Deep Networks 3

10-405
Summary

• The basic architecture:
  – Matrix operations implemented on GPU
  – Minibatch SGD optimization
  – Autodiff to get gradients

• This lets you write expressive models easily
  – ...but some models are hard to train...
  – Which models should you use, and why?
Deep Network Training Tricks
Recap: weight updates for multilayer ANN

For nodes $k$ in output layer $L$:
\[
\delta^L_k \equiv (t_k - a_k) \ a_k \ (1 - a_k)
\]

For nodes $j$ in hidden layer $h$:
\[
\delta^h_j \equiv \sum_k (\delta^{h+1}_j \ w_{kj}) \ a_j \ (1 - a_j)
\]

What happens as the layers get further and further from the output layer? E.g., what’s gradient for the bias term with several layers after it?

\[
\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}
\]
Gradients are unstable

Max at 1/4

If weights are less than 4 then we are multiplying by many numbers < 1 so the gradients get very small.

The vanishing gradient problem

What happens as the layers get further and further from the output layer? E.g., what’s gradient for the bias term with several layers after it in a trivial net?

\[
\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}
\]
Gradients are unstable

What happens as the layers get further and further from the output layer? E.g., what's gradient for the bias term with several layers after it in a trivial net?

\[
\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}
\]

If weights are large then we are multiplying by many numbers > 1 so the gradients get very big.

The **exploding gradient** problem (less common but possible)
Vanishing gradients

Histogram of gradients in a 5-layer network for an artificial image recognition task

Solution? “do as much as you can with networks with linear gradients” - wcohen
Understanding the difficulty of training deep feedforward neural networks

We will get to these tricks eventually....
Saturation and initialization

It’s easy for sigmoid units to saturate

Learning rate approaches zero and unit is “stuck”
It’s easy for sigmoid units to saturate

- Saturation visualization from Glorot & Bengio 2010

Closest-to-output hidden layer still stuck for first 2M examples
It’s easy for sigmoid units to saturate

For a big network there are lots of weighted inputs to each neuron. If any of them are too large then the neuron will saturate. So neurons get stuck with a few large inputs OR many small ones.
It’s easy for sigmoid units to saturate

• If there are 500 non-zero inputs initialized with a Gaussian \( \sim N(0,1) \) then the SD is \( \sqrt{500} \approx 22.4 \)

• Fix: initialize carefully! e.g.

\[
W_{ij} \sim U \left[ -\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}} \right]
\]

• Or (at level \( j \)):

\[
W \sim U \left[ -\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}} \right]
\]
First breakthrough deep learning results were based on clever pre-training initialization schemes, where deep networks were seeded with weights learned from unsupervised strategies.

Tanh vs Tanh N: only difference is initialization!
Cross-entropy loss

\[ C = \frac{1}{n} \sum_x [y \ln a + (1 - y) \ln(1 - a)], \]

\[ C = \frac{1}{n} \sum_x (y - a)^2 \]

1990’s ANNs often used quadratic loss at the output layer.

Figure 5: Cross entropy (black, surface on top) and quadratic (red, bottom surface) cost as a function of two weights (one at each layer) of a network with two layers, \( W_1 \) respectively on the first layer and \( W_2 \) on the second, output layer.
Network outputs a probability distribution!

Cross-entropy loss after a softmax layer gives a very simple, numerically stable gradient: \( Y - P \)
Softmax output layer

tf.nn.softmax_cross_entropy_with_logits

Inputs: X,W1,B1,W2,B2
Z1a = mul(X,W1)  // matrix mult
Z1b = add*(Z1a,B1)  // add bias vec
A1 = tanh(Z1b)  //element-wise
Z2a = mul(A1,W2)
Z2b = add*(Z2a,B2)
A2 = tanh(Z2b)  // element-wise
P = softMax(A2)  // vec to vec
C = crossEnt_{Y}(P)  // cost function
**Softmax output layer**

\[ \text{Inputs: } X, W_1, B_1, W_2, B_2 \]

\[ Z_{1a} = \text{mul}(X, W_1) \quad \text{matrix mult} \]

\[ Z_{1b} = \text{add}^{*}(Z_{1a}, B_1) \quad \text{add bias vec} \]

\[ A_1 = \tanh(Z_{1b}) \quad \text{element-wise} \]

\[ Z_{2a} = \text{mul}(A_1, W_2) \]

\[ Z_{2b} = \text{add}^{*}(Z_{2a}, B_2) \]

\[ A_2 = \tanh(Z_{2b}) \quad \text{element-wise} \]

\[ P = \text{softmax}(A_2) \quad \text{vec to vec} \]

\[ C = \text{crossEnt}_{Y}(P) \quad \text{cost function} \]
Alternative non-linearities

- A new change: modifying the nonlinearity
  - The logistic is not widely used in modern ANNs

Alternate non-linearities

- Zero’s don’t get you stuck

Alternate 1: tanh

Like logistic function but shifted to range \([-1, +1]\)
Understanding the difficulty of training deep feedforward neural networks

Alternative non-linearities

softsign(x) = x/(1 + |x|)
Alternative non-linearities

• A new change: modifying the nonlinearity — reLU often used in vision tasks

Alternate 2: rectified linear unit

Linear with a cutoff at zero

(Implement: clip the gradient when you pass zero)
Alternative non-linearities

- A new change: modifying the nonlinearity
  - reLU often used in vision tasks

Alternate 2: rectified linear unit

Soft version: $\log(\exp(x) + 1)$

Doesn’t saturate (at one end)
Sparsifies outputs
Helps with vanishing gradient
Deep Architectures
Word2Vec and GloVe Embeddings
Representing words in a deep network

"1 hot" vector

weights

hidden layer

1 aaliyeh 0
2 aardvark 0
3 aaai 0
... ... ...
18462 halloween 1
18463 hallows 0
... ... ...
... ... ...
29999 zymurgy 0
30000 zynga 0

The embeddings will be similar for words that behave similarly with respect to the downstream task.

\[ h = xW \]

But really \( h \) is the \( i \)-th row of \( W \). So learning \( W \) is just learning a hidden-layer encoding for each word in the vocabulary (embedding).
Representing words in a deep network

Potential downstream task: predict the words that co-occur with input word.
word2vec: skip-gram embeddings

A word in context “I had class on Halloween which seemed unfair” becomes an example
**word2vec: skip-gram embeddings**

**Training data:**
- **positive** examples are pairs of words $w(t), w(t+j)$ that co-occur.
- **negative** examples are samples of pairs of words $w(t), w(t+j)$ that don’t co-occur.

You want to train over a very large corpus (100M words+) and hundreds+ dimensions.

Often you initialize with this word representation and then “fine tune” - ie plug in the real downstream task and keep training....
GLOVE embeddings

\[ f(x) = \begin{cases} 
(x/x_{\text{max}})^{\alpha} & \text{if } x < x_{\text{max}} \\
1 & \text{otherwise}
\end{cases} \]

\[ J = \sum_{i,j=1}^{V} f(X_{ij}) \left( w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij} \right)^2 \]

how often words \( i \) and \( j \) co-occur in a corpus

how much to weight this word pair, based on frequency

embeddings for words \( i \) and \( j \)

biases for words \( i \) and \( j \)
RECURRENT NEURAL NETWORKS
Motivation: what about sequence prediction?

What can I do when input size and output size vary?
Motivation: what about sequence prediction?
Architecture for an RNN

Some information is passed from one subunit to the next

Sequence of outputs

Sequence of inputs

Start of sequence marker

End of sequence marker

http://colah.github.io/posts/2015-08-Understanding-LSTMs/
Architecture for an 1980’s RNN

Problem with this: it’s extremely deep and very hard to train
Architecture for an LSTM

“Bits of memory”

Decide what to forget
Decide what to insert

Longterm-short term model

σ: output in [0,1]
tanh: output in [-1,+1]

http://colah.github.io/posts/2015-08-Understanding-LSTMs/
Walkthrough

What part of memory to “forget” – zero means forget this bit

\[ f_t = \sigma \left( W_f \cdot [h_{t-1}, x_t] + b_f \right) \]

\( \sigma = \) logistic function, range [0,1]
What bits to insert into the next states

\[ i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i) \]

\[ \tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \]

What content to store into the next state

http://colah.github.io/posts/2015-08-Understanding-LSTMs/
Walkthrough

Next memory cell content – mixture of not-forgotten part of previous cell and insertion

\[ C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \]

\[(a'b)' = a'b + ab'\]
Walkthrough

What part of cell to output

tanh maps bits to [-1,+1] range

\[
o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)
\]

\[
h_t = o_t \times \tanh(C_t)
\]

http://colah.github.io/posts/2015-08-Understanding-LSTMs/
Architecture for an LSTM

(1)

\[ f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f) \]
\[ i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i) \]
\[ \tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \]

(2)

\[ C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \]

(3)

\[ o_t = \sigma (W_o \cdot [h_{t-1}, x_t] + b_o) \]
\[ h_t = o_t \cdot \tanh(C_t) \]
Implementing an LSTM

For $t = 1, \ldots, T$:

1. $f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$
   $i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i)$
   $\tilde{C}_t = \tanh (W_C \cdot [h_{t-1}, x_t] + b_C)$

2. $C_t = f_t \ast C_{t-1} + i_t \ast \tilde{C}_t$

3. $o_t = \sigma (W_o \cdot [h_{t-1}, x_t] + b_o)$
   $h_t = o_t \ast \tanh (C_t)$
SOME FUN LSTM EXAMPLES

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
LSTMs can be used for other sequence tasks

- **image captioning**
  - One to many

- **sequence classification**
  - Many to one

- **translation**
  - Many to many

- **named entity recognition**
  - Many to many

seq2seq

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Character-level language model

Test time:
• pick a seed character sequence
• generate the next character
• then the next
• then the next …

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Character-level language model

PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Character-level language model

First Citizen:
Nay, then, that was hers,
It speaks against your other service:
But since the
youth of the circumstance be spoken:
Your uncle and one Baptista's daughter.

SEBASTIAN:
Do I stand till the break off.

BIRON:
Hide thy head.

VENTIDIUS:
He purposeth to Athens: whither, with the vow
I made to handle you.

FALSTAFF:
My good knave.

Yoav Goldberg: order-10 unsmoothed character n-grams

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Character-level language model

-------------------- Recipe via Meal-Master (tm) v8.05

Title: BARBECUE RIBS
Categories: Chinese, Appetizers
Yield: 4 Servings

1 pk Seasoned rice
1 Beer -- cut into
cubes
1 ts Sugar
3/4 c Water
Chopped finels,
-up to 4 tblsp of chopped
2 pk Yeast Bread/over

--------------------FILLING--------------------

2 c Pineapple, chopped
1/3 c Milk
1/2 c Pecans
Cream of each
2 tb Balsamic cocoa
2 tb Flour
2 ts Lemon juice
Granulated sugar
2 tb Orange juice

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Character-level language model

LaTeX “almost compiles”

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m*} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on $X, U$ is a closed immersion of $S$, then $U \rightarrow T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \text{Spec}(R) = U \times_X U \times_X U$$

and the comparically in the fibre product covering we have to prove the lemma generated by $\bigsqcup Z \times_U U \rightarrow V$. Consider the maps $M$ along the set of points $Sch_{f_{ppf}}$ and $U \rightarrow U$ is the fibre category of $S$ in $U$ in Section, ?? and the fact that any $U$ affine, see Morphisms, Lemma ???. Hence we obtain a scheme $S$ and any open subset $W \subset U$ in $Sh(G)$ such that $\text{Spec}(R') \rightarrow S$ is smooth or an

$$U = \bigcup U_i \times_{S, U_i}$$

which has a nonzero morphism we may assume that $f_i$ is of finite presentation over $S$. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \rightarrow \mathcal{O}_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $GL_{S'}(x'/S'')$ and we win.

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\tilde{M}^\bullet = \mathcal{T}^\bullet \otimes_{\text{Spec}(k)} \mathcal{O}_{S,s} - i_{X}^{-1} \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

Arrows = $(Sch/S)_{f_{ppf}}^{opp}, (Sch/S)_{f_{ppf}}$

and

$$V = \Gamma(S, \mathcal{O}) \hookrightarrow (U, \text{Spec}(A))$$

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Character-level language model

```c
/**
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
    int error;
    if (fd == MARN_EPT) {
        /*
         * The kernel blank will coeld it to userspace.
         */
        if (ss->segment < mem_total)
            unblock_graph_and_set_blocked();
        else
            ret = 1;
        goto bail;
    }
    segaddr = in_SB(in.addr);
    selector = seg / 16;
    setup_works = true;
    for (i = 0; i < blocks; i++) {
        seq = buf[i++];
        bpf = bd->bd.next + i * search;
        if (fd) {
            current = blocked;
        }
    }
}
```
CONVOLUTIONAL NEURAL NETWORKS
Model of vision in animals

[Hubel & Wiesel 1962]:
- **Simple cells** detect local features
- **Complex cells** “pool” the outputs of simple cells within a retinotopic neighborhood.

```
```

Retinotopic Feature Maps

```
```

Multiple convolutions

pooling subsampling

```
```

“Simple cells”

“Complex cells”
What’s a convolution?

• Basic idea:
  – Pick a 3-3 matrix F of weights
  – Slide this over an image and compute the “inner product” (similarity) of F and the corresponding field of the image, and replace the pixel in the center of the field with the output of the inner product operation

• Key point:
  – Different convolutions extract different types of low-level “features” from an image
  – All that we need to vary to generate these different features is the weights of F
How do we convolve an image with an ANN?

Note that the parameters in the matrix defining the convolution are tied across all places that it is used.
What’s a convolution?

http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo
What’s a convolution?

http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo
What’s a convolution?

http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo
What’s a convolution?

http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo
What’s a convolution?

http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo
What’s a convolution?

http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo
What’s a convolution?

• Basic idea:
  – Pick a 3-3 matrix $F$ of weights
  – Slide this over an image and compute a “feature map” for that image

• Key point:
  – Different convolutions extract different types of low-level “features” from an image
  – Weights of convolutions will be learned
Usually we construct many “feature maps” for each image...
Example: 6 convolutions of a digit

http://scs.ryerson.ca/~aharley/vis/conv/
CNNs typically alternate convolutions, non-linearity, and then downsampling

Downsampling is usually averaging or (more common in recent CNNs) max-pooling
Why do max-pooling?

- Saves space
- Reduces overfitting?
- Because I’m going to add more convolutions after it!
  - Allows the short-range convolutions to extend over larger subfields of the images
    - So we can spot larger objects
    - Eg, a long horizontal line, or a corner, or …
- At some point the feature maps start to get very sparse and blobby – they are indicators of some semantic property, not a recognizable transformation of the image
- Then just use them as features in a “normal” ANN
Why do max-pooling?

• Saves space
• Reduces overfitting?
• Because I’m going to add more convolutions after it!
  – Allows the short-range convolutions to extend over larger subfields of the images
    • So we can spot larger objects
    • Eg, a long horizontal line, or a corner, or …

PROC. OF THE IEEE, NOVEMBER 1998

Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.
Alternating convolution and downsampling

5 layers up

The subfield in a large dataset that gives the strongest output for a neuron
Using RNNs and CNNs
LSTMs can be used for other tasks

<table>
<thead>
<tr>
<th>Encoder/Decoder</th>
<th>seq2seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>image captioning</td>
<td>sequence classification</td>
</tr>
<tr>
<td>one to many</td>
<td>many to one</td>
</tr>
<tr>
<td></td>
<td>many to many</td>
</tr>
<tr>
<td></td>
<td>many to many</td>
</tr>
</tbody>
</table>

Image

[CNN]

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
ANN Tricks for NLP

• Common tricks
  – represent words with embeddings
  – represent words in context with RNN hidden state
  – represent a sentence with the last hidden state
    • or pool all hidden states with MAX or SUM
  – biLSTM: run an LSTM in both directions
    • represent with first + last hidden state

  – feed representations into a deeper network....
Example: reasoning about entailment

A large annotated corpus for learning natural language inference

Samuel R. Bowman*†
sbowman@stanford.edu

Gabor Angeli†‡
angeli@stanford.edu

Christopher Potts*
cgpotts@stanford.edu

Christopher D. Manning*†‡
manning@stanford.edu

<table>
<thead>
<tr>
<th>Example</th>
<th>Annotation</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>A man inspects the uniform of a figure in some East Asian country.</td>
<td>contradiction</td>
<td>C C C C C</td>
</tr>
<tr>
<td>An older and younger man smiling.</td>
<td>neutral</td>
<td>N N E N N</td>
</tr>
<tr>
<td>A black race car starts up in front of a crowd of people.</td>
<td>contradiction</td>
<td>C C C C C</td>
</tr>
<tr>
<td>A soccer game with multiple males playing.</td>
<td>entailment</td>
<td>E E E E E</td>
</tr>
<tr>
<td>A smiling costumed woman is holding an umbrella.</td>
<td>neutral</td>
<td>N N E C N</td>
</tr>
<tr>
<td>The man is sleeping</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Two men are smiling and laughing at the cats playing on the floor.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A man is driving down a lonely road.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Some men are playing a sport.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A happy woman in a fairy costume holds an umbrella.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
RNNs for entailment

<table>
<thead>
<tr>
<th>Sentence model</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>100d Sum of words</td>
<td>79.3</td>
<td>75.3</td>
</tr>
<tr>
<td>100d RNN</td>
<td>73.1</td>
<td>72.2</td>
</tr>
<tr>
<td>100d LSTM RNN</td>
<td>84.8</td>
<td>77.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System</th>
<th>SNLI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Distance Based</td>
<td>71.9</td>
</tr>
<tr>
<td>Classifier Based</td>
<td>72.2</td>
</tr>
<tr>
<td>+ Lexical Resources</td>
<td>75.0</td>
</tr>
</tbody>
</table>
Example: question answering

**LSTM-based Deep Learning Models for non-factoid answer selection**

Ming Tan, Cicero dos Santos, Bing Xiang & Bowen Zhou
IBM Watson Core Technologies
Yorktown Heights, NY, USA
{mingtan,cicerons,bingxia,zhou}@us.ibm.com

Common trick: train network to make representations similar/dissimilar, not to classify
Example: question answering

Adding **attention**:  
• classify the hidden states $h_1, \ldots h_m$ of the answer according to relevance to the question  
• when you pool, weight by the classifier’s score  
• classifier is based on question representation $o_q$ and hidden state $h_i$
Example: question answering

<table>
<thead>
<tr>
<th>Model</th>
<th>Validation</th>
<th>Test1</th>
<th>Test2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Bag-of-word</td>
<td>31.9</td>
<td>32.1</td>
<td>32.2</td>
</tr>
<tr>
<td>B. Metzler-Bendersky IR model</td>
<td>52.7</td>
<td>55.1</td>
<td>50.8</td>
</tr>
<tr>
<td>C. Architecture-II in (Feng et al., 2015)</td>
<td>61.8</td>
<td>62.8</td>
<td>59.2</td>
</tr>
<tr>
<td>D. Architecture-II with GESD</td>
<td>65.4</td>
<td>65.3</td>
<td>61.0</td>
</tr>
<tr>
<td>A  QA-LSTM basic-model(head/tail)</td>
<td>54.0</td>
<td>53.1</td>
<td>51.2</td>
</tr>
<tr>
<td>B  QA-LSTM basic-model(avg pooling)</td>
<td>58.5</td>
<td>58.2</td>
<td>54.0</td>
</tr>
<tr>
<td>C  QA-LSTM basic-model(max pooling)</td>
<td>64.3</td>
<td>63.1</td>
<td>58.0</td>
</tr>
<tr>
<td>G  QA-LSTM with attention (max pooling)</td>
<td>66.5</td>
<td>63.7</td>
<td>60.3</td>
</tr>
<tr>
<td>H  QA-LSTM with attention (avg pooling)</td>
<td><strong>68.4</strong></td>
<td><strong>68.1</strong></td>
<td>62.2</td>
</tr>
</tbody>
</table>
Example: recommendation

Rose Catherine & Cohen, RecSys 2017
Example: recommendation

Rose Catherine & Cohen, RecSys 2017
# Example: recommendation

Rose Catherine & Cohen, RecSys 2017

---

## Dataset Performance

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DeepCoNN + Test Reviews</th>
<th>MF</th>
<th>DeepCoNN</th>
<th>DeepCoNN-rev$_{AB}$</th>
<th>TransNet</th>
<th>TransNet-Ext</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yelp17</td>
<td>1.2106</td>
<td>1.8661</td>
<td>1.8984</td>
<td>1.7045</td>
<td>1.6387</td>
<td>1.5913</td>
</tr>
<tr>
<td>AZ-Elec</td>
<td>0.9791</td>
<td>1.8898</td>
<td>1.9704</td>
<td>2.0774</td>
<td>1.8380</td>
<td>1.7781</td>
</tr>
<tr>
<td>AZ-CSJ</td>
<td>0.7747</td>
<td>1.5212</td>
<td>1.5487</td>
<td>1.7044</td>
<td>1.4487</td>
<td>1.4780</td>
</tr>
<tr>
<td>AZ-Mov</td>
<td>0.9392</td>
<td>1.4324</td>
<td>1.3611</td>
<td>1.5276</td>
<td>1.3599</td>
<td>1.2691</td>
</tr>
</tbody>
</table>