Character-Aware Neural Language Models

Yoon Kim Yacine Jernite David Sontag Alexander Rush

Harvard SEAS New York University

To appear in AAAI 2016

Code: https://github.com/yoonkim/lstm-char-cnn

Language Model

Language Model (LM): probability distribution over a sequence of words.

$p(w_1, \ldots, w_T)$ for any sequence of length T from a vocabulary $\mathcal{V}$ (with $w_i \in \mathcal{V}$ for all i).

Important for many downstream applications:

- machine translation
- speech recognition
- text generation

Count-based Language Models

By the chain rule, any distribution can be factorized as:

$$p(w_1, \dots, w_T) = \prod_{t=1}^{T} p(w_t | w_1, \dots, w_{t-1})$$

n-gram language models make a Markov assumption:

$$p(w_t | w_1, \dots, w_{t-1}) \approx p(w_t | w_{\mathbf{t-n}}, \dots, w_{t-1})$$

Needs smoothing to deal with rare *n*-grams.

Neural Language Models (NLM)

- Represent words as dense vectors in $\mathbb{R}^n$ (word embeddings).

  $\mathbf{w}_t \in \mathbb{R}^{|\mathcal{V}|}$: One-hot representation of word $\in \mathcal{V}$ at time t
  $\Rightarrow \mathbf{x}_t = \mathbf{X}\mathbf{w}_t$: Word embedding ($\mathbf{X} \in \mathbb{R}^{n \times |\mathcal{V}|}, n < |\mathcal{V}|$)

Neural Language Models

Neural Language Models (NLM)

- Represent words as dense vectors in $\mathbb{R}^n$ (word embeddings).

  $\mathbf{w}_t \in \mathbb{R}^{|\mathcal{V}|}$: One-hot representation of word $\in \mathcal{V}$ at time t
  $\Rightarrow \mathbf{x}_t = \mathbf{X}\mathbf{w}_t$: Word embedding ($\mathbf{X} \in \mathbb{R}^{n \times |\mathcal{V}|}, n < |\mathcal{V}|$)

- Train a neural net that composes history to predict next word.

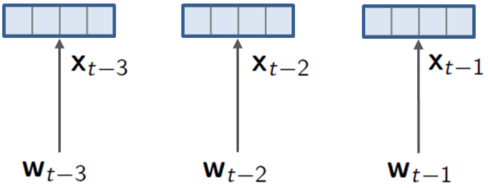
$$p(w_t = j | w_1, \dots, w_{t-1}) = \frac{\exp(\mathbf{p}^j \cdot g(\mathbf{x}_1, \dots, \mathbf{x}_{t-1}) + q^j)}{\sum_{j' \in \mathcal{V}} \exp(\mathbf{p}^{j'} \cdot g(\mathbf{x}_1, \dots, \mathbf{x}_{t-1}) + q^{j'})}$$
$$= \text{softmax}(\mathbf{P} g(\mathbf{x}_1, \dots, \mathbf{x}_{t-1}) + \mathbf{q})$$

$\mathbf{p}^j \in \mathbb{R}^m, q^j \in \mathbb{R}$: Output word embedding/bias for word $j \in \mathcal{V}$
g : Composition function

Feed-forward NLM (Bengio, Ducharme, and Vincent 2003)

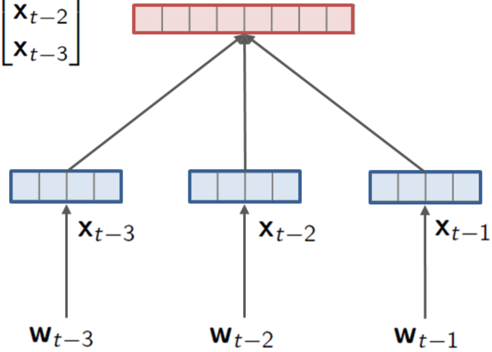
$\mathbf{w}_{t-3}$ $\mathbf{w}_{t-2}$ $\mathbf{w}_{t-1}$

Feed-forward NLM (Bengio, Ducharme, and Vincent 2003)



Feed-forward NLM (Bengio, Ducharme, and Vincent 2003)

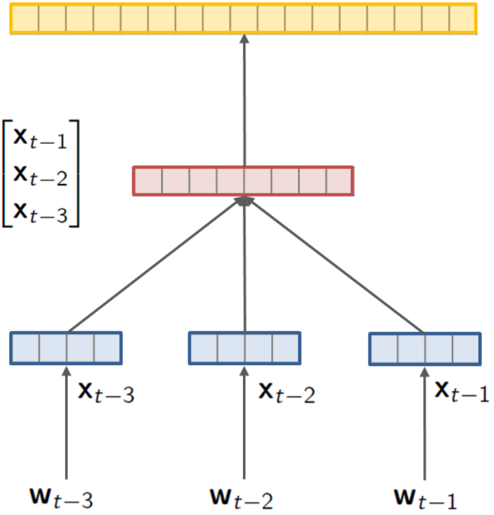
$$\mathbf{h}_{t-1} = \mathbf{W} \begin{bmatrix} \mathbf{x}_{t-1} \\ \mathbf{x}_{t-2} \\ \mathbf{x}_{t-3} \end{bmatrix}$$



Feed-forward NLM (Bengio, Ducharme, and Vincent 2003)

$$p(w_t | w_1, \dots, w_{t-1}) = \text{softmax}(\mathbf{P}\mathbf{h}_{t-1} + \mathbf{q})$$

$$\mathbf{h}_{t-1} = \mathbf{W} \begin{bmatrix} \mathbf{x}_{t-1} \\ \mathbf{x}_{t-2} \\ \mathbf{x}_{t-3} \end{bmatrix}$$



Recurrent Neural Network LM (Mikolov et al. 2011)

Maintain a hidden state vector $\mathbf{h}_t$ that is recursively calculated.

Recurrent Neural Network LM (Mikolov et al. 2011)

Maintain a hidden state vector $\mathbf{h}_t$ that is recursively calculated.

$$\mathbf{h}_t = f(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b})$$

$$\begin{aligned} \mathbf{h}_t \in \mathbb{R}^m &: \text{Hidden state at time } t \text{ (summary of history)} \\ \mathbf{W} \in \mathbb{R}^{m \times n} &: \text{Input-to-hidden transformation} \\ \mathbf{U} \in \mathbb{R}^{m \times m} &: \text{Hidden-to-hidden transformation} \\ f(\cdot) &: \text{Non-linearity} \end{aligned}$$

Recurrent Neural Network LM (Mikolov et al. 2011)

Maintain a hidden state vector $\mathbf{h}_t$ that is recursively calculated.

$$\mathbf{h}_t = f(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b})$$

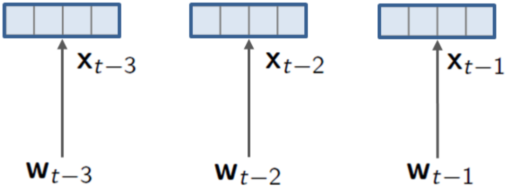
$$\begin{aligned} \mathbf{h}_t \in \mathbb{R}^m &: \text{Hidden state at time } t \text{ (summary of history)} \\ \mathbf{W} \in \mathbb{R}^{m \times n} &: \text{Input-to-hidden transformation} \\ \mathbf{U} \in \mathbb{R}^{m \times m} &: \text{Hidden-to-hidden transformation} \\ f(\cdot) &: \text{Non-linearity} \end{aligned}$$

Apply softmax to $\mathbf{h}_t$.

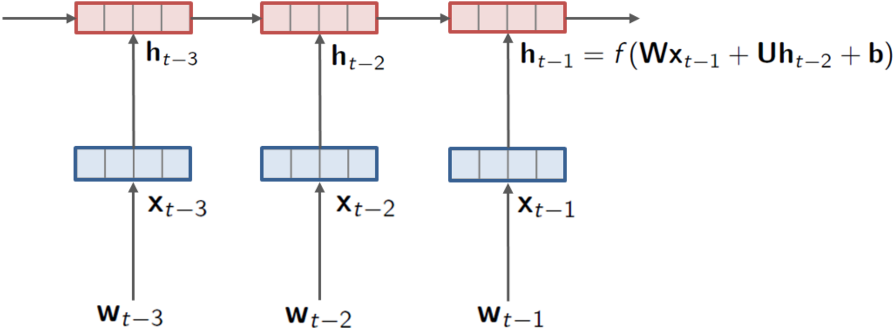
Recurrent Neural Network LM (Mikolov et al. 2011)

$\mathbf{w}_{t-3}$ $\mathbf{w}_{t-2}$ $\mathbf{w}_{t-1}$

Recurrent Neural Network LM (Mikolov et al. 2011)

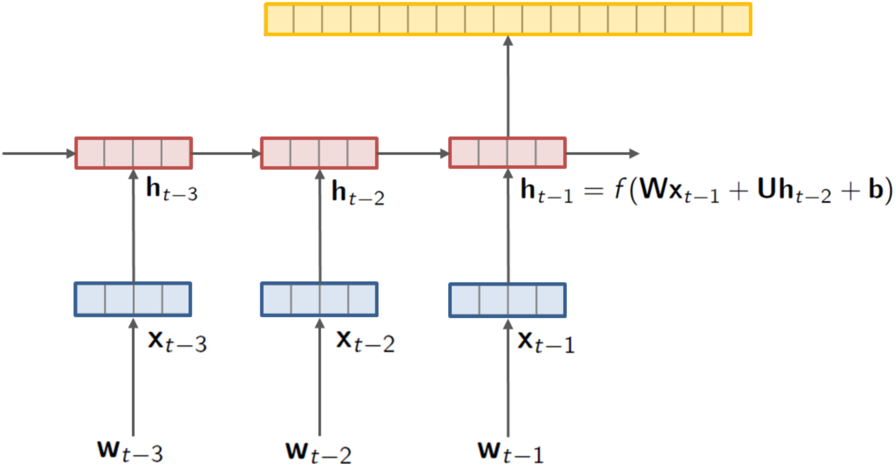


Recurrent Neural Network LM (Mikolov et al. 2011)



Recurrent Neural Network LM (Mikolov et al. 2011)

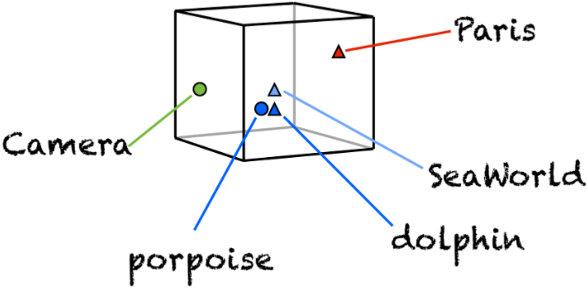
$$p(w_t | w_1, \dots, w_{t-1}) = \mathrm{softmax}(\mathbf{P}\mathbf{h}_{t-1} + \mathbf{q})$$



Word Embeddings (Collobert et al. 2011; Mikolov et al. 2012)

Key ingredient in Neural Language Models.

After training, similar words are close in the vector space.



(Not unique to NLMs)

NLM Performance (on Penn Treebank)

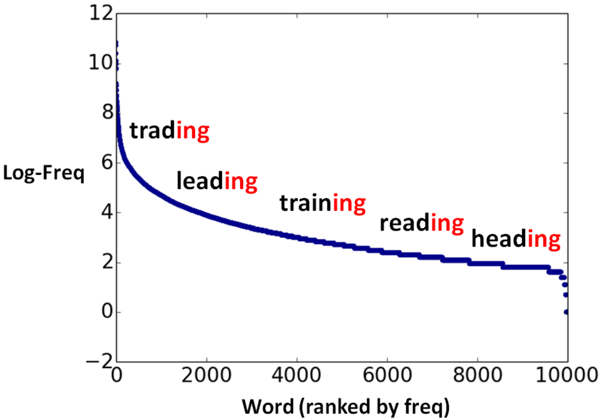
Difficult/expensive to train, but performs well.

Language Model	Perplexity
5-gram count-based (Mikolov and Zweig 2012)	141.2
RNN (Mikolov and Zweig 2012)	124.7
Deep RNN (Pascanu et al. 2013)	107.5
LSTM (Zaremba, Sutskever, and Vinyals 2014)	78.4

Renewed interest in language modeling.

NLM Issue

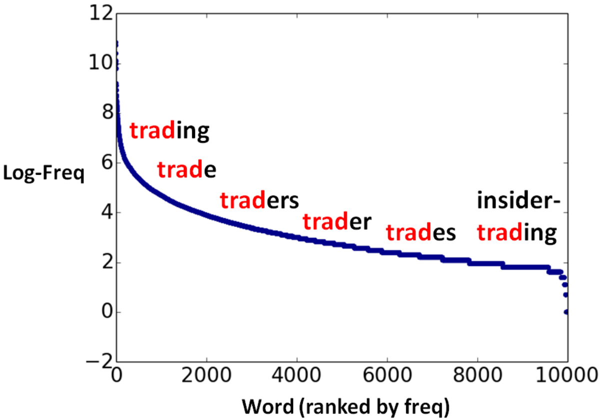
Issue: The fundamental unit of information is still the **word**



Separate embeddings for "trading", "leading", "training", etc.

NLM Issue

Issue: The fundamental unit of information is still the **word**



Separate embeddings for “trading”, “trade”, “trades”, etc.

NLM Issue

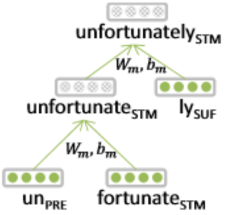
- No parameter sharing across orthographically similar words (e.g., spelled similarly).
- Orthography contains much semantic/syntactic information.
- How can we leverage subword information for language modeling?
- Can we exploit this to perform better language modeling with rare words?

Previous (NLM-based) Work

Use morphological segmenter as a preprocessing step

$$\text{unfortunately} \Rightarrow \text{un}_{\text{PRE}} \cdot \text{fortunate}_{\text{STM}} \cdot \text{ly}_{\text{SUF}}$$

- Luong, Socher, and Manning 2013: Recursive Neural Network over morpheme embeddings



- Botha and Blunsom 2014: Sum over word/morpheme embeddings

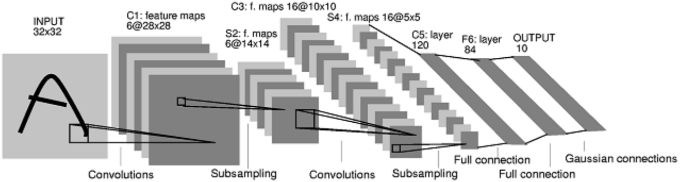
This Work

Main Idea: No explicit morphology, use characters directly.

This Work

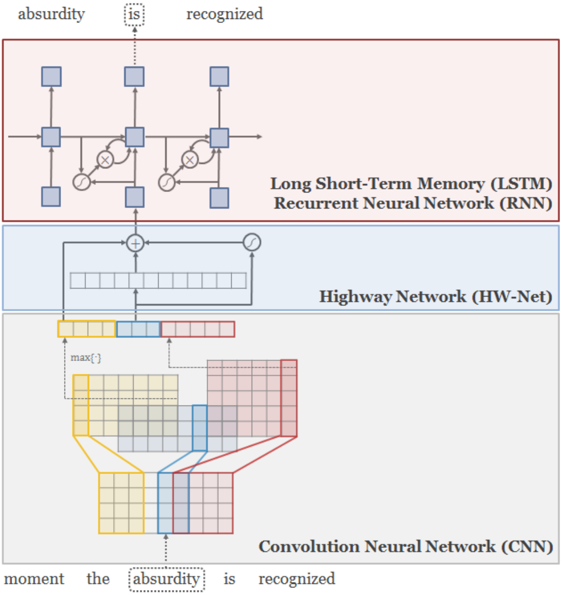
Main Idea: No explicit morphology, use characters directly.

Convolutional Neural Networks (CNN) (LeCun et al. 1989)

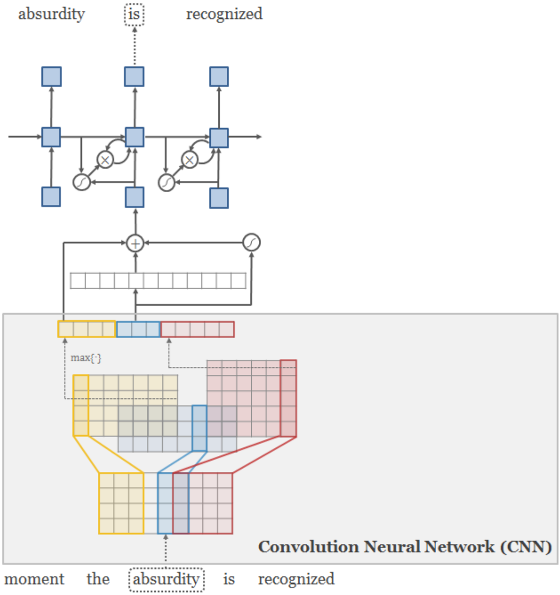


- Central to deep learning systems in vision.
- Shown to be effective for NLP tasks (Collobert et al. 2011).
- CNNs in NLP typically involve temporal (rather than spatial) convolutions over words.

Network Architecture: Overview



Character-level CNN (CharCNN)



Character-level CNN (CharCNN)

$$\mathbf{C} \in \mathbb{R}^{d \times l} : \text{Matrix representation of word (of length } l\text{)}$$

$$\mathbf{H} \in \mathbb{R}^{d \times w} : \text{Convolutional filter matrix}$$

$$d : \text{Dimensionality of character embeddings (e.g., 15)}$$

$$w : \text{Width of convolution filter (e.g., 1,2,3,4,5)}$$

Character-level CNN (CharCNN)

$\mathbf{C} \in \mathbb{R}^{d \times l}$: Matrix representation of word (of length l)

$\mathbf{H} \in \mathbb{R}^{d \times w}$: Convolutional filter matrix

d : Dimensionality of character embeddings (e.g., 15)

w : Width of convolution filter (e.g., 1,2,3,4,5)

1. Apply a convolution between $\mathbf{C}$ and $\mathbf{H}$ to obtain a vector $\mathbf{f} \in \mathbb{R}^{l-w+1}$

$$\mathbf{f}[i] = \langle \mathbf{C}[*, i : i + w - 1], \mathbf{H} \rangle$$

where $\langle \mathbf{A}, \mathbf{B} \rangle = \mathrm{Tr}(\mathbf{A}\mathbf{B}^T)$ is the Frobenius inner product.

Character-level CNN (CharCNN)

$\mathbf{C} \in \mathbb{R}^{d \times l}$: Matrix representation of word (of length l)

$\mathbf{H} \in \mathbb{R}^{d \times w}$: Convolutional filter matrix

d : Dimensionality of character embeddings (e.g., 15)

w : Width of convolution filter (e.g., 1,2,3,4,5)

1. Apply a convolution between $\mathbf{C}$ and $\mathbf{H}$ to obtain a vector $\mathbf{f} \in \mathbb{R}^{l-w+1}$

$$\mathbf{f}[i] = \langle \mathbf{C}[*, i : i + w - 1], \mathbf{H} \rangle$$

where $\langle \mathbf{A}, \mathbf{B} \rangle = \mathrm{Tr}(\mathbf{A}\mathbf{B}^T)$ is the Frobenius inner product.

2. Take the *max-over-time* (with bias and nonlinearity)

$$y = \tanh(\max_i \{\mathbf{f}[i]\} + b)$$

as the feature corresponding to the filter $\mathbf{H}$ (for a particular word).

Character-level CNN (CharCNN)

a b s u r d i t y

Character-level CNN (CharCNN)

$\mathbf{C} \in \mathbb{R}^{d \times l}$: Representation of *absurdity*

0.4	-0.8	2.2	0.1	0.5	-0.4	0.4	-0.4	0.1
0.1	1.2	1.5	-0.8	-1.5	0.2	0.1	1.2	0.7
0.2	0.1	-1.2	0.2	-0.2	0.3	0.2	-1.3	-0.1
-0.2	-0.5	0.1	0.2	-0.3	0.3	-0.1	1.0	-0.3

. . . .

a b s u r d i t y

Character-level CNN (CharCNN)

$\mathbf{H} \in \mathbb{R}^{d \times w}$: Convolutional filter matrix of width $w = 3$

-0.1	0.5	2.2
0.7	0.9	0.3
-0.2	-0.2	0.7
1.3	-0.1	-1.1

0.4	-0.8	2.2	0.1	0.5	-0.4	0.4	-0.4	0.1
0.1	1.2	1.5	-0.8	-1.5	0.2	0.1	1.2	0.7
0.2	0.1	-1.2	0.2	-0.2	0.3	0.2	-1.3	-0.1
-0.2	-0.5	0.1	0.2	-0.3	0.3	-0.1	1.0	-0.3

. . . .

a b s u r d i t y

Character-level CNN (CharCNN)

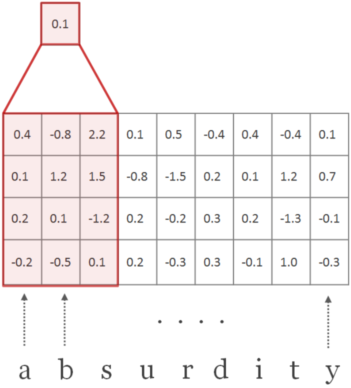
$$\mathbf{f}[1] = \langle \mathbf{C}[*, 1:3], \mathbf{H} \rangle$$

0.4	-0.8	2.2	0.1	0.5	-0.4	0.4	-0.4	0.1
0.1	1.2	1.5	-0.8	-1.5	0.2	0.1	1.2	0.7
0.2	0.1	-1.2	0.2	-0.2	0.3	0.2	-1.3	-0.1
-0.2	-0.5	0.1	0.2	-0.3	0.3	-0.1	1.0	-0.3

. . . .

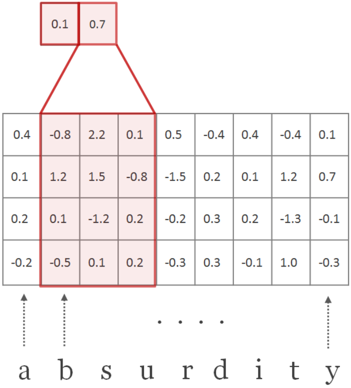
a b s u r d i t y

$$\mathbf{f}[1] = \langle \mathbf{C}[*, 1:3], \mathbf{H} \rangle$$



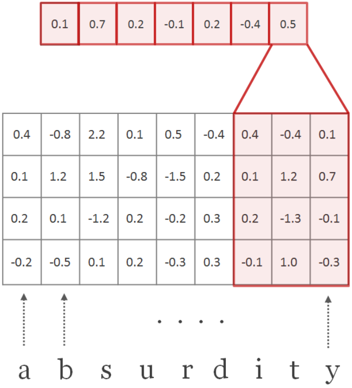
Character-level CNN (CharCNN)

$$\mathbf{f}[2] = \langle \mathbf{C}[*, 2:4], \mathbf{H} \rangle$$



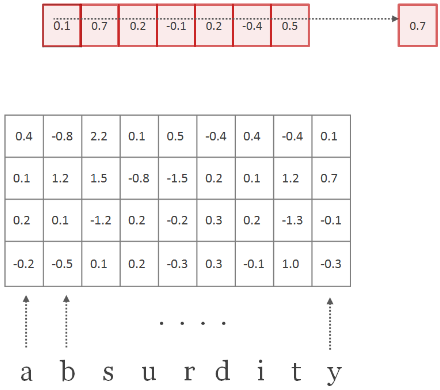
Character-level CNN (CharCNN)

$$\mathbf{f}[T-2] = \langle \mathbf{C}[*, T-2 : T], \mathbf{H} \rangle$$



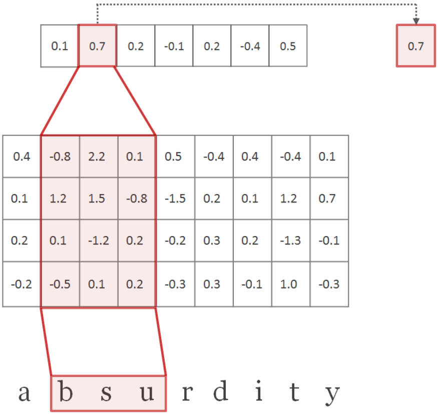
Character-level CNN (CharCNN)

$$y[1] = \max_i \{\mathbf{f}[i]\}$$



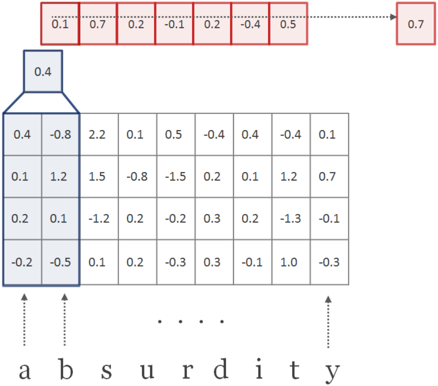
Character-level CNN (CharCNN)

Each filter picks out a character *n*-gram



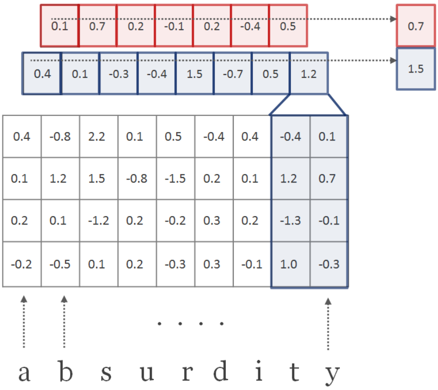
Character-level CNN (CharCNN)

$$\mathbf{f}'[1] = \langle \mathbf{C}[*, 1:2], \mathbf{H}' \rangle$$



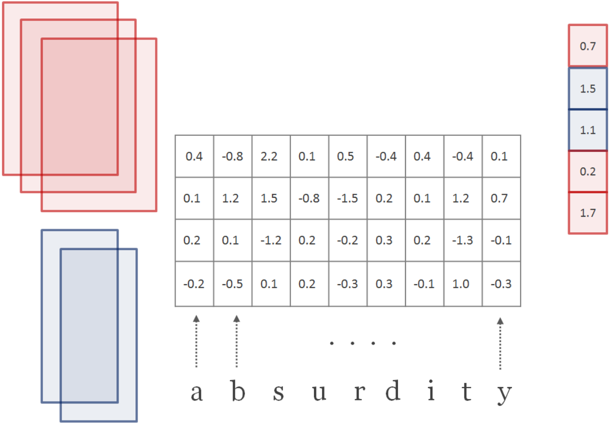
Character-level CNN (CharCNN)

$$y[2] = \max_i \{\mathbf{f}'[i]\}$$



Character-level CNN (CharCNN)

Many filter matrices (25–200) per width (1–7)



Character-level CNN (CharCNN)

Add bias, apply nonlinearity

$$\tanh\left(\begin{bmatrix}0.7\\1.5\\1.1\\0.2\\1.7\end{bmatrix} + \vec{b}\right) = \begin{bmatrix}0.8\\1.0\\0.9\\0.5\\1.1\end{bmatrix}$$

Character-level CNN (CharCNN)

For roughly the same number of parameters (20 million),

Before	Now
Word embedding	Output from CharCNN
PTB Perplexity: 85.4	PTB Perplexity: 84.6

CharCNN is slower, but convolution operations on GPU have been very optimized.

Character-level CNN (CharCNN)

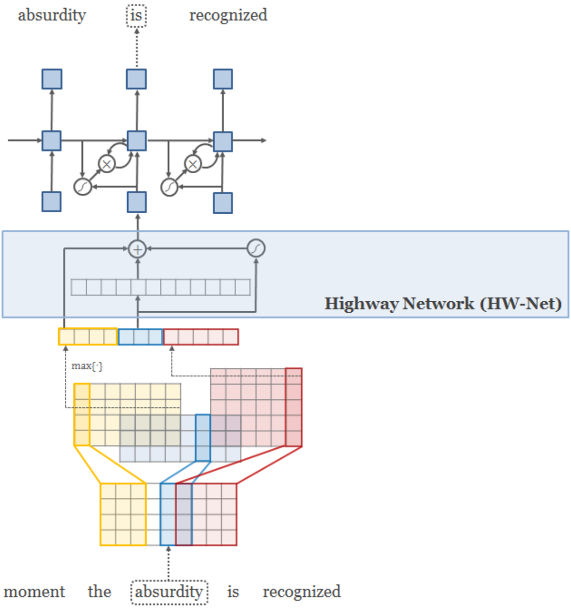
For roughly the same number of parameters (20 million),

Before	Now
Word embedding	Output from CharCNN
PTB Perplexity: 85.4	PTB Perplexity: 84.6

CharCNN is slower, but convolution operations on GPU have been very optimized.

Can we model more complex interactions between character *n*-grams picked up by the filters?

Highway Network



Highway Network

$\mathbf{y}$: output from CharCNN

Multilayer Perceptron

$$\mathbf{z} = g(\mathbf{W}\mathbf{y} + \mathbf{b})$$

Highway Network

$\mathbf{y}$: output from CharCNN

Multilayer Perceptron

$$\mathbf{z} = g(\mathbf{W}\mathbf{y} + \mathbf{b})$$

Highway Network

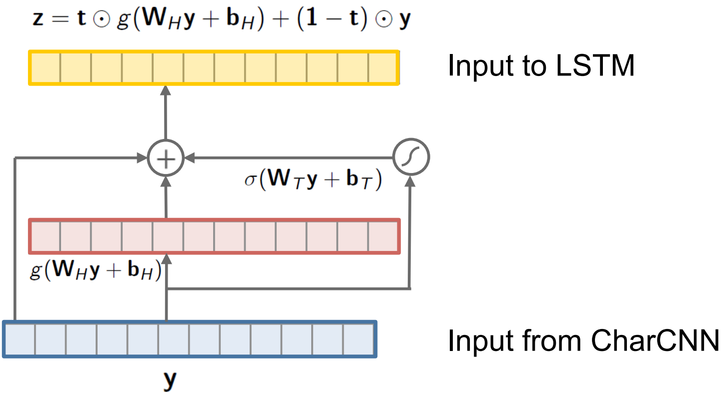
(Srivastava, Greff, and Schmidhuber 2015)

$$\mathbf{z} = \mathbf{t} \odot g(\mathbf{W}_H\mathbf{y} + \mathbf{b}_H) + (\mathbf{1} - \mathbf{t}) \odot \mathbf{y}$$

$$\begin{aligned} \mathbf{W}_H, \mathbf{b}_H &: \text{Affine transformation} \\ \mathbf{t} = \sigma(\mathbf{W}_T\mathbf{y} + \mathbf{b}_T) &: \textit{transform}\text{ gate} \\ \mathbf{1} - \mathbf{t} &: \textit{carry}\text{ gate} \end{aligned}$$

Hierarchical, adaptive composition of character *n*-grams.

Highway Network



Highway Network

Model	Perplexity
Word Model	85.4
No Highway Layers	84.6
One MLP Layer	92.6
One Highway Layer	79.7
Two Highway Layers	78.9

No more gains with 3+ layers.

Results: English Penn Treebank

	PPL	Size
KN-5 (Mikolov et al. 2012)	141.2	2 m
RNN (Mikolov et al. 2012)	124.7	6 m
Deep RNN (Pascanu et al. 2013)	107.5	6 m
Sum-Prod Net (Cheng et al. 2014)	100.0	5 m
LSTM-Medium (Zaremba, Sutskever, and Vinyals 2014)	82.7	20 m
LSTM-Huge (Zaremba, Sutskever, and Vinyals 2014)	78.4	52 m
LSTM-Word-Small	97.6	5 m
LSTM-Char-Small	92.3	5 m
LSTM-Word-Large	85.4	20 m
LSTM-Char-Large	78.9	19 m

What about morphologically rich languages?

	Data-s			Data-l		
	$\|\mathcal{V}\|$	$\|\mathcal{C}\|$	T	$\|\mathcal{V}\|$	$\|\mathcal{C}\|$	T
English (En)	10 k	51	1 m	60 k	197	20 m
Czech (Cs)	46 k	101	1 m	206 k	195	17 m
German (De)	37 k	74	1 m	339 k	260	51 m
Spanish (Es)	27 k	72	1 m	152 k	222	56 m
French (Fr)	25 k	76	1 m	137 k	225	57 m
Russian (Ru)	62 k	62	1 m	497 k	111	25 m

$|\mathcal{V}|$ = Word vocab Size
$|\mathcal{C}|$ = Character vocab size
T = number of tokens in training set.

What about morphologically rich languages?

	Data-s			Data-l		
	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{C}\rvert$	T	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{C}\rvert$	T
English (En)	10 k	51	1 m	60 k	197	20 m
Czech (Cs)	46 k	101	1 m	206 k	195	17 m
German (De)	37 k	74	1 m	339 k	260	51 m
Spanish (Es)	27 k	72	1 m	152 k	222	56 m
French (Fr)	25 k	76	1 m	137 k	225	57 m
Russian (Ru)	62 k	62	1 m	497 k	111	25 m

$\lvert\mathcal{V}\rvert$ varies quite a bit by language.

(effectively use the full vocabulary)

Baselines

Kneser-Ney LM: Count-based baseline

Word LSTM: Word embeddings as input

Morpheme LBL (Botha and Blunsom 2014)

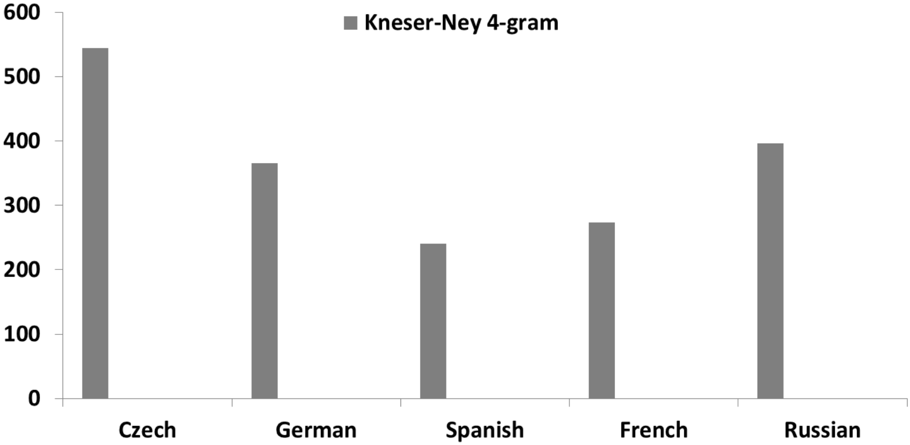
Input for word k is

$$\underbrace{\mathbf{x}^k}_{\text{word embedding}} + \underbrace{\sum_{j \in \mathcal{M}_k} \mathbf{m}^j}_{\text{morpheme embeddings}}$$

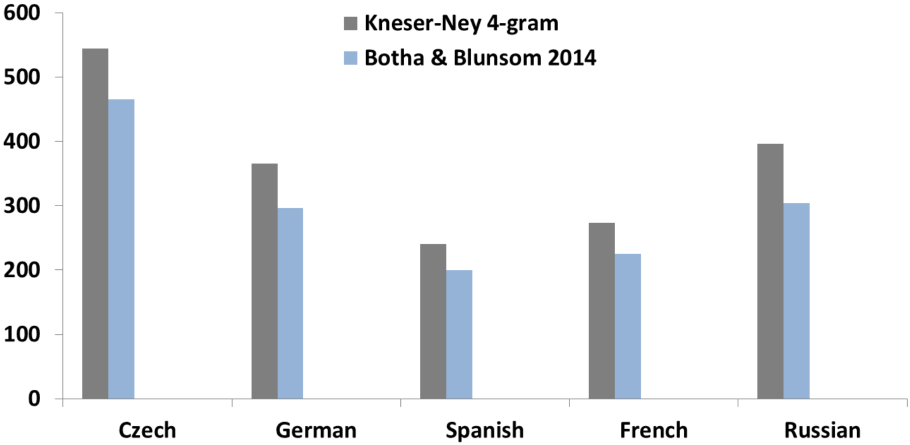
Morpheme LSTM: Same input as above, but with LSTM architecture

Morphemes obtained from running an unsupervised morphological tagger Morfessor Cat-MAP (Creutz and Lagus 2007).

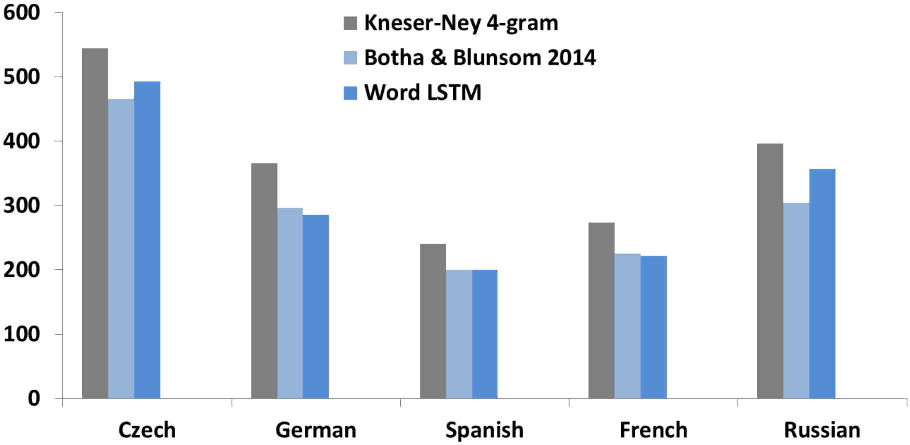
Perplexity on Data-S (1 M Tokens)



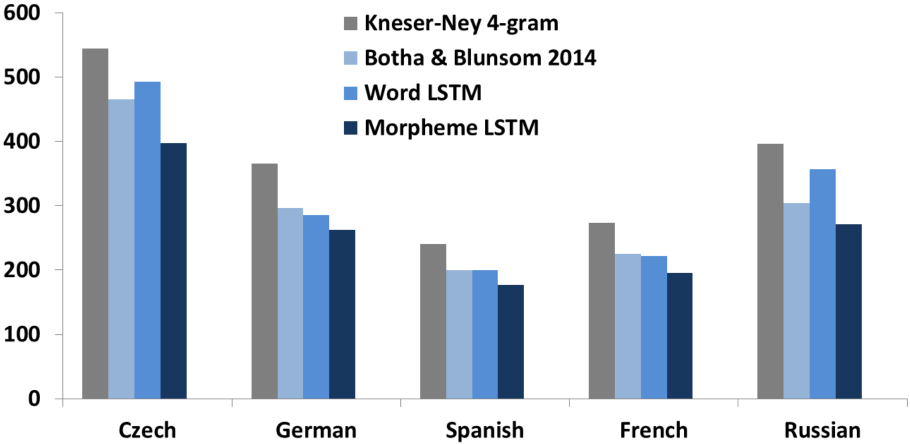
Perplexity on Data-S (1 M Tokens)



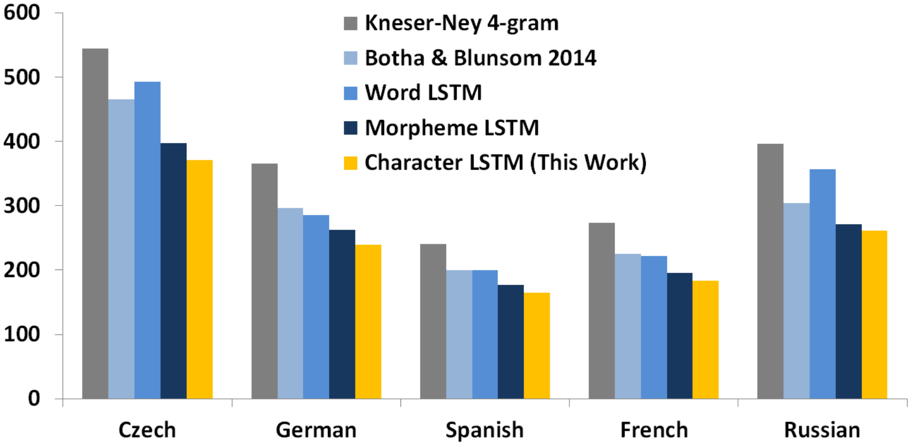
Perplexity on Data-S (1 M Tokens)



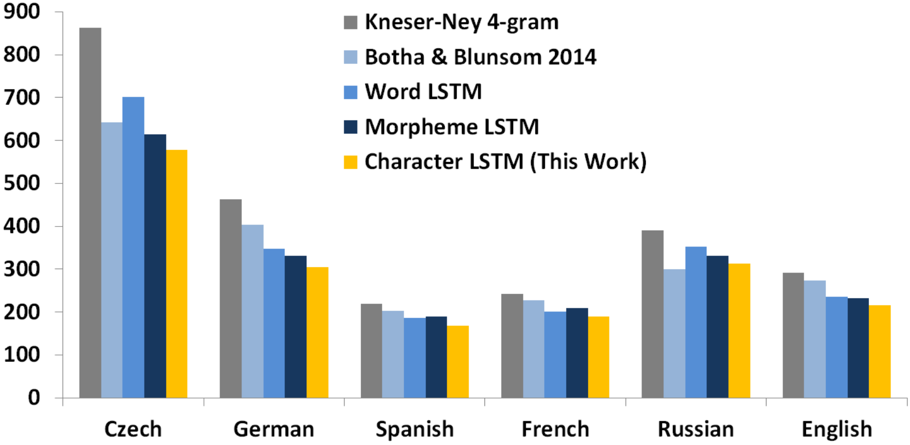
Perplexity on Data-S (1 M Tokens)



Perplexity on Data-S (1 M Tokens)



Perplexity on Data-L (17-57 M Tokens)



How does performance vary with corpus/vocab size?

Experiment on German large dataset:

- Take the most frequent K words as the vocabulary and replace rest with $<$unk$>$
- Compare % perplexity reduction going from word to character LSTM.

How does performance vary with corpus/vocab size?

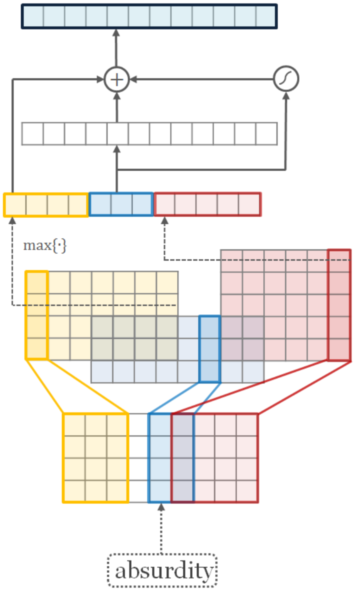
Experiment on German large dataset:

- Take the most frequent K words as the vocabulary and replace rest with <unk>
- Compare % perplexity reduction going from word to character LSTM.

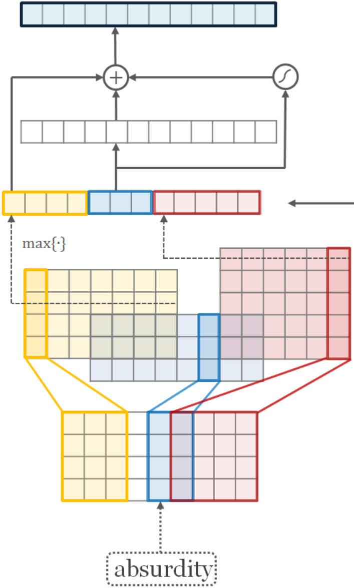
		Vocabulary Size			
		10 k	25 k	50 k	100 k
	1 m	17	16	21	–
Training	5 m	8	14	16	21
Size	10 m	9	9	12	15
	25 m	9	8	9	10

Character model outperforms word model in all scenarios.

Learned Word Representations

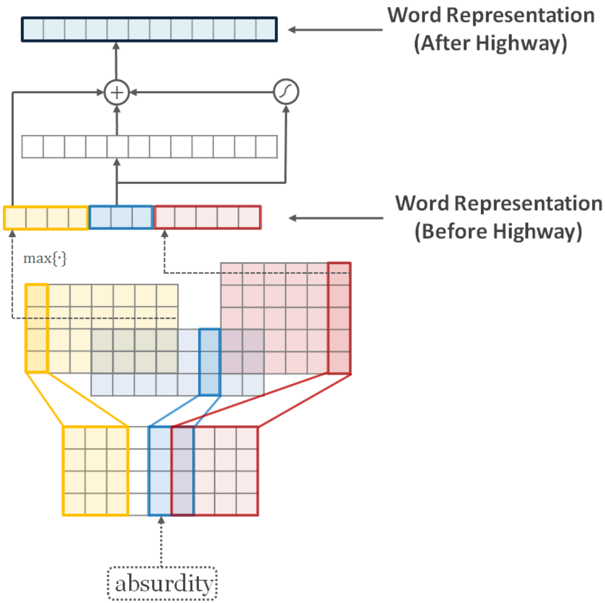


Learned Word Representations



Word Representation
(Before Highway)

Learned Word Representations



Learned Word Representations (In Vocab)

(Based on cosine similarity)

	In Vocabulary				
	while	*his*	*you*	*richard*	*trading*
Word Embedding	*although*	*your*	*conservatives*	*jonathan*	*advertised*
	letting	*her*	*we*	*robert*	*advertising*
	though	*my*	*guys*	*neil*	*turnover*
	minute	*their*	*i*	*nancy*	*turnover*

Learned Word Representations (In Vocab)

(Based on cosine similarity)

	In Vocabulary				
	while	*his*	*you*	*richard*	*trading*
Word Embedding	*although*	*your*	*conservatives*	*jonathan*	*advertised*
	letting	*her*	*we*	*robert*	*advertising*
	though	*my*	*guys*	*neil*	*turnover*
	minute	*their*	*i*	*nancy*	*turnover*
Characters (before highway)	*chile*	*this*	*your*	*hard*	*heading*
	whole	*hhs*	*young*	*rich*	*training*
	meanwhile	*is*	*four*	*richer*	*reading*
	white	*has*	*youth*	*richter*	*leading*

Learned Word Representations (In Vocab)

(Based on cosine similarity)

	In Vocabulary				
	while	*his*	*you*	*richard*	*trading*
Word Embedding	*although* *letting* *though* *minute*	*your* *her* *my* *their*	*conservatives* *we* *guys* *i*	*jonathan* *robert* *neil* *nancy*	*advertised* *advertising* *turnover* *turnover*
Characters (before highway)	*chile* *whole* *meanwhile* *white*	*this* *hhs* *is* *has*	*your* *young* *four* *youth*	*hard* *rich* *richer* *richter*	*heading* *training* *reading* *leading*
Characters (after highway)	*meanwhile* *whole* *though* *nevertheless*	*hhs* *this* *their* *your*	*we* *your* *doug* *i*	*eduard* *gerard* *edward* *carl*	*trade* *training* *traded* *trader*

Learned Word Representations (In Vocab)

(Based on cosine similarity)

	In Vocabulary				
	while	*his*	*you*	*richard*	*trading*
Word Embedding	*although* *letting* *though* *minute*	*your* *her* *my* *their*	*conservatives* *we* *guys* *i*	*jonathan* *robert* *neil* *nancy*	*advertised* *advertising* *turnover* *turnover*
Characters (before highway)	*chile* *whole* *meanwhile* *white*	*this* *hhs* *is* *has*	*your* *young* *four* *youth*	*hard* *rich* *richer* *richter*	*heading* *training* *reading* *leading*
Characters (after highway)	*meanwhile* *whole* *though* *nevertheless*	*hhs* *this* *their* *your*	*we* *your* *doug* *i*	*eduard* *gerard* *edward* *carl*	*trade* *training* *traded* *trader*

Learned Word Representations (In Vocab)

(Based on cosine similarity)

	In Vocabulary				
	while	*his*	*you*	*richard*	*trading*
Word Embedding	*although*	*your*	*conservatives*	*jonathan*	*advertised*
	letting	*her*	*we*	*robert*	*advertising*
	though	*my*	*guys*	*neil*	*turnover*
	minute	*their*	*i*	*nancy*	*turnover*
Characters (before highway)	*chile*	*this*	*your*	*hard*	*heading*
	whole	*hhs*	*young*	*rich*	*training*
	meanwhile	*is*	*four*	*richer*	*reading*
	white	*has*	*youth*	*richter*	*leading*
Characters (after highway)	*meanwhile*	*hhs*	*we*	*eduard*	*trade*
	whole	*this*	*your*	*gerard*	*training*
	though	*their*	*doug*	*edward*	*traded*
	nevertheless	*your*	*i*	*carl*	*trader*

Learned Word Representations (In Vocab)

(Based on cosine similarity)

	In Vocabulary				
	while	*his*	*you*	*richard*	*trading*
Word Embedding	*although*	*your*	*conservatives*	*jonathan*	*advertised*
	letting	*her*	*we*	*robert*	*advertising*
	though	*my*	*guys*	*neil*	*turnover*
	minute	*their*	*i*	*nancy*	*turnover*
Characters (before highway)	*chile*	*this*	*your*	*hard*	*heading*
	whole	*hhs*	*young*	*rich*	*training*
	meanwhile	*is*	*four*	*richer*	*reading*
	white	*has*	*youth*	*richter*	*leading*
Characters (after highway)	*meanwhile*	*hhs*	*we*	*eduard*	*trade*
	whole	*this*	*your*	*gerard*	*training*
	though	*their*	*doug*	*edward*	*traded*
	nevertheless	*your*	*i*	*carl*	*trader*

Learned Word Representations (In Vocab)

(Based on cosine similarity)

	In Vocabulary				
	while	*his*	*you*	*richard*	*trading*
Word Embedding	*although*	*your*	*conservatives*	*jonathan*	*advertised*
	letting	*her*	*we*	*robert*	*advertising*
	though	*my*	*guys*	*neil*	*turnover*
	minute	*their*	*i*	*nancy*	*turnover*
Characters (before highway)	*chile*	*this*	*your*	*hard*	*heading*
	whole	*hhs*	*young*	*rich*	*training*
	meanwhile	*is*	*four*	*richer*	*reading*
	white	*has*	*youth*	*richter*	*leading*
Characters (after highway)	*meanwhile*	*hhs*	*we*	*eduard*	*trade*
	whole	*this*	*your*	*gerard*	*training*
	though	*their*	*doug*	*edward*	*traded*
	nevertheless	*your*	*i*	*carl*	*trader*

Learned Word Representations (OOV)

	Out-of-Vocabulary		
	computer-aided	*misinformed*	*looooook*
Characters (before highway)	*computer-guided*	*informed*	*look*
	computerized	*performed*	*cook*
	disk-drive	*transformed*	*looks*
	computer	*inform*	*shook*
Characters (after highway)	*computer-guided*	*informed*	*look*
	computer-driven	*performed*	*looks*
	computerized	*outperformed*	*looked*
	computer	*transformed*	*looking*

Learned Word Representations (OOV)

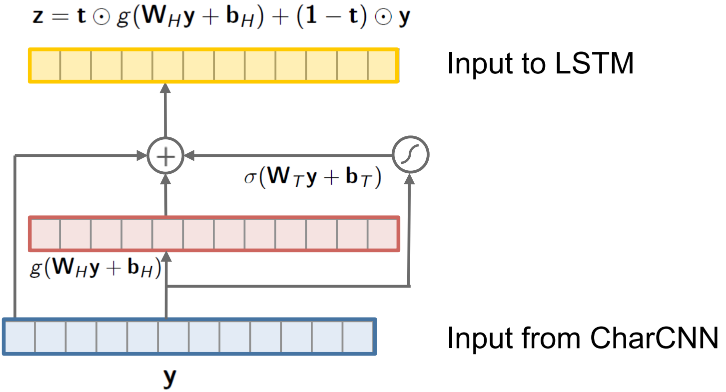
	Out-of-Vocabulary		
	computer-aided	*misinformed*	*looooook*
Characters (before highway)	*computer-guided*	*informed*	*look*
	computerized	*performed*	*cook*
	disk-drive	*transformed*	*looks*
	computer	*inform*	*shook*
Characters (after highway)	*computer-guided*	*informed*	*look*
	computer-driven	*performed*	*looks*
	computerized	*outperformed*	*looked*
	computer	*transformed*	*looking*

Learned Word Representations (OOV)

	Out-of-Vocabulary		
	computer-aided	*misinformed*	*looooook*
Characters (before highway)	*computer-guided*	*informed*	*look*
	computerized	*performed*	*cook*
	disk-drive	*transformed*	*looks*
	computer	*inform*	*shook*
Characters (after highway)	*computer-guided*	*informed*	*look*
	computer-driven	*performed*	*looks*
	computerized	*outperformed*	*looked*
	computer	*transformed*	*looking*

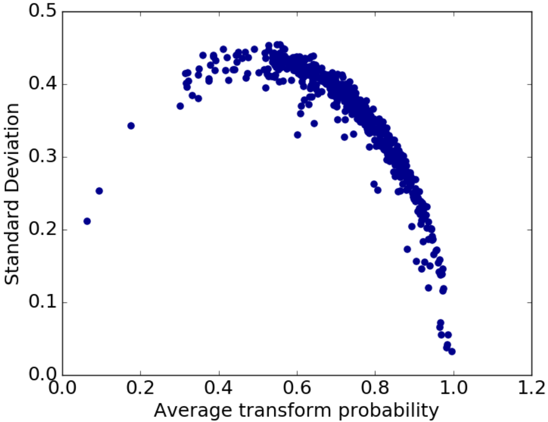
What is the highway network doing?

Q: Might we simply be learning to carry some dimensions and to combine others? Is the transform gate truly a function of the input word?



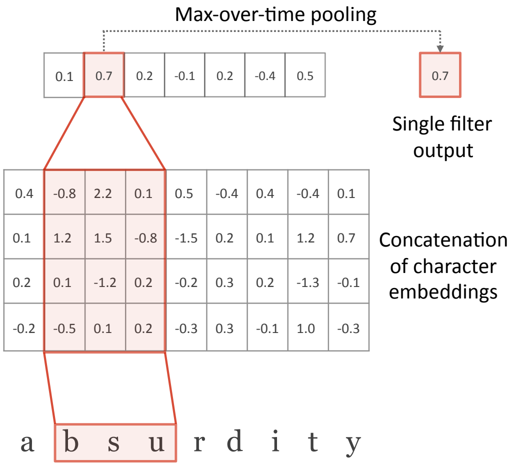
What is the highway network doing?

A: No. For all dimensions, on some words $\sigma(\cdot) \approx 0$, and for others ≈ 1.



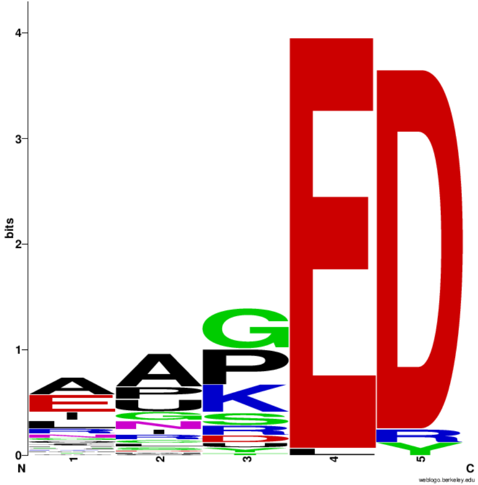
What is the convolutional layer doing?

Q: Does each filter truly pick out a character n-gram?



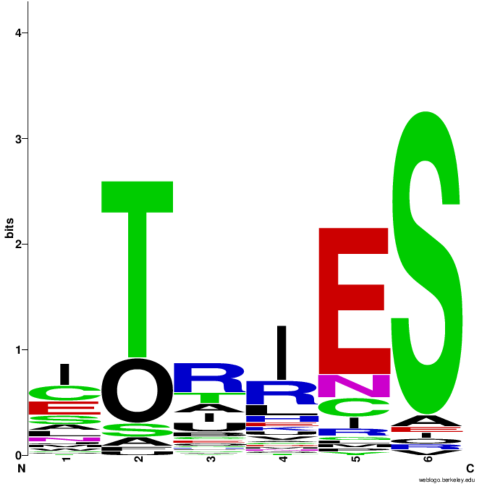
What is the convolutional layer doing?

For each length-6 filter, the 100 substrings with highest filter response.



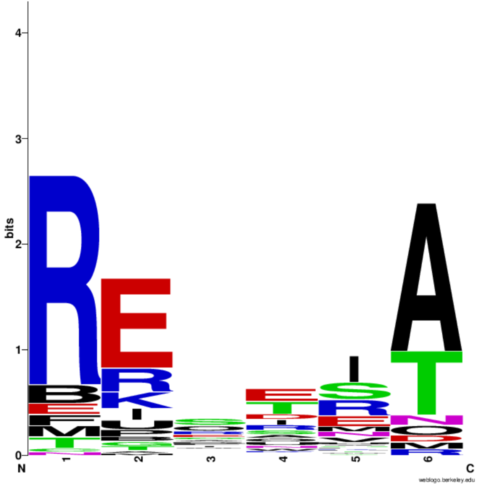
What is the convolutional layer doing?

For each length-6 filter, the 100 substrings with highest filter response.



What is the convolutional layer doing?

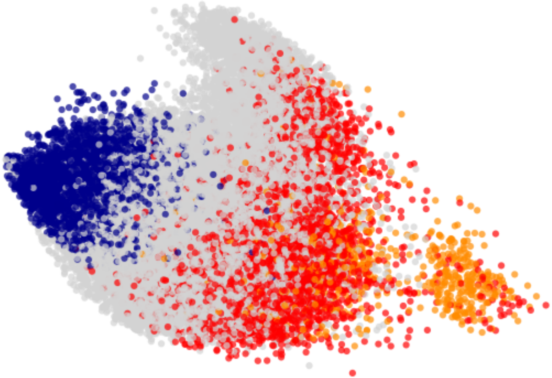
For each length-6 filter, the 100 substrings with highest filter response.



Conclusion

- A **character-aware** language model that relies only on character-level inputs: CharCNN + Highway network + LSTM.
- Outperforms strong word/morpheme LSTM baselines.
- Much recent work on character inputs:
  - Santos and Zadrozny 2014: CNN over characters concatenated with word embeddings into CRF.
  - Zhang and LeCun 2015: Deep CNN over characters for document classification.
  - Ballesteros, Dyer, and Smith 2015: LSTM over characters for parsing.
  - Ling et al. 2015: LSTM over characters into another LSTM for language modeling/POS-tagging.
- More broadly, suggests new ways to think about representation learning.

Appendix: Character *N*-gram Representations



Prefixes, Suffixes, Hyphenated, Others

Prefixes: character *n*-grams that start with 'start-of-word' character, such as {*un*, {*mis*. Suffixes defined similarly.

Appendix: Hyperparameters

		Small	Large
CNN	d	15	15
	w	$[1, 2, 3, 4, 5, 6]$	$[1, 2, 3, 4, 5, 6, 7]$
	h	$[25 \cdot w]$	$[\min\{200, 50 \cdot w\}]$
	f	tanh	tanh
HW-Net	l	1	2
	g	ReLU	ReLU
LSTM	l	2	2
	m	300	650

Appendix: Results on Data-S

		Cs	De	Es	Fr	Ru
B&B	KN-4	545	366	241	274	396
	MLBL	465	296	200	225	304
Small	Word	503	305	212	229	352
	Morph	414	278	197	216	290
	Char	401	260	182	189	278
Large	Word	493	286	200	222	357
	Morph	398	263	177	196	271
	Char	**371**	**239**	**165**	**184**	**261**

Appendix: Results on Data-L

		Cs	De	Es	Fr	Ru	En
B&B	KN-4	862	463	219	243	390	291
	MLBL	643	404	203	227	**300**	273
Small	Word	701	347	186	202	353	236
	Morph	615	331	189	209	331	233
	Char	**578**	**305**	**169**	**190**	313	**216**

Appendix: Effect of Highway Layers (PTB)

	Small	Large
No Highway Layers	100.3	84.6
One Highway Layer	92.3	79.7
Two Highway Layers	90.1	78.9
Multilayer Perceptron	111.2	92.6

No more gains with 2+ layers (may be language dependent).

References I

Bengio, Yoshua, Rejean Ducharme, and Pascal Vincent (2003). "A Neural Probabilistic Language Model". In: *Journal of Machine Learning Research* 3, pp. 1137–1155.

Mikolov, Tomas et al. (2011). "Empirical Evaluation and Combination of Advanced Language Modeling Techniques". In: *Proceedings of INTERSPEECH*.

Collobert, Ronan et al. (2011). "Natural Language Processing (almost) from Scratch". In: *Journal of Machine Learning Research* 12, pp. 2493–2537.

Mikolov, Tomas et al. (2012). "Subword Language Modeling with Neural Networks". In: *preprint: www.fit.vutbr.cz/˜imikolov/rnnlm/char.pdf*.

Mikolov, Tomas and Geoffrey Zweig (2012). "Context Dependent Recurrent Neural Network Language Model". In: *Proceedings of SLT*.

Pascanu, Razvan et al. (2013). "How to Construct Deep Neural Networks". In: *arXiv:1312.6026*.

References II

Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals (2014). "Recurrent Neural Network Regularization". In: *arXiv:1409.2329.*

Luong, Minh-Thang, Richard Socher, and Chris Manning (2013). "Better Word Representations with Recursive Neural Networks for Morphology". In: *Proceedings of CoNLL.*

Botha, Jan and Phil Blunsom (2014). "Compositional Morphology for Word Representations and Language Modelling". In: *Proceedings of ICML.*

LeCun, Yann et al. (1989). "Handwritten Digit Recognition with a Backpropagation Network". In: *Proceedings of NIPS.*

Srivastava, Rupesh Kumar, Klaus Greff, and Jurgen Schmidhuber (2015). "Training Very Deep Networks". In: *arXiv:1507.06228.*

Cheng, Wei Chen et al. (2014). "Language Modeling with Sum-Product Networks". In: *Proceedings of INTERSPEECH.*

References III

Creutz, Mathias and Krista Lagus (2007). "Unsupervised Models for Morpheme Segmentation and Morphology Learning". In: *Proceedings of the ACM Transations on Speech and Language Processing*.

Santos, Cicero Nogueira dos and Bianca Zadrozny (2014). "Learning Character-level Representations for Part-of-Speech Tagging". In: *Proceedings of ICML*.

Zhang, Xiang and Yann LeCun (2015). "Text Understanding From Scratch". In: *arXiv:1502.01710*.

Ballesteros, Miguel, Chris Dyer, and Noah A. Smith (2015). "Improved Transition-Based Parsing by Modeling Characters instead of Words with LSTMs". In: *Proceedings of EMNLP 2015*.

Ling, Wang et al. (2015). "Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation". In: *Proceedings of EMNLP*.

References IV

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9, pp. 1735–1780.