Neural Network Models for Text Classification

Hongwei Wang

18/11/2016

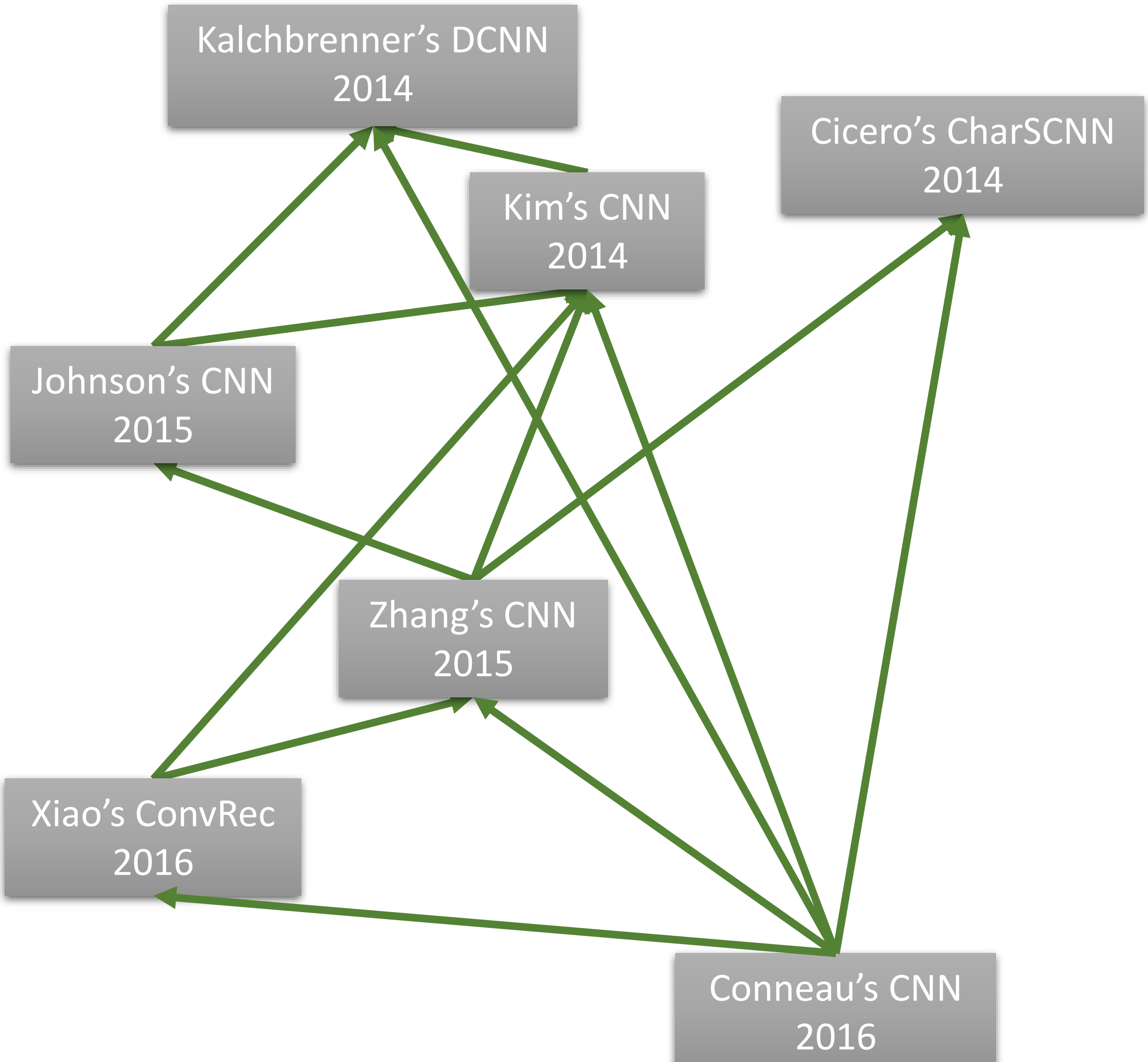
Deep Learning in NLP

- **Feedforward Neural Network**
  - The most basic form of NN
- **Convolutional Neural Network (CNN)**
  - Quite successful in computer vision
  - Extract local features
  - Most popular and effective in sentence classification
- **Recursive Neural Network**
  - Rely on parser tree of the sentence
- **Recurrent Neural Network (RNN)**
  - Designed for sequential data
  - The most popular version: **Long Short-Term Memory (LSTM)**
  - Further variants: Bidirectional-LSTM, Deep-LSTM …

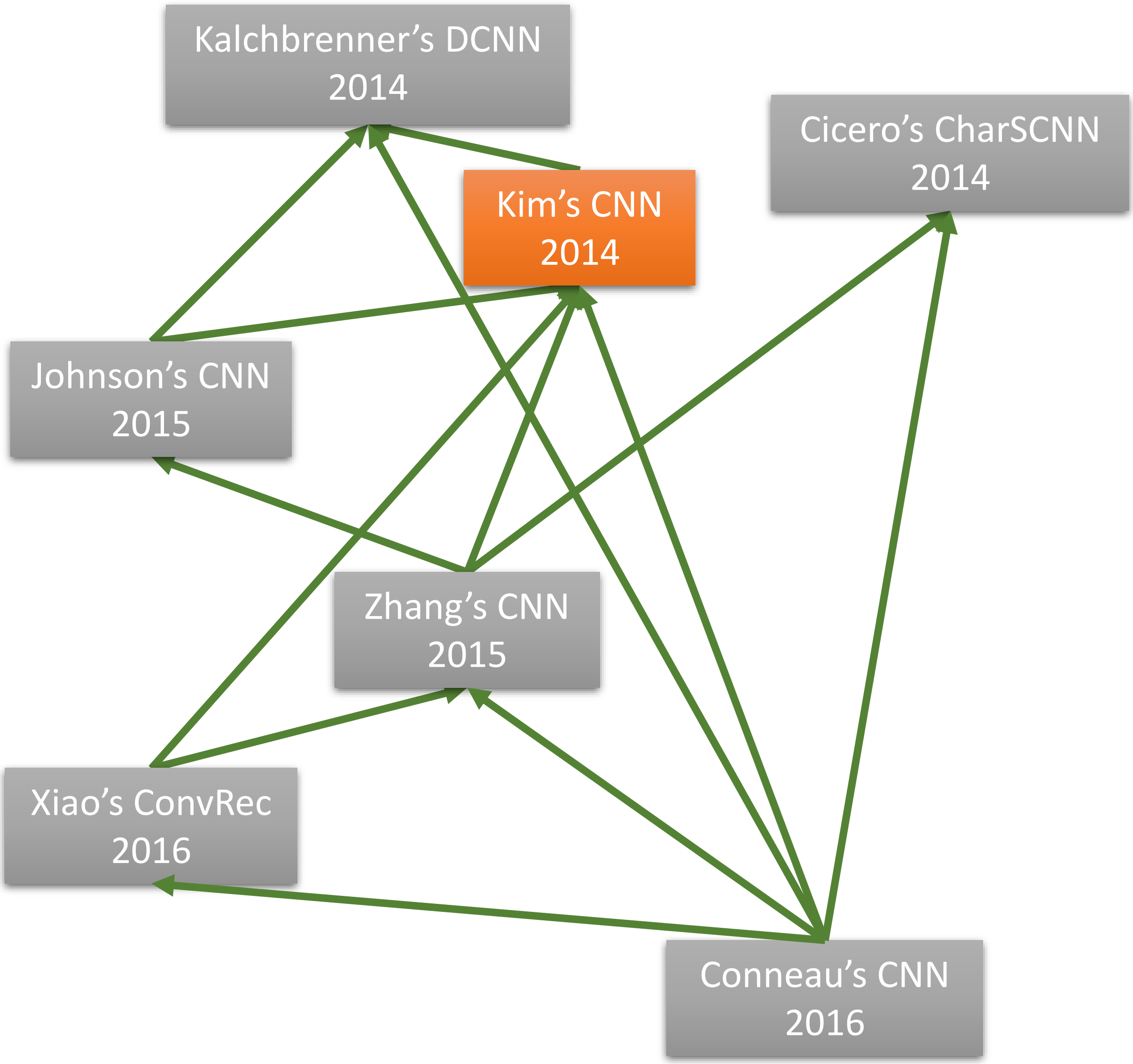
Deep Learning in NLP

- Feedforward Neural Network
  - The most basic form of NN
- **Convolutional Neural Network (CNN)**
  - Quite successful in computer vision
  - Extract local features
  - Most popular and effective in sentence classification
- Recursive Neural Network
  - Rely on parser tree of the sentence
- Recurrent Neural Network (RNN)
  - Designed for sequential data
  - The most popular version: **Long Short-Term Memory (LSTM)**
  - Further variants: Bidirectional-LSTM, Deep-LSTM …

Paper roadmap



Paper roadmap



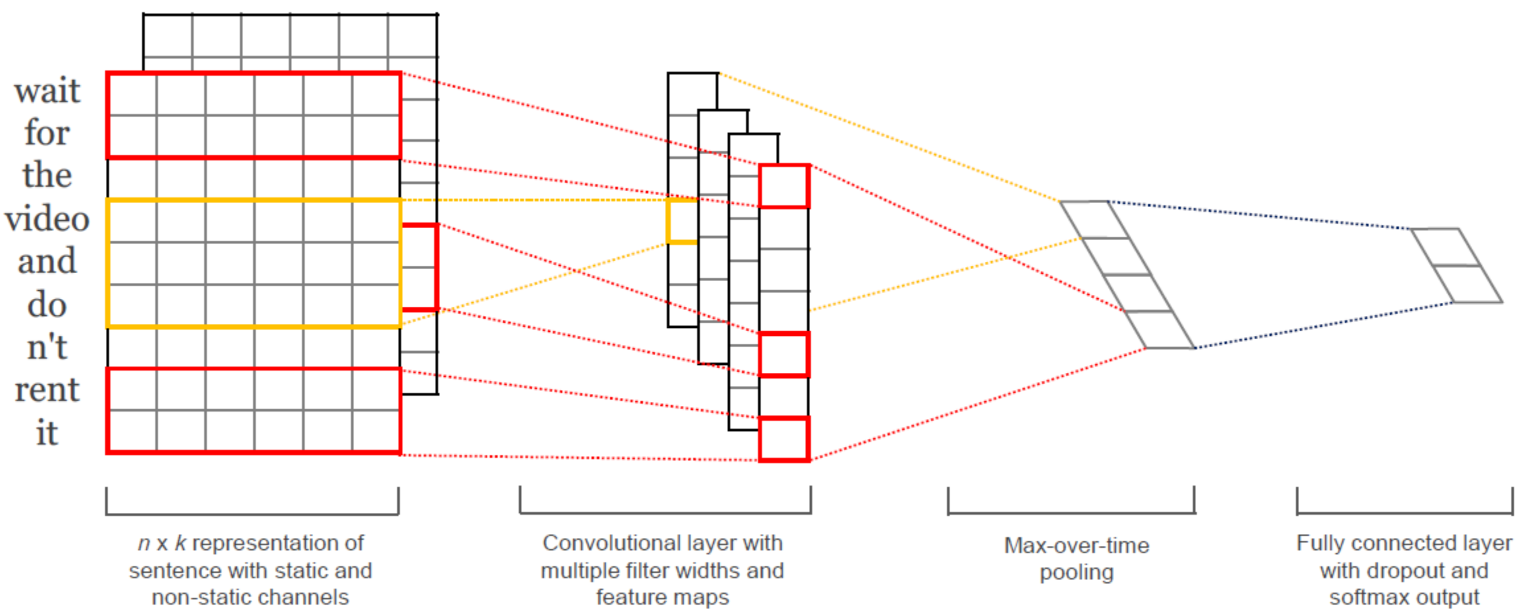
Kim's CNN

Convolutional Neural Networks for Sentence Classification

Yoon Kim
arXiv, 2014

Kim's CNN

Model architecture

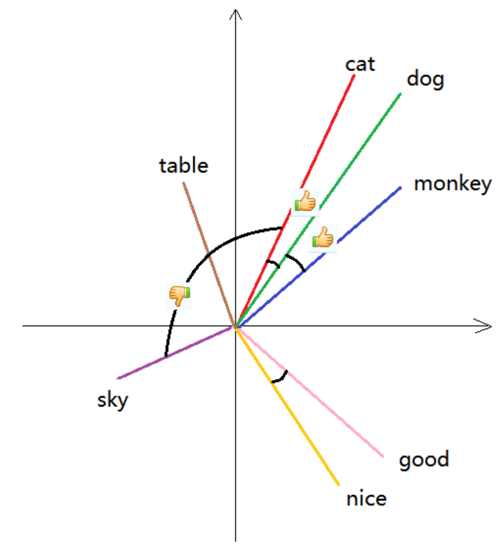


Kim's CNN

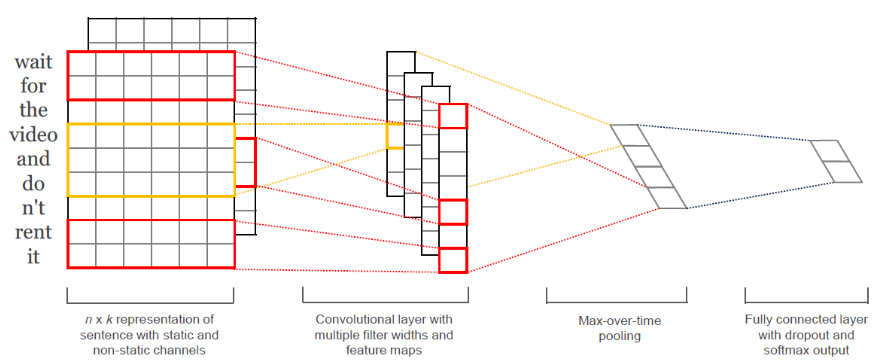
Preparation

- ❑ **Word embedding**
  - ❑ Sparse, 1-of-V encoding → lower dimensional vector space
  - ❑ Semantically close words are likewise close

- ❑ **Padding to size *n* if necessary**



Projection of the embedding vectors to 2-D

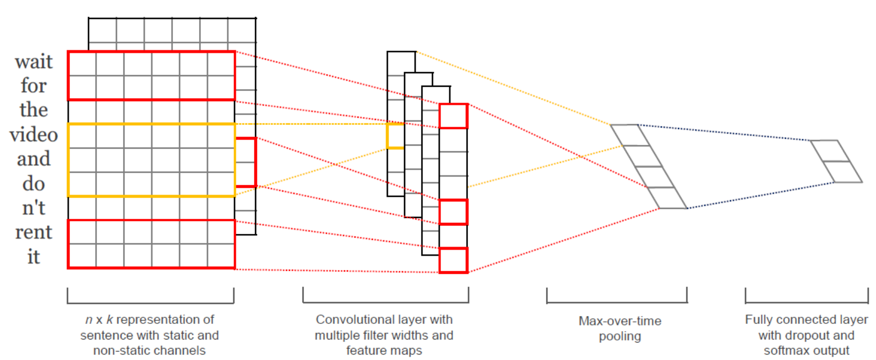


Kim's CNN

CNN layers

- **Convolution layer**
  - Input sentence: $\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \ldots \oplus \mathbf{x}_n$
  - Output local feature: $c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b)$
  - Feature map: $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}]$
- **Max-pooling layer**

  $\hat{c} = \max\{\mathbf{c}\}$
- **Fully connected layer with softmax output**



Kim's CNN

Regularization on the FC layer

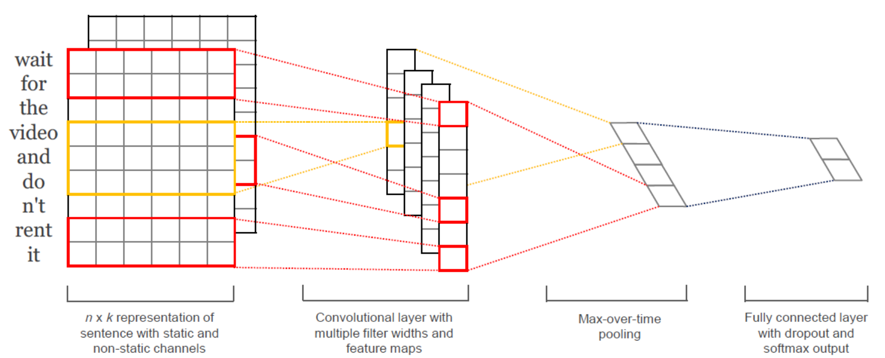
- **Dropout**
  - Using **r** (Bernouli random variables w.p. *p* of being 1) as mask when training:

$$y = \mathbf{w} \cdot (\mathbf{z} \circ \mathbf{r}) + b$$

  - Scale learned vectors by p when testing:

$$\hat{\mathbf{w}} = p\mathbf{w}$$

- **l_2-norms constraints**
  - Rescale w to have $||\mathbf{w}||_2 = s$



Kim’s CNN

Model variations

- **CNN-rand (baseline)**
  - All words are randomly initialized and modified during training
- **CNN-static**
  - Pre-trained word vectors via word2vec
  - Kept static during training
- **CNN-non-static**
  - Pre-trained word vectors via word2vec
  - Fine-tuned for each task during training
- **CNN-multichannel**
  - Both channels are initialized with word2vec
  - One channel static, the other fine-tuned
  - Each filter is applied to both channels and the results are added

Kim's CNN

Hyper-parameters and training

- **Hyper-parameters** (via grid search on the SST-2 set)
  - Rectified linear units
  - Filter windows: 3, 4, 5 with 100 feature maps each
  - Dropout rate: 0.5
  - L_2 constraint: 3
  - Minibatch size: 50
  - Dimensionality of word embedding: 300

- **During training——**
  - Early stopping
  - Stochastic gradient descent over shuffled mini-batches

Kim's CNN

Datasets

Data	c	l	N	$\|V\|$	$\|V_{pre}\|$	*Test*
MR	2	20	10662	18765	16448	CV
SST-1	5	18	11855	17836	16262	2210
SST-2	2	19	9613	16185	14838	1821
Subj	2	23	10000	21323	17913	CV
TREC	6	10	5952	9592	9125	500
CR	2	19	3775	5340	5046	CV
MPQA	2	3	10606	6246	6083	CV

- c: number of target classes;
  l: average sentence length;
  N: dataset size;
  |V|: vocabulary size;
  $|V_{pre}|$: number of word present in the set of pre-trained word vectors;
  Test: test set size (CV means 10-fold CV was used)

Kim's CNN

Results (accuracy)

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-rand	76.1	45.0	82.7	89.6	91.2	79.8	83.4
CNN-static	81.0	45.5	86.8	93.0	92.8	84.7	**89.6**
CNN-non-static	**81.5**	48.0	87.2	93.4	93.6	84.3	89.5
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
RAE (Socher et al., 2011)	77.7	43.2	82.4	–	–	–	86.4
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	–	–	–	–
RNTN (Socher et al., 2013)	–	45.7	85.4	–	–	–	–
DCNN (Kalchbrenner et al., 2014)	–	48.5	86.8	–	93.0	–	–
Paragraph-Vec (Le and Mikolov, 2014)	–	**48.7**	87.8	–	–	–	–
CCAE (Hermann and Blunsom, 2013)	77.8	–	–	–	–	–	87.2
Sent-Parser (Dong et al., 2014)	79.5	–	–	–	–	–	86.3
NBSVM (Wang and Manning, 2012)	79.4	–	–	93.2	–	81.8	86.3
MNB (Wang and Manning, 2012)	79.0	–	–	**93.6**	–	80.0	86.3
G-Dropout (Wang and Manning, 2013)	79.0	–	–	93.4	–	82.1	86.1
F-Dropout (Wang and Manning, 2013)	79.1	–	–	**93.6**	–	81.9	86.3
Tree-CRF (Nakagawa et al., 2010)	77.3	–	–	–	–	81.4	86.1
CRF-PR (Yang and Cardie, 2014)	–	–	–	–	–	82.7	–
SVM_S (Silva et al., 2011)	–	–	–	–	**95.0**	–	–

Kim's CNN

Discussion

- Multichannel vs. single channel
- Additional 2%-4% gain by dropout
- word2vec gave far superior performance compared with another word embedding method

Kim's CNN

Discussion (cont.)

- Static vs non-static

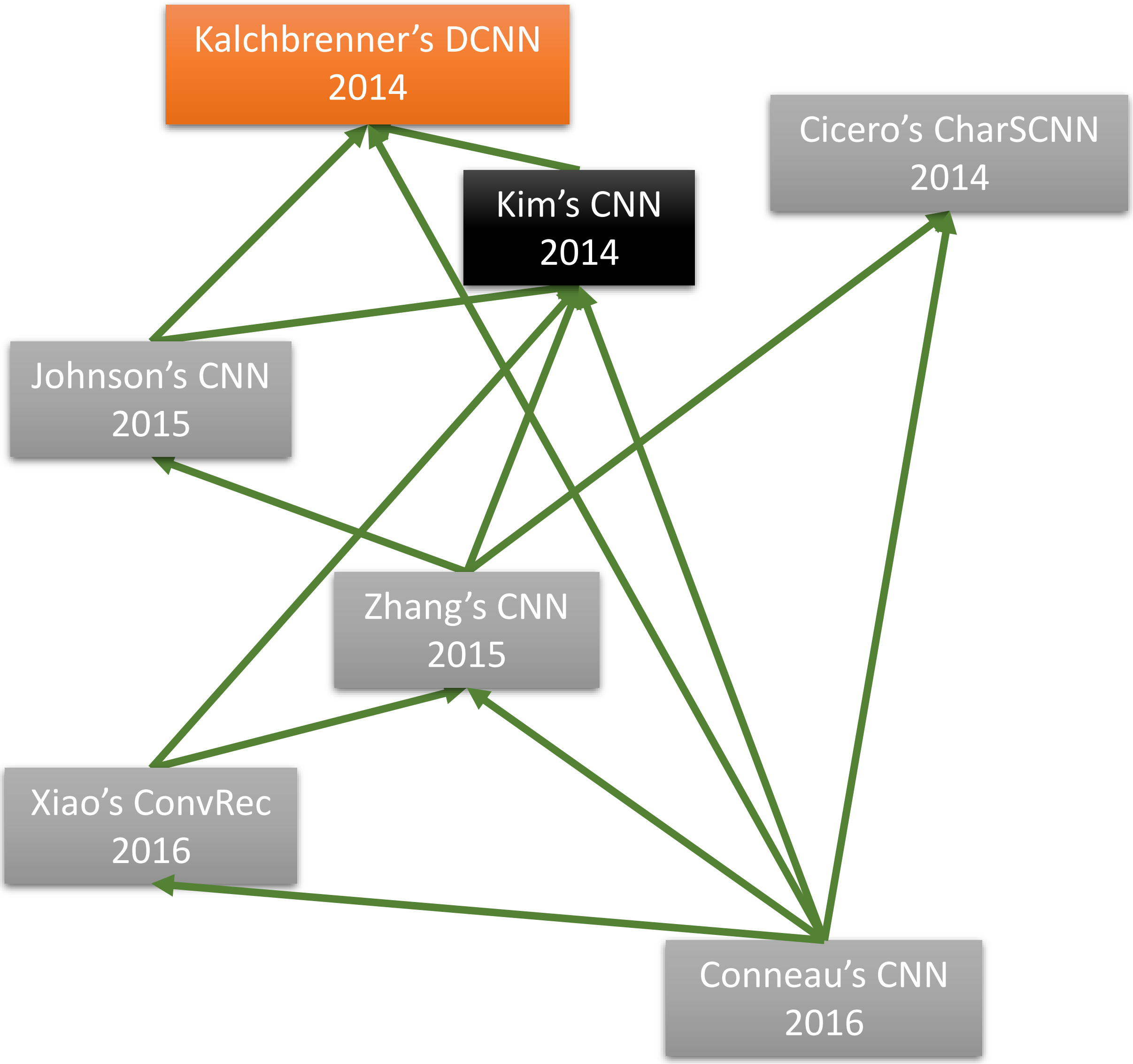
	Most Similar Words for	
	Static Channel	Non-static Channel
bad	*good* *terrible* *horrible* *lousy*	*terrible* *horrible* *lousy* *stupid*
good	*great* *bad* *terrific* *decent*	*nice* *decent* *solid* *terrific*
n't	*os* *ca* *ireland* *wo*	*not* *never* *nothing* *neither*
!	*2,500* *entire* *jez* *changer*	*2,500* *lush* *beautiful* *terrific*
,	*decasia* *abysmally* *demise* *valiant*	*but* *dragon* *a* *and*

Kim's CNN

Summary

Network type	CNN
# of layers	3 (1 con + 1 pool + 1 fc)
Word or character level	word level
Embedding	word2vec / self-learned
# of embedding dimension	300
Padding	yes
# of feature maps	3 * 100
Size of window	3, 4, 5
Pooling type	max
Non-linear function	ReLU
Regularization	dropout l_2-norm constraint

Paper roadmap



Kalchbrenner's DCNN

A Convolutional Neural Network for Modeling Sentences

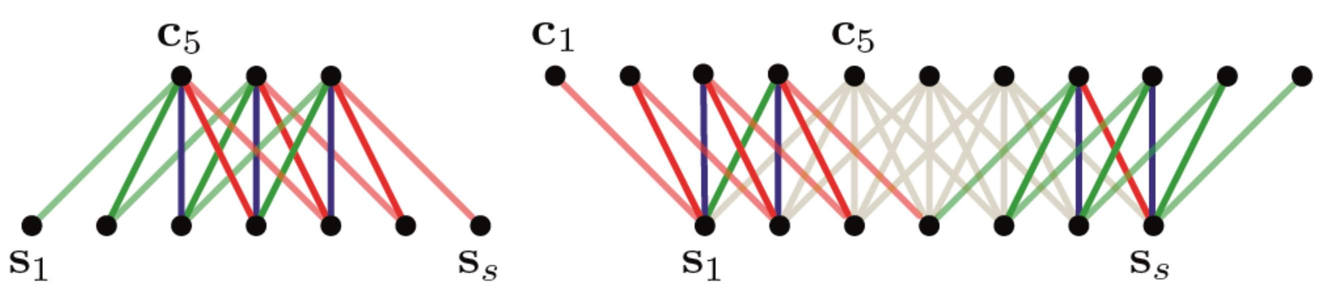
Nal Kalchbrenner, Edward Grefenstette, Phil Blunsom
arXiv, 2014

Kalchbrenner's DCNN

First we should know …

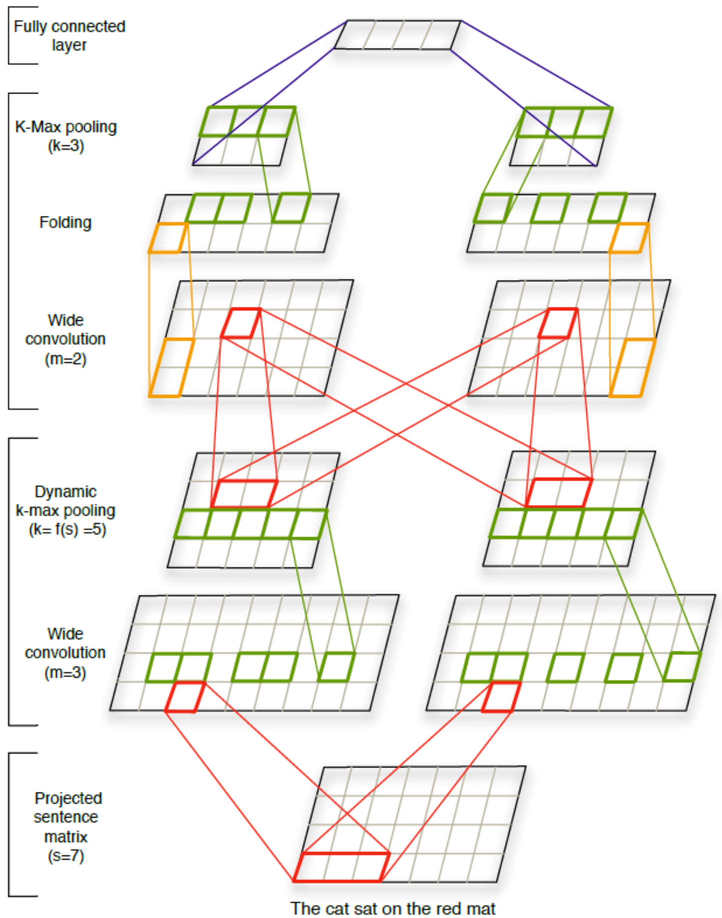
- **Disadvantages of max pooling**
  - Cannot distinguish the occurrence time of specific feature
  - Pooling factor can be excessive

- **Narrow & wide convolution**



Kalchbrenner's DCNN

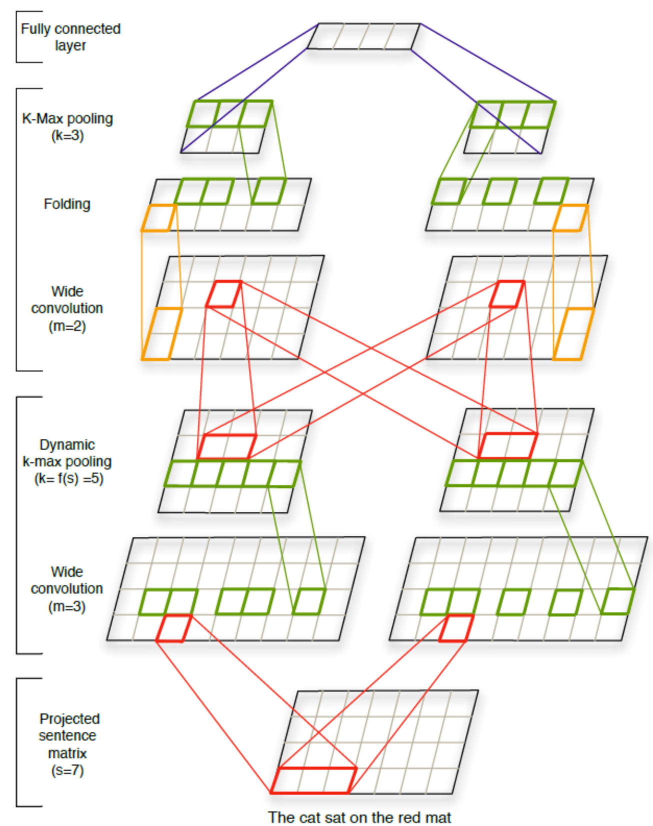
Model architecture



Kalchbrenner's DCNN

What is new

- **Wide convolution**
- **Dimension-wise convolution**
  - Embedding dimension — d; window size — m;
  - #d convolution vector of size m;
- **Dynamic *k*-Max pooling**
  - Largest *k* values selected, order preserved
  - $k_l = \max(\, k_{top},\ \lceil \frac{L-l}{L} s \rceil\,)$
- **Folding**
  - After convolution layer, before *k*-max pooling layer
  - Sum every two rows in a feature map component-wise



Kalchbrenner's DCNN

Results (accuracy)

SST1 & SST2

Classifier	Fine-grained (%)	Binary (%)
NB	41.0	81.8
BiNB	41.9	83.1
SVM	40.7	79.4
RecNTN	45.7	85.4
Max-TDNN	37.4	77.1
NBoW	42.4	80.5
DCNN	**48.5**	**86.8**

TREC

Classifier	Features	Acc. (%)
Hier	unigram, POS, head chunks NE, semantic relations	91.0
MaxEnt	unigram, bigram, trigram POS, chunks, NE, supertags CCG parser, WordNet	92.6
MaxEnt	unigram, bigram, trigram POS, wh-word, head word word shape, parser hypernyms, WordNet	93.6
SVM	unigram, POS, wh-word head word, parser hypernyms, WordNet 60 hand-coded rules	95.0
Max-TDNN	unsupervised vectors	84.4
NBoW	unsupervised vectors	88.2
DCNN	unsupervised vectors	93.0

Twitter sentiment

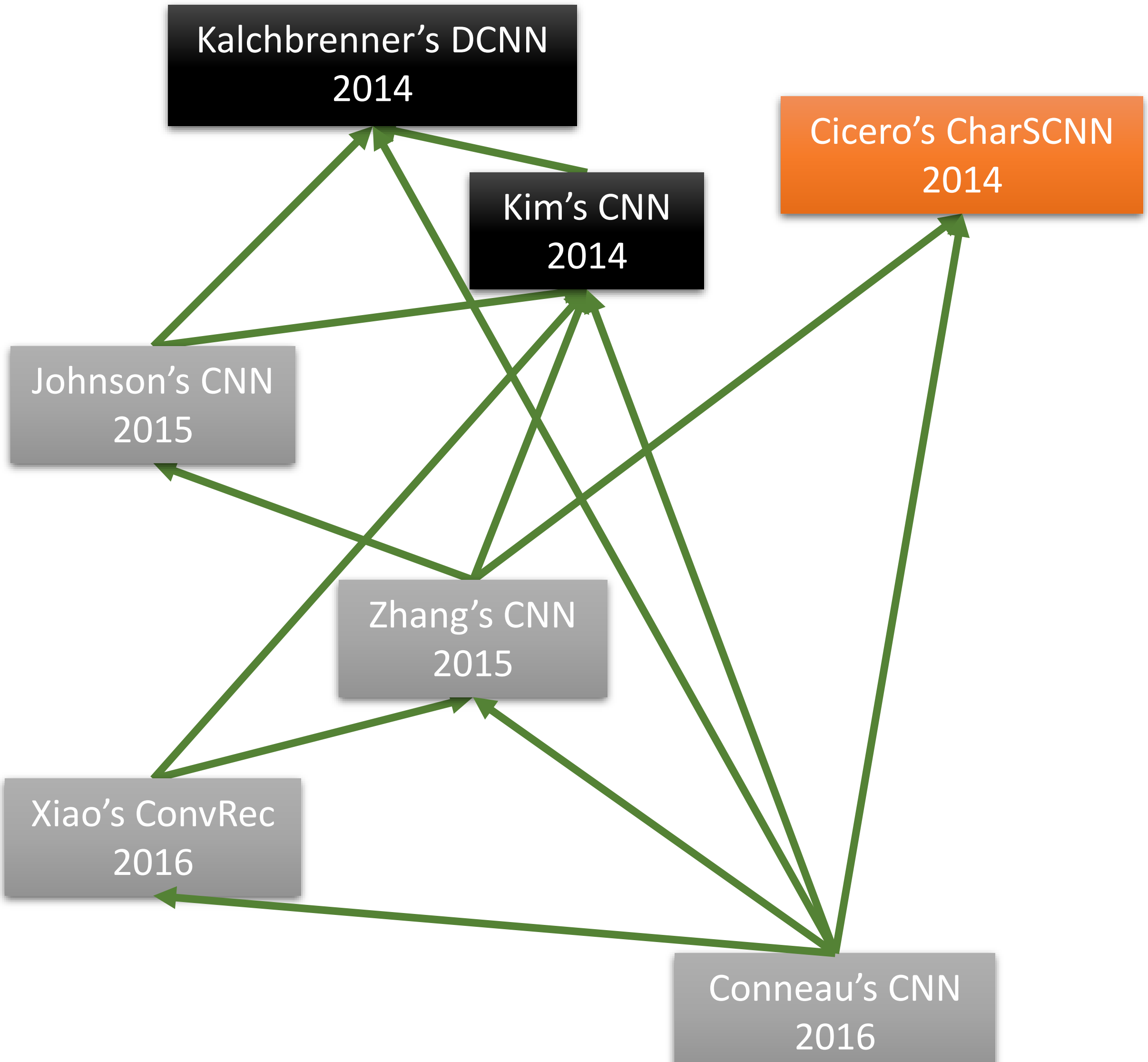
Classifier	Accuracy (%)
SVM	81.6
BiNB	82.7
MaxEnt	83.0
Max-TDNN	78.8
NBoW	80.9
DCNN	**87.4**

Kalchbrenner's DCNN

Summary

Experiment	SST1	SST2	TREC	Twitter
Network type	CNN			
# of layers	7 ((1 con + 1 fold + 1 pool) * 2 + 1 fc)			
Word or character level	word level			
Embedding	self-learned			
# of embedding dimension	48		32	60
Padding	no			
# of feature maps	6, 14	6, 12	8, 5	/
Size of window	7, 5	10, 7	/	/
Pooling type	dynamic *k*-max			
Non-linear function	tanh			
Regularization	L_2			

Paper roadmap



Cicero's CharSCNN

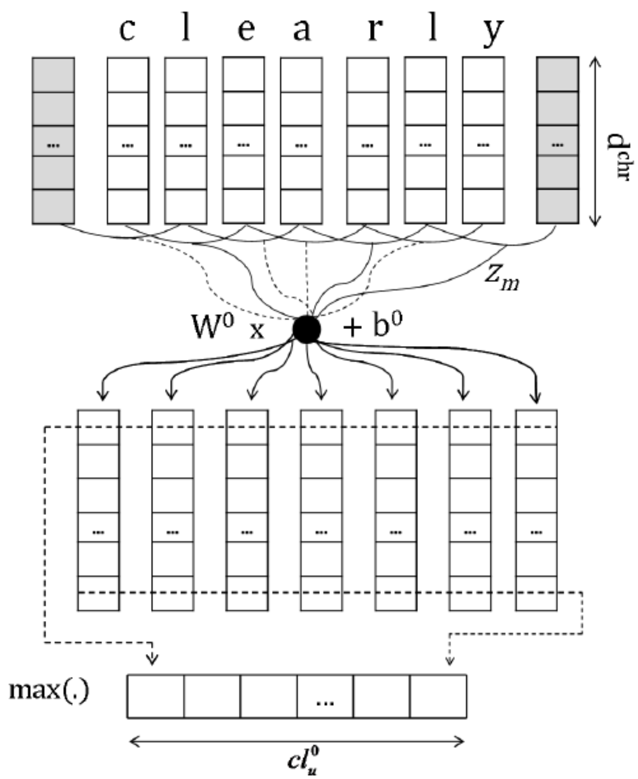
Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts

Cicero Nogueira dos Santos, Maira Gatti
Coling, 2014

Cicero's CharSCNN

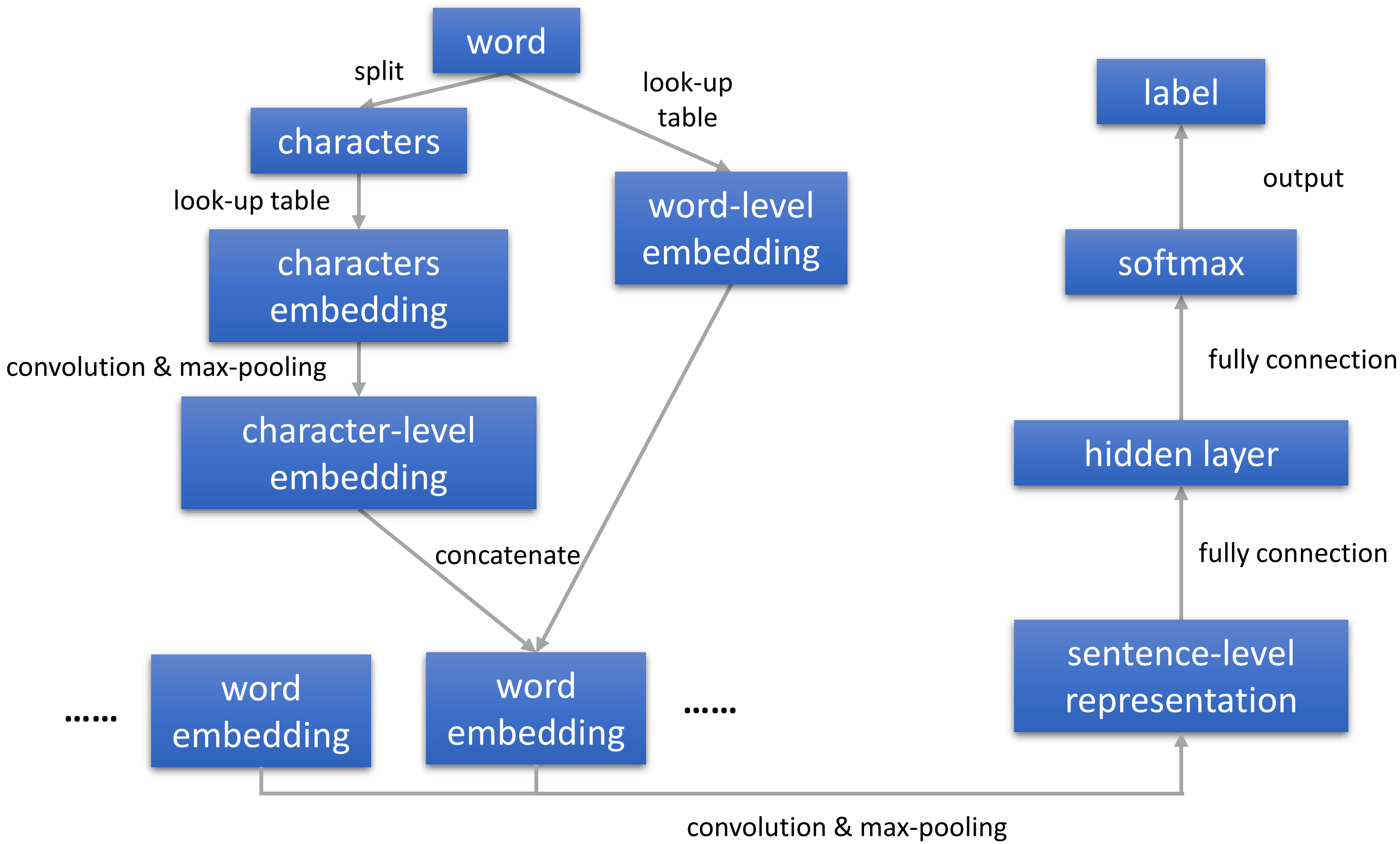
What is new

- **Word-level + character-level embedding**



Cicero's CharSCNN

Model Architecture



Cicero's CharSCNN

Results (accuracy)

SST

Model	Phrases	Fine-Grained	Positive/Negative
CharSCNN	yes	**48.3**	**85.7**
SCNN	yes	**48.3**	85.5
CharSCNN	no	43.5	82.3
SCNN	no	43.5	82.0
RNTN (Socher et al., 2013b)	yes	45.7	85.4
MV-RNN (Socher et al., 2013b)	yes	44.4	82.9
RNN (Socher et al., 2013b)	yes	43.2	82.4
NB (Socher et al., 2013b)	yes	41.0	81.8
SVM (Socher et al., 2013b)	yes	40.7	79.4

Twitter sentiment

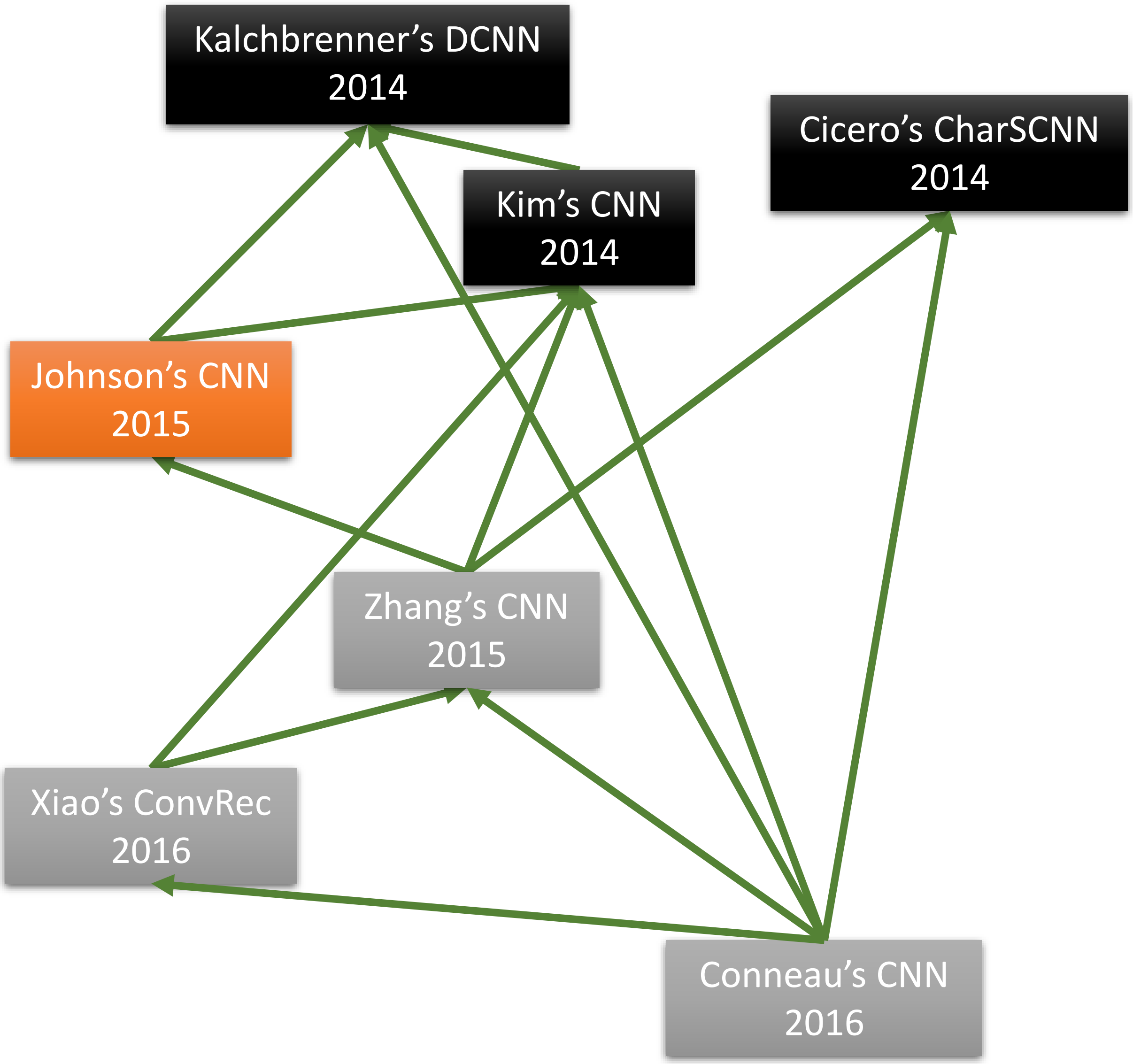
Model	Accuracy (unsup. pre-training)	Accuracy (random word embeddings)
CharSCNN	**86.4**	81.9
SCNN	85.2	82.2
LProp (Speriosu et al., 2011)	84.7	
MaxEnt (Go et al., 2009)	83.0	
NB (Go et al., 2009)	82.7	
SVM (Go et al., 2009)	82.2	

Cicero's CharSCNN

Summary

Experiment	SST	Twitter
Network type	CNN	
# of layers	6 ((1 con + 1 pool) * 2 + 2 fc)	
Word or character level	word level + character level	
Embedding	word2vec (word) / self-learned (char)	
# of embedding dimension	5 (char), 30 (word)	
Padding	yes	
# of feature maps	10 (char), 300 (word)	50 (char), 300 (word)
Size of window	3 (char), 5 (word)	
Pooling type	max	
Non-linear function	tanh	
Regularization	/	

Paper roadmap



Johnson's CNN

Effective Use of Word Order for Text Categorization with Convolutional Neural Networks

Rie Johnson, Tong Zhang
arXiv, 2015

Johnson's CNN

What is new

- **No word-embedding**
  - Word embedding is just s special case of convolution layer with window size 1
  - Directly learn features of text without embedding learning
- **seq-CNN & bow-CNN**
  - V = {"don't", "hate", "I", "it", "love"}, D = "I love it"
  - $\mathbf{x} = [0\ 0\ 1\ 0\ 0\ |\ 0\ 0\ 0\ 0\ 1\ |\ 0\ 0\ 0\ 1\ 0]^{T}$

$$\mathbf{r}_0(\mathbf{x}) = \begin{bmatrix} 0 \\ 0 \\ \mathbf{1} \\ 0 \\ 0 \\ \text{—} \\ 0 \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \end{bmatrix} \begin{array}{l} \text{don't} \\ \text{hate} \\ \textbf{I} \\ \text{it} \\ \text{love} \\ \\ \text{don't} \\ \text{hate} \\ \text{I} \\ \text{it} \\ \textbf{love} \end{array} \quad \mathbf{r}_1(\mathbf{x}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \\ \text{—} \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \\ 0 \end{bmatrix} \begin{array}{l} \text{don't} \\ \text{hate} \\ \text{I} \\ \text{it} \\ \textbf{love} \\ \\ \text{don't} \\ \text{hate} \\ \text{I} \\ \textbf{it} \\ \text{love} \end{array}$$

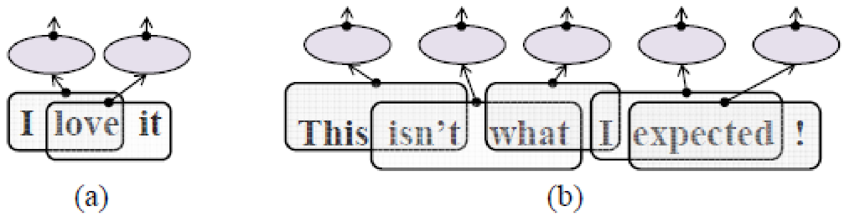
seq-CNN

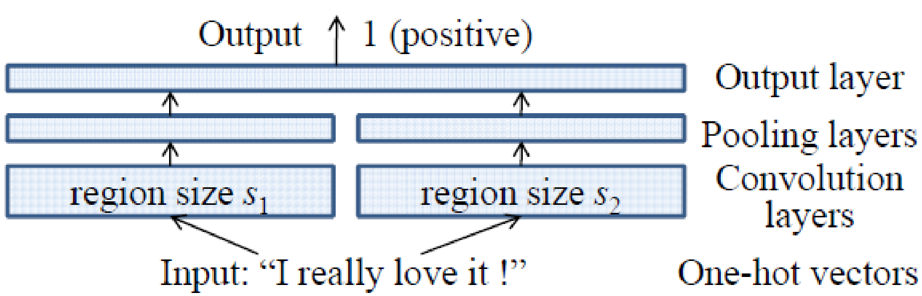
$$\mathbf{r}_0(\mathbf{x}) = \begin{bmatrix} 0 \\ 0 \\ \mathbf{1} \\ 0 \\ \mathbf{1} \end{bmatrix} \begin{array}{l} \text{don't} \\ \text{hate} \\ \textbf{I} \\ \text{it} \\ \textbf{love} \end{array} \quad \mathbf{r}_1(\mathbf{x}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix} \begin{array}{l} \text{don't} \\ \text{hate} \\ \text{I} \\ \textbf{it} \\ \textbf{love} \end{array}$$

bow-CNN

Johnson's CNN

Model Architecture





Johnson's CNN

Results (error)

methods	IMDB	Elec	RCV1
SVM bow3 (30K)	10.14	9.16	10.68
SVM bow1 (all)	11.36	11.71	10.76
SVM bow2 (all)	9.74	9.05	10.59
SVM bow3 (all)	9.42	8.71	10.69
NN bow3 (all)	9.17	8.48	10.67
NB-LM bow3 (all)	8.13	8.11	13.97
bow-CNN	8.66	8.39	**9.33**
seq-CNN	8.39	7.64	9.96
seq2-CNN	8.04	7.48	–
seq2-bow$n$-CNN	**7.67**	**7.14**	–

Johnson's CNN

Summary

Experiment	IMDB & Elec	RCV1
Network type	CNN	
# of layers	3 (1 con + 1 pool + + 1 fc)	
Word or character level	word level	
Embedding	none	
# of embedding dimension	N.A.	
Padding	yes	
# of feature maps	1000 (seq, bow) 2000 (seq2) 2020(seq2-bow*n*)	1000
Size of window	3	20
Pooling type	max	avg (10 units)
Non-linear function	ReLU	
Regularization	dropout L_2	

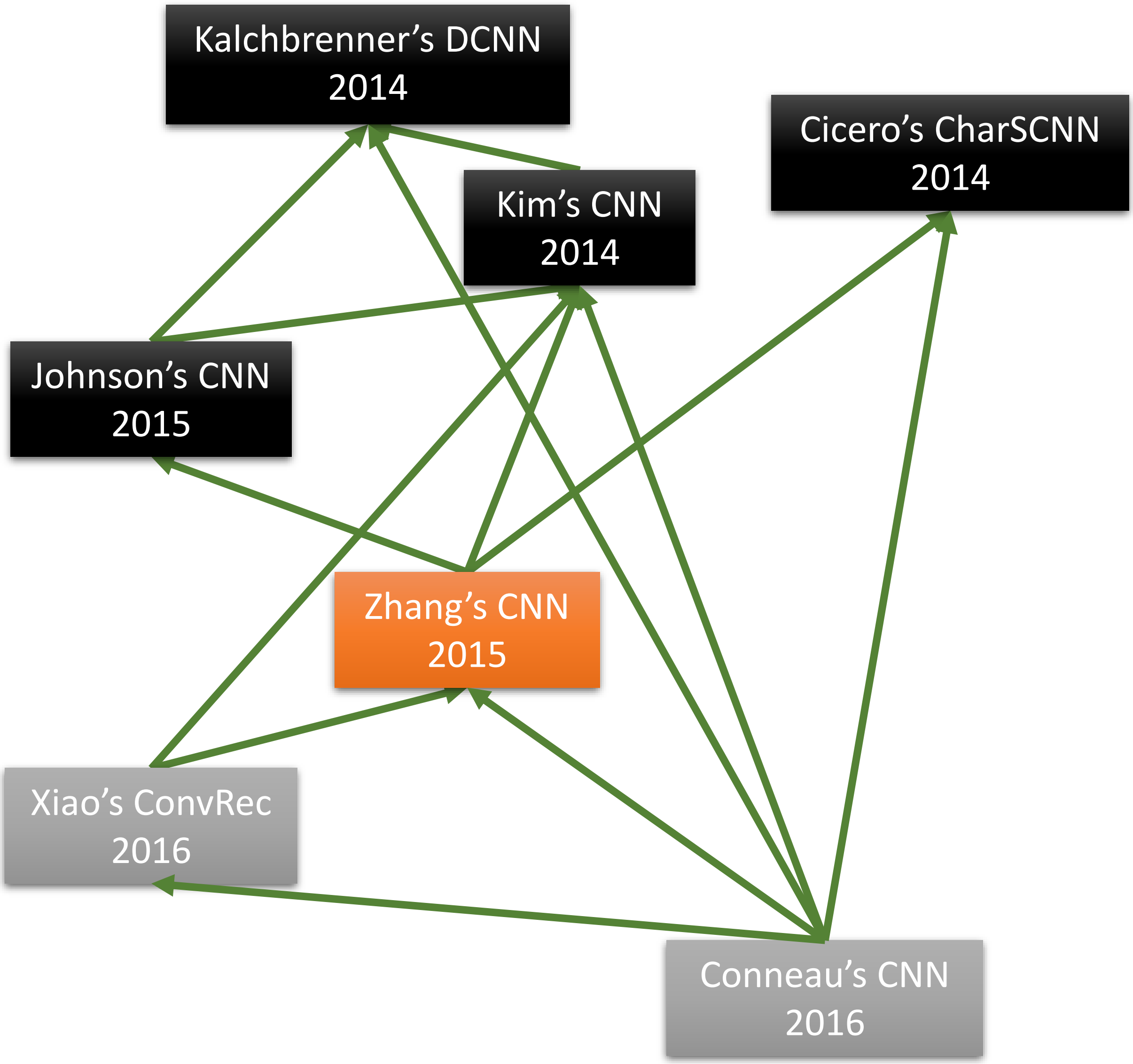
Johnson's CNN

What's more …

- **tv-embedding**
  - tv stands for "two-view"
  - Region embedding from unlabeled data
- **Supervised CNN + tv-embedding = semi-supervised CNN**
  - Published in NIPS 2015
  - Even better results

			IMDB	Elec	RCV1
linear SVM with 1-3grams [11]			10.14	9.16	10.68
linear TSVM with 1-3grams			9.99	16.41	10.77
[13]'s CNN			9.17	8.03	10.44
One-hot CNN (simple) [11]			8.39	7.64	9.17
One-hot CNN (simple) co-training best			(8.06)	(7.63)	(8.73)
Our CNN	unsup-tv.	100-dim	7.12	6.96	8.10
		200-dim	6.81	6.69	7.97
	parsup-tv.	100-dim	7.12	6.58	8.19
		200-dim	7.13	6.57	7.99
	unsup3-tv.	100-dim	7.05	6.66	8.13
		200-dim	6.96	6.84	8.02
	all three	100×3	**6.51**	**6.27**	**7.71**

Paper roadmap



Zhang's CNN

Character-level Convolutional Networks for Text Classification

Xiang Zhang, Junbo Zhao, Yann LeCun
NIPS, 2015

Zhang's CNN

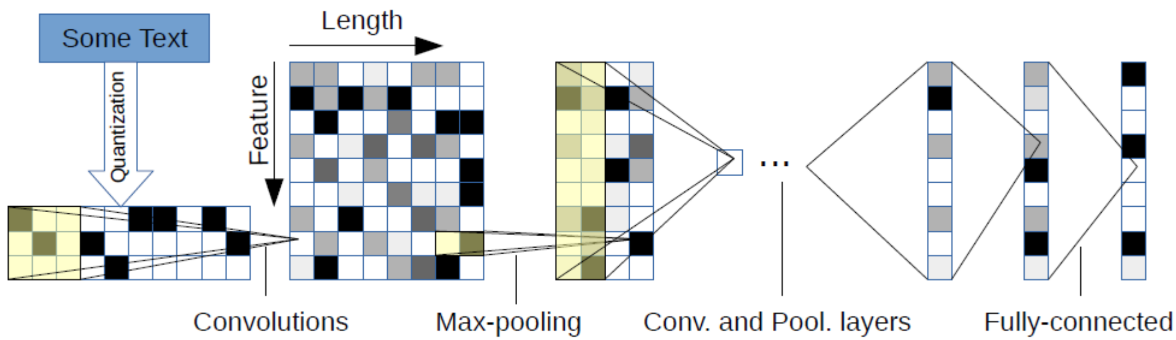
What is new

- **Apply CNN only on characters**
  - Do not require knowledge of words
  - Do not require knowledge about syntactic or semantic structure

- **Somewhat deep …**

Zhang's CNN

Model Architecture



Layer	Large Feature	Small Feature	Kernel	Pool
1	1024	256	7	3
2	1024	256	7	3
3	1024	256	3	N/A
4	1024	256	3	N/A
5	1024	256	3	N/A
6	1024	256	3	3

Layer	Output Units Large	Output Units Small
7	2048	1024
8	2048	1024
9	Depends on the problem	

Zhang's CNN

Datasets

Dataset	Classes	Train Samples	Test Samples	Epoch Size
AG's News	4	120,000	7,600	5,000
Sogou News	5	450,000	60,000	5,000
DBPedia	14	560,000	70,000	5,000
Yelp Review Polarity	2	560,000	38,000	5,000
Yelp Review Full	5	650,000	50,000	5,000
Yahoo! Answers	10	1,400,000	60,000	10,000
Amazon Review Full	5	3,000,000	650,000	30,000
Amazon Review Polarity	2	3,600,000	400,000	30,000

Zhang's CNN

Results (error)

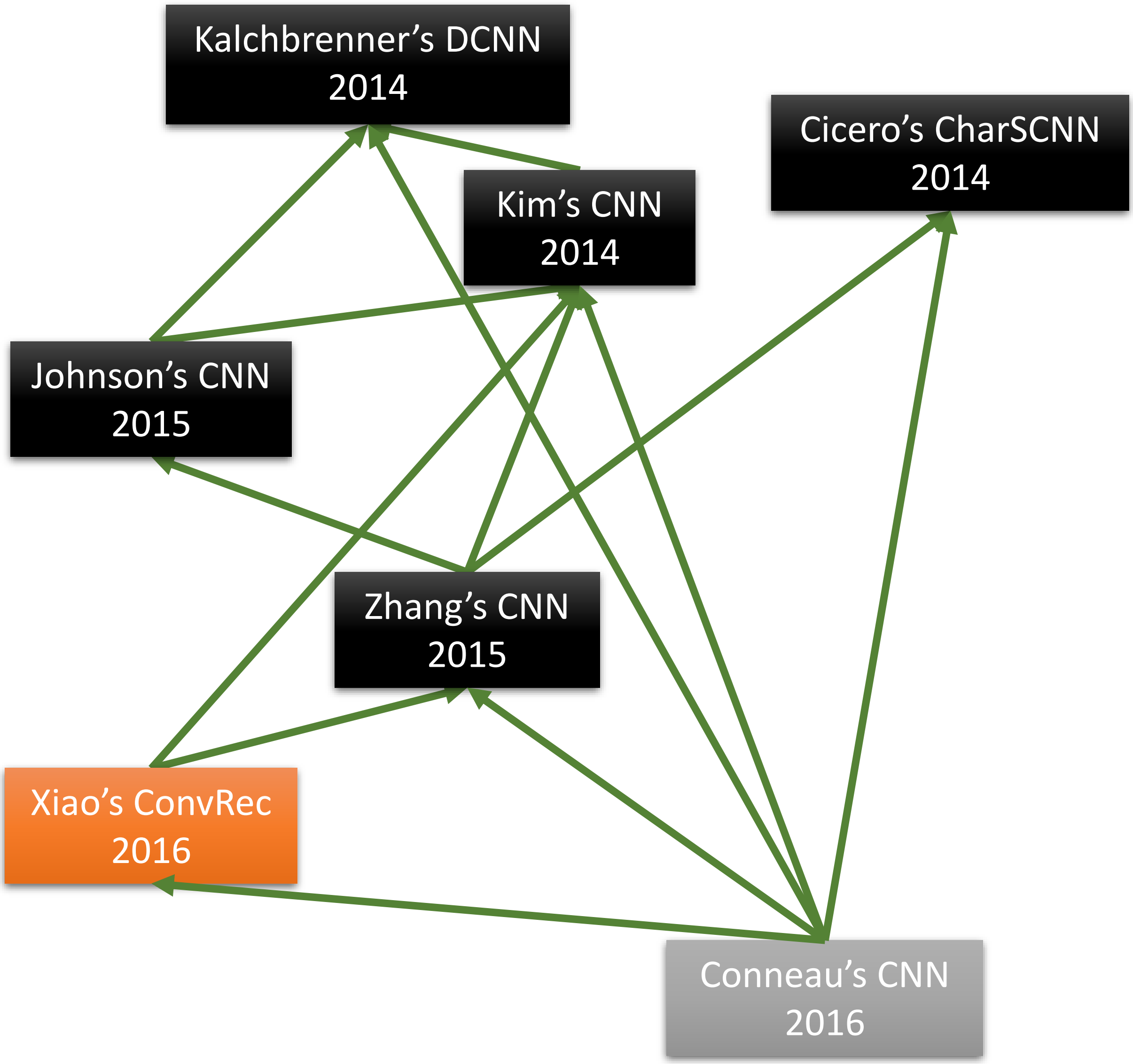
Model	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW	11.19	7.15	3.39	7.76	42.01	31.11	45.36	9.60
BoW TFIDF	10.36	6.55	2.63	6.34	40.14	28.96	44.74	9.00
ngrams	7.96	2.92	1.37	**4.36**	43.74	31.53	45.73	7.98
ngrams TFIDF	**7.64**	**2.81**	**1.31**	4.56	45.20	31.49	47.56	8.46
Bag-of-means	**16.91**	**10.79**	**9.55**	**12.67**	**47.46**	**39.45**	**55.87**	**18.39**
LSTM	13.94	4.82	1.45	5.26	41.83	29.16	40.57	6.10
Lg. w2v Conv.	9.92	4.39	1.42	4.60	40.16	31.97	44.40	5.88
Sm. w2v Conv.	11.35	4.54	1.71	5.56	42.13	31.50	42.59	6.00
Lg. w2v Conv. Th.	9.91	-	1.37	4.63	39.58	31.23	43.75	5.80
Sm. w2v Conv. Th.	10.88	-	1.53	5.36	41.09	29.86	42.50	5.63
Lg. Lk. Conv.	8.55	4.95	1.72	4.89	40.52	29.06	45.95	5.84
Sm. Lk. Conv.	10.87	4.93	1.85	5.54	41.41	30.02	43.66	5.85
Lg. Lk. Conv. Th.	8.93	-	1.58	5.03	40.52	28.84	42.39	5.52
Sm. Lk. Conv. Th.	9.12	-	1.77	5.37	41.17	28.92	43.19	5.51
Lg. Full Conv.	9.85	8.80	1.66	5.25	38.40	29.90	40.89	5.78
Sm. Full Conv.	11.59	8.95	1.89	5.67	38.82	30.01	40.88	5.78
Lg. Full Conv. Th.	9.51	-	1.55	4.88	38.04	29.58	40.54	5.51
Sm. Full Conv. Th.	10.89	-	1.69	5.42	**37.95**	29.90	40.53	5.66
Lg. Conv.	12.82	4.88	1.73	5.89	39.62	29.55	41.31	5.51
Sm. Conv.	15.65	8.65	1.98	6.53	40.84	29.84	40.53	5.50
Lg. Conv. Th.	13.39	-	1.60	5.82	39.30	**28.80**	40.45	**4.93**
Sm. Conv. Th.	14.80	-	1.85	6.49	40.16	29.84	**40.43**	5.67

Zhang's CNN

Summary

Network type	CNN
# of layers	12 (6 con + 3 pool + 3 fc)
Word or character level	character level
Embedding	None
# of embedding dimension	N.A.
Padding	yes
# of feature maps	1024 / 256
Size of window	7, 3
Pooling type	max (window size 3)
Non-linear function	ReLU
Regularization	dropout data augmentation (synonym replacement)

Paper roadmap



Xiao's ConvRec

Efficient Character-level Document Classification by Combining Convolutional and Recurrent Layers

Yijun Xiao, Kyunghyun Cho
arXiv, 2016

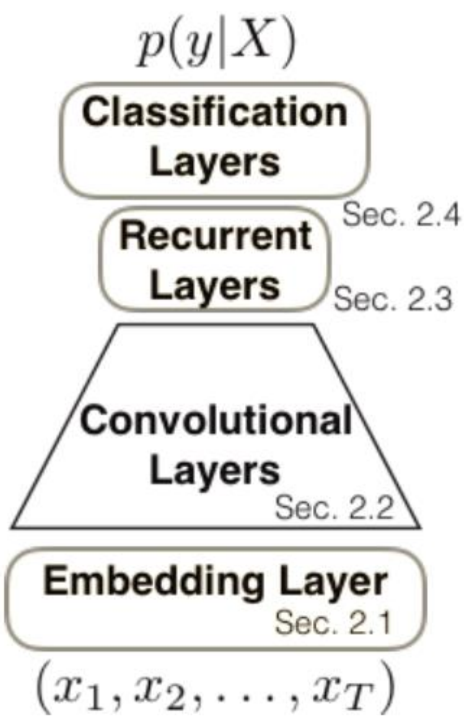
Xiao's ConvRec

What is new

- **CNN + RNN**
  - RNN can efficiently capture long-term dependencies
  - RNN can reduce the number of convolutional layers

Xiao's ConvRec

Model Architecture



$$H_{\text{forward}} = (\overrightarrow{\mathbf{h}}_1, \overrightarrow{\mathbf{h}}_2, \ldots, \overrightarrow{\mathbf{h}}_{T'})$$
$$H_{\text{reverse}} = (\overleftarrow{\mathbf{h}}_1, \overleftarrow{\mathbf{h}}_2, \ldots, \overleftarrow{\mathbf{h}}_{T'}).$$
$$\mathbf{h} = \left[\overrightarrow{\mathbf{h}}_{T'}; \overleftarrow{\mathbf{h}}_1\right]$$

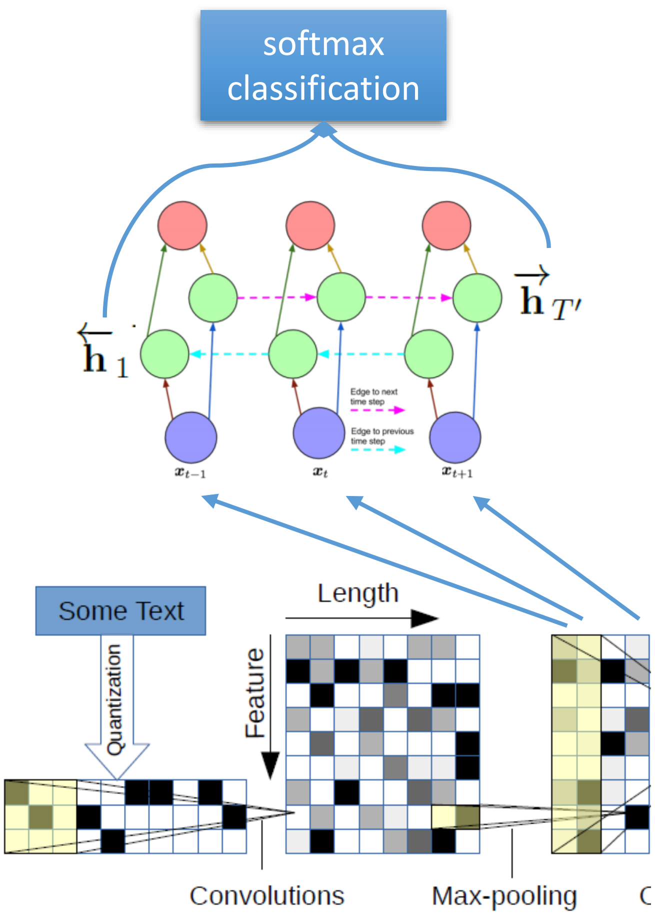
$$F = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_{T'})$$

$$E = (\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_T)$$

$$X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$$

Xiao's ConvRec

Model Architecture



$$H_{\text{forward}} = (\overrightarrow{\mathbf{h}}_1, \overrightarrow{\mathbf{h}}_2, \ldots, \overrightarrow{\mathbf{h}}_{T'})$$
$$H_{\text{reverse}} = (\overleftarrow{\mathbf{h}}_1, \overleftarrow{\mathbf{h}}_2, \ldots, \overleftarrow{\mathbf{h}}_{T'}).$$

$$\mathbf{h} = \left[\overrightarrow{\mathbf{h}}_{T'}; \overleftarrow{\mathbf{h}}_1\right]$$

$$F = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_{T'})$$

$$E = (\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_T)$$

$$X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$$

Xiao's ConvRec

Model Details

Model	Embedding Layer Sec. 2.1		Convolutional Layer Sec. 2.2				Recurrent Layer Sec. 2.3
	$\|V\|$	d	d'	r	r'	ϕ	d'
C2R1DD	96	8	D	5,3	2,2	ReLU	D
C3R1DD				5,5,3	2,2,2		
C4R1DD				5,5,3,3	2,2,2,2		
C5R1DD				5,5,3,3,3	2,2,2,1,2		

Xiao's ConvRec

Results (error)

Data set	# Ex.	# Cl.	Our Model: Network	Our Model: # Params	Our Model: Error (%)	(Zhang et al., 2015): Network	(Zhang et al., 2015): # Params	(Zhang et al., 2015): Error (%)
AG	120k	4	C2R1D1024	20M	8.39/**8.64**	C6F2D1024	27M	-/9.85
Sogou	450k	5	C3R1D128	.4M	4.82/**4.83**	C6F2D1024*	27M	-/4.88
DBPedia	560k	14	C2R1D128	.3M	1.46/**1.43**	C6F2D1024	27M	-/1.66
Yelp P.	560k	2	C2R1D128	.3M	5.50/5.51	C6F2D1024	27M	-/**5.25**
Yelp F.	650k	5	C2R1D128	.3M	38.00/**38.18**	C6F2D1024	27M	-/38.40
Yahoo A.	1.4M	10	C2R1D1024	20M	28.62/**28.26**	C6F2D1024*	27M	-/29.55
Amazon P.	3.6M	2	C3R1D128	.4M	5.64/5.87	C6F2D256*	2.7M	-/**5.50**
Amazon F.	3.0M	5	C3R1D128	.4M	40.30/40.77	C6F2D256*	2.7M	-/**40.53**

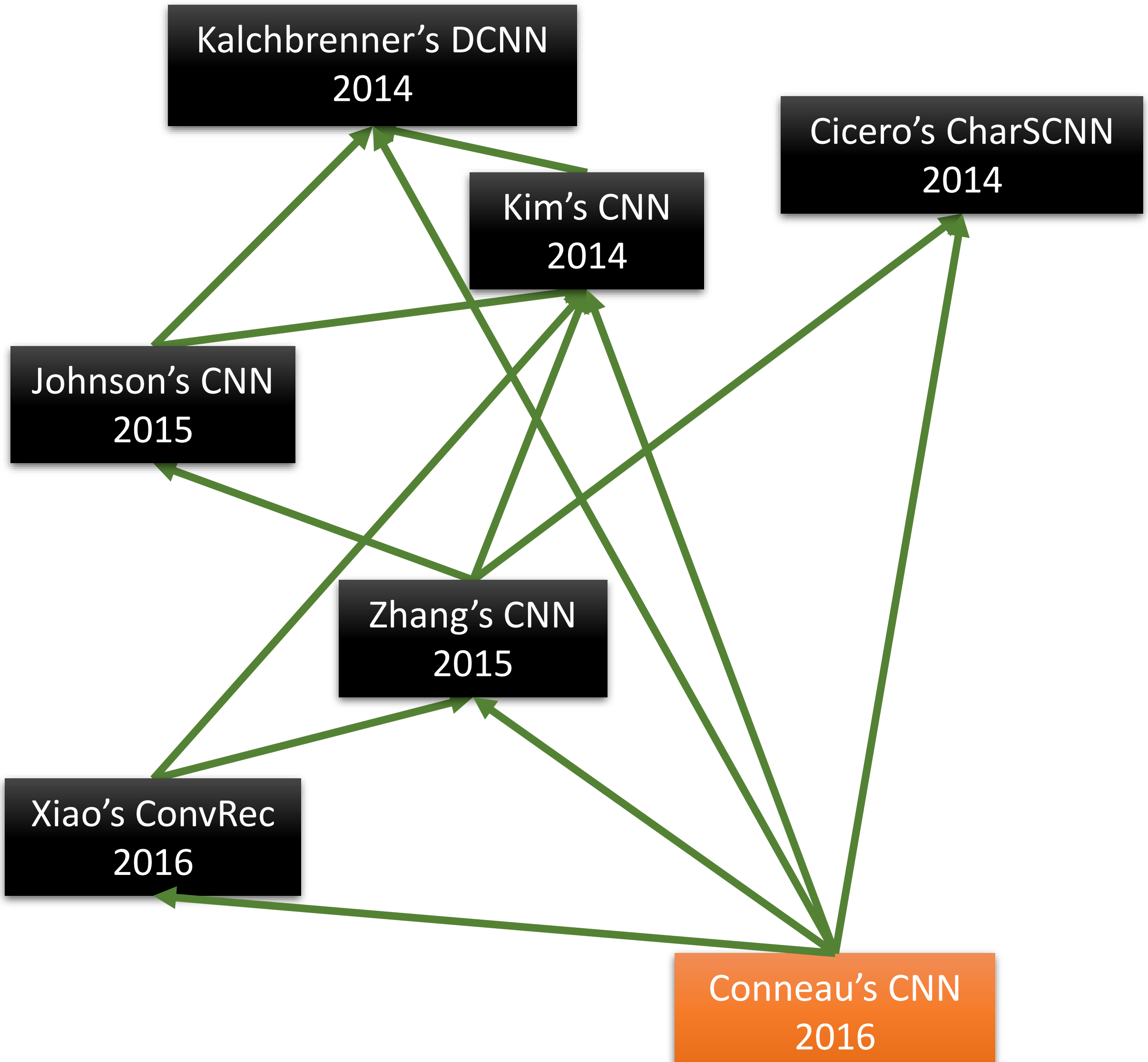
Table 3: Results on character-level document classification. $CCRRFFDD$ refers to a network with C convolutional layers, R recurrent layers, F fully-connected layers and D dimensional feature vectors. $\star$ denotes a model which does not distinguish between lower-case and upper-case letters. We only considered the character-level models without using Thesaraus-based data augmentation. We report both the validation and test errors. In our case, the network architecture for each dataset was selected based on the validation errors. The numbers of parameters are approximate.

Xiao's ConvRec

Summary

Network type	CNN + RNN
# of layers	4-12 (2-5 con + 2-5 pool + 1 rec + 1 fc)
Word or character level	character level
Embedding	/
# of embedding dimension	8
Padding	yes
# of feature maps	128, 1024
Size of window	5, 3
Pooling type	max (window size 2, 1)
Non-linear function	ReLU
Regularization	dropout L_2 early stopping

Paper roadmap



Conneau's CNN

Very Deep Convolutional Networks for Natural Language Processing

Alexis Conneau, Holger Schwenk, Yan LeCun, Loic Barreau
arXiv, 2016

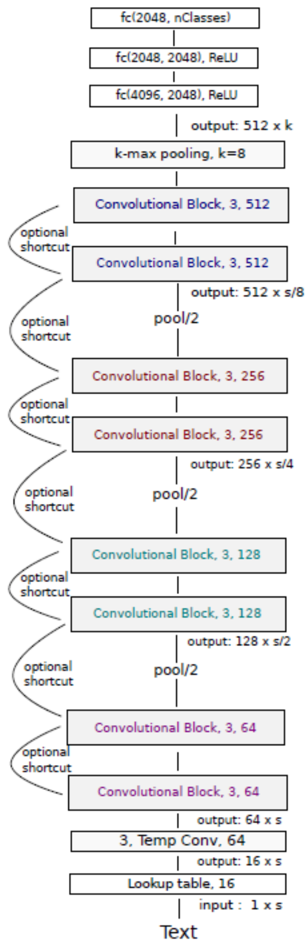
Conneau's CNN

What is new

- **Very deep …**
  - No one uses more than 6 convolutional layers before for sentence classification
  - Significant improvement have been reported using much deeper networks in computer vision

Conneau's CNN

Model Architecture



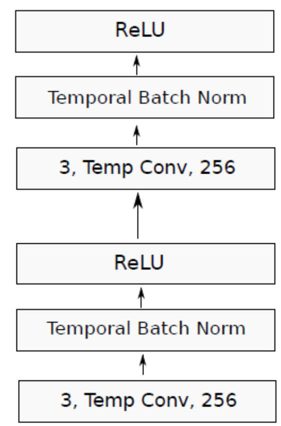


Figure 2: Detailed architecture of a convolutional block.

Conneau's CNN

Model Details

Table 1: Number of convolutional layers for each depth.

Depth:	9	17	29	49
conv block 512	2	4	4	6
conv block 256	2	4	4	10
conv block 128	2	4	10	16
conv block 64	2	4	10	16
First conv. layer	1	1	1	1
#params [in M]	2.2	4.3	4.6	7.8

Conneau's CNN

Results (error)

Best results from previous work

Corpus:	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
Method	n-TFIDF	n-TFIDF	n-TFIDF	ngrams	Conv	Conv+RNN	Conv	Conv
Author	[Zhang]	[Zhang]	[Zhang]	[Zhang]	[Zhang]	[Xiao]	[Zhang]	[Zhang]
Error	7.64	2.81	1.31	4.36	37.95*	28.26	40.43*	4.93*

Very deep CNN

Depth	Pooling	AG	Sogou	DBP.	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
9	Convolution	10.17	4.22	1.64	5.01	37.63	28.10	38.52	4.94
9	KMaxPooling	9.83	3.58	1.56	5.27	38.04	28.24	39.19	5.69
9	MaxPooling	9.17	3.70	1.35	4.88	36.73	27.60	37.95	4.70
17	Convolution	9.29	3.94	1.42	4.96	36.10	27.35	37.50	4.53
17	KMaxPooling	9.39	3.51	1.61	5.05	37.41	28.25	38.81	5.43
17	MaxPooling	8.88	3.54	1.40	4.50	36.07	27.51	37.39	4.41
29	Convolution	9.36	3.61	1.36	4.35	**35.28**	27.17	37.58	**4.28**
29	KMaxPooling	**8.67**	**3.18**	1.41	4.63	37.00	27.16	38.39	4.94
29	MaxPooling	8.73	3.36	**1.29**	**4.28**	35.74	**26.57**	**37.00**	4.31

Conneau's CNN

Results (error) cont.

Effect of shortcut

	without shortcut	with shortcut
depth 9	33.08 / 37.63	33.51 / 40.27
depth 17	30.85 / 36.10	35.80 / 39.18
depth 29	29.57 / 35.28	30.59 / 36.01
depth 49	35.54 / 37.41	32.28 / 36.15

Inspired by the best paper of CVPR 2016:

He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[J]. arXiv preprint arXiv:1512.03385, 2015.

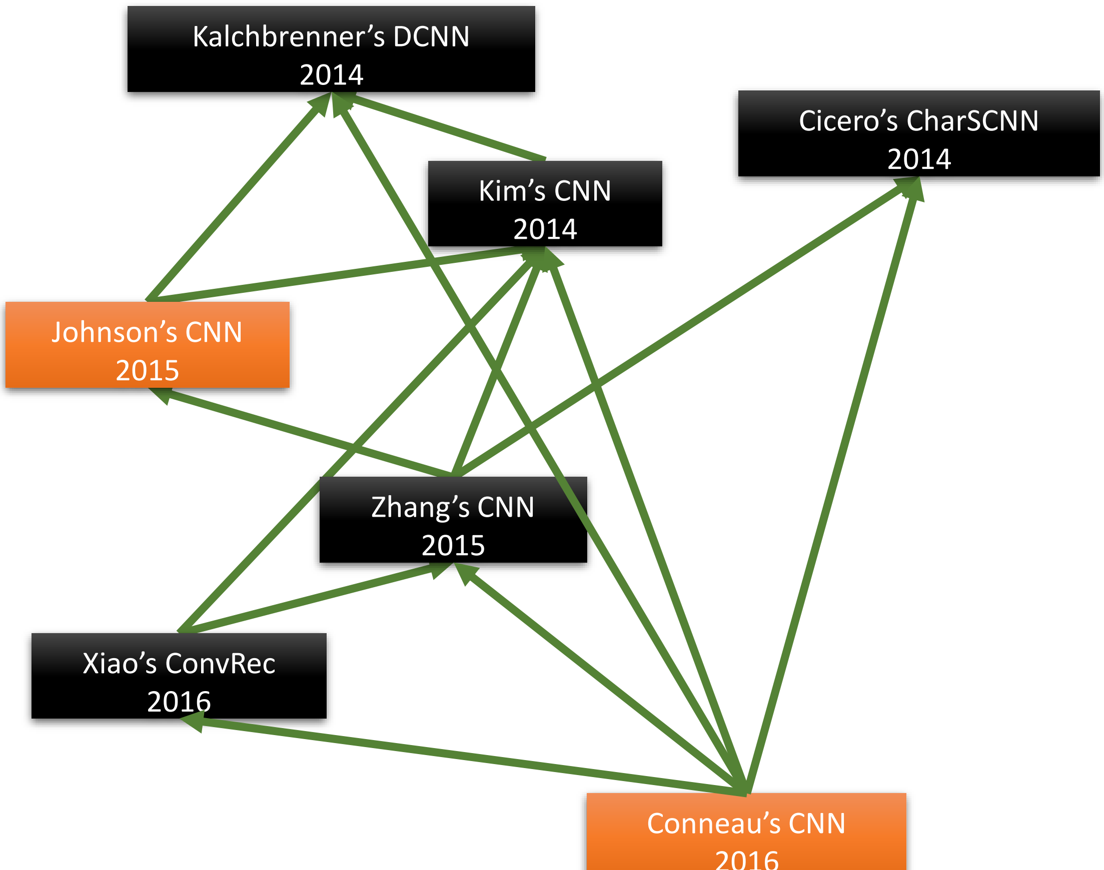
Conneau's CNN

Summary

Network type	CNN
# of layers	16-56 (9-49 con + 4 pool + 3 fc)
Word or character level	character level
Embedding	/
# of embedding dimension	16
Padding	yes
# of feature maps	64, 128, 256, 512
Size of window	3
Pooling type	max (halve or *k*-max)
Non-linear function	ReLU
Regularization	short cut temporal batch norm

Johnson vs. Conneau

What's more …



Johnson vs. Conneau

Word-level vs. char-level

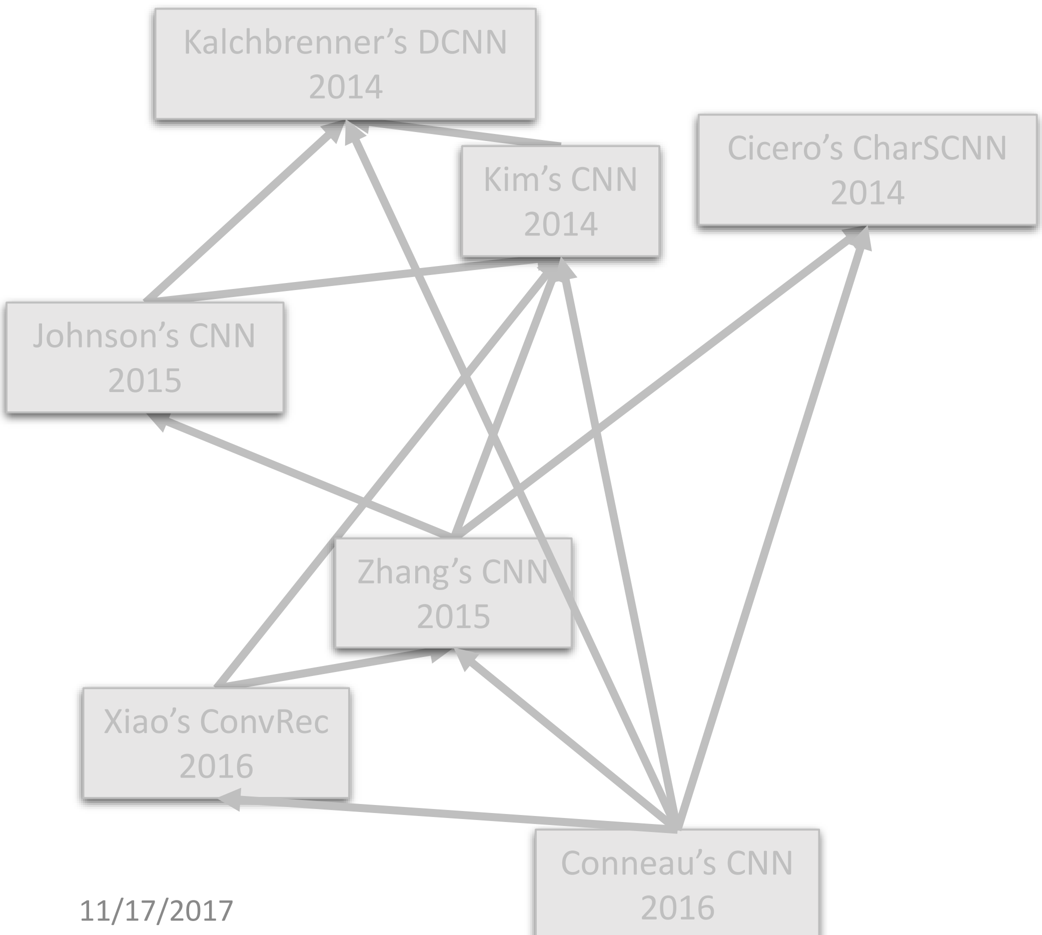
Error rate

Models	depth	AG	Sogou	Dbpedia	Yelp.p	Yelp.f	Yahoo	Ama.f	Ama.p
Linear model *best* [9]	0	7.64	2.81	1.31	4.36	40.14	28.96	44.74	7.98
char-CNN *best* [1]	9+2	9.17	3.58	1.35	4.88	36.73	27.60	37.95	4.70
	29+2	8.67	3.18	1.29	4.28	35.28	26.57	*37.00*	4.28
word-CNN w/o tv-embed.	1	*6.95*	*2.21*	*1.12*	*3.44*	*34.21*	*26.06*	37.51	*4.27*
word-CNN w/ tv (300-dim)	2	**6.57**	**1.89**	**0.84**	**2.90**	**32.39**	**24.85**	**36.24**	**3.79**

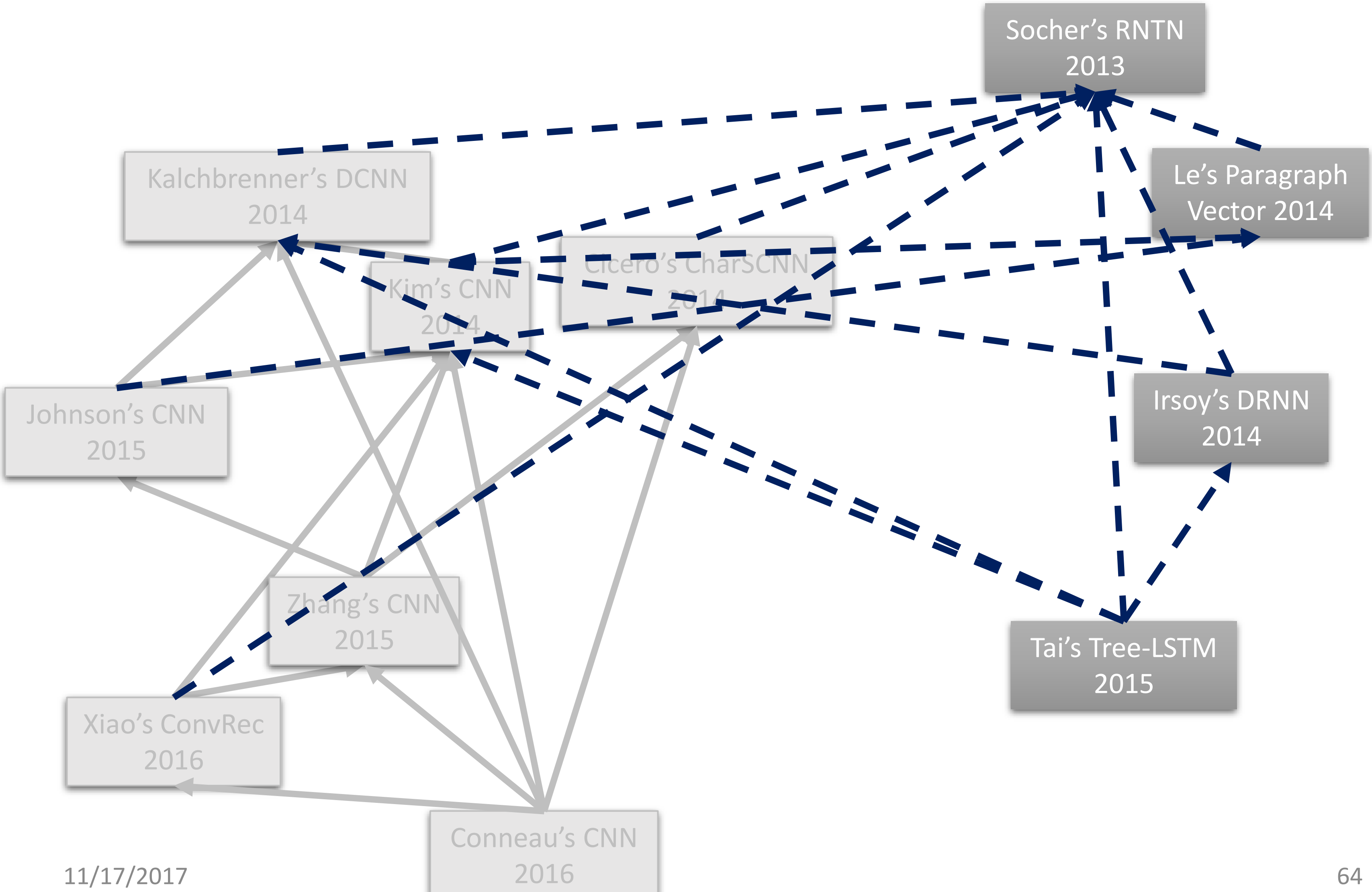
Model size & computation time

		Depth	Dimensionality of layer outputs (#layers)	#param	Time	Error rate(%)
char-CNN [1]		9+2	16(1), 64(3), 128(2), 256(2), 512(2), 2048(2)	2.2M	†215	36.73
		29+2	16(1), 64(11), 128(10), 256(4), 512(4), 2048(2)	4.6M	‡700	35.28
word-CNN	w/o tv-embed.	1	500(1)	45M	6	34.21
	w/ 2 tv (100-dim)	2	100(2), 500(1)	68M	21	32.77
	w/ 4 tv (100-dim)	2	100(4), 500(1)	91M	36	32.55
	w/ 4 tv (300-dim)	2	300(4), 500(1)	184M	72	32.39

Paper roadmap



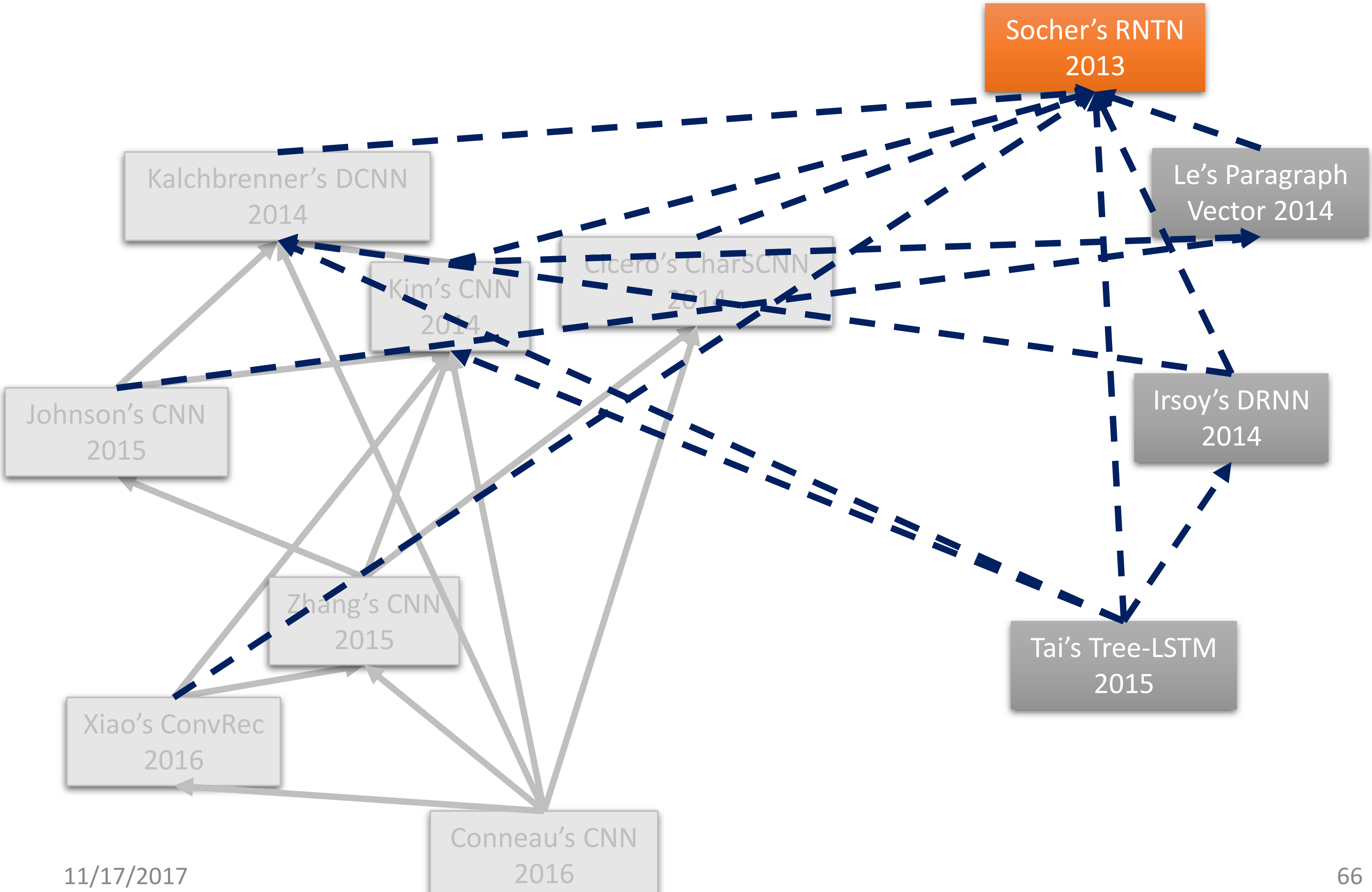
Paper roadmap



Deep Learning in NLP

- **Feedforward Neural Network**
  - The most basic form of NN
- **Convolutional Neural Network (CNN)**
  - Quite successful in computer vision
  - Extract local features
  - Most popular and effective in sentence classification
- **Recursive Neural Network**
  - Rely on parser tree of the sentence
- **Recurrent Neural Network (RNN)**
  - Designed for sequential data
  - The most popular version: **Long Short-Term Memory (LSTM)**
  - Further variants: Bidirectional-LSTM, Deep-LSTM …

Paper roadmap



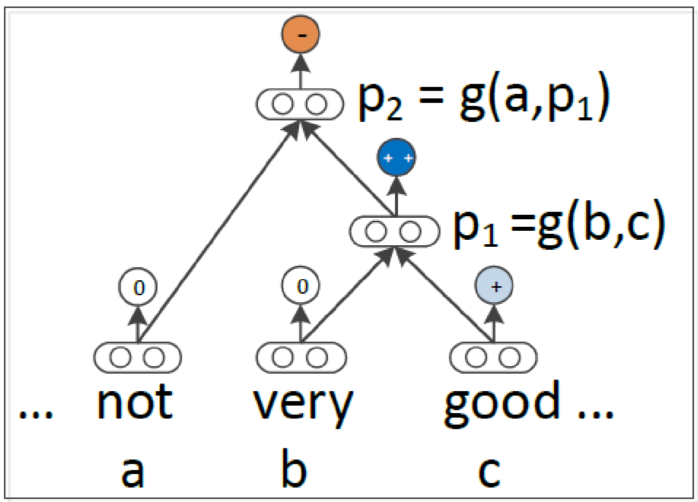
Socher's RNTN

Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, Christopher Potts
EMNLP, 2013

Socher's RNTN

Model Architecture



Compute the posterior probability over labels given the word vector *a* via:

$$y^a = \mathrm{softmax}(W_s a)$$

Recursive Neural Network

$$p_1 = f\left(W\begin{bmatrix} b \\ c \end{bmatrix}\right), p_2 = f\left(W\begin{bmatrix} a \\ p_1 \end{bmatrix}\right)$$

RNTN (Recursive Neural Tensor Network)

$$p_1 = f\left(\begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]}\begin{bmatrix} b \\ c \end{bmatrix} + W\begin{bmatrix} b \\ c \end{bmatrix}\right)$$

$$p_2 = f\left(\begin{bmatrix} a \\ p_1 \end{bmatrix}^T V^{[1:d]}\begin{bmatrix} a \\ p_1 \end{bmatrix} + W\begin{bmatrix} a \\ p_1 \end{bmatrix}\right)$$

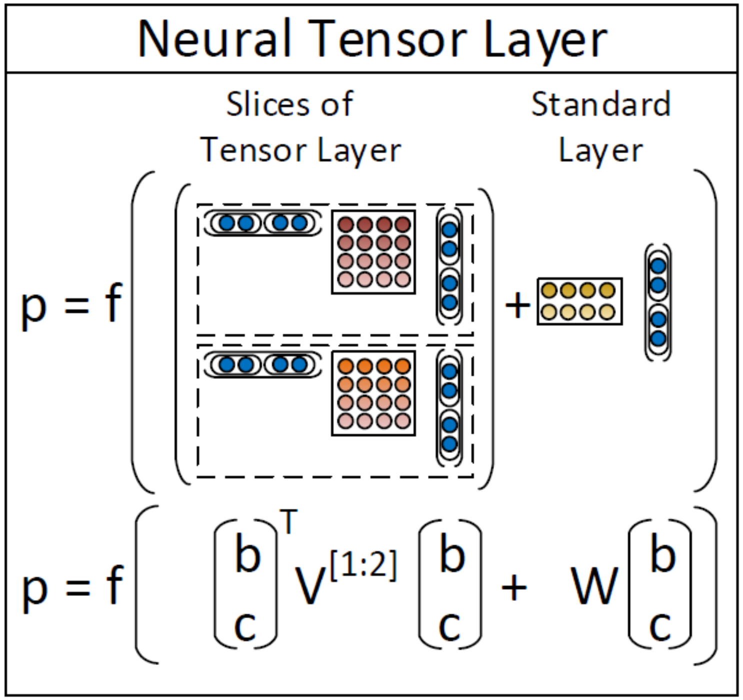
$$h = \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]}\begin{bmatrix} b \\ c \end{bmatrix}; h_i = \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[i]}\begin{bmatrix} b \\ c \end{bmatrix}$$

Socher's RNTN

Model Architecture

RNTN (Recursive Neural Tensor Network)

$$p_1 = f\left(\begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix} + W \begin{bmatrix} b \\ c \end{bmatrix}\right)$$



Socher's RNTN

Results (accuracy)

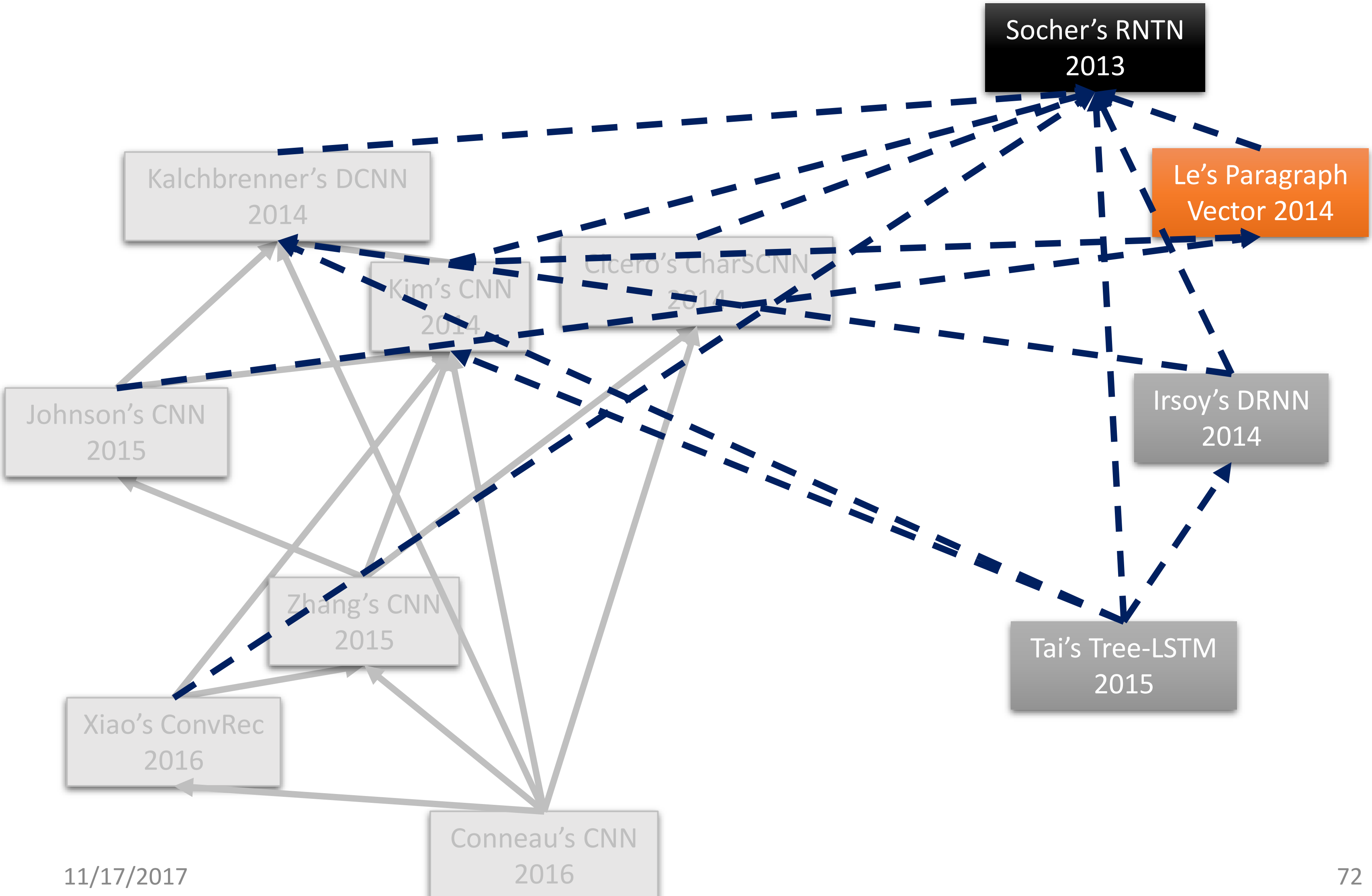
Model	Fine-grained		Positive/Negative	
	All	Root	All	Root
NB	67.2	41.0	82.6	81.8
SVM	64.3	40.7	84.6	79.4
BiNB	71.0	41.9	82.7	83.1
VecAvg	73.3	32.7	85.1	80.1
RNN	79.0	43.2	86.1	82.4
MV-RNN	78.7	44.4	86.8	82.9
RNTN	**80.7**	**45.7**	**87.6**	**85.4**

Socher's RNTN

Summary

Network type	Recursive NN
Word or character level	word level
Embedding	self learned
# of embedding dimension	25-35
# of features	25-35
Non-linear function	tanh
Regularization	L_2

Paper roadmap



Le's Paragraph Vector

Distibuted Representation of Sentences and Documents

Quoc Le, Tomas Mikolov
ICML, 2014

Le's Paragraph Vector

Model architecture

- **An unsupervised algorithm that learns representation for piece of texts**
  - Sentences, paragraphs, documents …
  - Word embedding + paragraph vector

- **Learned by predicting a word given the other words in a context**

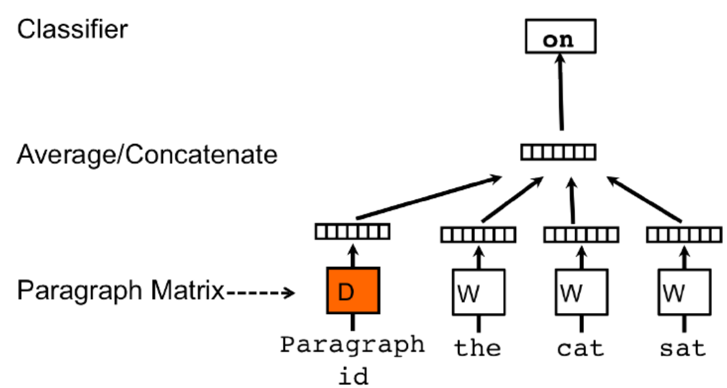
Maximize the average log probability:

$$\frac{1}{T}\sum_{t=k}^{T-k}\log p(w_t|w_{t-k},\dots,w_{t+k})$$

where

$$p(w_t|w_{t-k},\dots,w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

$$y = b + Uh(w_{t-k},\dots,w_{t+k},D;W)$$



Le's Paragraph Vector

Results (error)

- Trained by gradient descent
- Apply logistic regression after learning representations
- Dataset: SST

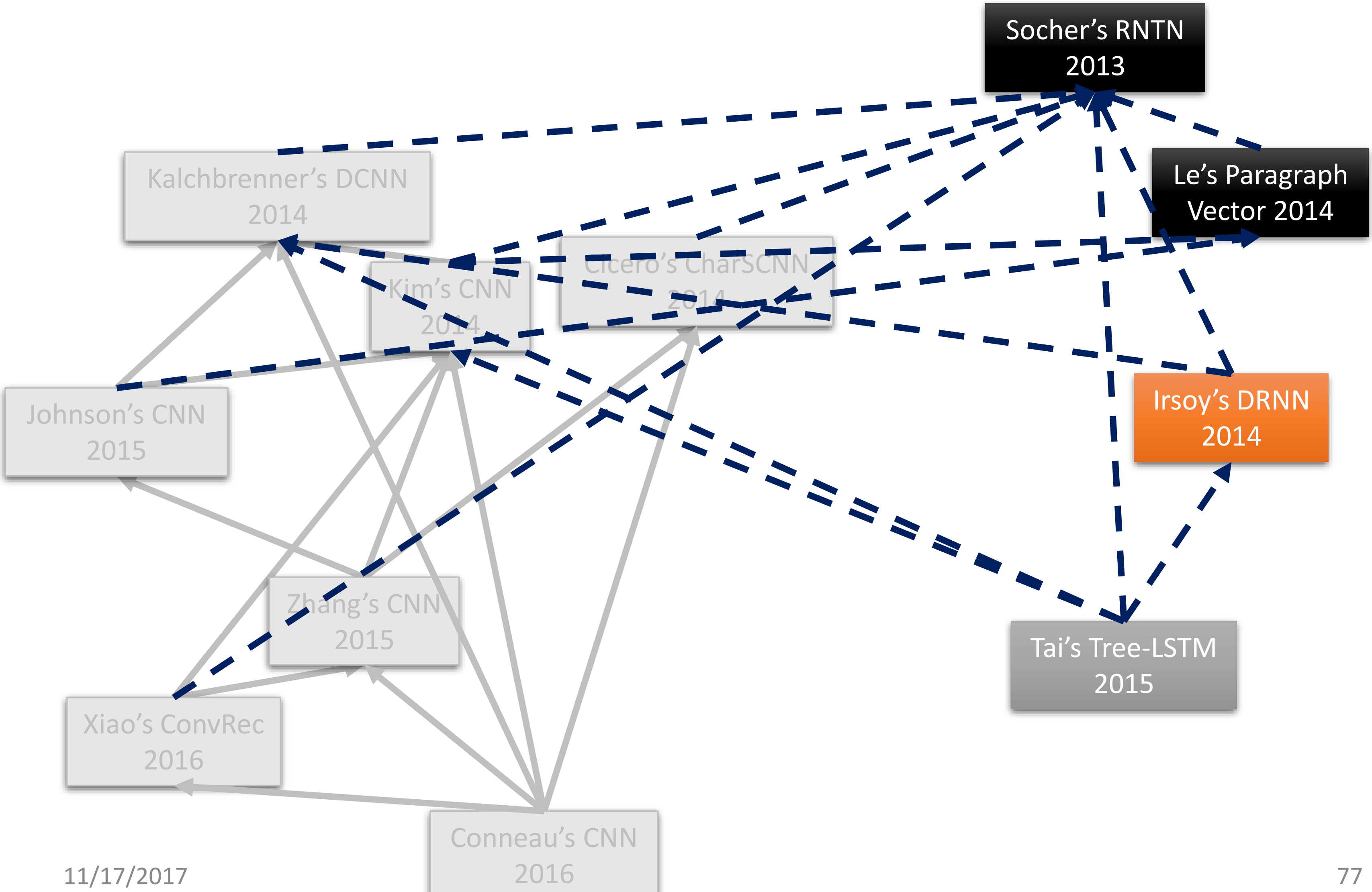
Model	Error rate (Positive/ Negative)	Error rate (Fine-grained)
Naïve Bayes (Socher et al., 2013b)	18.2 %	59.0%
SVMs (Socher et al., 2013b)	20.6%	59.3%
Bigram Naïve Bayes (Socher et al., 2013b)	16.9%	58.1%
Word Vector Averaging (Socher et al., 2013b)	19.9%	67.3%
Recursive Neural Network (Socher et al., 2013b)	17.6%	56.8%
Matrix Vector-RNN (Socher et al., 2013b)	17.1%	55.6%
Recursive Neural Tensor Network (Socher et al., 2013b)	14.6%	54.3%
Paragraph Vector	**12.2%**	**51.3%**

Le's Paragraph Vector

Summary

Network type	feedforward NN
Word or character level	word level
Embedding	self learned
# of embedding dimension	400
# of features	N.A.
Non-linear function	/
Regularization	/

Paper roadmap



Irsoy's DRNN

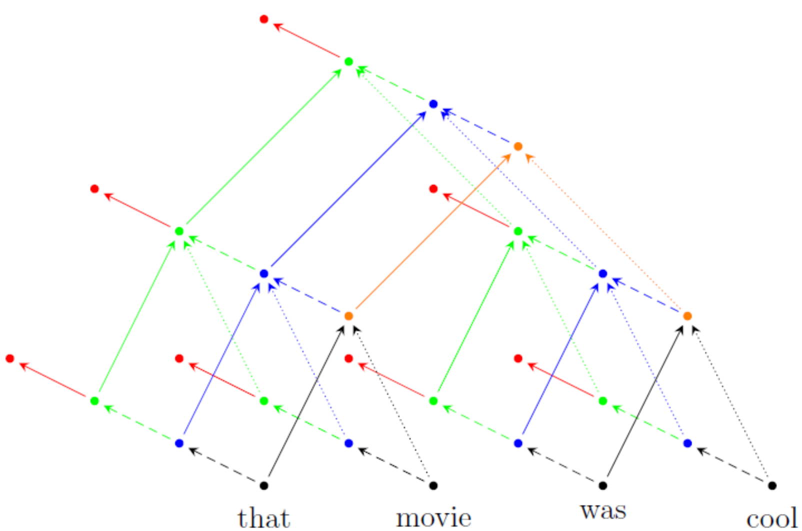
Deep Recursive Neural Networks for Compositionality in Language

Ozan Irsoy, Claire Cardie
NIPS, 2014

Irsoy's DRNN

Model architecture (cont.)

- **Not only deep in structure, but also deep in space**
  - Capture hierarchy representations
- **Untie the transforming matrix between leaves and internal nodes**



To compute the embedding vector of a node, previously we did this:

$$h_\eta = f(W_L^{l(\eta)} h_{l(\eta)} + W_R^{r(\eta)} h_{r(\eta)} + b)$$

Now:

$$h_\eta^{(i)} = f(W_L^{(i)} h_{l(\eta)}^{(i)} + W_R^{(i)} h_{r(\eta)}^{(i)} + V^{(i)} h_\eta^{(i-1)} + b^{(i)})$$

Irsoy's DRNN

Results (error)

- Dataset: SST

ℓ	$\|h\|$	Fine-grained	Binary
1	50	46.1	85.3
2	45	48.0	85.5
3	40	43.1	83.5
1	340	48.1	86.4
2	242	48.3	86.4
3	200	49.5	**86.7**
4	174	**49.8**	86.6
5	157	49.0	85.5

(a) Results for RNNs. ℓ and $|h|$ denote the depth and width of the networks, respectively.

Method	Fine-grained	Binary
Bigram NB	41.9	83.1
RNN	43.2	82.4
MV-RNN	44.4	82.9
RNTN	45.7	85.4
DCNN	48.5	86.8
Paragraph Vectors	48.7	**87.8**
DRNN (4, 174)	**49.8**	86.6

(b) Results for previous work and our best model (DRNN).

Irsoy's DRNN

Discussion

		not great	
1	as great	nothing good	not very informative
2	a great	not compelling	not really funny
3	is great	only good	not quite satisfying
4	Is n't it great	too great	thrashy fun
5	be great	completely numbing experience	fake fun

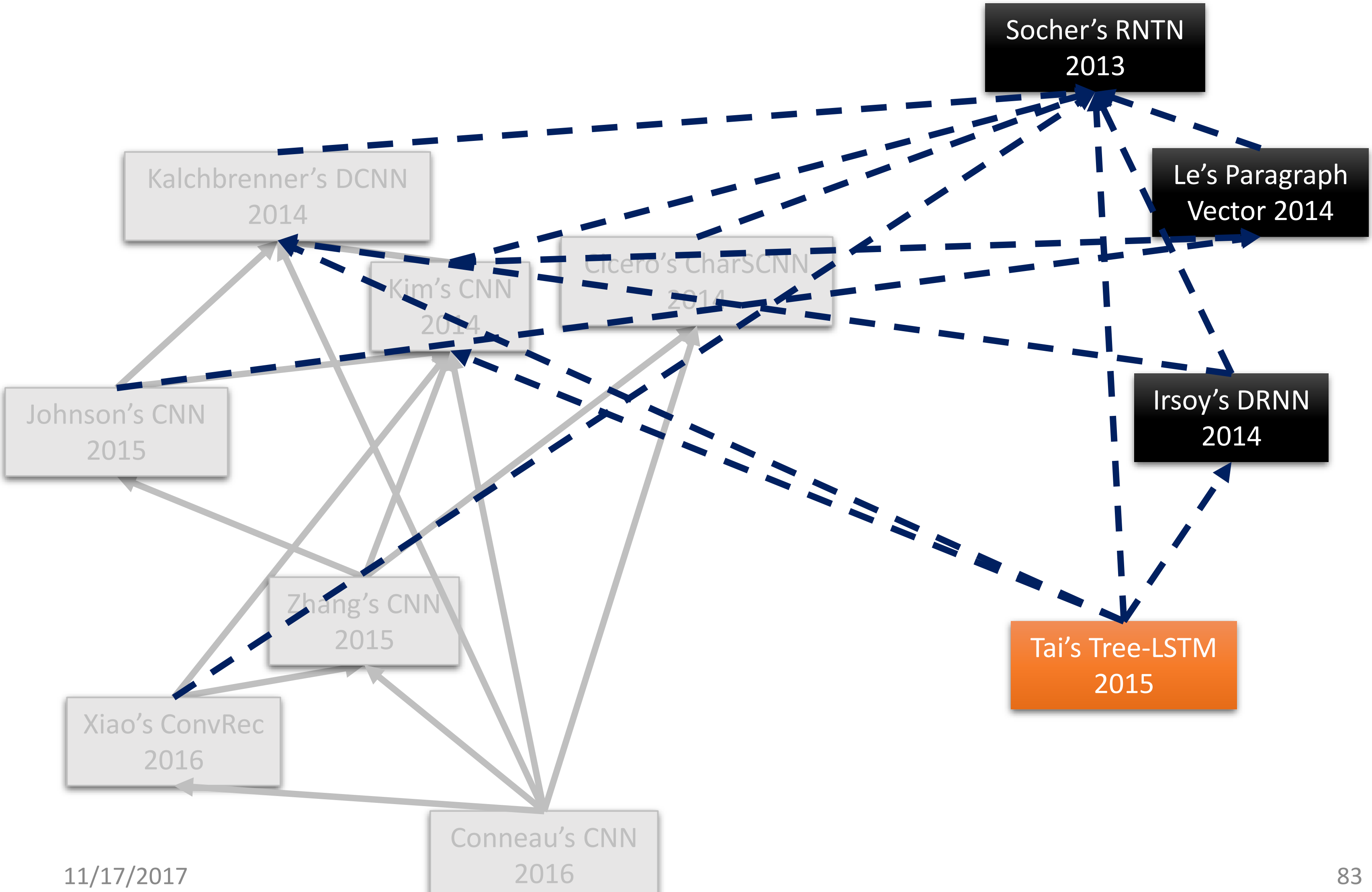
Table 2: Example shortest phrases and their nearest neighbors across three layers.

Irsoy's DRNN

Summary

Network type	Recursive NN
Word or character level	word level
Embedding	word2vec
# of embedding dimension	300
# of features	/
Non-linear function	ReLU
Regularization	Dropout L_2

Paper roadmap



Tai’s Tree-LSTM

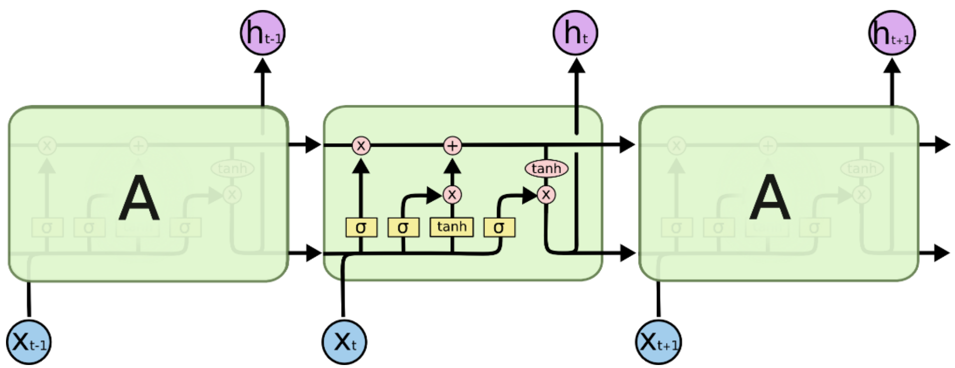
Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks

Kai Sheng Tai, Richard Socher, Christopher D. Manning
CoRR, 2015

Tai's Tree-LSTM

LSTM

- Long Short-Term Memory



$$i_t = \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}\right),$$
$$f_t = \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}\right),$$
$$o_t = \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}\right),$$
$$u_t = \tanh\left(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}\right),$$
$$c_t = i_t \odot u_t + f_t \odot c_{t-1},$$
$$h_t = o_t \odot \tanh(c_t),$$

Tai's Tree-LSTM

Model architecture

- **Apply LSTM to tree-structured network topologies**

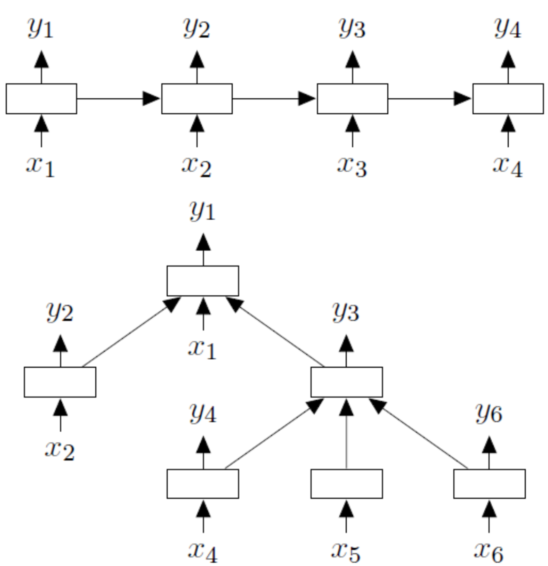


Figure 1: **Top:** A chain-structured LSTM network. **Bottom:** A tree-structured LSTM network with arbitrary branching factor.

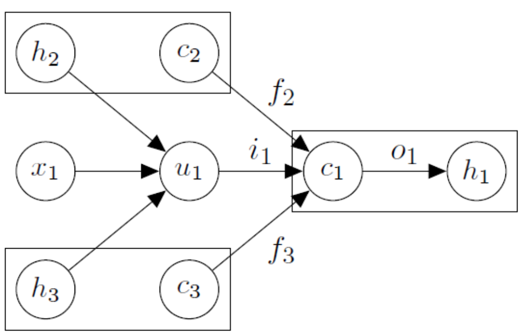


Figure 2: Composing the memory cell c_1 and hidden state h_1 of a Tree-LSTM unit with two children (subscripts 2 and 3). Labeled edges correspond to gating by the indicated gating vector, with dependencies omitted for compactness.

Tai's Tree-LSTM

Model architecture (cont.)

- **Dependency Tree-LSTMs**
  - Treat all children of a node as equivalent and unordered

- **Constituency Tree-LSTMs**
  - Treat all children of a node as distinct and ordered

$$\tilde{h}_j = \sum_{k \in C(j)} h_k, \tag{2}$$

$$i_j = \sigma\left(W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)}\right), \tag{3}$$

$$f_{jk} = \sigma\left(W^{(f)} x_j + U^{(f)} h_k + b^{(f)}\right), \tag{4}$$

$$o_j = \sigma\left(W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)}\right), \tag{5}$$

$$u_j = \tanh\left(W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)}\right), \tag{6}$$

$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k, \tag{7}$$

$$h_j = o_j \odot \tanh(c_j), \tag{8}$$

Dependency Tree-LSTMs

$$i_j = \sigma\left(W^{(i)} x_j + \sum_{\ell=1}^{N} U_\ell^{(i)} h_{j\ell} + b^{(i)}\right), \tag{9}$$

$$f_{jk} = \sigma\left(W^{(f)} x_j + \sum_{\ell=1}^{N} U_{k\ell}^{(f)} h_{j\ell} + b^{(f)}\right), \tag{10}$$

$$o_j = \sigma\left(W^{(o)} x_j + \sum_{\ell=1}^{N} U_\ell^{(o)} h_{j\ell} + b^{(o)}\right), \tag{11}$$

$$u_j = \tanh\left(W^{(u)} x_j + \sum_{\ell=1}^{N} U_\ell^{(u)} h_{j\ell} + b^{(u)}\right), \tag{12}$$

$$c_j = i_j \odot u_j + \sum_{\ell=1}^{N} f_{j\ell} \odot c_{j\ell}, \tag{13}$$

$$h_j = o_j \odot \tanh(c_j), \tag{14}$$

Constituency Tree-LSTMs

Tai's Tree-LSTM

Results (error)

- Dataset: SST

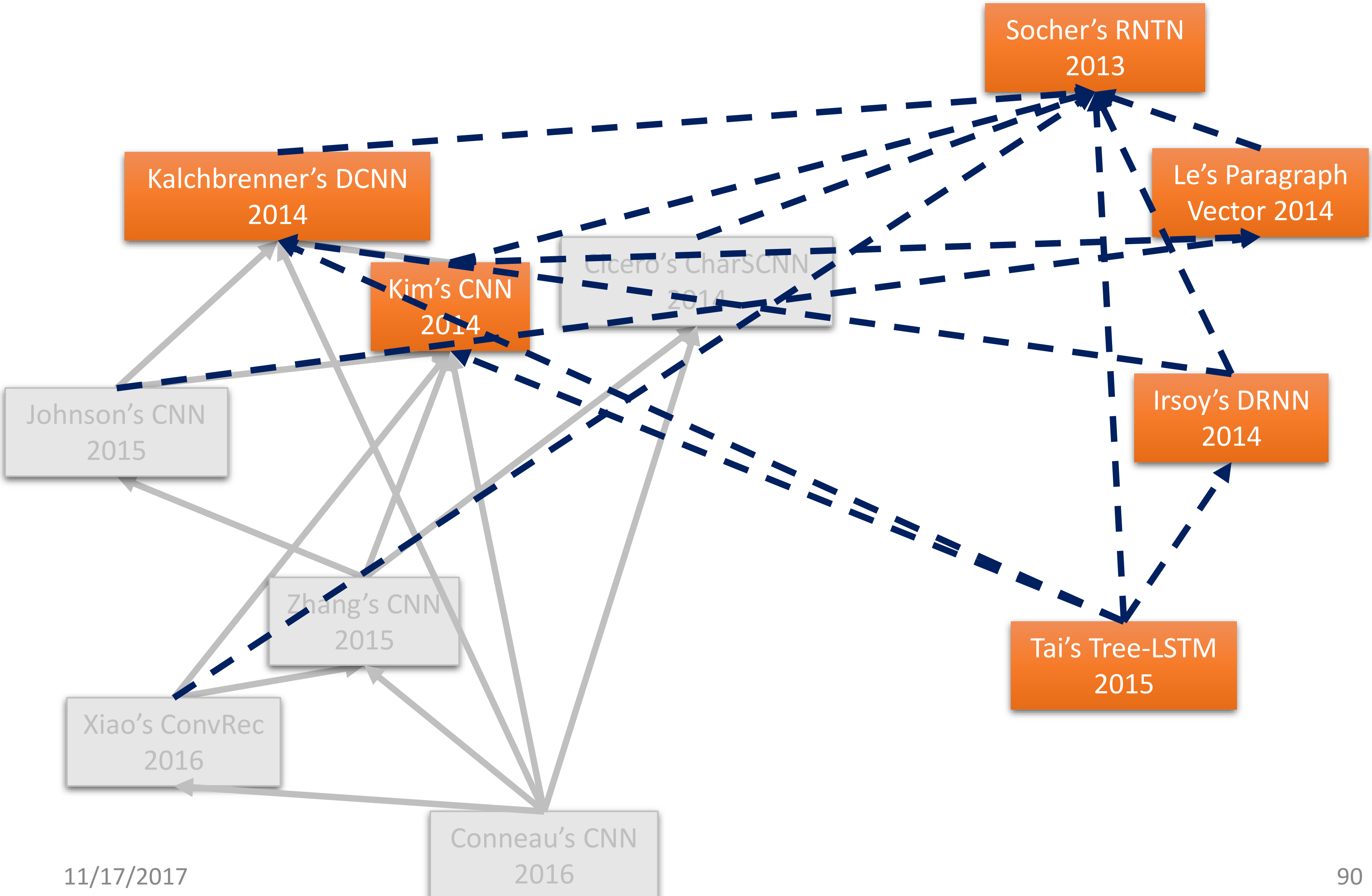
Method	Fine-grained	Binary
RAE (Socher et al., 2013)	43.2	82.4
MV-RNN (Socher et al., 2013)	44.4	82.9
RNTN (Socher et al., 2013)	45.7	85.4
DCNN (Blunsom et al., 2014)	48.5	86.8
Paragraph-Vec (Le and Mikolov, 2014)	48.7	87.8
CNN-non-static (Kim, 2014)	48.0	87.2
CNN-multichannel (Kim, 2014)	47.4	**88.1**
DRNN (Irsoy and Cardie, 2014)	49.8	86.6
LSTM	46.4 (1.1)	84.9 (0.6)
Bidirectional LSTM	49.1 (1.0)	87.5 (0.5)
2-layer LSTM	46.0 (1.3)	86.3 (0.6)
2-layer Bidirectional LSTM	48.5 (1.0)	87.2 (1.0)
Dependency Tree-LSTM	48.4 (0.4)	85.7 (0.4)
Constituency Tree-LSTM		
– randomly initialized vectors	43.9 (0.6)	82.0 (0.5)
– Glove vectors, fixed	49.7 (0.4)	87.5 (0.8)
– Glove vectors, tuned	**51.0** (0.5)	88.0 (0.3)

Tai's Tree-LSTM

Summary

Network type	RNN
Word or character level	word level
Embedding	Glove + fine-tuned
# of embedding dimension	300
# of features	120, 150, 168
Non-linear function	tanh
Regularization	Dropout L_2

Final PK on SST



Final PK on SST

Results (accuracy)

- **Dataset: SST**

Model	Fine (5-class)	Binary
DCNN (Blunsom, et al. 2014)	0.485	0.868
RNTN (Socher, et al. 2013)	0.457	0.854
CNN-non-static (Kim, 2014)	0.480	0.872
CNN-multi-channel (Kim, 2014)	0.474	0.881
DRNN w. pretrained word-embeddings (Irsoy and Cardie, 2014)	0.498	0.866
Paragraph Vector (Le and Mikolov. 2014)	0.487	0.878
Dependency Tree-LSTM (Tai, et al, 2015)	0.484	0.857
Constituency Tree-LSTM (Tai, et al, 2015)	0.439	0.820
Constituency Tree-LSTM (Glove vectors) (Tai, et al, 2015)	0.510	0.880
Paragraph Vector	0.391	0.798
LSTM	0.456	0.843
Deep Recursive-NN	0.469	0.847

Hong J, Fang M. Sentiment Analysis with Deeply Learned Distributed Representations of Variable Length Texts[J].

References

1. Kim Y. Convolutional neural networks for sentence classification[J]. arXiv preprint arXiv:1408.5882, 2014.
2. Kalchbrenner N, Grefenstette E, Blunsom P. A convolutional neural network for modelling sentences[J]. arXiv preprint arXiv:1404.2188, 2014.
3. dos Santos C N, Gatti M. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts[C]//COLING. 2014: 69-78.
4. Johnson R, Zhang T. Effective use of word order for text categorization with convolutional neural networks[J]. arXiv preprint arXiv:1412.1058, 2014.
5. Johnson R, Zhang T. Semi-supervised convolutional neural networks for text categorization via region embedding[C]//Advances in neural information processing systems. 2015: 919-927.
6. Zhang X, Zhao J, LeCun Y. Character-level convolutional networks for text classification[C]//Advances in Neural Information Processing Systems. 2015: 649-657.
7. Xiao Y, Cho K. Efficient Character-level Document Classification by Combining Convolution and Recurrent Layers[J]. arXiv preprint arXiv:1602.00367, 2016.
8. Conneau A, Schwenk H, Barrault L, et al. Very Deep Convolutional Networks for Natural Language Processing[J]. arXiv preprint arXiv:1606.01781, 2016.
9. Johnson R, Zhang T. Convolutional Neural Networks for Text Categorization: Shallow Word-level vs. Deep Character-level[J]. arXiv preprint arXiv:1609.00718, 2016.
10. Socher R, Perelygin A, Wu J Y, et al. Recursive deep models for semantic compositionality over a sentiment treebank[C]//Proceedings of the conference on empirical methods in natural language processing (EMNLP). 2013, 1631: 1642.
11. Le Q V, Mikolov T. Distributed Representations of Sentences and Documents[C]//ICML. 2014, 14: 1188-1196.
12. Irsoy O, Cardie C. Deep recursive neural networks for compositionality in language[C]//Advances in Neural Information Processing Systems. 2014: 2096-2104.
13. Tai K S, Socher R, Manning C D. Improved semantic representations from tree-structured long short-term memory networks[J]. arXiv preprint arXiv:1503.00075, 2015.
14. Hong J, Fang M. Sentiment Analysis with Deeply Learned Distributed Representations of Variable Length Texts[J].

References

Supplementaries

Word2vec

Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space[J]. arXiv preprint arXiv:1301.3781, 2013.

Dropout

Hinton G E, Srivastava N, Krizhevsky A, et al. Improving neural networks by preventing co-adaptation of feature detectors[J]. arXiv preprint arXiv:1207.0580, 2012.

AdaGrad

Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization[J]. Journal of Machine Learning Research, 2011, 12(Jul): 2121-2159.

Deep Convolutional Neural Network

Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.

Deep Residual Learning

He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[J]. arXiv preprint arXiv:1512.03385, 2015.

THANKS!