Convolutional Networks 1

CS 287

Quiz

Neural language models can be poor at assigning very high probability to high confidence decisions, for instance `major league baseball` or `united states of america`.

Give a high-level explanation of why this might occur compared to an n-gram model. Describe a variant of the Bengio model that is able to incorporate extra parameters to allow for rare cases that should have high probability.

Answer

Dense + Sparse with important conjunction features.

Contents

Answer I

Sentiment

Good Sentences

- A thoughtful, provocative, insistently humanizing film.
- Occasionally melodramatic, it's also extremely effective.
- Guaranteed to move anyone who ever shook, rattled, or rolled.

Bad Sentences

- A sentimental mess that never rings true.
- This 100-minute movie only has about 25 minutes of decent material.
- Here, common sense flies out the window, along with the hail of bullets, none of which ever seem to hit Sascha.

Review Linear Models for Classification

Linear model,

$$\hat{\mathbf{y}} = f(\mathbf{x}\mathbf{W} + \mathbf{b})$$

- $\mathbf{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}, \mathbf{b} \in \mathbb{R}^{1 \times d_{\text{out}}}$; model parameters
- $f : \mathbb{R}^{d_{\text{out}}} \mapsto \mathbb{R}^{d_{\text{out}}}$; activation function
- Sometimes $\mathbf{z} = \mathbf{x}\mathbf{W} + \mathbf{b}$ informally "score" vector.
- Note $\mathbf{z}$ and $\hat{\mathbf{y}}$ are not one-hot.

Class prediction,

$$\hat{c} = \operatorname*{arg\,max}_{i \in \mathcal{C}} \hat{y}_i = \operatorname*{arg\,max}_{i \in \mathcal{C}} (\mathbf{x}\mathbf{W} + \mathbf{b})_i$$

Features 1: Sparse Bag-of-Words Features

Representation is counts of input words,

- $\mathcal{F}$; the vocabulary of the language.
- $\mathbf{x} = \sum_i \delta(f_i)$

Example: Movie review input,

```
A sentimental mess
```

$$\begin{aligned} \mathbf{x} &= \delta(\texttt{word:A}) + \delta(\texttt{word:sentimental}) \\ &+ \delta(\texttt{word:mess}) \end{aligned}$$

$$\mathbf{x}^\top = \begin{bmatrix} 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix} = \begin{array}{c|l} \left[\begin{matrix} 1 \\ \vdots \\ 1 \\ 1 \end{matrix}\right] & \begin{matrix} \texttt{word:A} \\ \vdots \\ \texttt{word:mess} \\ \texttt{word:sentimental} \end{matrix} \end{array}$$

Features 2: Sparse Bag-of-Bigrams Features

Representation is counts of input bigrams,

- $\mathcal{F}$; the vocabulary of the bigram language.
- $\mathbf{x} = \sum_i \delta(f_i)$

Example: Movie review input,

```
A sentimental mess
```

$$\begin{aligned}\mathbf{x} &= \delta(\texttt{word:A}) + \delta(\texttt{bigram:A:sentimental}) \\ &+ \delta(\texttt{word:sentimental}) + \delta(\texttt{bigram:sentimental:mess}) \\ &+ \delta(\texttt{word:mess})\end{aligned}$$

Features 3: Continuous Bag-of-Words Features

$$\mathbf{x} = \sum_{i=1}^{k} v(f_i; \theta) = \sum_{i=1}^{k} \delta(f_i)\mathbf{W}^0$$

- $\mathcal{F}$; the vocabulary of the language.
- $\mathbf{x} = \sum_i \delta(f_i)$

Example: Movie review input,

$$\mathbf{x} = v(\texttt{word:A}) + v(\texttt{word:sentimental}) + v(\texttt{word:mess})$$

$$\mathbf{x}^\top = \begin{bmatrix} 0.2 \\ \vdots \\ 1.2 \\ -0.5 \end{bmatrix} + \begin{bmatrix} 0.8 \\ \vdots \\ 1.0 \\ -1.0 \end{bmatrix} + \begin{bmatrix} 0.1 \\ \vdots \\ 9.2 \\ -2.0 \end{bmatrix} = \begin{bmatrix} 1.1 \\ \vdots \\ 11.4 \\ -3.5 \end{bmatrix}$$

Features 4: Continuous Bag-of-Bigrams Features?

Representation is counts of input bigrams,

- $\mathcal{F}$; the vocabulary of the bigram language.
- $\mathbf{x} = \sum_i \delta(f_i)$

Example: Movie review input,

`A sentimental mess`

$$\begin{aligned} \mathbf{x} &= v(\texttt{word:A}) + v_2(\texttt{bigram:A:sentimental}) \\ &+ v(\texttt{word:sentimental}) + v_2(\texttt{bigram:sentimental:mess}) \\ &+ v(\texttt{word:mess}) \end{aligned}$$

Neural Network

One-layer multi-layer perceptron architecture,

$$NN_{MLP1}(\mathbf{x}) = g(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)W^2 + \mathbf{b}^2$$

- $\mathbf{x}\mathbf{W} + \mathbf{b}$; *perceptron*
- $\mathbf{x}$ is the dense representation in $\mathbb{R}^{1 \times d_{\mathrm{in}}}$
- $\mathbf{W}^1 \in \mathbb{R}^{d_{\mathrm{in}} \times d_{\mathrm{hid}}}, \mathbf{b}^1 \in \mathbb{R}^{1 \times d_{\mathrm{hid}}}$; first affine transformation
- $\mathbf{W}^2 \in \mathbb{R}^{d_{\mathrm{hid}} \times d_{\mathrm{out}}}, \mathbf{b}^2 \in \mathbb{R}^{1 \times d_{\mathrm{out}}}$; second affine transformation
- $g : \mathbb{R}^{d_{\mathrm{hid}} \times d_{\mathrm{hid}}}$ is an *activation non-linearity* (often pointwise)
- $g(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)$ is the *hidden layer*

Contents

Windowed Classification

Alternative method, windows into MLP.

Goal: predict t_5.

- Windowed word model.

$$w_1 \; w_2 \; \left[w_3 \; w_4 \; w_5 \; w_6 \; w_7 \right] \; w_8$$

- w_3, w_4; left context
- w_5; Word of interest
- w_6, w_7; right context
- d_{win}; size of window ($d_{\mathrm{win}} = 5$)

All Window for Classification

Idea: Use window at each location.

$$\left[w_1\ w_2\ w_3\ w_4\ w_5\right]\ w_6\ w_7\ w_8$$

$$w_1\ \left[w_2\ w_3\ w_4\ w_5\ w_6\right]\ w_7\ w_8$$

$$w_1\ w_2\ \left[w_3\ w_4\ w_5\ w_6\ w_7\right]\ w_8$$

$$\vdots$$

Each maps from window of embeddings to d_{hid}

Convolution Formally

Let our input be the embeddings of the full sentence, $\mathbf{X} \in \mathbb{R}^{n \times d^0}$

$$\mathbf{X} = [v(w_1), v(w_2), v(w_3), \ldots, v(w_n)]$$

Define a window model as $NN_{window} : \mathbb{R}^{1 \times (d_{\mathrm{win}} d^0)} \mapsto \mathbb{R}^{1 \times d_{\mathrm{hid}}}$,

$$NN_{window}(\mathbf{x}_{win}) = \mathbf{x}_{win}\mathbf{W}^1 + \mathbf{b}^1$$

The convolution is defined as $NN_{conv} : \mathbb{R}^{n \times d^0} \mapsto \mathbb{R}^{(n - d_{\mathrm{win}} + 1) \times d_{\mathrm{hid}}}$,

$$NN_{conv}(\mathbf{X}) = \tanh \begin{bmatrix} NN_{window}(\mathbf{X}_{1:d_{\mathrm{win}}}) \\ NN_{window}(\mathbf{X}_{2:d_{\mathrm{win}}+1}) \\ \vdots \\ NN_{window}(\mathbf{X}_{n-d_{\mathrm{win}}:n}) \end{bmatrix}$$

Pooling

- Unfortunately $NN_{conv} : \mathbb{R}^{n \times d^0} \mapsto \mathbb{R}^{(n - d_{\text{win}} + 1) \times d_{\text{hid}}}$.
- Need to map down to d_{out} for different n
- Recall pooling operations.
- Pooling "over-time" operations $f : \mathbb{R}^{n \times m} \mapsto \mathbb{R}^{1 \times m}$
  1. $f_{max}(\mathbf{X})_{1,j} = \max_i X_{i,j}$
  2. $f_{min}(\mathbf{X})_{1,j} = \min_i X_{i,j}$
  3. $f_{mean}(\mathbf{X})_{1,j} = \sum_i X_{i,j} / n$

$$f(\mathbf{X}) = \begin{bmatrix} \Downarrow & \Downarrow & \dots \\ \Downarrow & \Downarrow & \dots \\ & \vdots & \\ \Downarrow & \Downarrow & \dots \end{bmatrix} = \begin{bmatrix} & \dots & \end{bmatrix}$$

Putting it together

$$\hat{y} = \mathrm{softmax}(f_{max}(NN_{conv}(\mathbf{X}))\mathbf{W}^2 + \mathbf{b}^2)$$

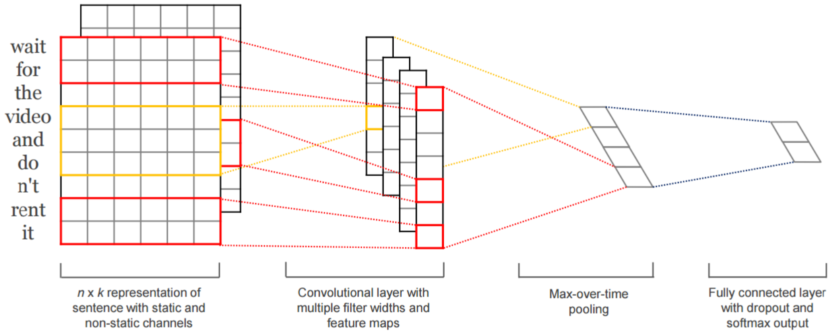
- $\mathbf{W}^2 \in \mathbb{R}^{d_{\mathrm{hid}} \times d_{\mathrm{out}}}$, $\mathbf{b}^2 \in \mathbb{R}^{1 \times d_{\mathrm{out}}}$
- Final linear layer $\mathbf{W}^2$ uses learned window features

Multiple Convolutions

$$\hat{y} = \mathrm{softmax}([f(NN^1_{conv}(\mathbf{X})), f(NN^2_{conv}(\mathbf{X})), \ldots, f(NN^f_{conv}(\mathbf{X}))]\mathbf{W}^2 + \mathbf{b}^2)$$

- Concat several convolutions together.
- Each NN^1, NN^2, etc uses a different d_{win}
- Allows for different window-sizes (similar to multiple n-grams)

Convolution Diagram (Kim, 2014)

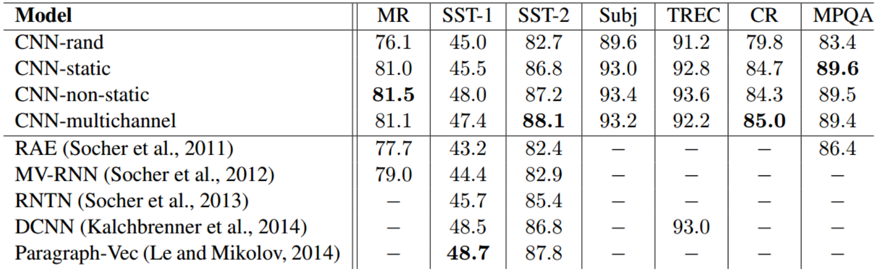

wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

- $n = 9$, $d_{\text{hid}} = 4$, $d_{\text{out}} = 2$
- red- $d_{\text{win}} = 2$, blue- $d_{\text{win}} = 3$, (ignore back channel)

Convolutional Vocabulary

- **kernel size** or **filter width** ; window size d_{win}
- **filter**; column of matrix $\mathbf{W}^1$ in $\mathbb{R}^{(d^0 \times d_{\mathrm{win}}) \times 1}$
- **feature map**; column of NN_{conv}, d_{hid} of these
- **fully-connected layer**; affine or linear $+$ activation
- **random**, **static**, **non-static**; embedding layer setup

Classification Results



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	–	–	–	–

Contents

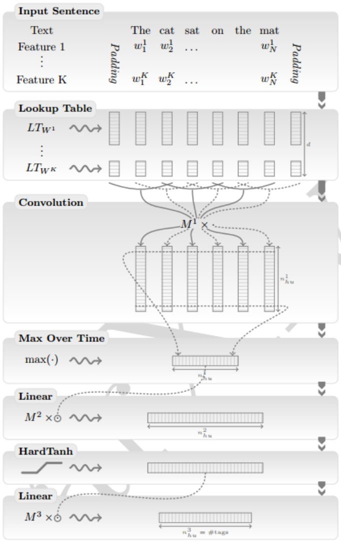
Language Applications: Semantic Role Labeling

He would n't accept anything of value from those he was writing about

[A0 He] [AM-MOD would] [AM-NEG n't] [V accept] [A1 anything of value] from [A2 those he was writing about]

- V: verb
- A0: acceptor
- A1: thing accepted
- A2: accepted-from
- A3:attribute
- AM-MOD: modal
- AM-NEG: negation

Other Language Applications (Collobert et al. 2011)



C&W SRL

- First given a verb w_i e.g. `accept`.
- Then consider a word w_j e.g. `n't`
- For a word w_k features are

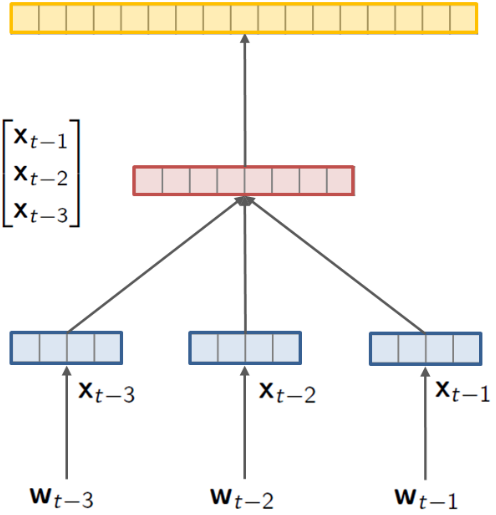
$$v(w_k), v_2(cap(w_k)), v_3(i-k), v_4(j-k)$$

- Convolution over sentence is used to predict role.
- $O(n \times |verbs|)$ convolutions per sentence

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

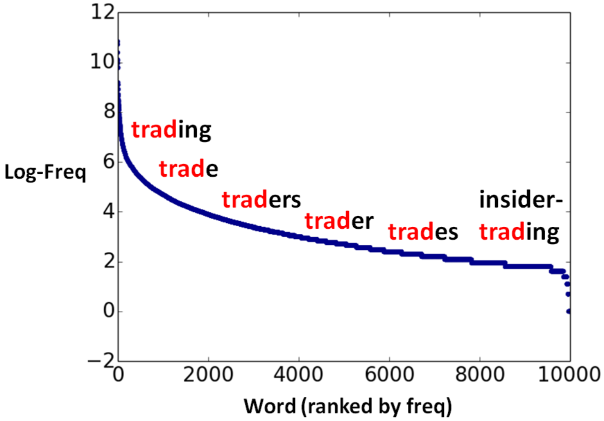
$$p(w_t | w_1, \ldots, w_{t-1}) = \text{softmax}(\mathbf{P}\mathbf{h}_{t-1} + \mathbf{q})$$

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



NLM Issue

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



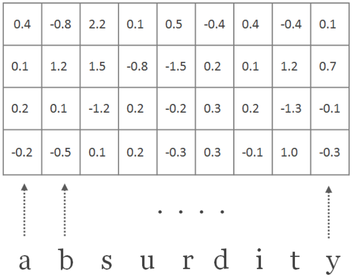
Separate embeddings for "trading", "trade", "trades", etc.

Character-level CNN (CharCNN)

a b s u r d i t y

Character-level CNN (CharCNN)

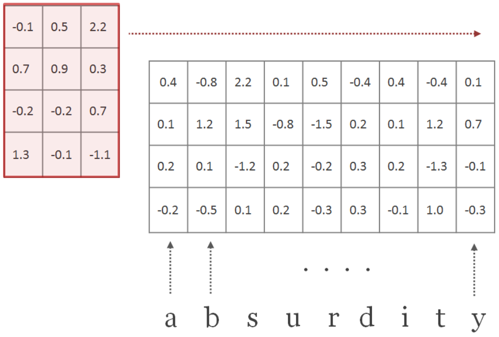
$\mathbf{C} \in \mathbb{R}^{d \times l}$: Representation of *absurdity*



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

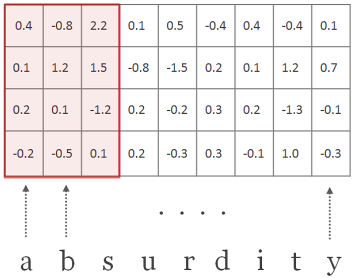
Character-level CNN (CharCNN)

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



Character-level CNN (CharCNN)

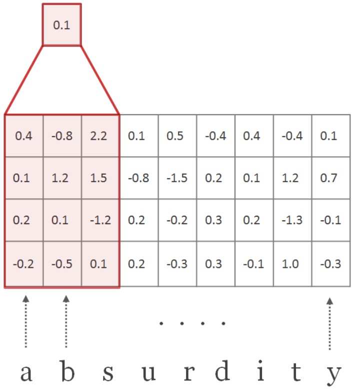
$$\mathbf{f}[1] = \langle \mathbf{C}[*, 1:3], \mathbf{H} \rangle$$



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

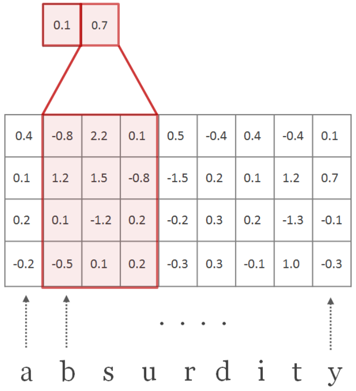
Character-level CNN (CharCNN)

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



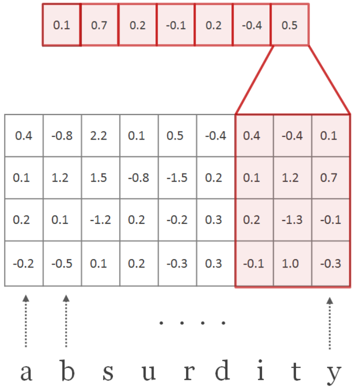
Character-level CNN (CharCNN)

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



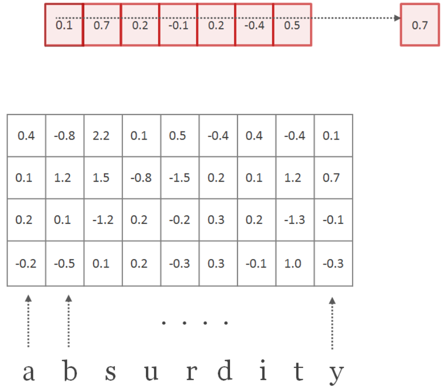
Character-level CNN (CharCNN)

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



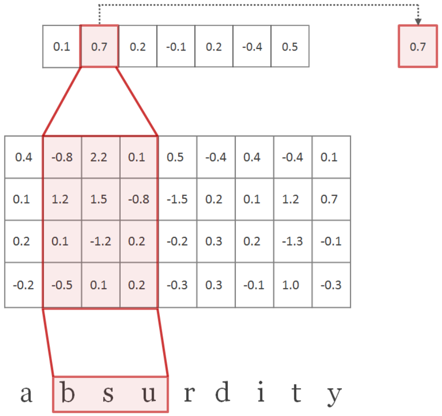
Character-level CNN (CharCNN)

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



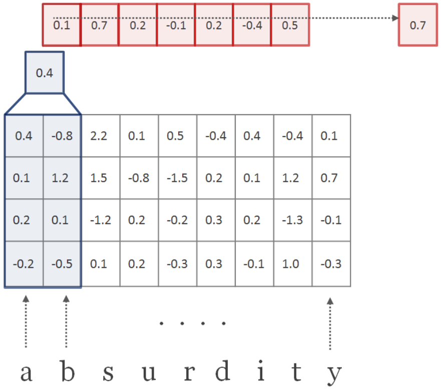
Character-level CNN (CharCNN)

Each filter picks out a character *n*-gram



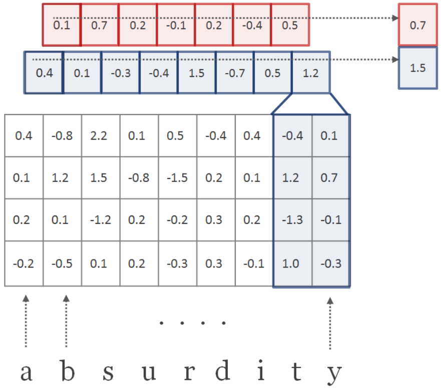
Character-level CNN (CharCNN)

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



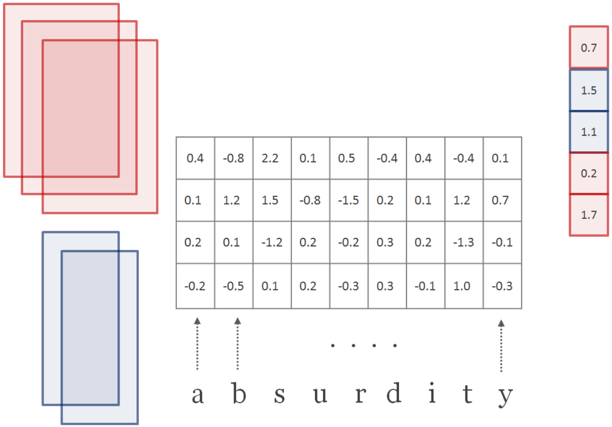
Character-level CNN (CharCNN)

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



Character-level CNN (CharCNN)

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



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)

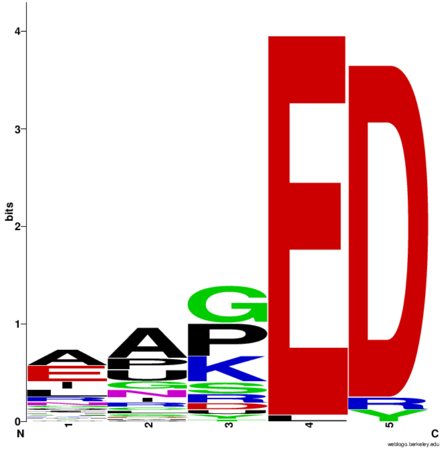
	Out-of-Vocabulary		
	computer-aided	*misinformed*	*looooook*
Characters (before highway)	*computer-guided*	*informed*	*look*
	computerized	*performed*	*cook*
	disk-drive	*transformed*	*looks*
	computer	*inform*	*shook*
Characters (after highway)	*computer-guided*	*informed*	*look*
	computer-driven	*performed*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

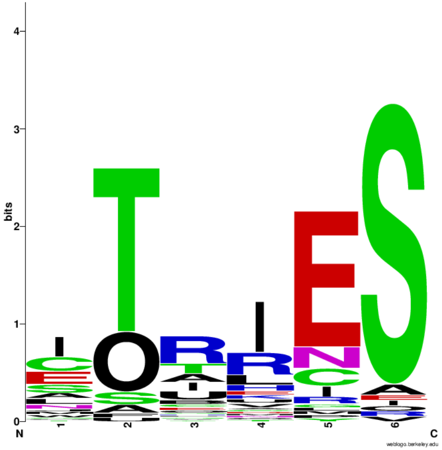
Convolutional Filters

For each filter, visualize 100 substrings with the highest filter response

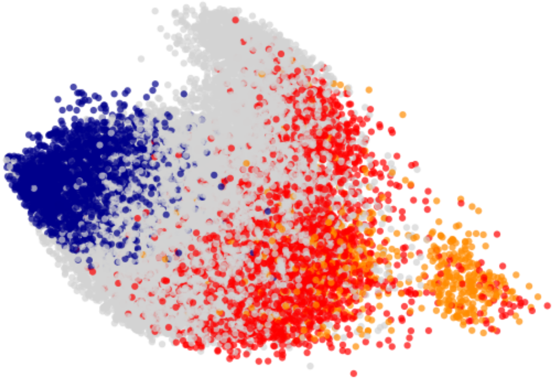


Convolutional Filters

For each filter, visualize 100 substrings with the highest filter response



Character *N*-gram Representations

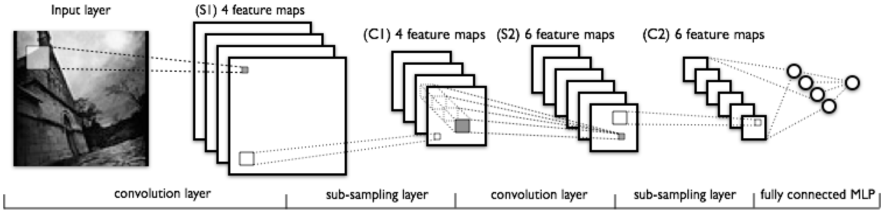


Prefixes, Suffixes, Hyphenated, Others

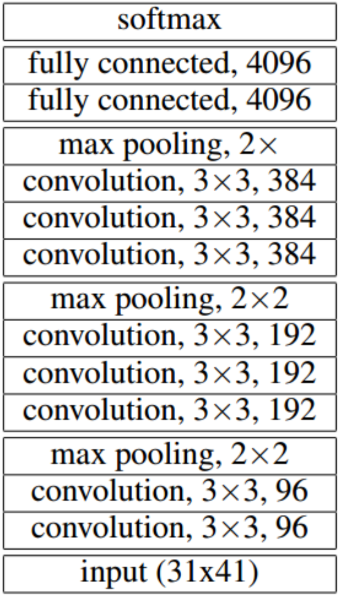
Prefixes: character *n*-grams that start with 'start-of-word' character, such as {*un*, {*mis*. Suffixes defined similarly.

Contents

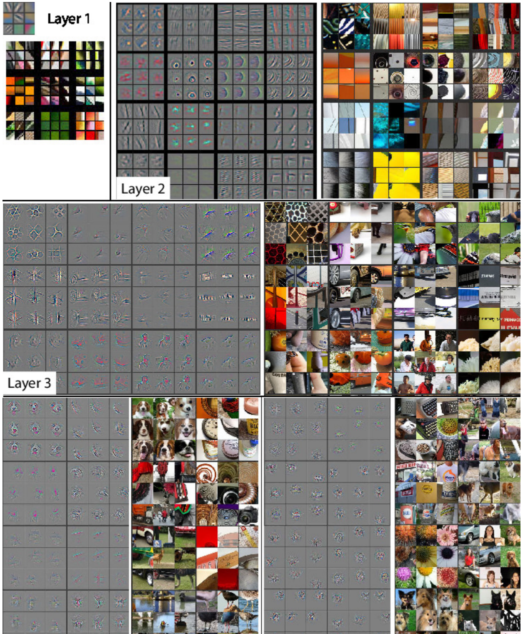
Visual Classification



Speech Convolutions



Visual Convolution


Layer 1
Layer 2
Layer 3