CS395T: Structured Models for NLP
Lecture 17: CNNs

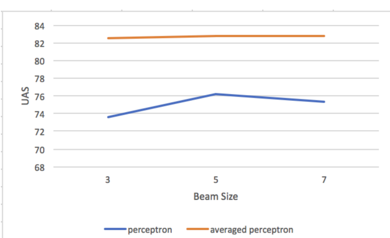
Greg Durrett

Project 2 Results

Top 3 scores:

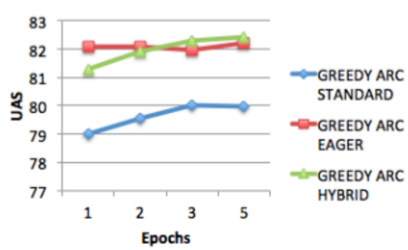
- Su Wang: 90.13 UAS
  Greedy logistic regression with extended feature set, trained for 30 epochs with Adagrad with a weight decay schedule
- Yasumasa Onoe: 89.58 UAS
  Greedy averaged perceptron, features looking at children + grandchildren on the stack, also three-way conjunctive POS features
- Prateek Shrishail Kolhar: 89.42 UAS
  Global model with beam size 5 + averaged perceptron, feature engineering with distance, valency, etc.

Project 2 Results



Subham Ghosh

- Model averaging helps a lot
- LR better than SVM for many students

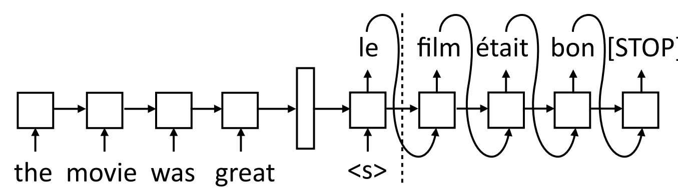


Tanya Goyal

- Other transition systems usually better than arc-standard

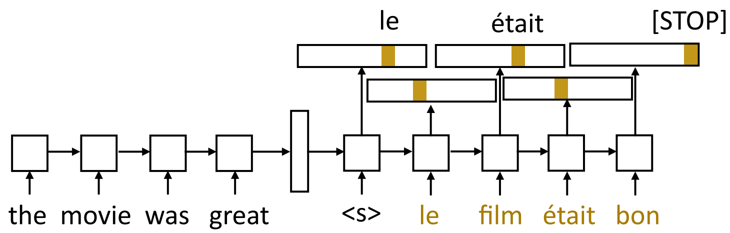
Recall: Seq2seq Models

- Generate next word conditioned on previous word as well as hidden state



- During inference: need to compute the argmax over the word predictions and then feed that to the next RNN state
- Need to actually evaluate computation graph up to this point to form input for the next state
- Decoder is advanced one state at a time until [STOP] is reached

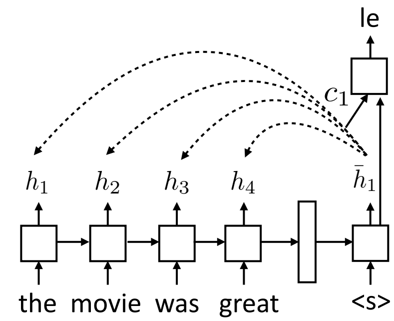
Recall: Seq2seq Training



- Objective: maximize $\log P(w_i^* | \mathbf{x}, w_{i-1}^*)$
- One loss term for each target-sentence word, feed the correct word regardless of model's prediction
- Length of gold sequence is known, can run the whole encoder-decoder in one computation graph and compute losses

Recall: Attention

- For each decoder state, compute a weighted sum of input states reflecting what's most important right now



$e_{ij} = f(\bar{h}_i, h_j)$
- Unnormalized scalar weight

$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$
- Normalized scalar weight

$c_i = \sum_j \alpha_{ij} h_j$
- Weighted sum of input hidden states (vector)

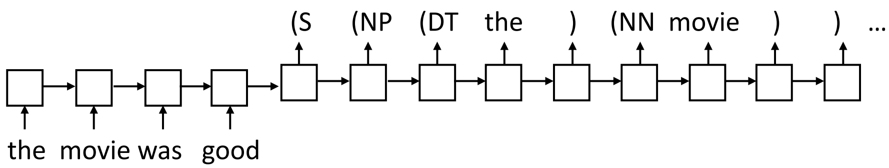
This Lecture

- Other RNN applications (finish up)
- CNNs
- CNNs for Sentiment
- Dilated CNNs for MT

Other RNN Applications

Parsing

- Parsing: input is a sentence, output is a bracketed sentence

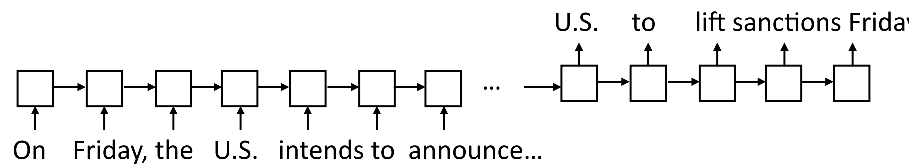


- Attention is essential: <70 F1 without it, 88.3 F1 / 90.5 F1 (ensemble) with it
- The best parsers still use some structure — we'll come back to these

Vinyals et al. (2014)

Summarization

- Summarization/compression: input is an article/sentence, output is a summary of the input



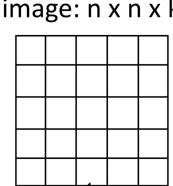
- Long articles, hard to deal with even with attention
- Speech recognition/text-to-speech: neural nets are good at dealing with continuous speech signals!

CNNs

Convolutional Layer

- Applies a *filter* over patches of the input and returns that filter's activations
- Convolution: take dot product of filter with a patch of the input

image: n x n x k



filter: m x m x k

sum over dot products

$$\text{activation}_{ij} = \sum_{i_o=0}^{k-1} \sum_{j_o=0}^{k-1} \text{image}(i+i_o, j+j_o) \cdot \text{filter}(i_o, j_o)$$

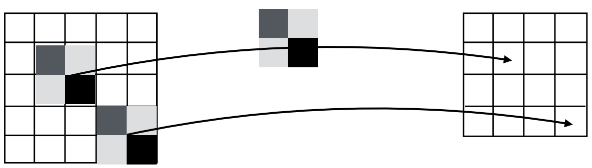
offsets

Each of these cells is a vector with multiple values
Images: RGB values (3 dim); text: word vector (50+ dim)

Convolutional Layer

- Applies a *filter* over patches of the input and returns that filter's activations
- Convolution: take dot product of filter with a patch of the input

image: n x n x k   filter: m x m x k   activations: (n - m + 1) x (n - m + 1) x 1



can generally have more filters

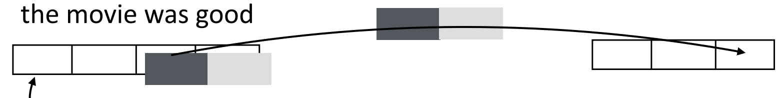
- "Narrow convolution" reduces input size, but can also preserve it

Convolutions for NLP

- Input and filter are 2-dimensional instead of 3-dimensional

sentence: n words x k vec dim   filter: m x k   activations: (n - m + 1) x 1

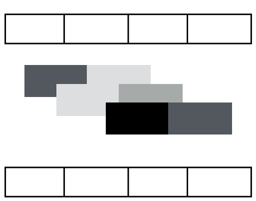
the movie was good



vector for each word

- Combines evidence locally in a sentence and produces a new (but still variable-length) representation
- Filters are like basis vectors: each filter computes each n-gram's value for that coordinate in the basis

Compare: LSTMs vs. CNNs

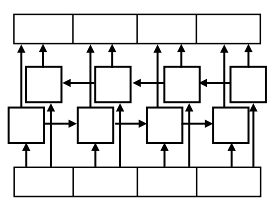


n x c

c filters,
m x k each

n x k

the movie was good



n x k

BiLSTM

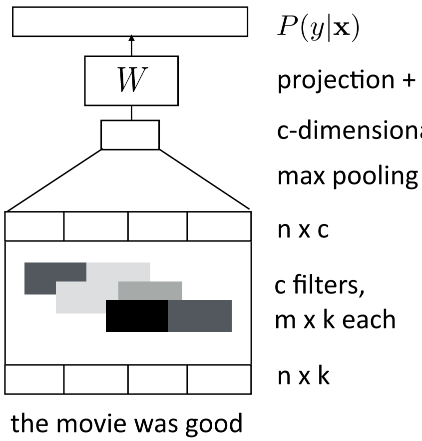
n x k

the movie was good

- Both LSTMs and convolutional layers transform the input using context
- LSTM: "global" in that it looks at the whole sentence (but largely local for many problems)
- CNN: local depending on filter width + number of layers

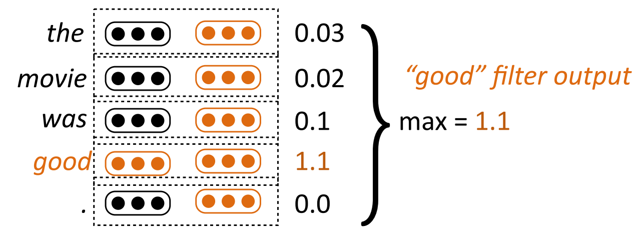
CNNs for Sentiment

CNNs for Sentiment Analysis



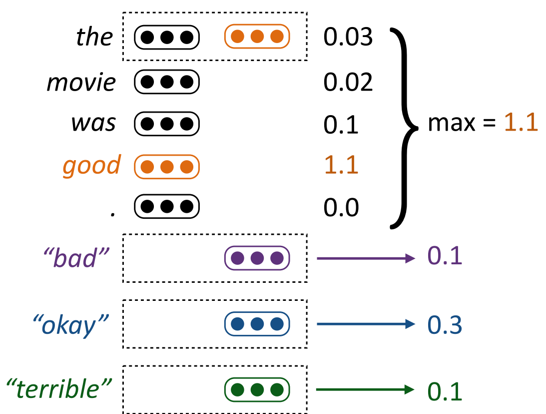
- Max pooling: return the max activation of a given filter over the entire sentence; like a logical OR (sum pooling is like logical AND)

Understanding CNNs for Sentiment

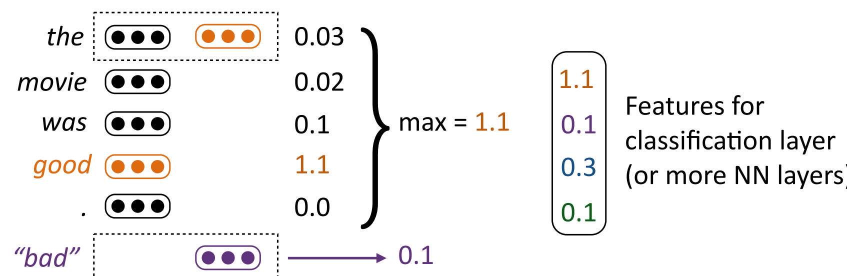


- Filter "looks like" the things that will cause it to have high activation

Understanding CNNs for Sentiment

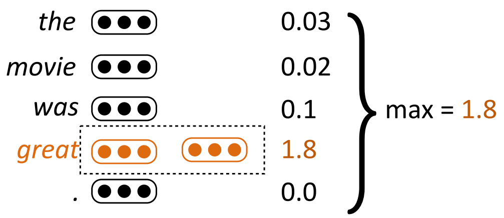


Understanding CNNs for Sentiment



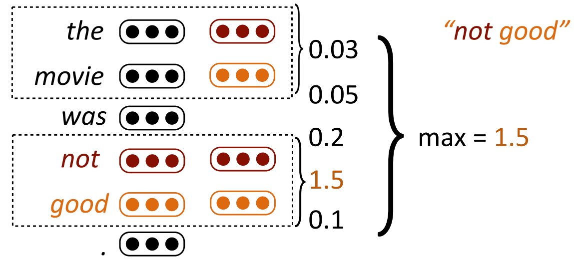
- Takes variable-length input and turns it into fixed-length output
- Filters are initialized randomly and then learned

Understanding CNNs for Sentiment



- Word vectors for similar words are similar, so convolutional filters will have similar outputs

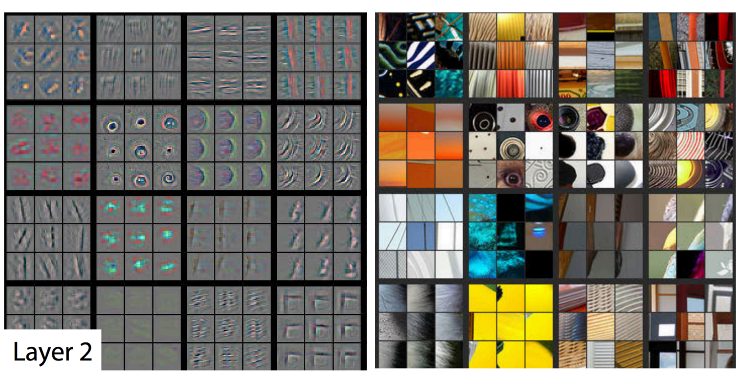
Understanding CNNs for Sentiment



- Analogous to bigram features in bag-of-words models
- Indicator feature of text containing bigram <-> max pooling of a filter that matches that bigram

Deep Convolutional Networks

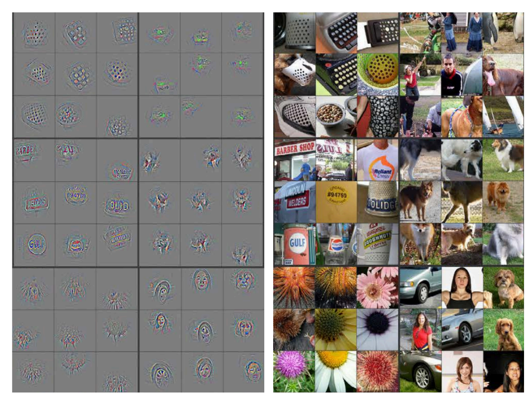
- Low-level filters: extract low-level features from the data



Zeiler and Fergus (2014)

Deep Convolutional Networks

- High-level filters: match larger and more "semantic patterns"



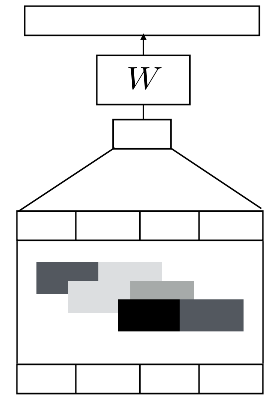
Zeiler and Fergus (2014)

CNNs: Implementation

- Input is batch_size x n x k matrix, filters are c x m x k matrix (c filters)
- Typically use filters with m ranging from 1 to 5 or so (multiple filter widths in a single convnet)
- Filters are initialized randomly, need to learn to pick up on appropriate patterns
- All computation graph libraries support efficient convolution operations

CNNs for Sentence Classification

- Question classification, sentiment, etc.
- Conv+pool, then use feedforward layers to classify
- Can use multiple types of input vectors (fixed initializer and learned)



the movie was good

Kim (2014)

Sentence Classification

movie review sentiment

subjectivity/objectivity detection

product reviews

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-multichannel	81.1	47.4	**88.1**	93.2	92.2	**85.0**	89.4
NBSVM (Wang and Manning, 2012)	79.4	–	–	93.2	–	81.8	86.3

question type classification

- Also effective at document-level text classification

Kim (2014)

Entity Linking

- CNNs can produce good representations of both sentences and documents like typical bag-of-words features
- Can distill topic representations for use in entity linking

that they had disqualified **Armstrong** from his seven consecutive

cycling domain

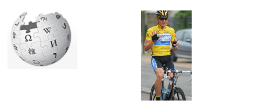
Lance Armstrong

geopolitical domain

Armstrong County

Entity Linking

Although he originally won the event, the United States Anti-Doping Agency announced in August 2012 that they had disqualified **Armstrong** from his seven consecutive Tour de France wins from 1999–2005.



Lance Edward Armstrong is an American former professional road cyclist

Armstrong County is a county in Pennsylvania…

CNN

Document topic vector d

CNN

Article topic vector a_{Lance}

CNN

Article topic vector a_{County}

$s_{\text{Lance}} = d \cdot a_{\text{Lance}}$ $\quad$ $s_{\text{County}} = d \cdot a_{\text{County}}$

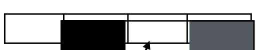
$P(y|\mathbf{x}) = \text{softmax}(\mathbf{s})$

Francis-Landau et al. (2016)

Dilated CNNs for MT

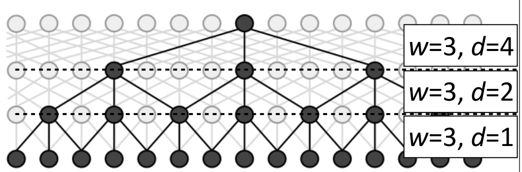
Dilated Convolutions

- Standard convolution: looks at every token under the filter
- Dilated convolution with gap *d*: looks at every *d*th token



w = 2, *d* =2: gap in the filter

- Can chain successive dilated convolutions together to get a wide receptive field (see a lot of the sentence)

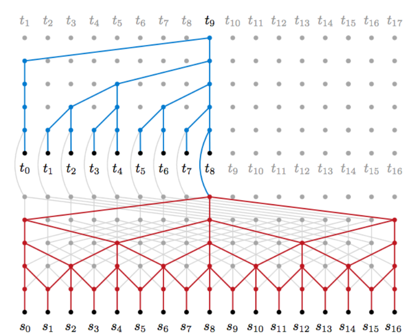


- Top nodes see lots of the sentence, but with different processing

Strubell et al. (2017)

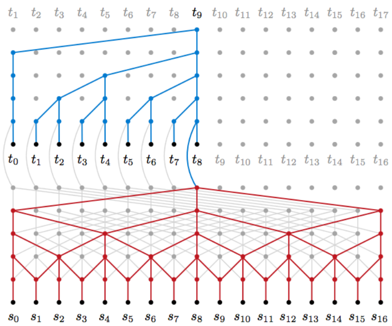
CNNs for Machine Translation

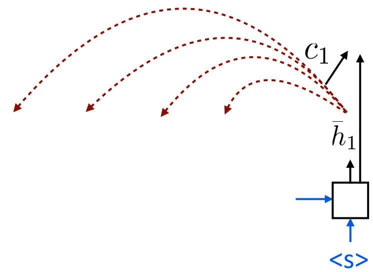
- "ByteNet": operates over characters (bytes)
- Encode source sequence w/dilated convolutions
- Predict *n*th target character by looking at the *n*th position in the source and a dilated convolution over the *n*-1 target tokens so far
- To deal with divergent lengths, t_n actually looks at $s_{n\alpha}$ where α is a heuristically-chosen parameter
- Assumes mostly monotonic translation



Kalchbrenner et al. (2016)

Compare: CNNs vs. LSTMs

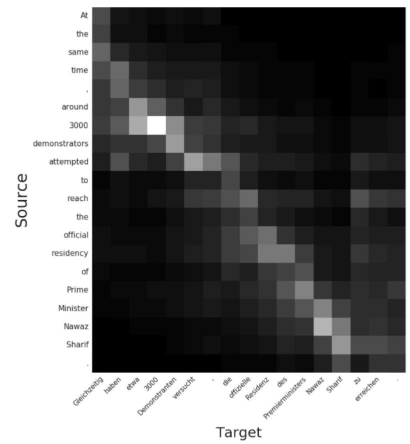




- CNN: source encoding at this position gives us "attention", target encoding gives us decoder context
- LSTM: looks at previous word + hidden state, attention over input

Kalchbrenner et al. (2016)

Attention from CNN

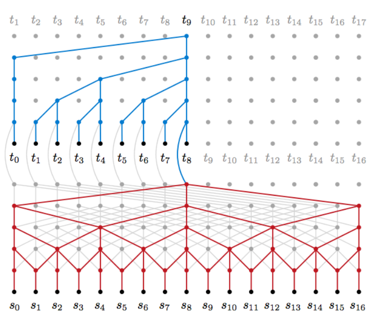


- Model is character-level, this visualization shows which words's characters impact the convolutional encoding the most
- Largely monotonic but does consult other information

Kalchbrenner et al. (2016)

Advantages of CNNs

- LSTM with attention is quadratic: compute attention over the whole input for each decoded token
- CNN is linear!
- CNN is shallower too in principle but the conv layers are very sophisticated (3 layers each)



Kalchbrenner et al. (2016)

English-German MT Results

Model	Inputs	Outputs	WMT Test '14
Phrase Based MT (Freitag et al., 2014; Williams et al., 2015)	phrases	phrases	20.7
RNN Enc-Dec (Luong et al., 2015)	words	words	11.3
Reverse RNN Enc-Dec (Luong et al., 2015)	words	words	14.0
RNN Enc-Dec Att (Zhou et al., 2016)	words	words	20.6
RNN Enc-Dec Att (Luong et al., 2015)	words	words	20.9
GNMT (RNN Enc-Dec Att) (Wu et al., 2016a)	word-pieces	word-pieces	**24.61**
RNN Enc-Dec Att (Chung et al., 2016b)	BPE	BPE	19.98
RNN Enc-Dec Att (Chung et al., 2016b)	BPE	char	21.33
GNMT (RNN Enc-Dec Att) (Wu et al., 2016a)	char	char	22.62
ByteNet	char	char	**23.75**

Kalchbrenner et al. (2016)

Up Next

- Next lecture: Ye will talk about using neural networks in lower-resource settings
- After that: advanced neural network structures
  - Tree-structured RNNs
  - Neural CRFs
  - Memory networks, etc.