CS388: Natural Language Processing
Lecture 9: CNNs, Neural CRFs

Greg Durrett

Administrivia

- Project 1 due today at 5pm
- Mini 2 out tonight, due in two weeks

This Lecture

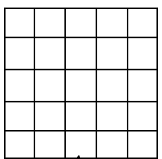
- CNNs
- CNNs for Sentiment
- Neural CRFs

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

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

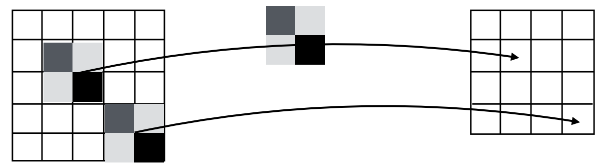
offsets

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

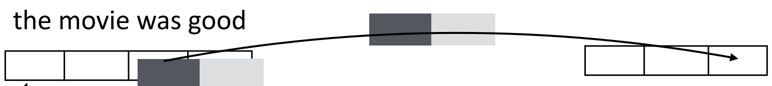


Convolutions for NLP

- 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

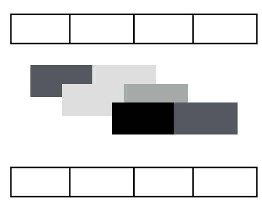
the movie was good



vector for each word

- Combines evidence locally in a sentence and produces a new (but still variable-length) representation

Compare: CNNs vs. LSTMs

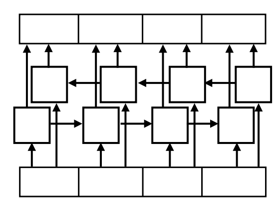


O(n) x c

c filters,
m x k each

n x k

the movie was good



n x 2c

BiLSTM with
hidden size c

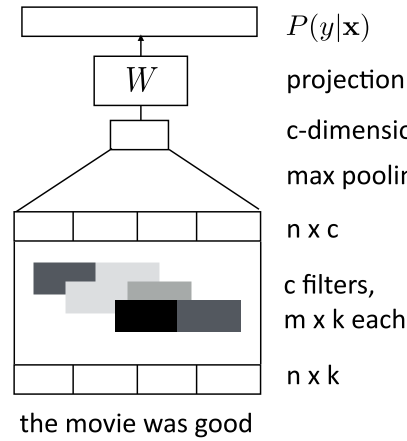
n x k

the movie was good

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

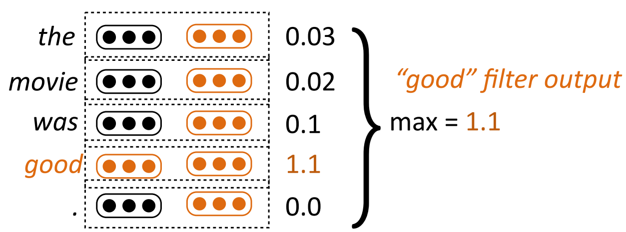
CNNs for Sentiment

CNNs for Sentiment Analysis



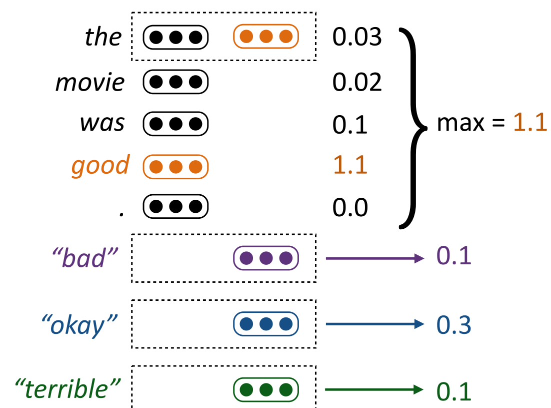
- Max pooling: return the max activation of a given filter over the entire sentence; like a logical OR (sum pooling is like logical AND)

Understanding CNNs for Sentiment

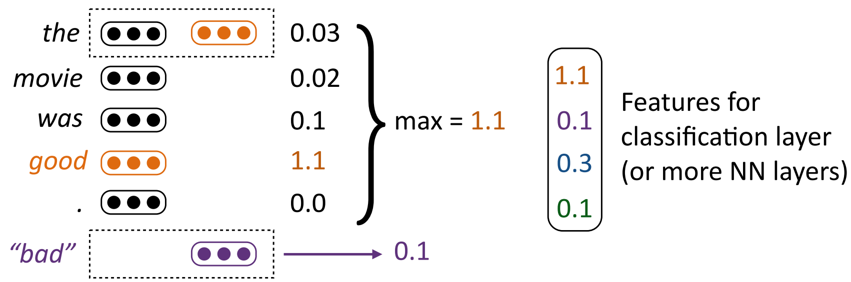


- Filter "looks like" the things that will cause it to have high activation

Understanding CNNs for Sentiment

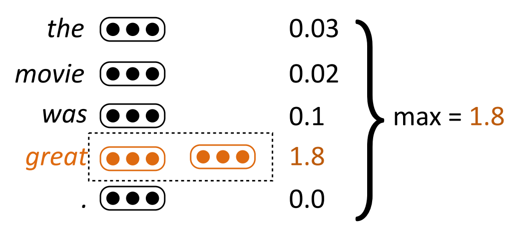


Understanding CNNs for Sentiment



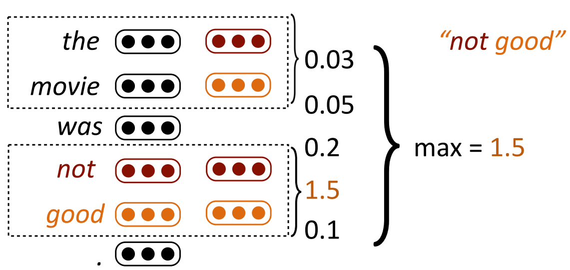
- Takes variable-length input and turns it into fixed-length output
- Filters are initialized randomly and then learned

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

What can CNNs learn?

the movie was not good

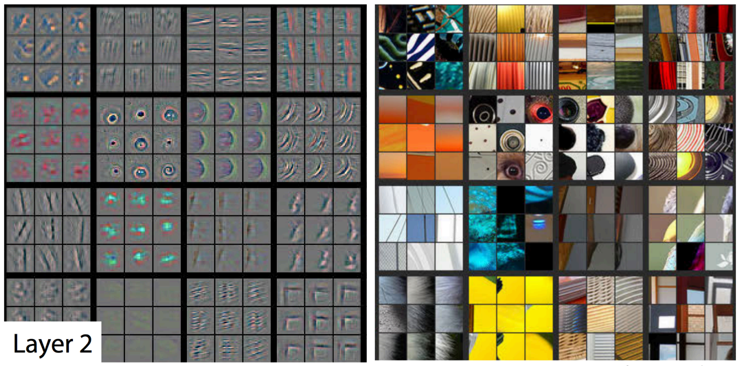
the movie was not really all that good

the cinematography was good, the music great, but the movie was bad

I entered the theater in the bloom of youth and left as an old man

Deep Convolutional Networks

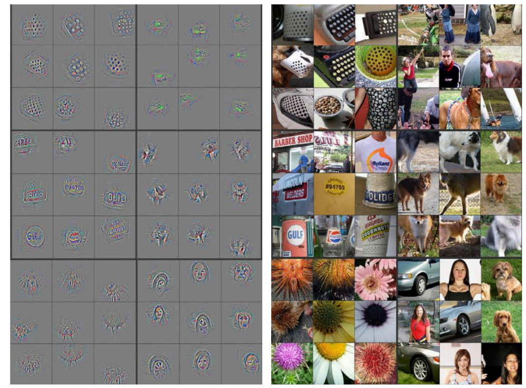
- Low-level filters: extract low-level features from the data



Zeiler and Fergus (2014)

Deep Convolutional Networks

- High-level filters: match larger and more “semantic patterns”



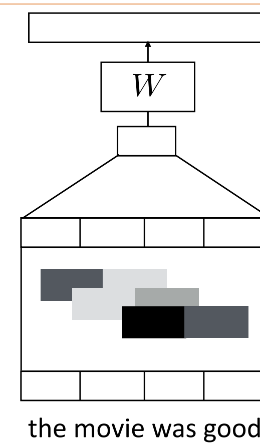
Zeiler and Fergus (2014)

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)
- 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)



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)

Neural CRF Basics

NER Revisited

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON LOC ORG

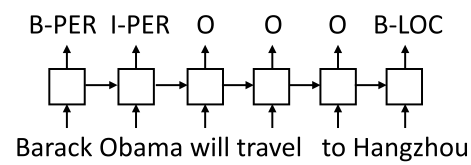
- Features in CRFs: I[tag=B-LOC & curr_word=*Hangzhou*], I[tag=B-LOC & prev_word=*to*], I[tag=B-LOC & curr_prefix=*Han*]
- Linear model over features
- Downsides:
  - Lexical features mean that words need to be seen in the training data
  - Linear model can't capture feature conjunctions as effectively (can't look at more than 2 words with a single feature)

LSTMs for NER

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON LOC ORG



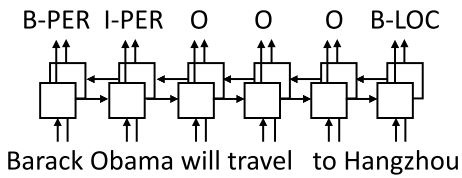
- Transducer (LM-like model)
- What are the strengths and weaknesses of this model compared to CRFs?

LSTMs for NER

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON LOC ORG



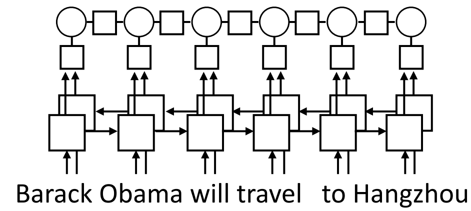
- Bidirectional transducer model
- What are the strengths and weaknesses of this model compared to CRFs?

Neural CRFs

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

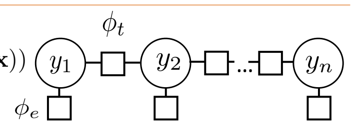
PERSON LOC ORG



- Neural CRFs: bidirectional LSTMs (or some NN) compute emission potentials, capture structural constraints in transition potentials

Neural CRFs

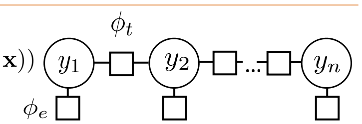
$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\prod_{i=2}^{n}\exp(\phi_t(y_{i-1}, y_i))\prod_{i=1}^{n}\exp(\phi_e(y_i, i, \mathbf{x}))$$



- Conventional: $\phi_e(y_i, i, \mathbf{x}) = w^\top f_e(y_i, i, \mathbf{x})$
- Neural: $\phi_e(y_i, i, \mathbf{x}) = W_{y_i}^\top f(i, \mathbf{x})$ *W* is a num_tags x len(*f*) matrix
- *f*(*i*, ***x***) could be the output of a feedforward neural network looking at the words around position *i*, or the *i*th output of an LSTM, …
- Neural network computes unnormalized potentials that are consumed and "normalized" by a structured model
- Inference: compute *f*, use Viterbi

Computing Gradients

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\prod_{i=2}^{n}\exp(\phi_t(y_{i-1}, y_i))\prod_{i=1}^{n}\exp(\phi_e(y_i, i, \mathbf{x}))$$



- Conventional: $\phi_e(y_i, i, \mathbf{x}) = w^\top f_e(y_i, i, \mathbf{x})$
- Neural: $\phi_e(y_i, i, \mathbf{x}) = W_{y_i}^\top f(i, \mathbf{x})$

$$\frac{\partial \mathcal{L}}{\partial \phi_{e,i}} = -P(y_i = s|\mathbf{x}) + I[s \text{ is gold}]$$ "error signal", compute with F-B

- For linear model: $\frac{\partial \phi_{e,i}}{w_i} = f_{e,i}(y_i, i, \mathbf{x})$

chain rule say to multiply together, gives our update

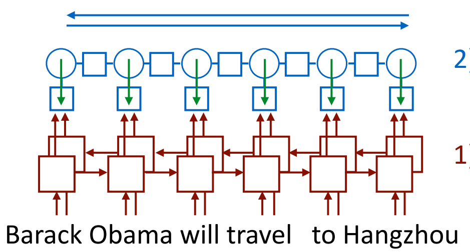
- For neural model: compute gradient of phi w.r.t. parameters of neural net

Neural CRFs

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON LOC ORG



2) Run forward-backward

3) Compute error signal

1) Compute f(**x**)

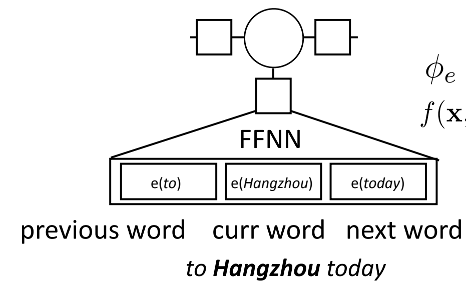
4) Backprop (no knowledge of sequential structure required)

FFNN Neural CRF for NER

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON LOC ORG



$$\phi_e = W g(V f(\mathbf{x}, i))$$

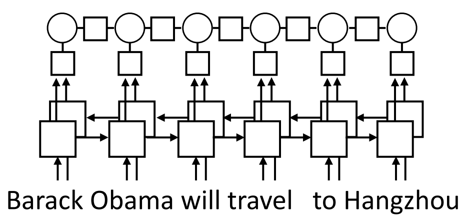
$$f(\mathbf{x}, i) = [\mathrm{emb}(\mathbf{x}_{i-1}), \mathrm{emb}(\mathbf{x}_i), \mathrm{emb}(\mathbf{x}_{i+1})]$$

LSTM Neural CRFs

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON LOC ORG



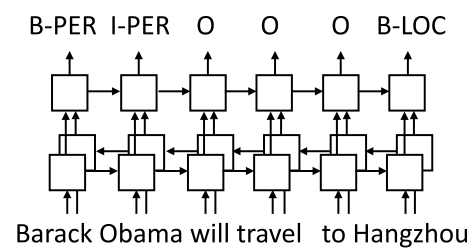
- Bidirectional LSTMs compute emission (or transition) potentials

LSTMs for NER

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

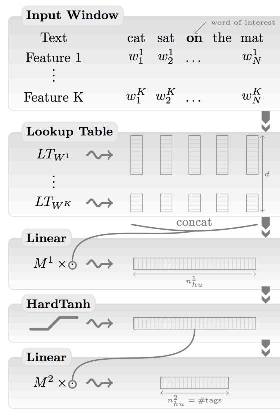
PERSON LOC ORG



- How does this compare to neural CRF?

"NLP (Almost) From Scratch"

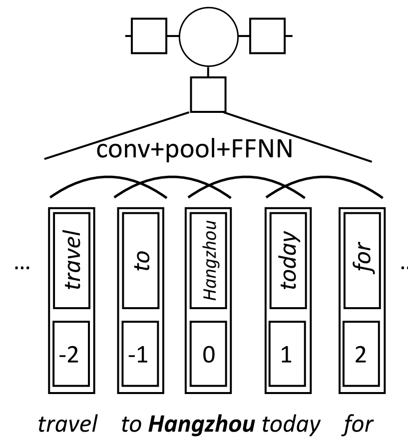
Approach	POS (PWA)	CHUNK (F1)	NER (F1)	SRL (F1)
Benchmark Systems	97.24	94.29	89.31	77.92
NN+WLL	96.31	89.13	79.53	55.40
NN+SLL	96.37	90.33	81.47	70.99
NN+WLL+LM1	97.05	91.91	85.68	58.18
NN+SLL+LM1	97.10	93.65	87.58	73.84
NN+WLL+LM2	97.14	92.04	86.96	58.34
NN+SLL+LM2	97.20	93.63	88.67	74.15



- WLL: independent classification; SLL: neural CRF
- LM2: word vectors learned from a precursor to word2vec/GloVe, trained for 2 weeks (!) on Wikipedia

Collobert, Weston, et al. 2008, 2011

CNN Neural CRFs



- Append to each word vector an *embedding of the relative position* of that word
- Convolution over the sentence produces a position-dependent representation

CNN NCRFs vs. FFNN NCRFs

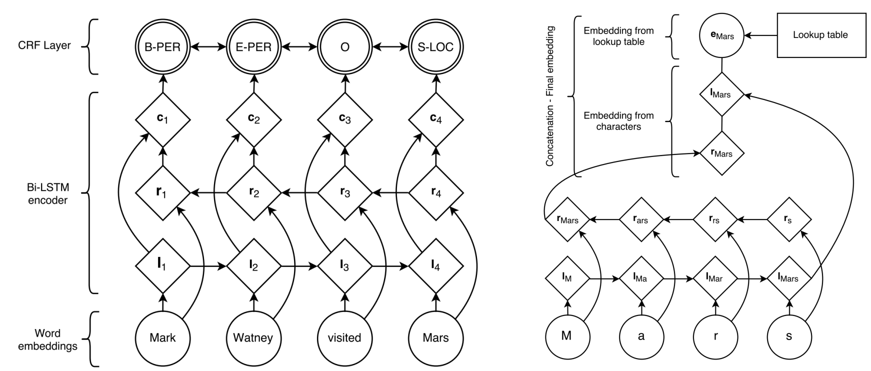
Approach	POS (PWA)	CHUNK (F1)	NER (F1)	SRL (F1)
Benchmark Systems	97.24	94.29	89.31	77.92
		Window Approach		
NN+SLL+LM2	97.20	93.63	88.67	–
		Sentence Approach		
NN+SLL+LM2	97.12	93.37	88.78	74.15

- Sentence approach (CNNs) is comparable to window approach (FFNNs) except for SRL where they claim it works much better

Collobert and Weston 2008, 2011

Neural CRFs with LSTMs

- Neural CRF using character LSTMs to compute word representations



Chiu and Nichols (2015), Lample et al. (2016)

Neural CRFs with LSTMs

- Chiu+Nichols: character CNNs instead of LSTMs
- Lin/Passos/Luo: use external resources like Wikipedia
- LSTM-CRF captures the important aspects of NER: word context (LSTM), sub-word features (character LSTMs), outside knowledge (word embeddings)

Model	$\mathbf{F_1}$
Collobert et al. (2011)*	89.59
Lin and Wu (2009)	83.78
Lin and Wu (2009)*	90.90
Huang et al. (2015)*	90.10
Passos et al. (2014)	90.05
Passos et al. (2014)*	90.90
Luo et al. (2015)* + gaz	89.9
Luo et al. (2015)* + gaz + linking	**91.2**
Chiu and Nichols (2015)	90.69
Chiu and Nichols (2015)*	90.77
LSTM-CRF (no char)	90.20
LSTM-CRF	**90.94**

Chiu and Nichols (2015), Lample et al. (2016)

Takeaways

- CNNs are a flexible way of extracting features analogous to bag of n-grams, can also encode positional information
- All kinds of NNs can be integrated into CRFs for structured inference. Can be applied to NER, other tagging, parsing, …
- This concludes the ML/DL-heavy portion of the course. Starting Tuesday: syntax, then semantics