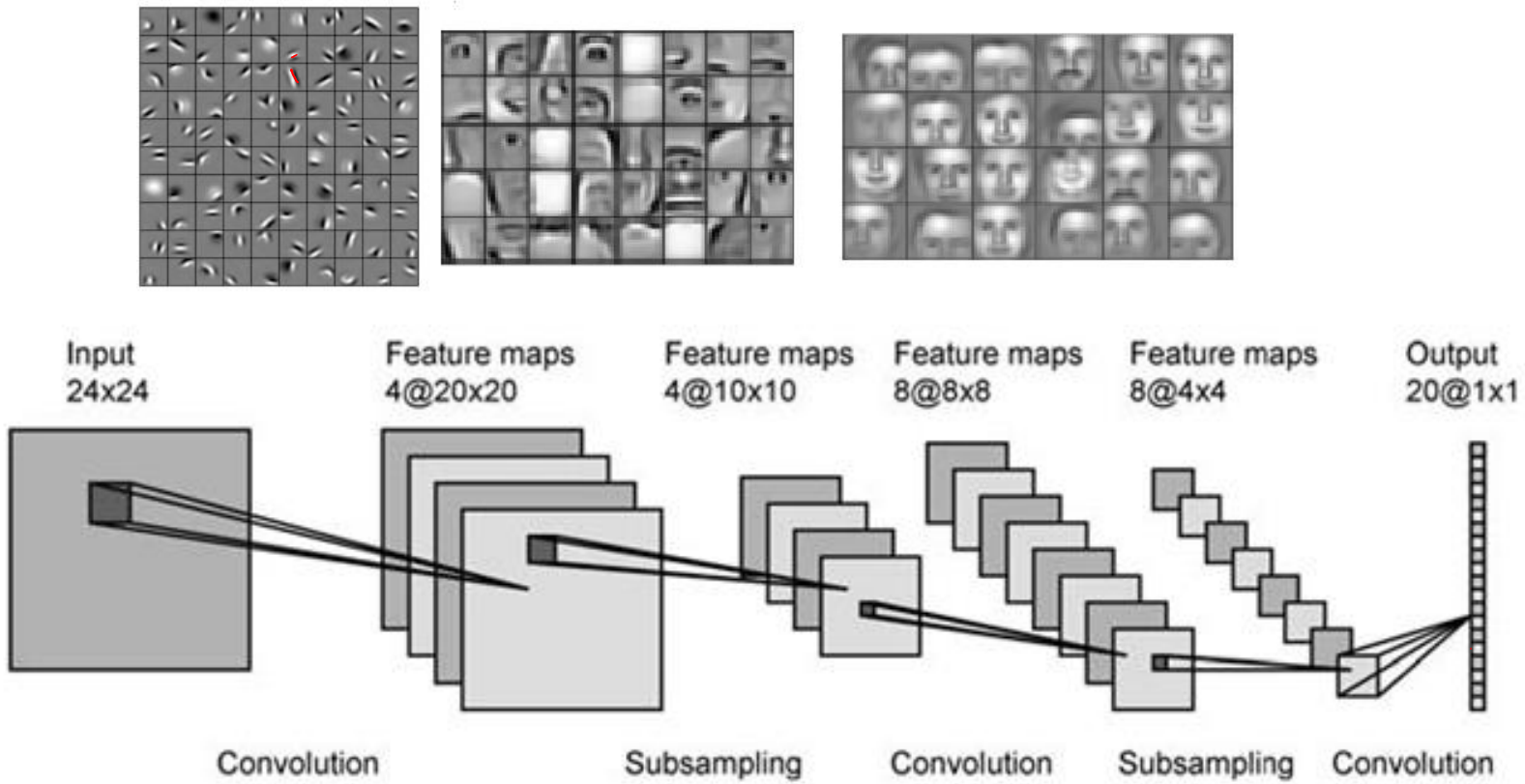
But what are we missing here?
Primarily order of words. It could beat RecNN but **not CNN**!



convolutional nets

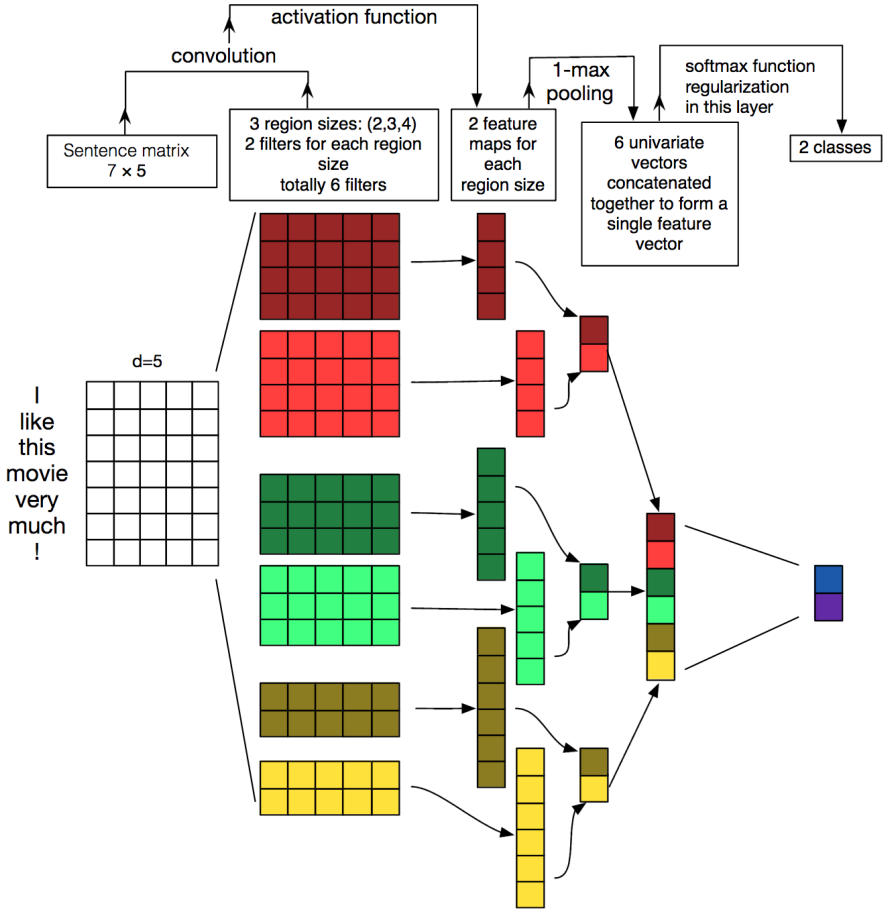


CNNs intuition



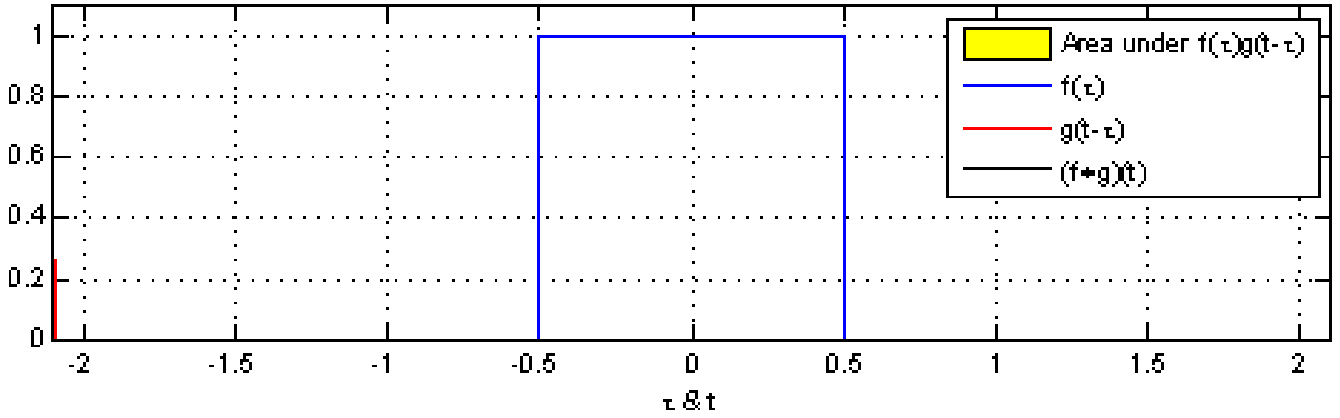
CNNs
feature maps

- *A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification*. Zhang Y. and Wallace B.



https://arxiv.org/pdf/1510.03820v2.pdf

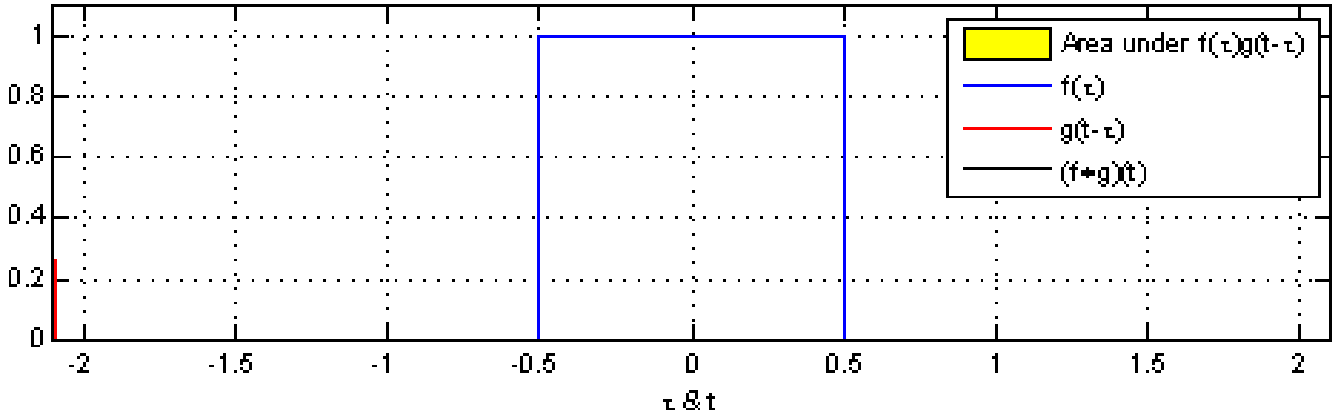
CNNs
the convolution operation



Wikipedia

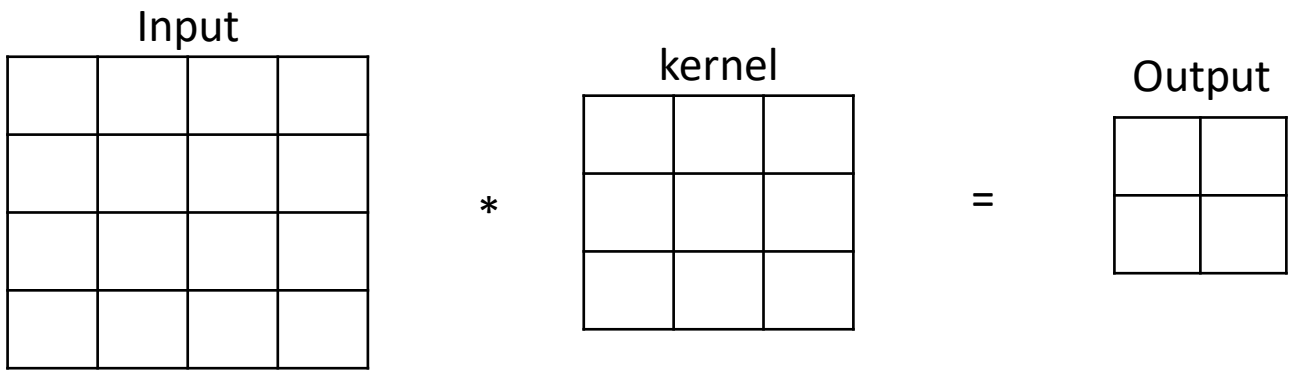
CNNs
the convolution operation

1 D

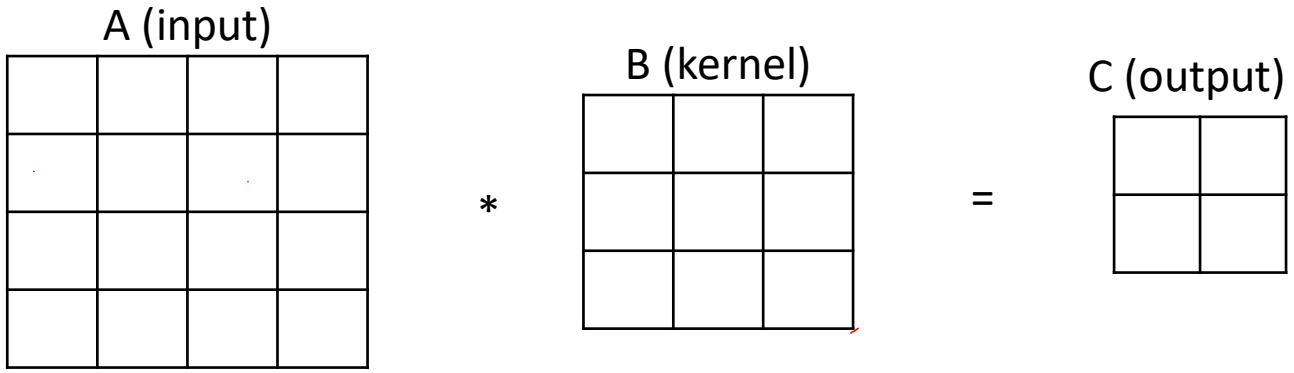


Wikipedia

2 D

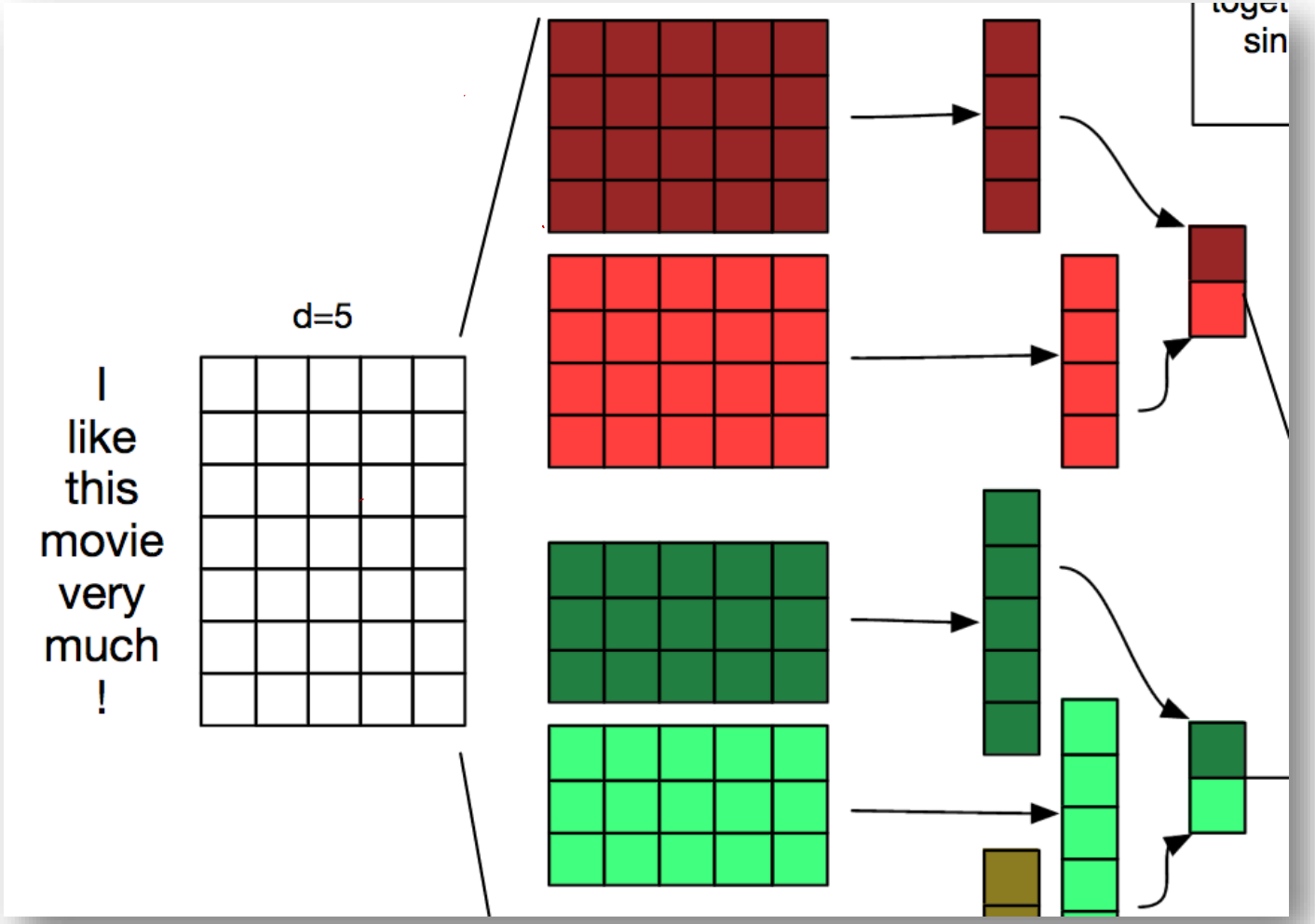


CNNs
the convolution operation

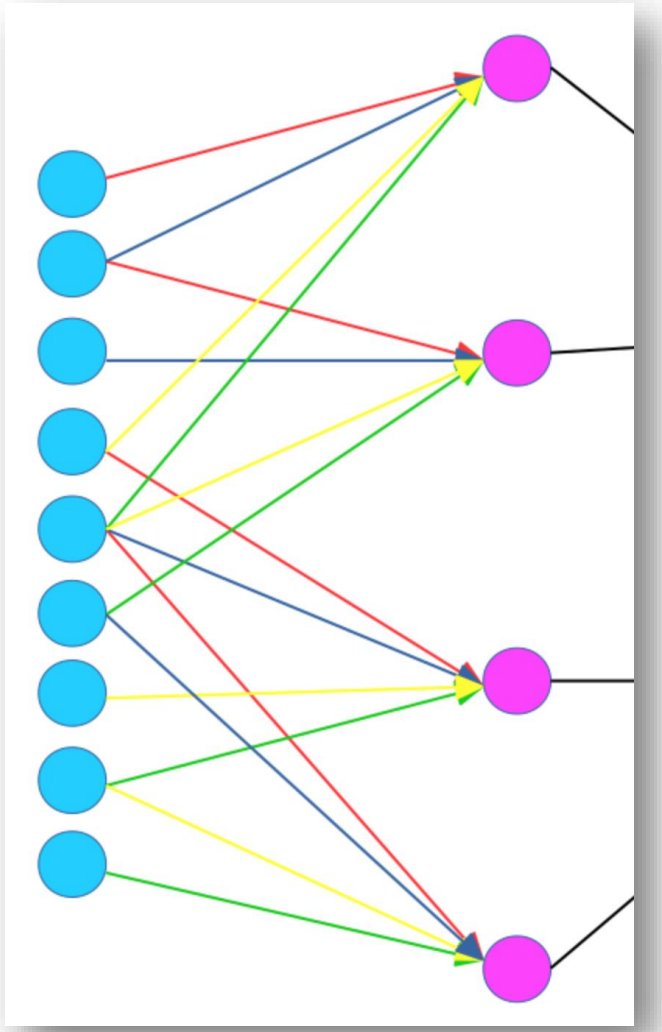


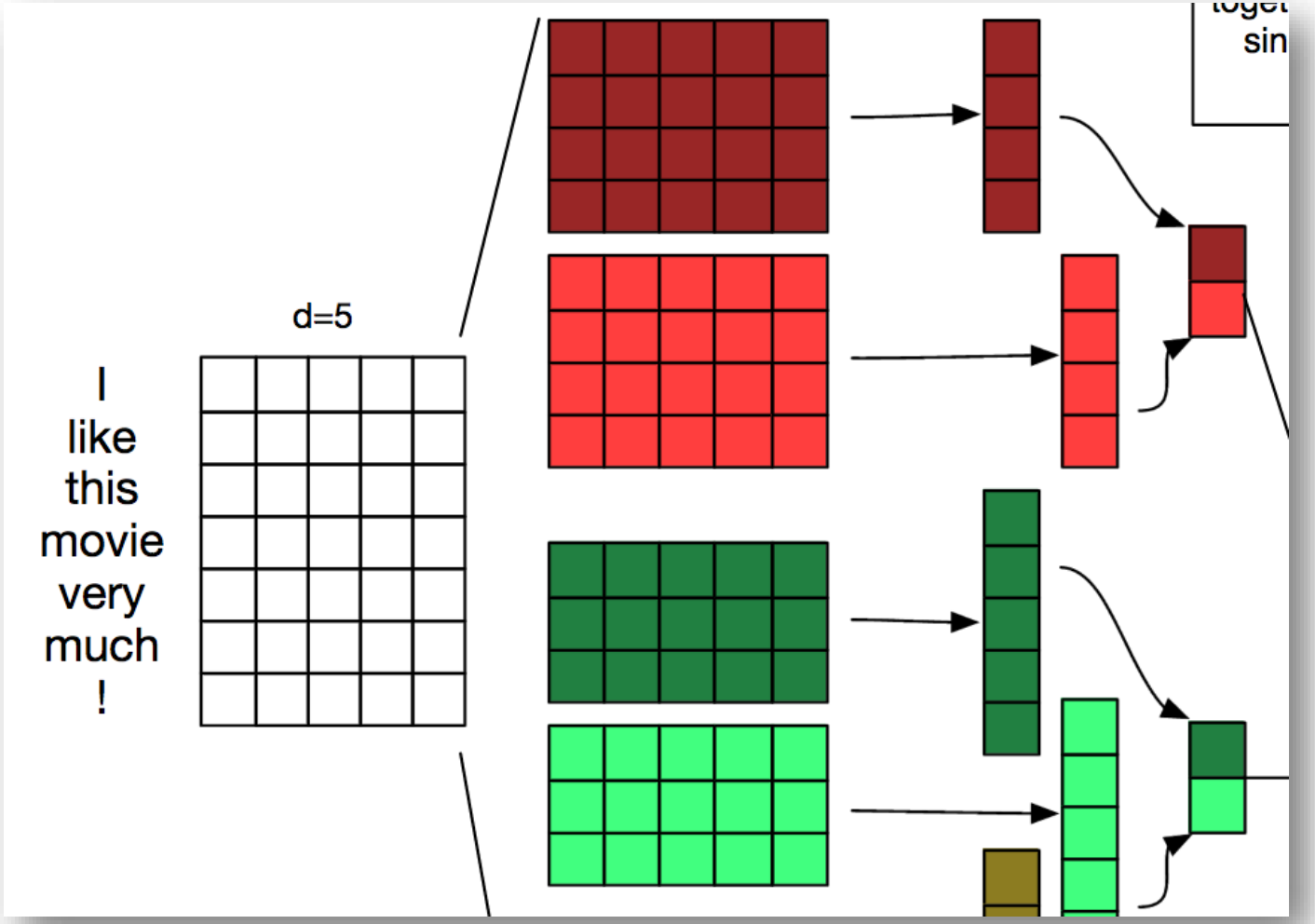
$$C[m,n] = (A * B)[m,n] = \sum_j \sum_k B[j,k].A[m+j,n+k]$$

CNNs



- Shared parameters
- Sparse neurons

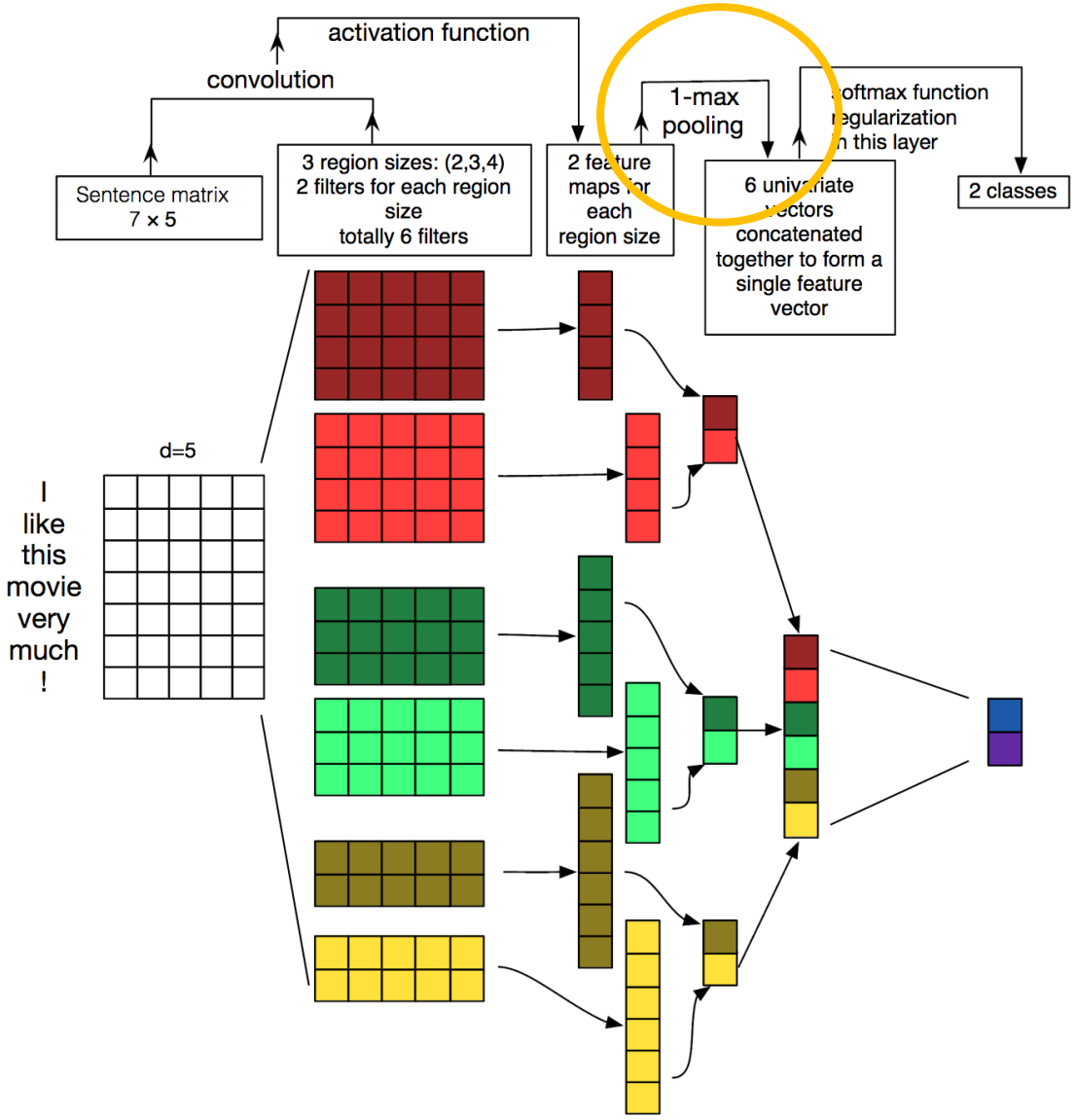




CNNs

Retain only the max values.

Why does this make sense?



https://arxiv.org/pdf/1510.03820v2.pdf

Convolutional neural networks

Dataset	bow-SVM	wv-SVM	bow+wv-SVM	gv-SVM	bow+gv-SVM
MR	78.24	78.53	79.67	78.09	80.07
SST-1	37.92	44.34	43.15	44.31	44.29
SST-2	80.54	81.97	83.30	81.92	83.52
Subj	89.13	90.94	91.74	92.61	92.22
TREC	87.95	83.61	87.33	80.28	88.71
CR	80.21	80.79	81.31	81.75	82.27
MPQA	85.38	89.27	89.70	88.97	89.27

Dataset	Non-static Word2vec-CNN	Static Word2vec-CNN
MR	81.24 (80.69, 81.56)	80.66 (80.16, 81.22)
SST-1	47.08 (46.42,48.01)	45.54 (45.03,46.27)
SST-2	85.49 (85.03, 85.90)	84.84 (84.34,85.20)
Subj	93.20 (92.97, 93.45)	92.84 (92.56,93.06)
TREC	91.54 (91.15, 91.92)	90.66 (90.02, 91.18)
CR	83.92 (82.95, 84.56)	83.57 (82.78, 84.28)
MPQA	89.32 (88.84, 89.73)	89.57 (89.18, 89.85)

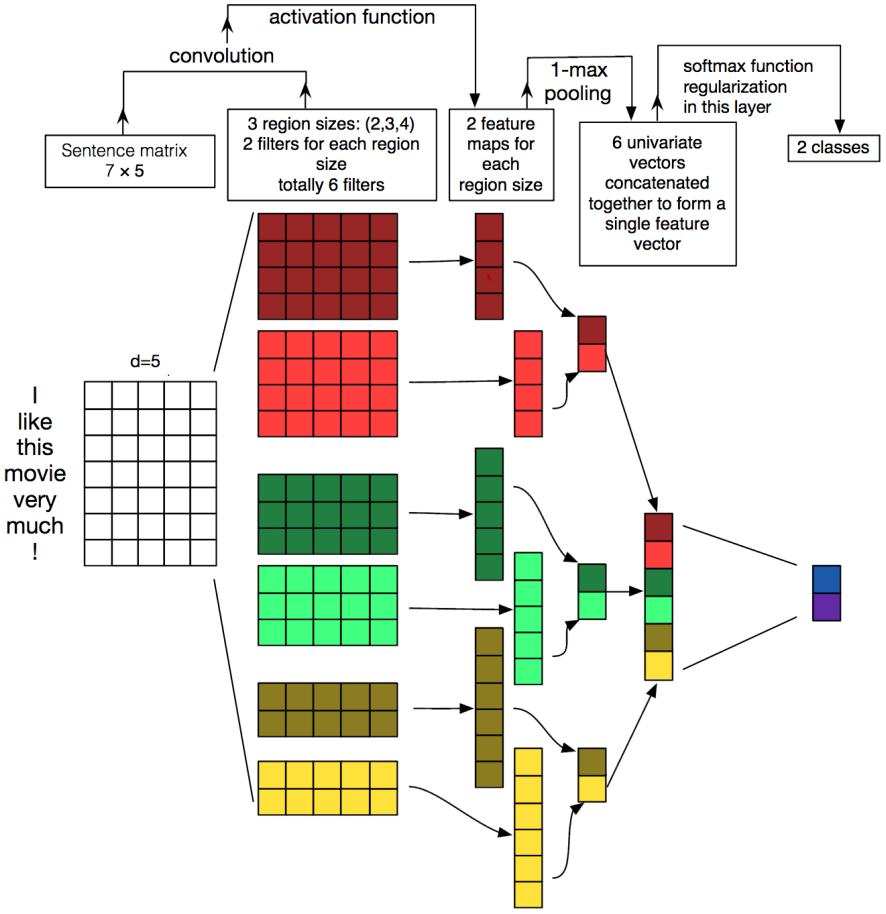
https://github.com/bwallace/CNN-for-text-classification/blob/master/CNN_text.py

CNNs

Now we have something that can take word order into consideration but we still have two problems:

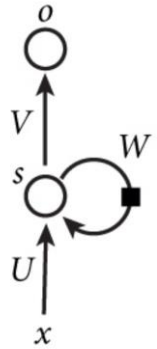
- Length of sentence
- Long distance dependencies

.. Let's talk about recurrent neural networks



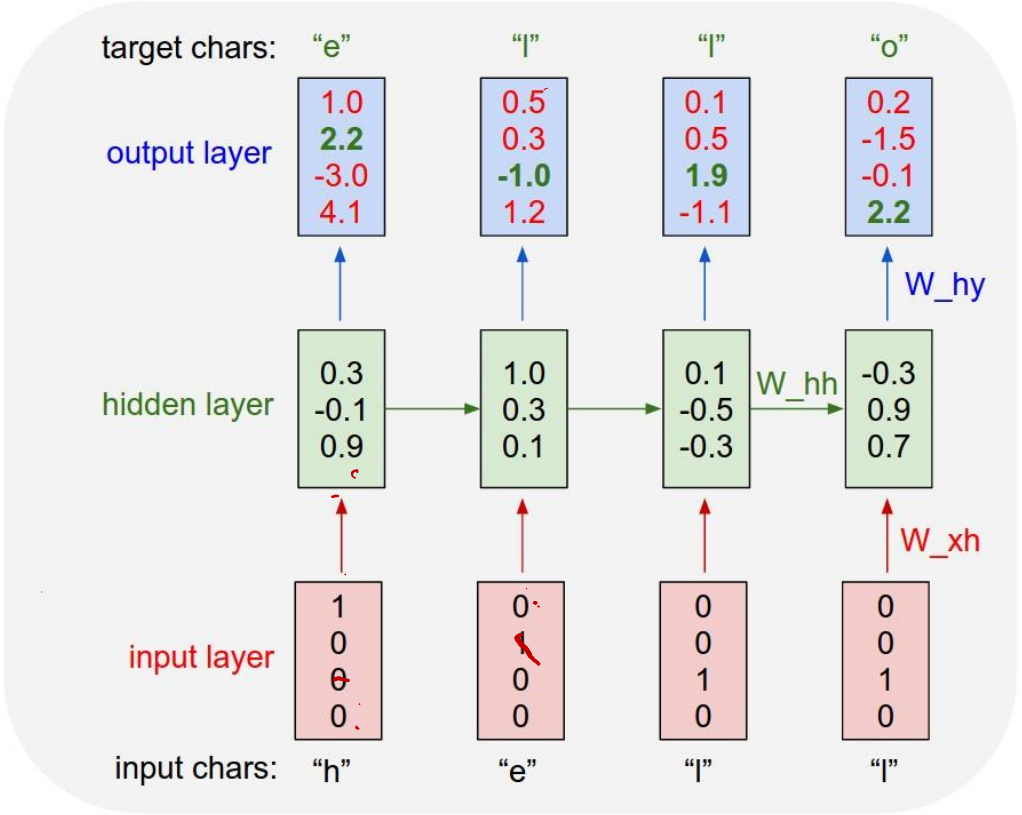
https://arxiv.org/pdf/1510.03820v2.pdf

recurrent nets



RNNs
Vanilla

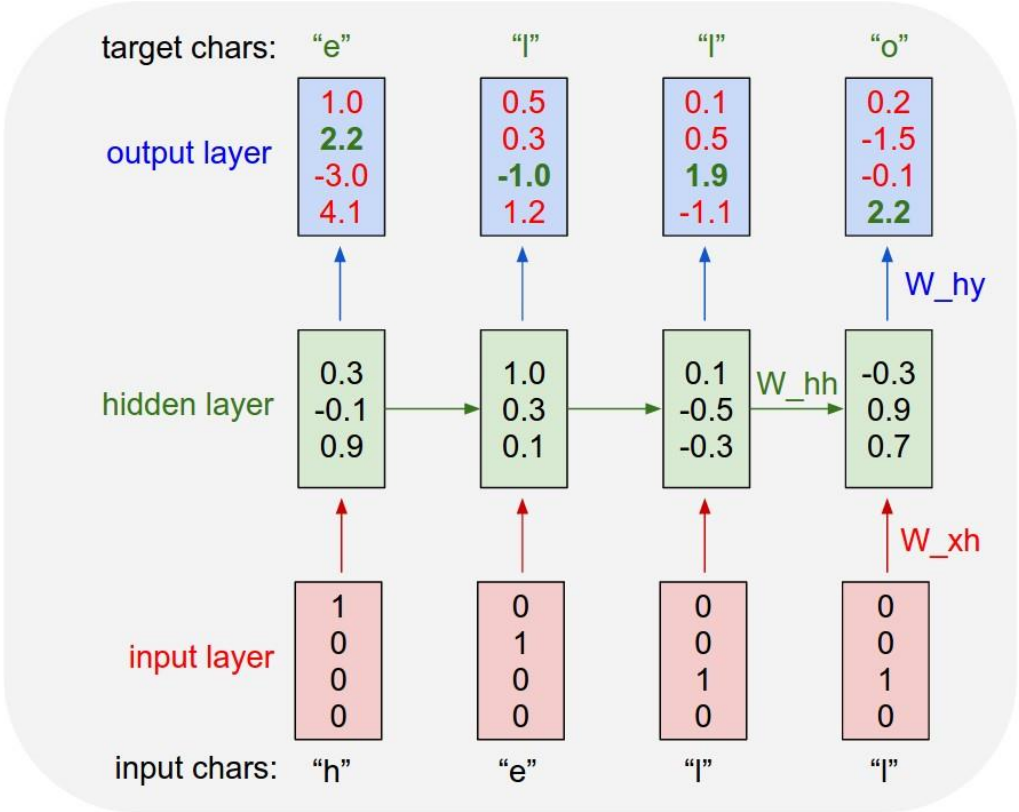
The Unreasonable Effectiveness of Recurrent Neural Networks



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNNs
Vanilla

The Unreasonable Effectiveness of Recurrent Neural Networks

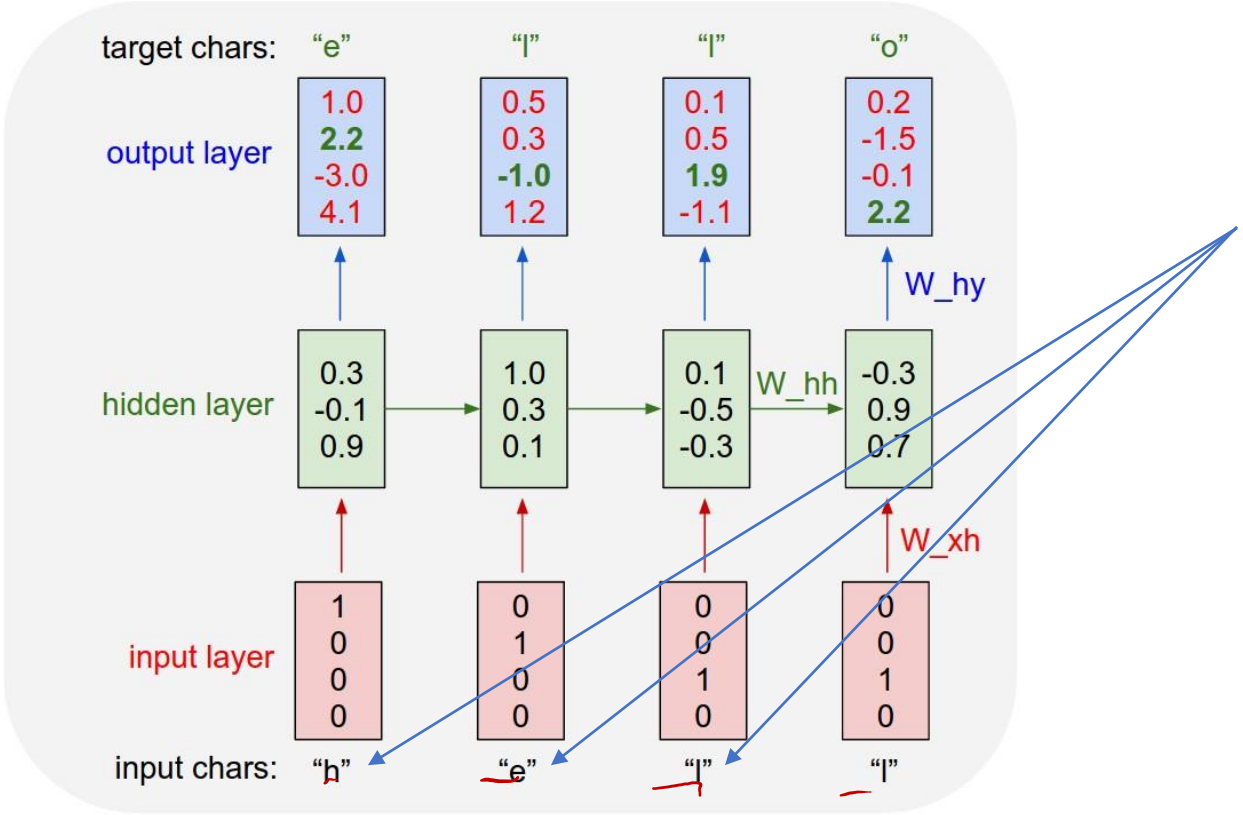


How can we do this with words instead of chars?

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNNs
Vanilla

The Unreasonable Effectiveness of Recurrent Neural Networks

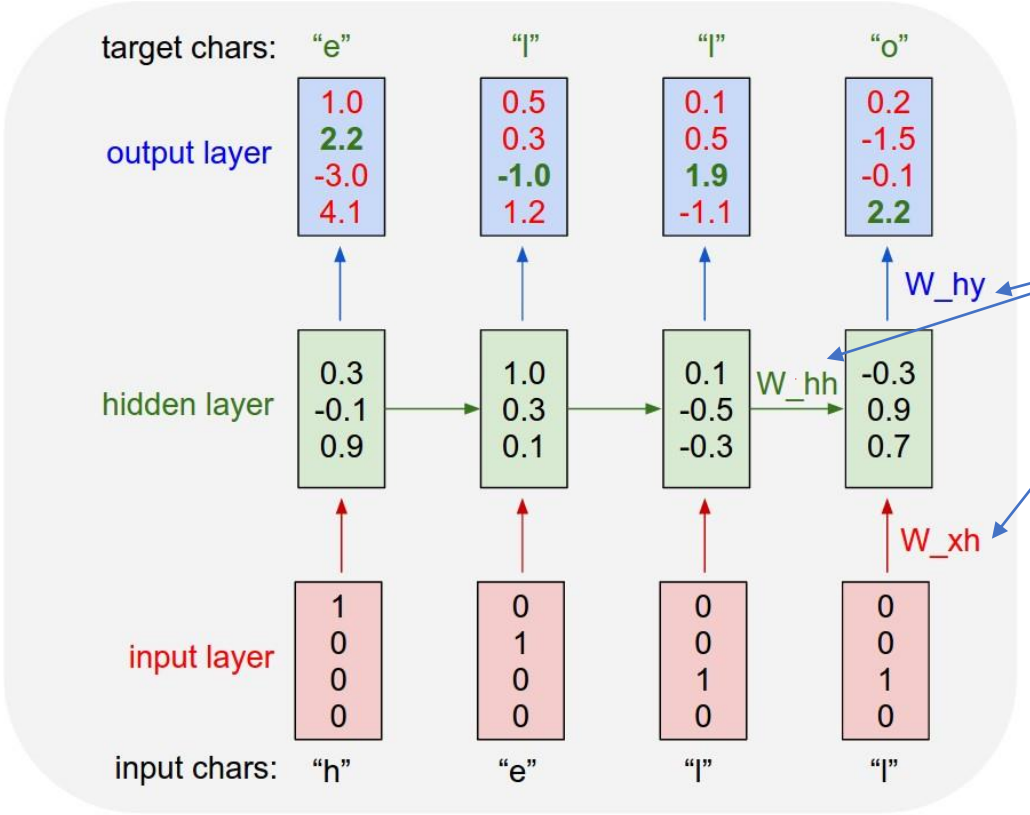


sequential

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNNs
Vanilla

The Unreasonable Effectiveness of Recurrent Neural Networks

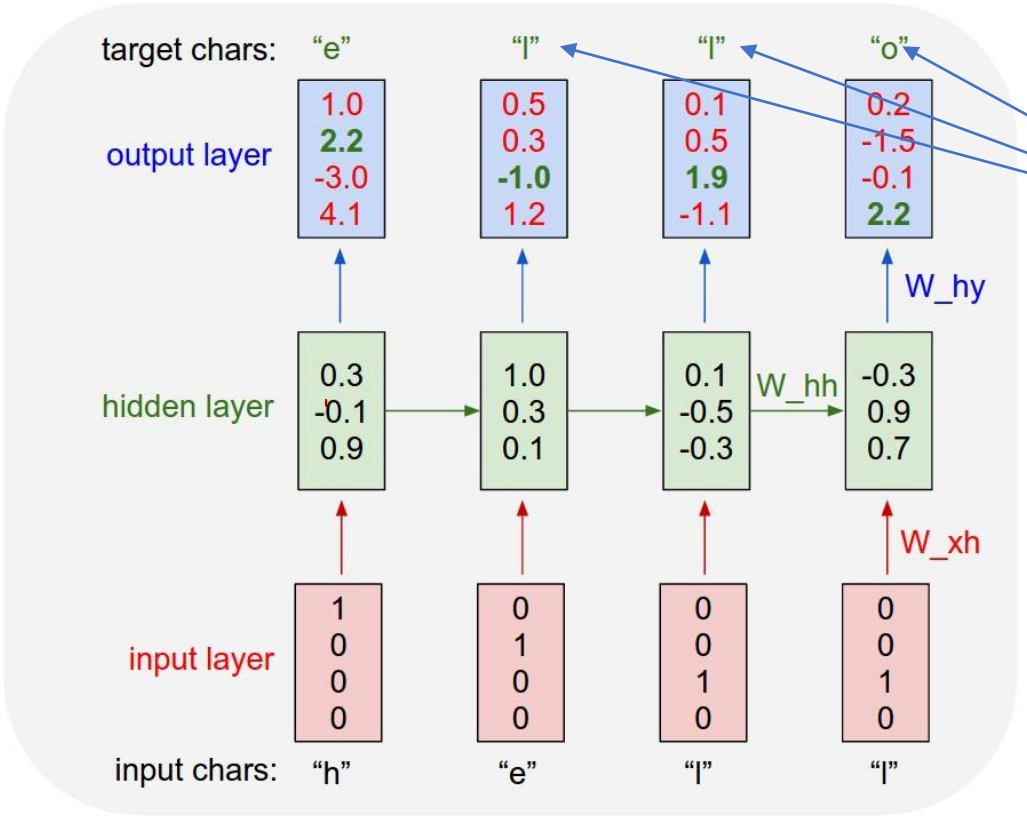


Shared params

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNNs
Vanilla

The Unreasonable Effectiveness of Recurrent Neural Networks

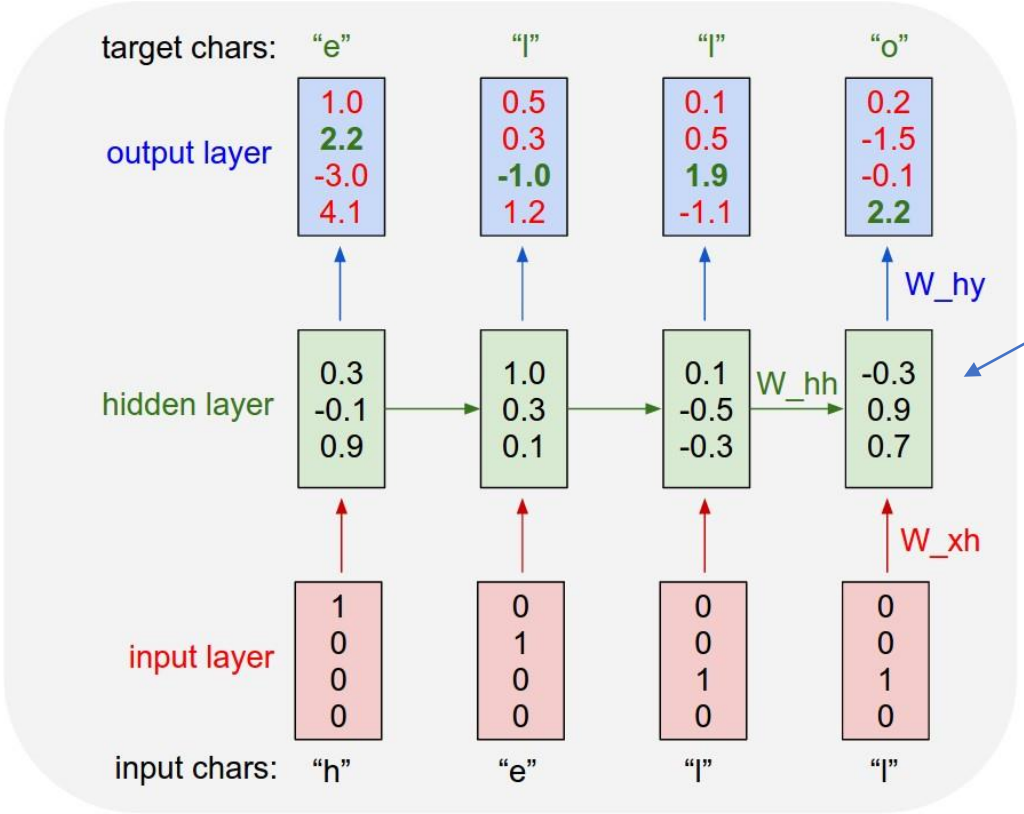


generative

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNNs
Vanilla

The Unreasonable Effectiveness of Recurrent Neural Networks

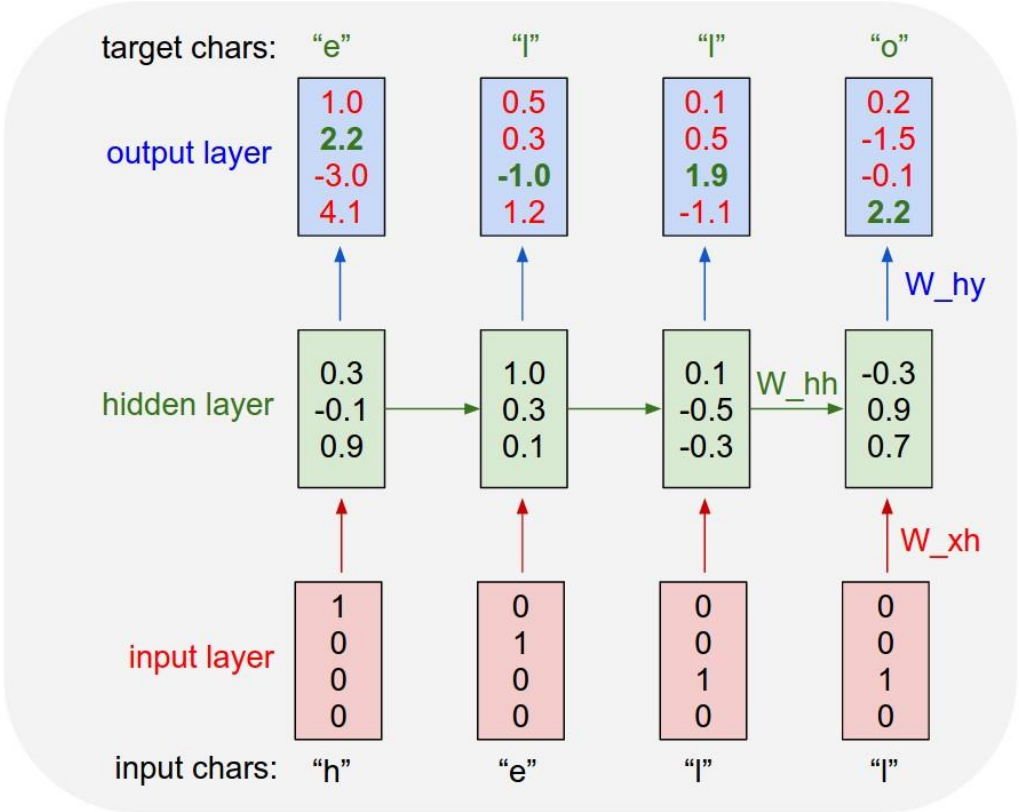


Last hidden state is sequence representation

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNNs
Vanilla

The Unreasonable Effectiveness of Recurrent Neural Networks



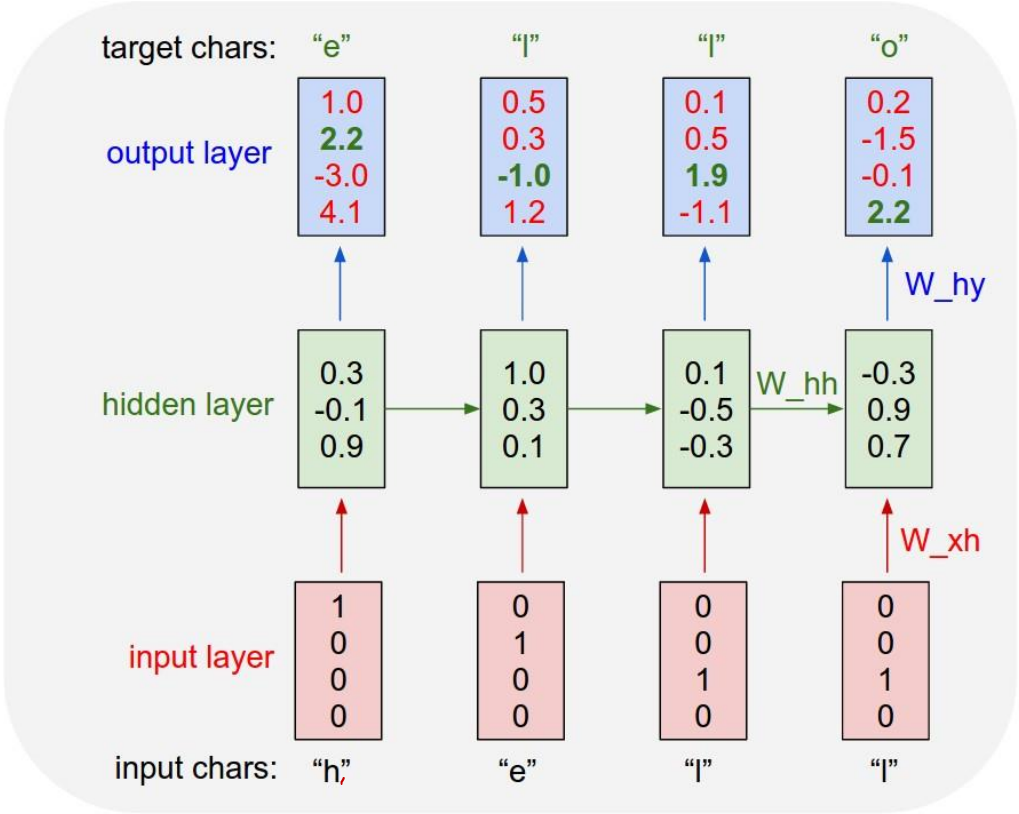
Can be used to classify either each element of the sequence or whole sequence

LMs too

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNNs
Vanilla

The Unreasonable Effectiveness of Recurrent Neural Networks



<u>Training:</u>

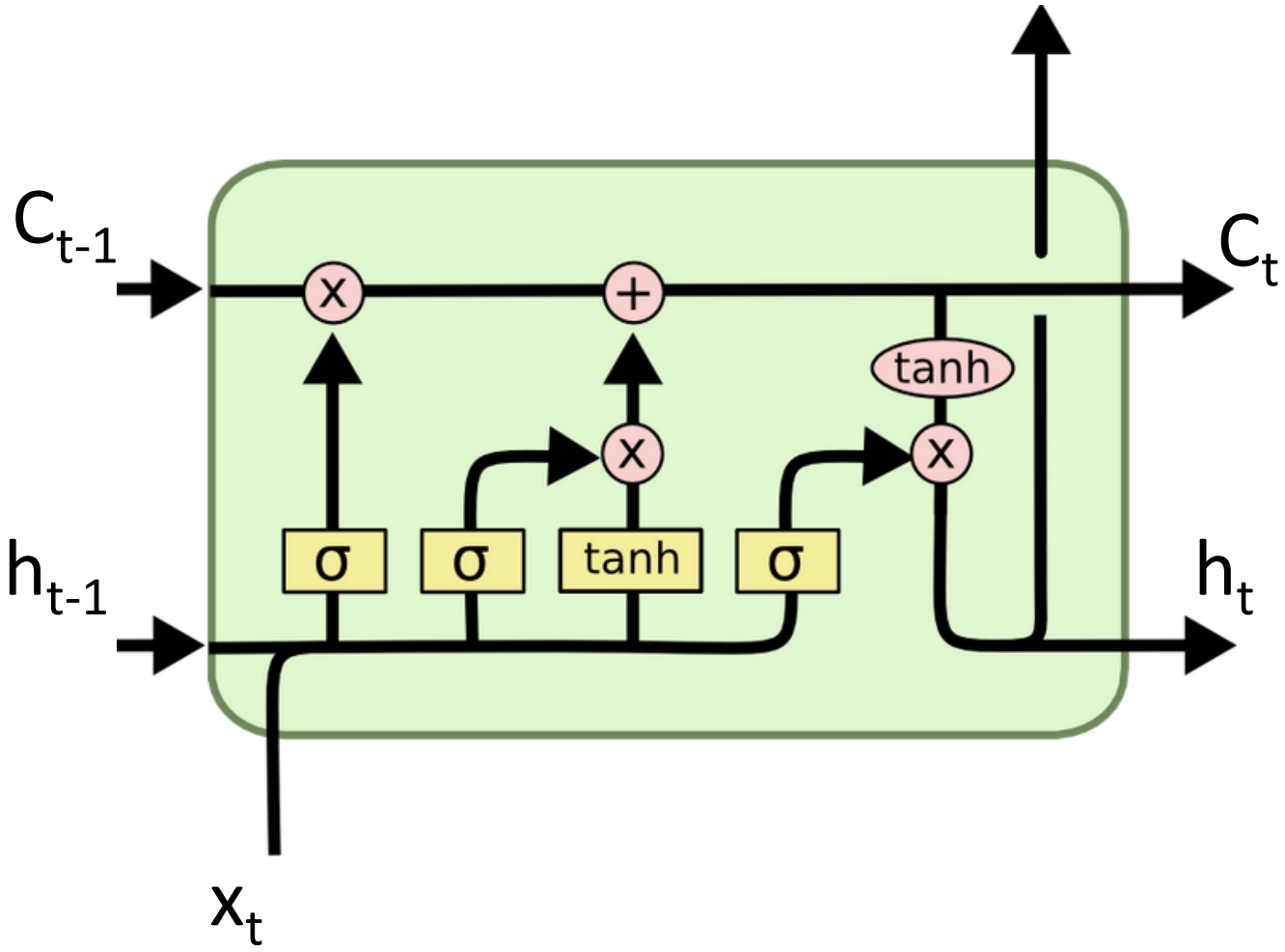
Algorithm: BTTBP

Issues: vanishing gradients

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

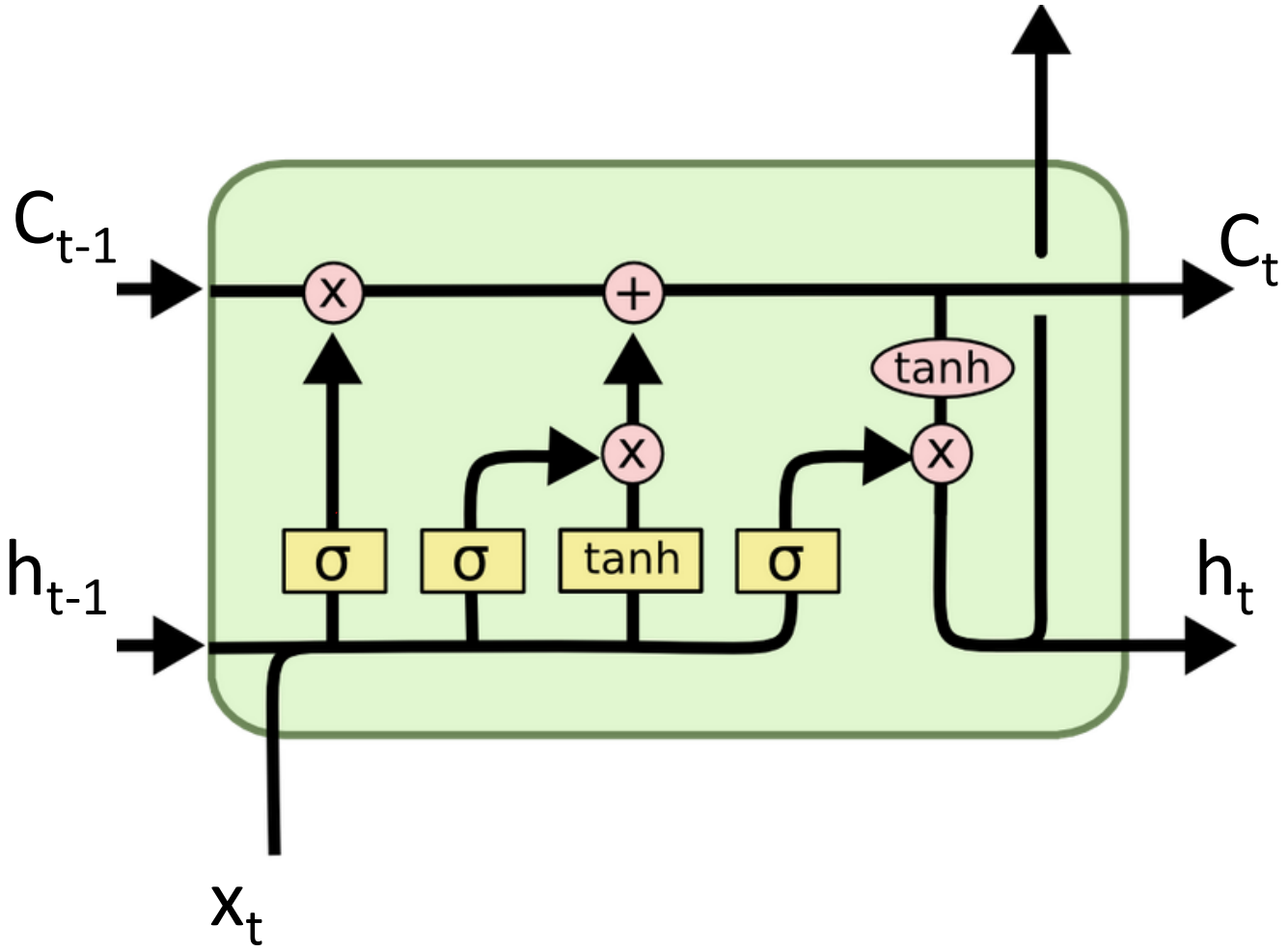
Gated RNNs: LSTMs

Gated RNNs: LSTMs



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated RNNs: LSTMs



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

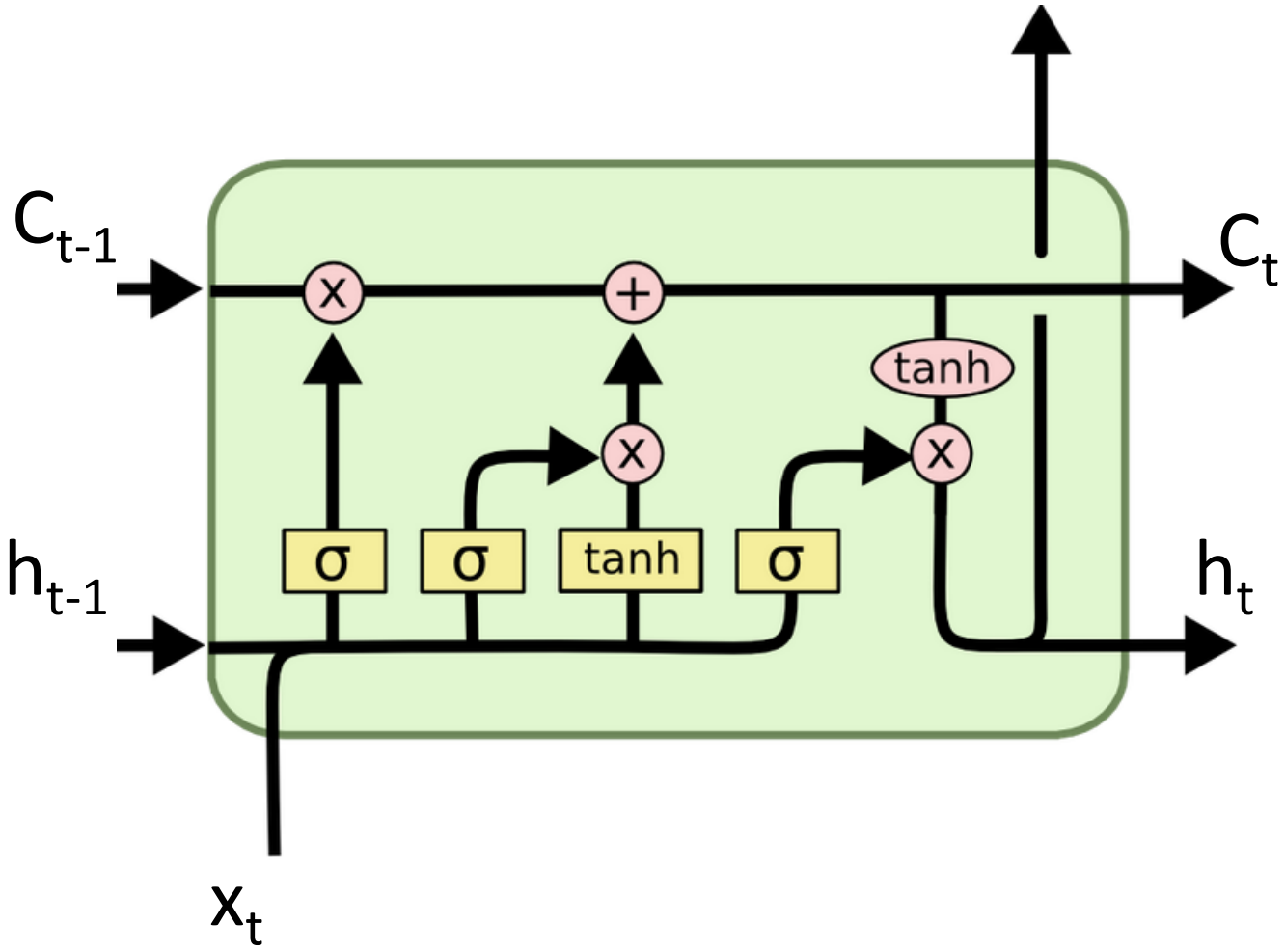
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Gated RNNs: LSTMs



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

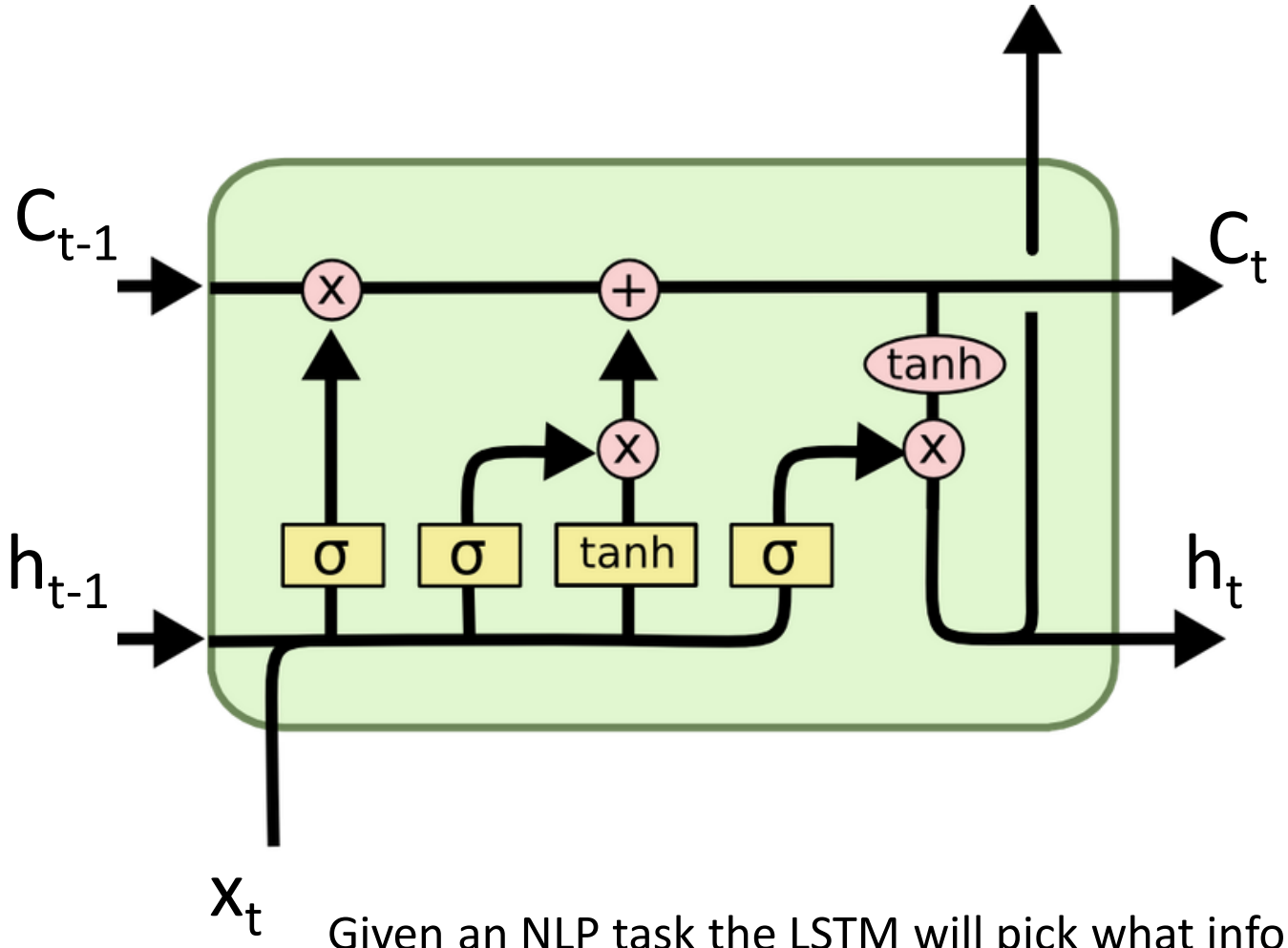
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Gated RNNs: LSTMs



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

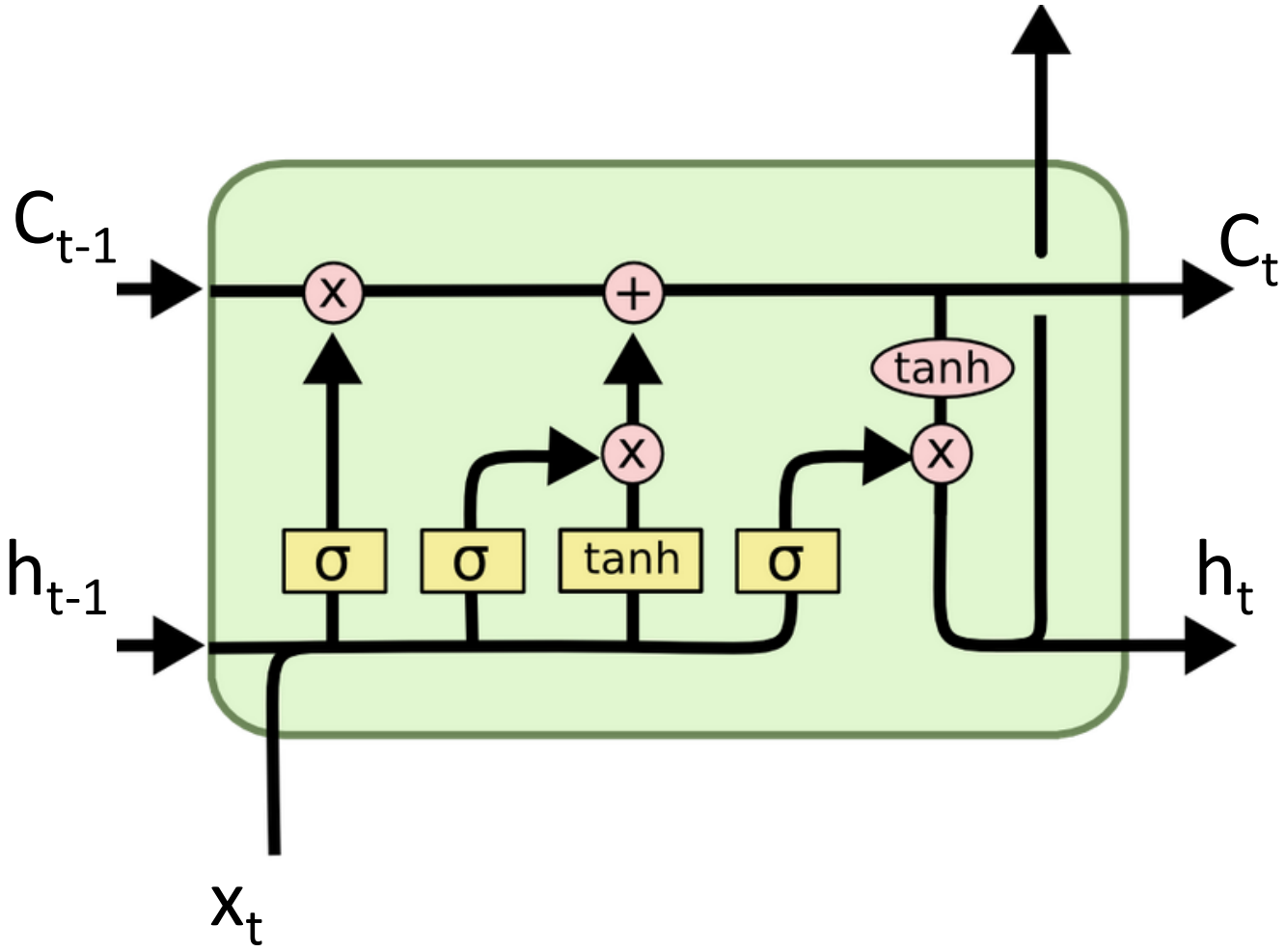
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Given an NLP task the LSTM will pick what information to remember and what information is irrelevant to build a sentence level rep: **h(n)** where ‘n’ is the length of the sentence

Gated RNNs: LSTMs



Works much better ..

What problems does it solve?:
- Vanishing gradient
- Longer memory

Gated RNNs: LSTMs

Assessing State-of-the-Art Sentiment Models on State-of-the-Art Sentiment Datasets

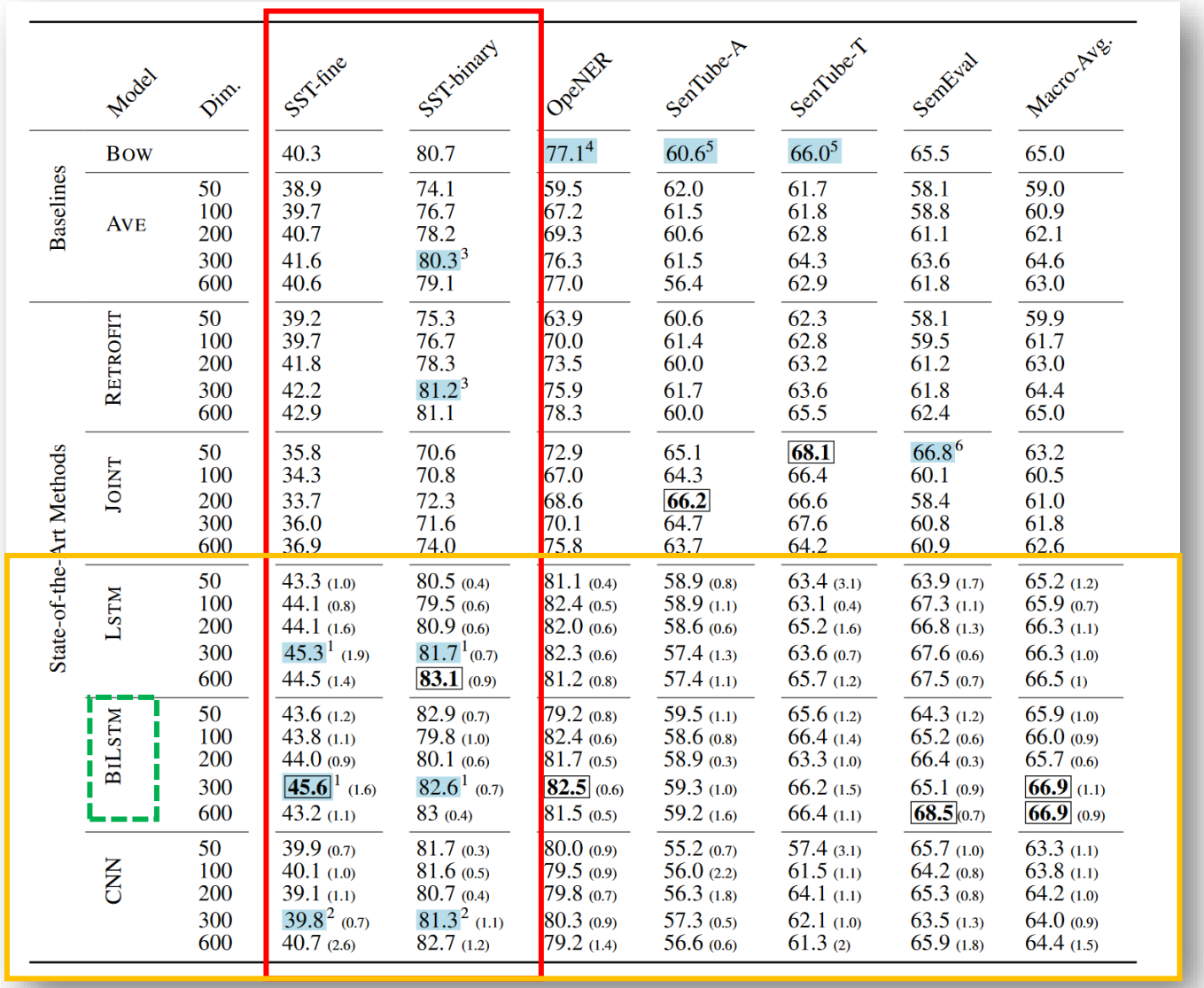
Jeremy Barnes, Roman Klinger, and **Sabine Schulte im Walde**
Institut für Maschinelle Sprachverarbeitung
University of Stuttgart
Pfaffenwaldring 5b, 70569 Stuttgart, Germany
{barnesjy,klinger,schulte}@ims.uni-stuttgart.de

Abstract

There has been a good amount of progress in sentiment analysis over the past 10 years, including the proposal of new methods and the creation of benchmark datasets. In

Internet (Forsyth and Martell, 2007). The first approaches in this field of research depended on the use of words at a symbolic level (unigrams, bigrams, bag-of-words features), where generalizing to new words was difficult (Pang et al., 2002; Riloff

Gated RNNs: LSTMs



	Model	Dim.	SST-fine	SST-binary	OpeNER	SenTube-A	SenTube-T	SemEval	Macro-Avg.
Baselines	BOW		40.3	80.7	77.1[4]	60.6[5]	66.0[5]	65.5	65.0
	AVE	50	38.9	74.1	59.5	62.0	61.7	58.1	59.0
		100	39.7	76.7	67.2	61.5	61.8	58.8	60.9
		200	40.7	78.2	69.3	60.6	62.8	61.1	62.1
		300	41.6	80.3[3]	76.3	61.5	64.3	63.6	64.6
		600	40.6	79.1	77.0	56.4	62.9	61.8	63.0
State-of-the-Art Methods	RETROFIT	50	39.2	75.3	63.9	60.6	62.3	58.1	59.9
		100	39.7	76.7	70.0	61.4	62.8	59.5	61.7
		200	41.8	78.3	73.5	60.0	63.2	61.2	63.0
		300	42.2	81.2[3]	75.9	61.7	63.6	61.8	64.4
		600	42.9	81.1	78.3	60.0	65.5	62.4	65.0
	JOINT	50	35.8	70.6	72.9	65.1	**68.1**	66.8[6]	63.2
		100	34.3	70.8	67.0	64.3	66.4	60.1	60.5
		200	33.7	72.3	68.6	**66.2**	66.6	58.4	61.0
		300	36.0	71.6	70.1	64.7	67.6	60.8	61.8
		600	36.9	74.0	75.8	63.7	64.2	60.9	62.6
	LSTM	50	43.3 (1.0)	80.5 (0.4)	81.1 (0.4)	58.9 (0.8)	63.4 (3.1)	63.9 (1.7)	65.2 (1.2)
		100	44.1 (0.8)	79.5 (0.6)	82.4 (0.5)	58.9 (1.1)	63.1 (0.4)	67.3 (1.1)	65.9 (0.7)
		200	44.1 (1.6)	80.9 (0.6)	82.0 (0.6)	58.6 (0.6)	65.2 (1.6)	66.8 (1.3)	66.3 (1.1)
		300	45.3[1] (1.9)	81.7[1] (0.7)	82.3 (0.6)	57.4 (1.3)	63.6 (0.7)	67.6 (0.6)	66.3 (1.0)
		600	44.5 (1.4)	**83.1** (0.9)	81.2 (0.8)	57.4 (1.1)	65.7 (1.2)	67.5 (0.7)	66.5 (1)
	BILSTM	50	43.6 (1.2)	82.9 (0.7)	79.2 (0.8)	59.5 (1.1)	65.6 (1.2)	64.3 (1.2)	65.9 (1.0)
		100	43.8 (1.1)	79.8 (1.0)	82.4 (0.6)	58.6 (0.8)	66.4 (1.4)	65.2 (0.6)	66.0 (0.9)
		200	44.0 (0.9)	80.1 (0.6)	81.7 (0.5)	58.9 (0.3)	63.3 (1.0)	66.4 (0.3)	65.7 (0.6)
		300	**45.6**[1] (1.6)	82.6[1] (0.7)	**82.5** (0.6)	59.3 (1.0)	66.2 (1.5)	65.1 (0.9)	**66.9** (1.1)
		600	43.2 (1.1)	83 (0.4)	81.5 (0.5)	59.2 (1.6)	66.4 (1.1)	**68.5** (0.7)	**66.9** (0.9)
	CNN	50	39.9 (0.7)	81.7 (0.3)	80.0 (0.9)	55.2 (0.7)	57.4 (3.1)	65.7 (1.0)	63.3 (1.1)
		100	40.1 (1.0)	81.6 (0.5)	79.5 (0.9)	56.0 (2.2)	61.5 (1.1)	64.2 (0.8)	63.8 (1.1)
		200	39.1 (1.1)	80.7 (0.4)	79.8 (0.7)	56.3 (1.8)	64.1 (1.1)	65.3 (0.8)	64.2 (1.0)
		300	39.8[2] (0.7)	81.3[2] (1.1)	80.3 (0.9)	57.3 (0.5)	62.1 (1.0)	63.5 (1.3)	64.0 (0.9)
		600	40.7 (2.6)	82.7 (1.2)	79.2 (1.4)	56.6 (0.6)	61.3 (2)	65.9 (1.8)	64.4 (1.5)

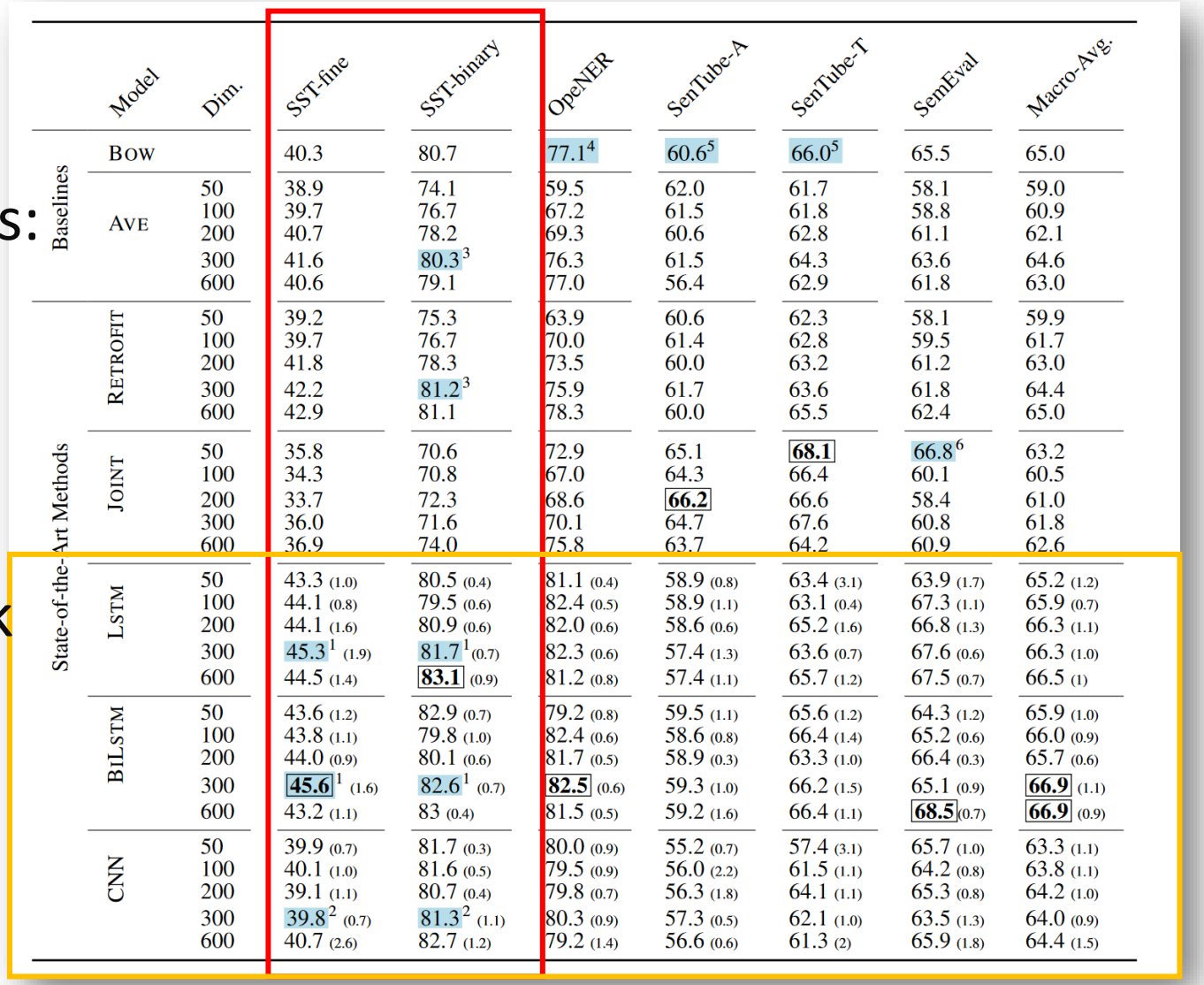
BiLSTM means bidirectional LSTM, basically two LSTMs one starting from the left and the other from the right. At each word we concatenate the content of the hidden layers

Gated RNNs: LSTMs

Let's discuss some limitations of LSTMs:

- Speed
- Accuracy of representation

Where do you think is the state of art today?



	Model	Dim.	SST-fine	SST-binary	OpeNER	SenTube-A	SenTube-T	SemEval	Macro-Avg.
Baselines	BOW		40.3	80.7	77.1[4]	60.6[5]	66.0[5]	65.5	65.0
	AVE	50	38.9	74.1	59.5	62.0	61.7	58.1	59.0
		100	39.7	76.7	67.2	61.5	61.8	58.8	60.9
		200	40.7	78.2	69.3	60.6	62.8	61.1	62.1
		300	41.6	80.3[3]	76.3	61.5	64.3	63.6	64.6
		600	40.6	79.1	77.0	56.4	62.9	61.8	63.0
State-of-the-Art Methods	RETROFIT	50	39.2	75.3	63.9	60.6	62.3	58.1	59.9
		100	39.7	76.7	70.0	61.4	62.8	59.5	61.7
		200	41.8	78.3	73.5	60.0	63.2	61.2	63.0
		300	42.2	81.2[3]	75.9	61.7	63.6	61.8	64.4
		600	42.9	81.1	78.3	60.0	65.5	62.4	65.0
	JOINT	50	35.8	70.6	72.9	65.1	**68.1**	66.8[6]	63.2
		100	34.3	70.8	67.0	64.3	66.4	60.1	60.5
		200	33.7	72.3	68.6	**66.2**	66.6	58.4	61.0
		300	36.0	71.6	70.1	64.7	67.6	60.8	61.8
		600	36.9	74.0	75.8	63.7	64.2	60.9	62.6
	LSTM	50	43.3 (1.0)	80.5 (0.4)	81.1 (0.4)	58.9 (0.8)	63.4 (3.1)	63.9 (1.7)	65.2 (1.2)
		100	44.1 (0.8)	79.5 (0.6)	82.4 (0.5)	58.9 (1.1)	63.1 (0.4)	67.3 (1.1)	65.9 (0.7)
		200	44.1 (1.6)	80.9 (0.6)	82.0 (0.6)	58.6 (0.6)	65.2 (1.6)	66.8 (1.3)	66.3 (1.1)
		300	45.3[1] (1.9)	81.7[1] (0.7)	82.3 (0.6)	57.4 (1.3)	63.6 (0.7)	67.6 (0.6)	66.3 (1.0)
		600	44.5 (1.4)	**83.1** (0.9)	81.2 (0.8)	57.4 (1.1)	65.7 (1.2)	67.5 (0.7)	66.5 (1)
	BILSTM	50	43.6 (1.2)	82.9 (0.7)	79.2 (0.8)	59.5 (1.1)	65.6 (1.2)	64.3 (1.2)	65.9 (1.0)
		100	43.8 (1.1)	79.8 (1.0)	82.4 (0.6)	58.6 (0.8)	66.4 (1.4)	65.2 (0.6)	66.0 (0.9)
		200	44.0 (0.9)	80.1 (0.6)	81.7 (0.5)	58.9 (0.3)	63.3 (1.0)	66.4 (0.3)	65.7 (0.6)
		300	**45.6**[1] (1.6)	82.6[1] (0.7)	**82.5** (0.6)	59.3 (1.0)	66.2 (1.5)	65.1 (0.9)	**66.9** (1.1)
		600	43.2 (1.1)	83 (0.4)	81.5 (0.5)	59.2 (1.6)	66.4 (1.1)	**68.5** (0.7)	**66.9** (0.9)
	CNN	50	39.9 (0.7)	81.7 (0.3)	80.0 (0.9)	55.2 (0.7)	57.4 (3.1)	65.7 (1.0)	63.3 (1.1)
		100	40.1 (1.0)	81.6 (0.5)	79.5 (0.9)	56.0 (2.2)	61.5 (1.1)	64.2 (0.8)	63.8 (1.1)
		200	39.1 (1.1)	80.7 (0.4)	79.8 (0.7)	56.3 (1.8)	64.1 (1.1)	65.3 (0.8)	64.2 (1.0)
		300	39.8[2] (0.7)	81.3[2] (1.1)	80.3 (0.9)	57.3 (0.5)	62.1 (1.0)	63.5 (1.3)	64.0 (0.9)
		600	40.7 (2.6)	82.7 (1.2)	79.2 (1.4)	56.6 (0.6)	61.3 (2)	65.9 (1.8)	64.4 (1.5)

Gated RNNs: LSTMs

Binary classification (SST-2, 56.4k examples):

Model	Accuracy	Paper / Source
MT-DNN (Liu et al., 2019)	95.6	Multi-Task Deep Neural Networks for Natural Language Understanding
Bidirectional Encoder Representations from Transformers (Devlin et al., 2018)	94.9	BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
Block-sparse LSTM (Gray et al., 2017)	93.2	GPU Kernels for Block-Sparse Weights
bmLSTM (Radford et al., 2017)	91.8	Learning to Generate Reviews and Discovering Sentiment
Single layer bilstm distilled from BERT (Tang et al., 2019)	90.7	Distilling Task-Specific Knowledge from BERT into Simple Neural Networks
BCN+Char+CoVe (McCann et al., 2017)	90.3	Learned in Translation: Contextualized Word Vectors
Neural Semantic Encoder (Munkhdalai and Yu, 2017)	89.7	Neural Semantic Encoders
BLSTM-2DCNN (Zhou et al., 2017)	89.5	Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling

http://nlpprogress.com/

	Model	Dim.	SST-fine	SST-binary	OpeNER	SenTube-A	SenTube-T	SemEval	Macro-Avg.
Baselines	BOW		40.3	80.7	77.1[4]	60.6[5]	66.0[5]	65.5	65.0
	AVE	50	38.9	74.1	59.5	62.0	61.7	58.1	59.0
		100	39.7	76.7	67.2	61.5	61.8	58.8	60.9
		200	40.7	78.2	69.3	60.6	62.8	61.1	62.1
		300	41.6	80.3[3]	76.3	61.5	64.3	63.6	64.6
		600	40.6	79.1	77.0	56.4	62.9	61.8	63.0
State-of-the-Art Methods	RETROFIT	50	39.2	75.3	63.9	60.6	62.3	58.1	59.9
		100	39.7	76.7	70.0	61.4	62.8	59.5	61.7
		200	41.8	78.3	73.5	60.0	63.2	61.2	63.0
		300	42.2	81.2[3]	75.9	61.7	63.6	61.8	64.4
		600	42.9	81.1	78.3	60.0	65.5	62.4	65.0
	JOINT	50	35.8	70.6	72.9	65.1	**68.1**	66.8[6]	63.2
		100	34.3	70.8	67.0	64.3	66.4	60.1	60.5
		200	33.7	72.3	68.6	**66.2**	66.6	58.4	61.0
		300	36.0	71.6	70.1	64.7	67.6	60.8	61.8
		600	36.9	74.0	75.8	63.7	64.2	60.9	62.6
	LSTM	50	43.3 (1.0)	80.5 (0.4)	81.1 (0.4)	58.9 (0.8)	63.4 (3.1)	63.9 (1.7)	65.2 (1.2)
		100	44.1 (0.8)	79.5 (0.6)	82.4 (0.5)	58.9 (1.1)	63.1 (0.4)	67.3 (1.1)	65.9 (0.7)
		200	44.1 (1.6)	80.9 (0.6)	82.0 (0.6)	58.6 (0.6)	65.2 (1.6)	66.8 (1.3)	66.3 (1.1)
		300	45.3[1] (1.9)	81.7[1] (0.7)	82.3 (0.6)	57.4 (1.3)	63.6 (0.7)	67.6 (0.6)	66.3 (1.0)
		600	44.5 (1.4)	**83.1** (0.9)	81.2 (0.8)	57.4 (1.1)	65.7 (1.2)	67.5 (0.7)	66.5 (1)
	BILSTM	50	43.6 (1.2)	82.9 (0.7)	79.2 (0.8)	59.5 (1.1)	65.6 (1.2)	64.3 (1.2)	65.9 (1.0)
		100	43.8 (1.1)	79.8 (1.0)	82.4 (0.6)	58.6 (0.8)	66.4 (1.4)	65.2 (0.6)	66.0 (0.9)
		200	44.0 (0.9)	80.1 (0.6)	81.7 (0.5)	58.9 (0.3)	63.3 (1.0)	66.4 (0.3)	65.7 (0.6)
		300	**45.6**[1] (1.6)	82.6[1] (0.7)	**82.5** (0.6)	59.3 (1.0)	66.2 (1.5)	65.1 (0.9)	**66.9** (1.1)
		600	43.2 (1.1)	83 (0.4)	81.5 (0.5)	59.2 (1.6)	66.4 (1.1)	**68.5** (0.7)	**66.9** (0.9)
	CNN	50	39.9 (0.7)	81.7 (0.3)	80.0 (0.9)	55.2 (0.7)	57.4 (3.1)	65.7 (1.0)	63.3 (1.1)
		100	40.1 (1.0)	81.6 (0.5)	79.5 (0.9)	56.0 (2.2)	61.5 (1.1)	64.2 (0.8)	63.8 (1.1)
		200	39.1 (1.1)	80.7 (0.4)	79.8 (0.7)	56.3 (1.8)	64.1 (1.1)	65.3 (0.8)	64.2 (1.0)
		300	39.8[2] (0.7)	81.3[2] (1.1)	80.3 (0.9)	57.3 (0.5)	62.1 (1.0)	63.5 (1.3)	64.0 (0.9)
		600	40.7 (2.6)	82.7 (1.2)	79.2 (1.4)	56.6 (0.6)	61.3 (2)	65.9 (1.8)	64.4 (1.5)

Gated RNNs: LSTMs

Fine-grained classification (SST-5, 94,2k examples):

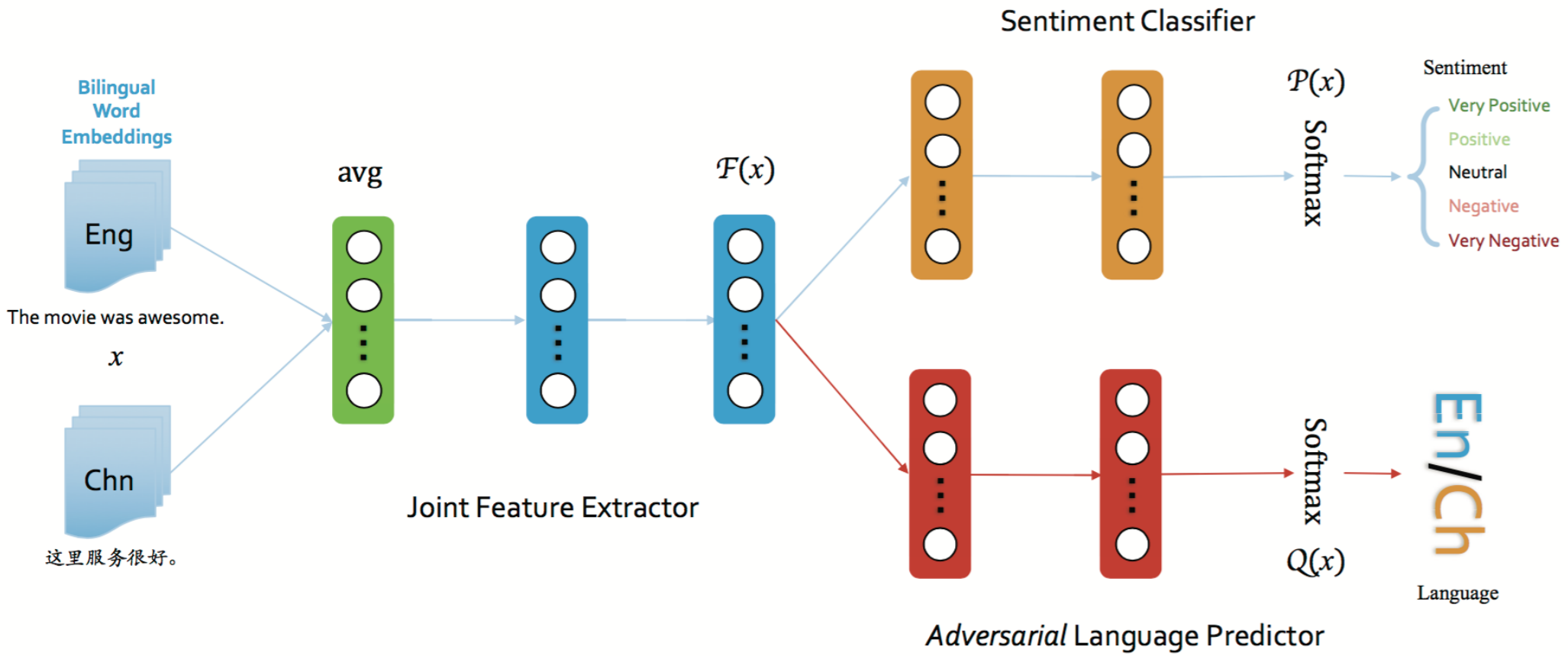
Model	Accuracy	Paper / Source
BCN+ELMo (Peters et al., 2018)	54.7	Deep contextualized word representations
BCN+Char+CoVe (McCann et al., 2017)	53.7	Learned in Translation: Contextualized Word Vectors

http://nlpprogress.com/

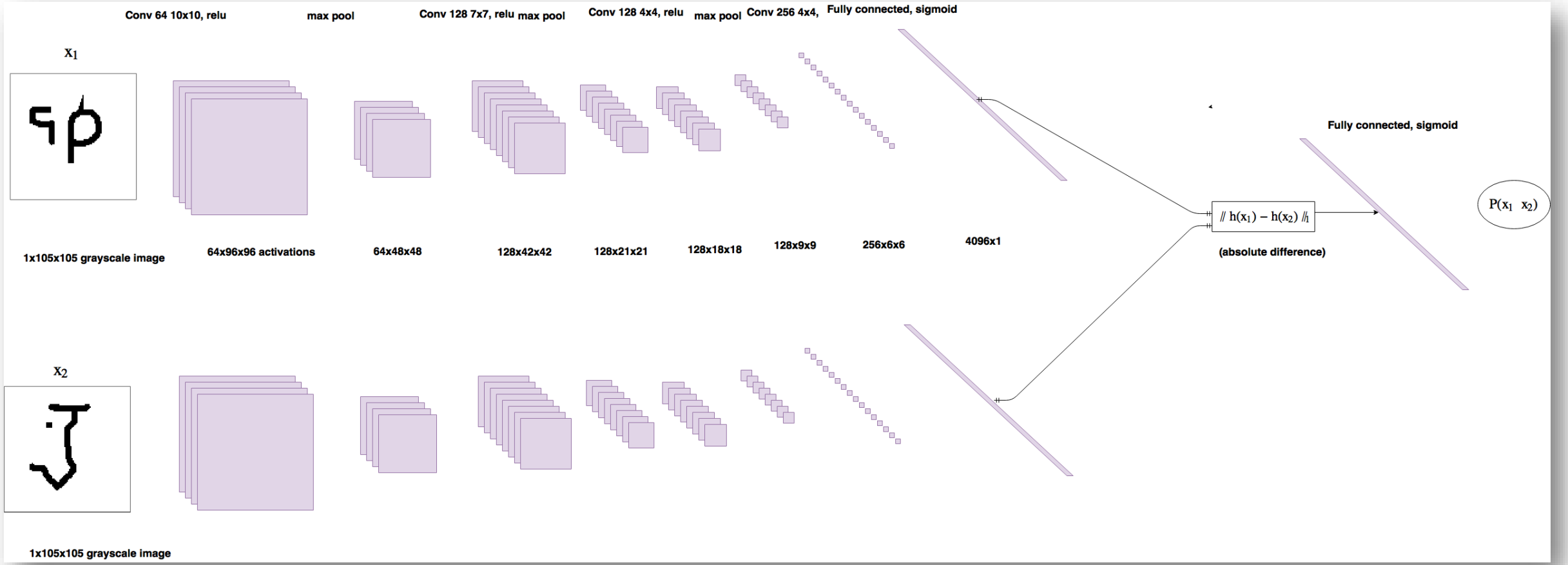
	Model	Dim.	SST-fine	SST-binary	OpeNER	SenTube-A	SenTube-T	SemEval	Macro-Avg.
Baselines	BOW		40.3	80.7	77.1[4]	60.6[5]	66.0[5]	65.5	65.0
	AVE	50	38.9	74.1	59.5	62.0	61.7	58.1	59.0
		100	39.7	76.7	67.2	61.5	61.8	58.8	60.9
		200	40.7	78.2	69.3	60.6	62.8	61.1	62.1
		300	41.6	80.3[3]	76.3	61.5	64.3	63.6	64.6
		600	40.6	79.1	77.0	56.4	62.9	61.8	63.0
State-of-the-Art Methods	RETROFIT	50	39.2	75.3	63.9	60.6	62.3	58.1	59.9
		100	39.7	76.7	70.0	61.4	62.8	59.5	61.7
		200	41.8	78.3	73.5	60.0	63.2	61.2	63.0
		300	42.2	81.2[3]	75.9	61.7	63.6	61.8	64.4
		600	42.9	81.1	78.3	60.0	65.5	62.4	65.0
	JOINT	50	35.8	70.6	72.9	65.1	**68.1**	66.8[6]	63.2
		100	34.3	70.8	67.0	64.3	66.4	60.1	60.5
		200	33.7	72.3	68.6	**66.2**	66.6	58.4	61.0
		300	36.0	71.6	70.1	64.7	67.6	60.8	61.8
		600	36.9	74.0	75.8	63.7	64.2	60.9	62.6
	LSTM	50	43.3 (1.0)	80.5 (0.4)	81.1 (0.4)	58.9 (0.8)	63.4 (3.1)	63.9 (1.7)	65.2 (1.2)
		100	44.1 (0.8)	79.5 (0.6)	82.4 (0.5)	58.9 (1.1)	63.1 (0.4)	67.3 (1.1)	65.9 (0.7)
		200	44.1 (1.6)	80.9 (0.6)	82.0 (0.6)	58.6 (0.6)	65.2 (1.6)	66.8 (1.3)	66.3 (1.1)
		300	45.3[1] (1.9)	81.7[1] (0.7)	82.3 (0.6)	57.4 (1.3)	63.6 (0.7)	67.6 (0.6)	66.3 (1.0)
		600	44.5 (1.4)	**83.1** (0.9)	81.2 (0.8)	57.4 (1.1)	65.7 (1.2)	67.5 (0.7)	66.5 (1)
	BILSTM	50	43.6 (1.2)	82.9 (0.7)	79.2 (0.8)	59.5 (1.1)	65.6 (1.2)	64.3 (1.2)	65.9 (1.0)
		100	43.8 (1.1)	79.8 (1.0)	82.4 (0.6)	58.6 (0.8)	66.4 (1.4)	65.2 (0.6)	66.0 (0.9)
		200	44.0 (0.9)	80.1 (0.6)	81.7 (0.5)	58.9 (0.3)	63.3 (1.0)	66.4 (0.3)	65.7 (0.6)
		300	**45.6**[1] (1.6)	82.6[1] (0.7)	**82.5** (0.6)	59.3 (1.0)	66.2 (1.5)	65.1 (0.9)	**66.9** (1.1)
		600	43.2 (1.1)	83 (0.4)	81.5 (0.5)	59.2 (1.6)	66.4 (1.1)	**68.5** (0.7)	**66.9** (0.9)
	CNN	50	39.9 (0.7)	81.7 (0.3)	80.0 (0.9)	55.2 (0.7)	57.4 (3.1)	65.7 (1.0)	63.3 (1.1)
		100	40.1 (1.0)	81.6 (0.5)	79.5 (0.9)	56.0 (2.2)	61.5 (1.1)	64.2 (0.8)	63.8 (1.1)
		200	39.1 (1.1)	80.7 (0.4)	79.8 (0.7)	56.3 (1.8)	64.1 (1.1)	65.3 (0.8)	64.2 (1.0)
		300	39.8[2] (0.7)	81.3[2] (1.1)	80.3 (0.9)	57.3 (0.5)	62.1 (1.0)	63.5 (1.3)	64.0 (0.9)
		600	40.7 (2.6)	82.7 (1.2)	79.2 (1.4)	56.6 (0.6)	61.3 (2)	65.9 (1.8)	64.4 (1.5)

Other architectures

ADANs


Sentiment Classifier
Bilingual Word Embeddings
Eng
The movie was awesome.
x
Chn
这里服务很好。
avg
$\mathcal{F}(x)$
Joint Feature Extractor
$\mathcal{P}(x)$
Softmax
Sentiment
Very Positive
Positive
Neutral
Negative
Very Negative
Softmax
$Q(x)$
En/Ch
Language
Adversarial Language Predictor

one-shot learning



Back to the intro

Rodney Brooks

From Wikipedia, the free encyclopedia

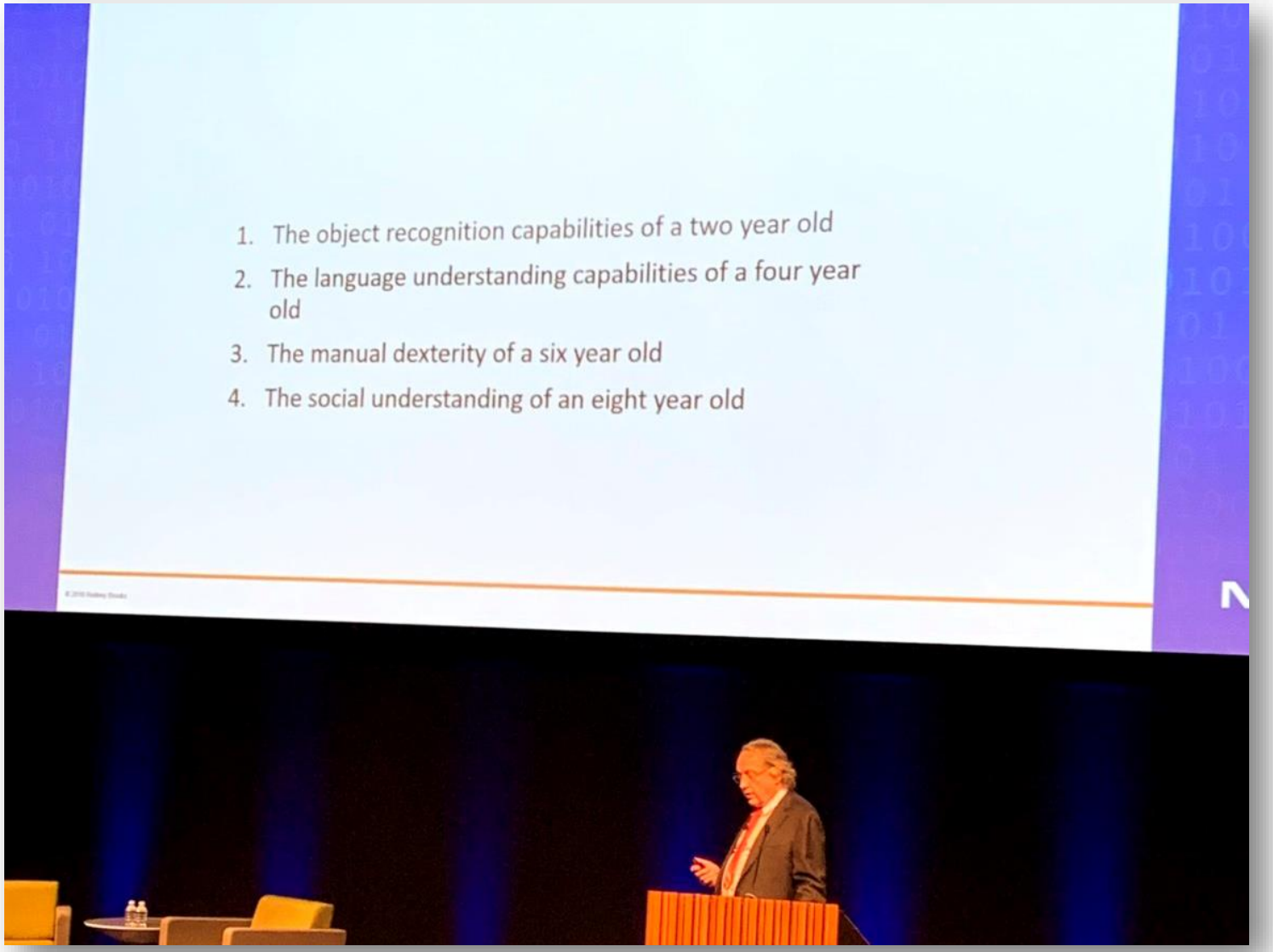
Rodney Allen Brooks (born 30 December 1954[1]) is an Australian roboticist, Fellow of the Australian Academy of Science, author, and robotics entrepreneur, most known for popularizing the actionist approach to robotics. He was a Panasonic Professor of Robotics at the Massachusetts Institute of Technology and former director of the MIT Computer Science and Artificial Intelligence Laboratory. He is a founder and former Chief Technical Officer of iRobot[2] and co-Founder, Chairman and Chief Technical Officer of Rethink Robotics (formerly Heartland Robotics). and his work, *Fast, Cheap & Out of Co*

Contents [hide]

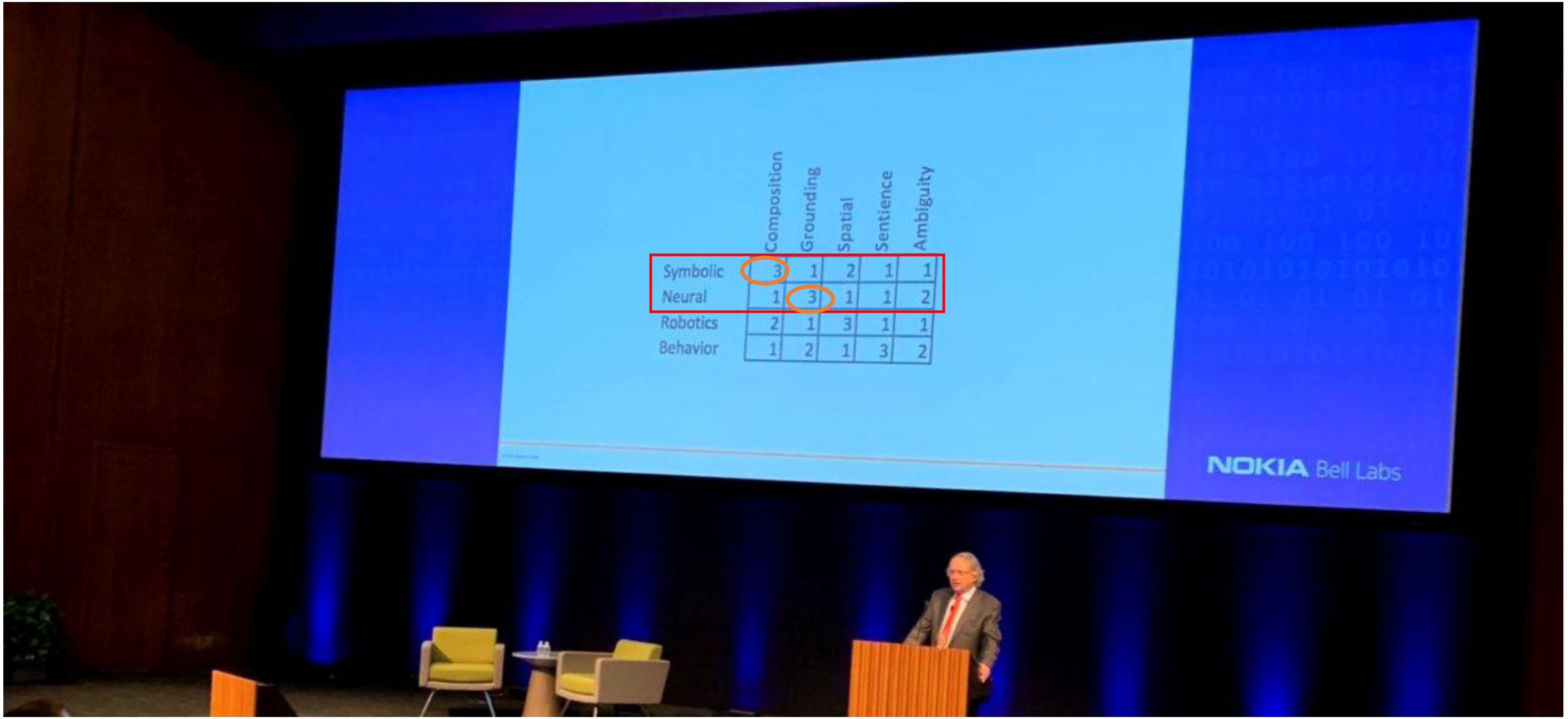
1 Life
2 Work
2.1 Academic work

Rodney Brooks

Back to the intro



Back to the intro



Back to the intro