Neural Networks for Information Retrieval
Neural Networks for Information Retrieval

Hosein Azarbonyad\textsuperscript{1} Alexey Borisov\textsuperscript{1,2} Mostafa Dehghani\textsuperscript{1} Tom Kenter\textsuperscript{3}
Bhaskar Mitra\textsuperscript{4,5} Maarten de Rijke\textsuperscript{1} Christophe Van Gysel\textsuperscript{1}

\textsuperscript{1}University of Amsterdam
\textsuperscript{2}Yandex
\textsuperscript{3}Booking.com
\textsuperscript{4}University College London
\textsuperscript{5}Microsoft

NN4IR@ECIR2018
Who’s here today and who are we?

Tom Kenter
Booking.com

Maarten de Rijke
U. Amsterdam

Bhaskar Mitra
Microsoft & UCL

Christophe Van Gysel
U. Amsterdam
Aims

Describe the use of neural network based methods in information retrieval

Compare architectures

Summarize where the field is, what seems to work, what seems to fail

Provide an overview of applications and directions for future development of neural methods in information retrieval
Structure of the tutorial

Morning
1. Preliminaries
2. Semantic matching
3. Learning to rank
4. Entities

Afternoon
5. Modeling user behavior
6. Generating responses
7. Recommender systems
8. Industry insights
9. Q & A
Materials

Slides available at http://nn4ir.com/ecir2018

Bibliography available at http://nn4ir.com/ecir2018

Survey 1 An introduction to neural information retrieval [Mitra and Craswell, 2017]
Survey 2 Neural information retrieval: at the end of the early years [Onal et al., 2017]
Outline

Morning program
   Preliminaries
   Semantic matching
   Learning to rank
   Entities

Afternoon program
   Modeling user behavior
   Generating responses
   Recommender systems
   Industry insights
   Q & A
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Outline

Morning program
Preliminaries
  Feedforward neural network
  Distributed representations
  Recurrent neural networks
  Sequence-to-sequence models
  Convolutional neural networks

Semantic matching
Learning to rank
Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
Q & A
Multi-layer perceptron a.k.a. feedforward neural network

Preliminaries

weights

hidden layer

output layer

input: $x$

weights

node $j$ at level $i$

output/prediction: $\hat{y}$

target: $y$

cost function
e.g.: $1/2 (\hat{y} - y)^2$

$w_{1,4}$

$\phi$: activation function
e.g.: sigmoid

$\frac{1}{1 + e^{o}}$

$\phi(o_{i,j})$

$o_{i,j} = \sum_{k=1}^{K} w_{i,k} \cdot x_{i-1,k}$

node $j$ at level $i$

$y_1$

$y_2$

$y_3$

$x_{1}$

$x_{2}$

$x_{3}$

$x_{4}$

$y_{1}$

$\hat{y}_{1}$

$\hat{y}_{2}$

$\hat{y}_{3}$

$x_{i-1,1}$

$x_{i-1,2}$

$x_{i-1,3}$

$x_{i-1,4}$

$y_1$ output/prediction:

$y_2$

$y_3$

target:

$y$

$\hat{y}$

output/prediction:

$\hat{y}$

cost function
e.g.: $1/2 (\hat{y} - y)^2$

weights
Multi-layer perceptron a.k.a. feedforward neural network

Preliminaries

ground truth: $\vec{y}$

$\begin{array}{ccc}
\hat{y}[0] & \hat{y}[1] & \hat{y}[2] \\
\end{array}$

output layer/predictions $\hat{\vec{x}}_3$

$\begin{array}{ccc}
\hat{x}_3[0] & \hat{x}_3[1] & \hat{x}_3[2] \\
\end{array}$

activation

$\hat{x}_3 = \sigma(\hat{o}_2) = [1 \times 4]$  
$\hat{x}_2 \cdot W_2 = \hat{o}_2$  
$[1 \times 4] \times [4 \times 4] = [1 \times 4]$  
$\hat{x}_2 = \sigma(\hat{o}_1) = [1 \times 4]$  
$\hat{x}_1 \cdot W_1 = \hat{o}_1$  
$[1 \times 4] \times [4 \times 4] = [1 \times 4]$  
$\phi: \text{activation function, e.g.: sigmoid}$  
$\frac{1}{1 + e^{-o}}$

node $j$ at level $i$

$\begin{array}{ccc}
x_{i-1,1} & x_{i-1,2} & x_{i-1,3} & x_{i-1,4} \\
\end{array}$

$\begin{array}{ccc}
x_{i,j} \\
\end{array}$

$\begin{array}{ccc}
o_{i,j} = \sum_{k=1}^{K} w_{i,k} \cdot x_{i-1,k} \\
\end{array}$

$\phi: \text{activation function, e.g.: sigmoid}$  
$\frac{1}{1 + e^{-o}}$
Outline

Morning program

Preliminaries
  Feedforward neural network
  Distributed representations
  Recurrent neural networks
  Sequence-to-sequence models
  Convolutional neural networks

Semantic matching
Learning to rank
Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Distributed representations

- Represent units, e.g., words, as vectors
- Goal: words that are similar, e.g., in terms of meaning, should get similar embeddings

Cosine similarity to determine how similar two vectors are:

$$\text{cosine}(\vec{v}, \vec{w}) = \frac{\vec{v}^\top \cdot \vec{w}}{\|\vec{v}\|_2 \|\vec{w}\|_2}$$

$$= \frac{\sum_{i=1}^{\|v\|} v_i \cdot w_i}{\sqrt{\sum_{i=1}^{\|v\|} v_i^2} \sqrt{\sum_{i=1}^{\|w\|} w_i^2}}$$

newspaper = <0.08, 0.31, 0.41>
magazine = <0.09, 0.35, 0.36>
biking = <0.59, 0.25, 0.01>
Distributed representations

How do we get these vectors?

- You shall know a word by the company it keeps [Firth, 1957]
- The vector of a word should be similar to the vectors of the words surrounding it

\[ \text{all} \quad \text{you} \quad \text{need} \quad \text{is} \quad \text{love} \]
Embedding methods

target distribution

vocabulary size probability distribution

vocabulary size layer

embedding size hidden layer

vocabulary size inputs

**Embedding size × vocabulary size weight matrix**

?? turn this into a probability distribution ??

**vocabulary size × embedding size weight matrix**
**Probability distributions**

**softmax** = normalize the logits

\[\text{softmax} = \frac{e^{\text{logits}[i]}}{\sum_{j=1}^{\text{logits}} e^{\text{logits}[j]}}\]

**cost** = cross entropy loss

\[\text{cost} = - \sum_{x} p(x) \log \hat{p}(x)\]

\[= - \sum_{i} p_{\text{ground truth}}(\text{word} = \text{vocabulary}[i]) \log p_{\text{predictions}}(\text{word} = \text{vocabulary}[i])\]

\[= - \sum_{i} y_i \log \hat{y}_i\]
Outline

Morning program

Preliminaries

- Feedforward neural network
- Distributed representations
- Recurrent neural networks
- Sequence-to-sequence models
- Convolutional neural networks

Semantic matching
Learning to rank
Entities

Afternoon program

- Modeling user behavior
- Generating responses
- Recommender systems
- Industry insights
- Q & A
Recurrent neural networks

- Recurrent neural networks (RNNs) are typically used in scenarios where a sequence of inputs and/or outputs is being modelled.
- RNNs have memory that captures information about what has been computed so far.
- RNNs are called recurrent because they perform the same task for every element of the sequence, with output dependent on previous computations.
- RNNs can, in theory, make use of information in arbitrarily long sequences – in practice, however, they are limited to looking back only a few steps.
Recurrent cell

**Preliminaries**

- **Recurrent cell**
- **rnn cell**
- **(language modelling)**
- **word vector**
- **input vector**
- **output vector**
- **output vector 2**
- **hidden state vector:** $f(\text{word vector}, \text{input vector})$
- **softmax**
- **distribution over vocabulary**
- **target distribution**
- **cross entropy loss**
- **next word**
- **distribution over vocabulary**
- **output vector 2**
- **target distribution**
- **cross entropy loss**
- **next word**
Recurrent cell

\begin{equation}
\text{hidden state vector: } f(\text{hidden state}_{t-1}, \text{input vector})
\end{equation}

\text{softmax}

\text{distribution over vocabulary}

\text{target distribution}

\text{cross entropy loss}

\text{output vector 2}

\text{next word}

\text{word vector}
Recurrent neural networks

- RNN being unrolled (or unfolded) into full network
- Unrolling: write out network for complete sequence

Formulas governing computation:

- $x_t$ input at time step $t$
- $s_t$ hidden state at time step $t$ – memory of the network, calculated based on
  - input at the current step
  - previous hidden state
- $f$ usually nonlinearity, e.g., tanh or ReLU
  $f$ can also be LSTM or GRU.

Image credits: Nature
Language modeling using RNNs

- **Language model** allows us to predict probability of observing sentence (in a given dataset) as: 
  \[ P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i | w_1, \ldots, w_{i-1}) \]

- In RNN, set \( o_t = x_{t+1} \): we want output at step \( t \) to be actual next word

- **Cross-entropy loss** as loss function

- Training RNN similar to training a traditional NN: backpropagation algorithm, but with small twist:
  parameters shared by all time steps, so gradient at each output depends on calculations of previous time steps: **Backpropagation Through Time**
Vanishing and exploding gradients

- For training RNNs, calculate gradients for $U$, $V$, $W$ – ok for $V$ but for $W$ and $U$ ...

- Gradients for $W$:

$$\frac{\partial L_3}{\partial W} = \frac{\partial L_3}{\partial o_3} \frac{\partial o_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W} = \sum_{k=0}^{3} \frac{\partial L_3}{\partial o_3} \frac{\partial o_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}$$

- More generally:

$$\frac{\partial L}{\partial s_t} = \frac{\partial L}{\partial s_m} \cdot \frac{\partial s_m}{\partial s_{m-1}} \cdot \frac{\partial s_{m-1}}{\partial s_{m-2}} \cdots \frac{\partial s_{t+1}}{\partial s_t} \Rightarrow \ll 1$$

$$< 1 \quad < 1 \quad < 1$$

- Gradient contributions from far away steps become zero: state at those steps doesn’t contribute to what you are learning

Long Short Term Memory [Hochreiter and Schmidhuber, 1997]

LSTMs designed to combat vanishing gradients through gating mechanism

\[
\begin{align*}
i_t &= \sigma(x_t U^i + s_{t-1} W^i) \\
f_t &= \sigma(x_t U^f + s_{t-1} W^f) \\
o_t &= \sigma(x_t U^o + s_{t-1} W^o) \\
g_t &= \tanh(x_t U^g + s_{t-1} W^g) \\
c_t &= c_{t-1} \circ f + g \circ i \\
s_t &= \tanh(c_t) \circ o
\end{align*}
\]

(\circ is elementwise multiplication)

Image credits:
https://commons.wikimedia.org/wiki/File:Peephole_Long_Short-Term_Memory.svg
Gated Recurrent Units

- GRU layer quite similar to that of LSTM layer, as are the equations:

\[
\begin{align*}
z &= \sigma(x_t U^z + s_{t-1} W^z) \\
r &= \sigma(x_t U^r + s_{t-1} W^r) \\
h &= \tanh(x_t U^h + (s_{t-1} \circ r) W^h) \\
s_t &= (1 - z) \circ h + z \circ s_{t-1}
\end{align*}
\]

- GRU has **two gates**: reset gate \( r \) and **update gate** \( z \).
  - Reset gate determines how to combine new input with previous memory; update gate defines how much of the previous memory to keep around
  - Set reset to all 1’s and update gate to all 0’s to get plain RNN model

- On many tasks, LSTMs and GRUs perform similarly
Bidirectional RNNs

- **Bidirectional RNNs** based on idea that output at time $t$ may depend on previous and future elements in sequence
  - Example: predict missing word in a sequence
- Bidirectional RNNs are two RNNs stacked on top of each other
- Output is computed based on hidden state of both RNNs

Outline

Morning program
  Preliminaries
    Feedforward neural network
    Distributed representations
    Recurrent neural networks
    Sequence-to-sequence models
    Convolutional neural networks
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Sequence-to-sequence models

Increasingly important: not just retrieval but also generation

- Machine translation, spoken results, chatbots, conversational interfaces, . . . , but also snippets, query suggestion, query correction, . . .

Basic sequence-to-sequence (seq2seq) model consists of two RNNs: an encoder that processes input and a decoder that generates output:

Each box represents cell of RNN (often GRU cell or LSTM cell). Encoder and decoder can share weights or, as is more common, use a different set of parameters

Image credits: Adapted from https://www.tensorflow.org/tutorials/seq2seq
Attention mechanism Bahdanau et al. [2014]

Normal decoder model:

\[ h_t = f(x, h_{t-1}; \theta), \]

Decoder with attention:

\[ h_{\text{dec}}^t = g(x^{\text{dec}}, H^{\text{encoder}}, h_{\text{dec}}^{t-1}) \]
Attention mechanism

Decoder with attention  Luong and Manning [2016], Vinyals et al. [2015]:

\[ h_{t}^{\text{dec}} = g(x_{t}^{\text{dec}}, H^{\text{encoder}}, h_{t-1}^{\text{dec}}) \]
\[ = W_{\text{proj}} \cdot d_{t} \Vert \hat{h}_{t}^{\text{dec}}, \]

\( \Vert \) is the concatenation operator, and
\[ h_{t}^{\text{dec}} = f(x_{t}^{\text{dec}}, h_{t-1}^{\text{dec}}; \theta^{\text{dec}}) \] from above.

\[ d_{t} \text{ is calculated from } H^{\text{encoder}} \text{ by:} \]
\[ d_{t} = \sum_{i=1}^{n} a_{t,i} h_{i}^{\text{encoder}} \]
\[ a_{t} = \text{softmax}(u_{t}) \]
\[ u_{t,i} = v^{T} \tanh(W_{1}h_{i}^{\text{encoder}} + W_{2}h_{t}^{\text{dec}}), \]
Outline

Morning program

Preliminaries
  Feedforward neural network
  Distributed representations
  Recurrent neural networks
  Sequence-to-sequence models
  Convolutional neural networks

Semantic matching
Learning to rank
Entities

Afternoon program

Modeling user behavior
Generating responses
Recommender systems
Industry insights
Q & A
Convolutinal neural networks

What is a convolution?  Intuition: sliding window function applied to a matrix

Example: convolution with $3 \times 3$ filter

Multiply values element-wise with original matrix, then sum. Slide over whole matrix.

Image credits:
http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution
Convolutional neural networks

- Each layer applies different filters and combines results
- Pooling (subsampling) layers
- During training, CNN learns values of filters
- First layer may learn to detect edges from raw pixels in first layer
- Use edges to detect simple shapes in second layer
- Use shapes to detect higher-level features, such as facial shapes in higher layers
- Last layer: classifier using high-level features
CNNs in text

- Instead of image pixels, inputs typically are word embeddings.
- For a 10 word sentence using a 100-dimensional embedding we would have a $10 \times 100$ matrix as our input.
- That's our “image”
- Typically 1-dimensional convolutions are used. I.e. filters slide over rows of the matrix (words).

Architecture for sentiment analysis Zhang and Wallace [2015]
Example uses in IR

- MSR: how to learn semantically meaningful representations of sentences that can be used for Information Retrieval
- Recommending potentially interesting documents to users based on what they are currently reading
- Sentence representations are trained based on search engine log data
- Modeling interestingness with deep neural networks Gao et al. [2014]
  A latent semantic model with convolutional-pooling structure for information retrieval [Shen et al., 2014]
Take aways

1. Information Retrieval (IR) systems help people find the right (most useful) information in the right (most convenient) format at the right time (when they need it).

2. Neural Network (NN) is a function $F(x; \Theta)$ with (a large number of) parameters $\Theta$ that maps an input object $x$ (which can be text, image or arbitrary vector of features) to an output object $y$ (class label, sequence of class labels, text, image).

3. There are three main architectures (classes of $F(x; \Theta)$): (i) feed-forward NN (FFNN), (ii) recurrent NN (RNN), (iii) convolutional NN (CNN).

4. Embeddings are vector representations of objects in a high-dimensional space that are learned during training. In many practical applications, these vectors reflect similarities between objects that are important for solving the task.

5. Other stuff, such as seq2seq, vanishing and exploding gradients, LSTM and GRU, attention mechanism, softmax, cross-entropy.
Outline

Morning program
  Preliminaries
  **Semantic matching**
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Semantic matching

Definition
"... conduct query/document analysis to represent the meanings of query/document with richer representations and then perform matching with the representations." - Li et al. [2014]

A promising area within neural IR, due to the success of semantic representations in NLP and computer vision.
Outline

Morning program
  Preliminaries
  Semantic matching
    Using pre-trained unsupervised representations for semantic matching
    Learning unsupervised representations for semantic matching
    Learning to match models
    Learning to match using pseudo relevance
  Toolkits
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Unsupervised semantic matching with pre-trained representations

Word embeddings have recently gained popularity for their ability to encode semantic and syntactic relations amongst words.

How can we use word embeddings for information retrieval tasks?
Word embedding

**Distributional Semantic Model (DSM):** A model for associating words with vectors that can capture their meaning. DSM relies on the distributional hypothesis.

**Distributional Hypothesis:** Words that occur in the same contexts tend to have similar meanings [Harris, 1954].

Statistics on observed contexts of words in a corpus is quantified to derive word vectors.

- The most common choice of context: The set of words that co-occur in a context window.
- Context-counting VS. Context-predicting [Baroni et al., 2014]
From word embeddings to query/document embeddings

Creating representations for compound units of text (e.g., documents) from representation of lexical units (e.g., words).
From word embeddings to query/document embeddings

Obtaining representations of compound units of text (in comparison to the atomic words).

Bag of embedded words: sum or average of word vectors.

▶ Averaging the word representations of query terms has been extensively explored in different settings. [Vulić and Moens, 2015, Zamani and Croft, 2016b]
  ▶ Effective but for small units of text, e.g. query [Mitra, 2015].

▶ Training word embeddings directly for the purpose of being averaged [Kenter et al., 2016].
Semantic matching

From word embeddings to query/document embeddings

- **Skip-Thought Vectors**
  - Conceptually similar to distributional semantics: a unit representation is a function of its neighbouring units, except units are sentences instead of words.
  - Similar to auto-encoding objective: encode sentence, but decode neighboring sentences.
  - Pair of LSTM-based seq2seq models with shared encoder.

- **Doc2vec (Paragraph2vec) [Le and Mikolov, 2014].**
  - You’ll hear more later about it on “Learning unsupervised representations from scratch”. (Also you might want to take a look at Deep Learning for Semantic Composition)
Using similarity amongst documents, queries and terms.

Given low-dimensional representations, integrate their similarity signal within IR.
Dual Embedding Space Model (DESM) [Nalisnick et al., 2016]

Word2vec optimizes IN-OUT dot product which captures the co-occurrence statistics of words from the training corpus:
- We can gain by using these two embeddings differently

▶ IN-IN and OUT-OUT cosine similarities are high for words that are similar by function or type (typical) and the
▶ IN-OUT cosine similarities are high between words that often co-occur in the same query or document (topical).

<table>
<thead>
<tr>
<th></th>
<th>yale</th>
<th>yale</th>
<th>yale</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN-IN</td>
<td>yale</td>
<td>yahoo</td>
<td>orientations</td>
</tr>
<tr>
<td>OUT-OUT</td>
<td>harvard</td>
<td>alumni</td>
<td>orientation</td>
</tr>
<tr>
<td>IN-OUT</td>
<td>cornell</td>
<td>nyt</td>
<td>haven</td>
</tr>
<tr>
<td></td>
<td>tulane</td>
<td>tufts</td>
<td>graduate</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>seahawks</th>
<th>seahawks</th>
<th>seahawks</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN-IN</td>
<td>seahawks</td>
<td>seahawks</td>
<td>seahawks</td>
</tr>
<tr>
<td>OUT-OUT</td>
<td>49ers</td>
<td>broncos</td>
<td>highlights</td>
</tr>
<tr>
<td>IN-OUT</td>
<td>packers</td>
<td>ntl</td>
<td>shirts</td>
</tr>
<tr>
<td></td>
<td>steelers</td>
<td>steelers</td>
<td>hats</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>eminem</th>
<th>eminem</th>
<th>eminem</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN-IN</td>
<td>rihanna</td>
<td>rihanna</td>
<td>rap</td>
</tr>
<tr>
<td>OUT-OUT</td>
<td>ludacris</td>
<td>dre</td>
<td>featuring</td>
</tr>
<tr>
<td>IN-OUT</td>
<td>kanye</td>
<td>kanye</td>
<td>tracklist</td>
</tr>
<tr>
<td></td>
<td>beyonce</td>
<td>beyonce</td>
<td>diss</td>
</tr>
<tr>
<td></td>
<td>2pac</td>
<td>tupac</td>
<td>performs</td>
</tr>
</tbody>
</table>
Pre-trained word embeddings for document retrieval and ranking

DESM [Nalisnick et al., 2016]: Using IN-OUT similarity to model document aboutness.

- A document is represented by the centroid of its word OUT-vectors:

\[
\vec{v}_{d,\text{OUT}} = \frac{1}{|d|} \sum_{t_d \in d} \frac{\vec{v}_{t_d,\text{OUT}}}{\|\vec{v}_{t_d,\text{OUT}}\|}
\]

- Query-document similarity is average of cosine similarity over query words:

\[
\text{DESM}_{\text{IN-OUT}}(q, d) = \frac{1}{q} \sum_{t_q \in q} \frac{\vec{v}_{t_q,\text{IN}}^\top \vec{v}_{t_d,\text{OUT}}}{\|\vec{v}_{t_q,\text{IN}}\| \|\vec{v}_{t_d,\text{OUT}}\|}
\]

- IN-OUT captures more topical notion of similarity than IN-IN and OUT-OUT.

- DESM is effective at, but only at, ranking at least somewhat relevant documents.
Pre-trained word embeddings for document retrieval and ranking

- NTLM [Zuccon et al., 2015]: Neural Translation Language Model
  - Translation Language Model: extending query likelihood:

\[
p(d|q) \sim p(q|d)p(d)
\]
\[
p(q|d) = \prod_{t_q \in q} p(t_q|d)
\]
\[
p(t_q|d) = \sum_{t_d \in d} p(t_q|t_d)p(t_d|d)
\]

- Uses the similarity between term embeddings as a measure for term-term translation probability \( p(t_q|t_d) \).

\[
p(t_q|t_d) = \frac{\cos(\vec{v}_{t_q}, \vec{v}_{t_d})}{\sum_{t \in V} \cos(\vec{v}_t, \vec{v}_{t_d})}
\]
Pre-trained word embeddings for document retrieval and ranking

GLM [Ganguly et al., 2015]: Generalized Language Model

▶ Terms in a query are generated by sampling them independently from either the document or the collection.

▶ The noisy channel may transform (mutate) a term $t$ into a term $t'$.

$$p(t_q|d) = \lambda p(t_q|d) + \alpha \sum_{t_d \in d} p(t_q,t_d|d) p(t_d) + \beta \sum_{t' \in N_t} p(t_q,t'|C) p(t') + 1 - \lambda - \alpha - \beta p(t_q|C)$$

$N_t$ is the set of nearest-neighbours of term $t$.

$$p(t', t|d) = \frac{\text{sim}(\vec{v}_{t'}, \vec{v}_t).\text{tf}(t', d)}{\sum_{t_1 \in d} \sum_{t_2 \in d} \text{sim}(\vec{v}_{t_1}, \vec{v}_{t_2}).|d|}$$
Pre-trained word embeddings for query term weighting

Term re-weighting using word embeddings [Zheng and Callan, 2015].
- Learning to map query terms to query term weights.

Constructing the feature vector \( \vec{x}_{t_{q}} \) for term \( t_{q} \) using its embedding and embeddings of other terms in the same query \( q \) as:

\[
\vec{x}_{t_{q}} = \vec{v}_{t_{q}} - \frac{1}{|q|} \sum_{t'_{q} \in q} \vec{v}_{t'_{q}}
\]

- \( \vec{x}_{t_{q}} \) measures the semantic difference of a term to the whole query.
- Learn a model to map the feature vectors the defined target term weights.
Pre-trained word embeddings for query expansion

- Identify expansion terms using word2vec cosine similarity [Roy et al., 2016].
  - pre-retrieval:
    - Taking nearest neighbors of query terms as the expansion terms.
  - post-retrieval:
    - Using a set of pseudo-relevant documents to restrict the search domain for the candidate expansion terms.
  - pre-retrieval incremental:
    - Using an iterative process of reordering and pruning terms from the nearest neighbors list.
      - Reorder the terms in decreasing order of similarity with the previously selected term.

- Works better than having no query expansion, but does not beat non-neural query expansion methods.
Pre-trained word embedding for query expansion

- **Embedding-based Query Expansion** [Zamani and Croft, 2016a]
  *Main goal:* Estimating a better language model for the query using embeddings.

- **Embedding-based Relevance Model:**
  *Main goal:* Semantic similarity in addition to term matching for PRF.
Pre-trained word embedding for query expansion

Query expansion with locally-trained word embeddings [Diaz et al., 2016].

- **Main idea**: Embeddings be learned on topically-constrained corpora, instead of large topically-unconstrained corpora.
- Training word2vec on documents from first round of retrieval.
- Fine-grained word sense disambiguation.
- A large number of embedding spaces can be cached in practice.
Outline

Morning program
  Preliminaries
  Semantic matching
    Using pre-trained unsupervised representations for semantic matching
    Learning unsupervised representations for semantic matching
    Learning to match models
    Learning to match using pseudo relevance
  Toolkits
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Learning unsupervised representations for semantic matching

Pre-trained word embeddings can be used to obtain
- a query/document representation through compositionality, or
- a similarity signal to integrate within IR frameworks.

Can we learn unsupervised query/document representations directly for IR tasks?
History of latent document representations

Latent representations of documents that are learned from scratch have been around since the early 1990s.

- Latent Semantic Indexing [Deerwester et al., 1990],
- Probabilistic Latent Semantic Indexing [Hofmann, 1999], and
- Latent Dirichlet Allocation [Blei et al., 2003].

These representations provide a semantic matching signal that is complementary to a lexical matching signal.

- Auto-encoder trained on frequency vectors.
- Documents are mapped to memory addresses in such a way that semantically similar documents are located at nearby bit addresses.
- Documents similar to a query document can then be found by accessing addresses that differ by only a few bits from the query document address.

Schematic representation of Semantic Hashing. Taken from Salakhutdinov and Hinton [2009].
Distributed Representations of Documents [Le and Mikolov, 2014]

- Learn document representations based on the words contained within each document.
- Reported to work well on a document similarity task.
- Attempts to integrate learned representations into standard retrieval models [Ai et al., 2016a,b].

Overview of the Distributed Memory document vector model. Taken from Le and Mikolov [2014].
Two Doc2Vec Architectures [Le and Mikolov, 2014]

Overview of the Distributed Memory document vector model. Taken from Le and Mikolov [2014].

Overview of the Distributed Bag of Words document vector model. Taken from Le and Mikolov [2014].
Neural Vector Spaces for Unsupervised IR [Van Gysel et al., 2018]

- Learns query (term) and document representations directly from the document collection.

- Outperforms existing latent vector space models and provides semantic matching signal complementary to lexical retrieval models.

- Learns a notion of term specificity.

- Luhn significance: mid-frequency words are more important for retrieval than infrequent and frequent words.

Relation between query term representation L2-norm within NVSM and its collection frequency. Taken from [Van Gysel et al., 2018].
Outline

Morning program
  Preliminaries
  Semantic matching
    Using pre-trained unsupervised representations for semantic matching
    Learning unsupervised representations for semantic matching
    Learning to match models
    Learning to match using pseudo relevance
    Toolkits
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Text matching as a supervised objective

Text matching is often formulated as a supervised objective where pairs of relevant or paraphrased texts are given.

In the next few slides, we’ll go over different architectures introduced for supervised text matching. Note that this is a mix of models originally introduced for (i) relevance ranking, (ii) paraphrase identification, and (iii) question answering among others.
Representation-based models

Representation-based models construct a fixed-dimensional vector representation for each text separately and then perform matching within the latent space.
(C)DSSM [Huang et al., 2013, Shen et al., 2014]

- Siamese network between query and document, performed on character trigrams.
- Originally introduced for learning from implicit feedback.
**ARC-I [Hu et al., 2014]**

- Similar to DSSM, perform 1D convolution on text representations separately.
- Originally introduced for paraphrasing task.
Interaction-based models

Interaction-based models compute the interaction between each individual term of both texts. An interaction can be **identity** or **syntactic/semantic similarity**.

The interaction matrix is subsequently summarized into a matching score.
DRMM [Guo et al., 2016]

- Compute term/document interactions and matching histograms using different strategies (count, relative count, log-count).
- Pass histograms through feed-forward network for every query term.
- Gating network that produces an attention weight for every query term; per-term scores are then aggregated into a relevance score using attention weights.
MatchPyramid [Pang et al., 2016]

- Interaction matrix between query/document terms, followed by convolutional layers.

- After convolutions, feed-forward layers determine matching score.
aNMM [Yang et al., 2016]

- Compute word interaction matrix.
- Aggregate similarities by running multiple kernels.
- Every kernel assigns a different weight to a particular similarity range.
- Similarities are aggregated to the kernel output by weighting them according to which bin they fall in.
Match-SRNN [Wan et al., 2016b]

- Word interaction layer, followed by a spatial recurrent NN.

- The RNN hidden state is updated using the current interaction coefficient, and the hidden state of the prefix.
K-NRM [Xiong et al., 2017b]

- Compute word-interaction matrix, apply k kernels to every query term row in interaction matrix.
- This results in k-dimensional vector.
- Aggregate the query term vectors into a fixed-dimensional query representation.
Hybrid models

Hybrid models consist of (i) a representation component that combines a sequence of words (e.g., a whole text, a window of words) into a fixed-dimensional representation and (ii) an interaction component.

These two components can occur (1) in serial or (2) in parallel.
ARC-II [Hu et al., 2014]

- Cascade approach where word representation are generated from context.
- Interaction matrix between sliding windows, where the interaction activation is computed using a non-linear mapping.
- Originally introduced for paraphrasing task.
MV-LSTM [Wan et al., 2016a]

- Cascade approach where input representations for the interaction matrix are generated using a bi-directional LSTM.

- Differs from pure interaction-based approaches as the LSTM builds a representation of the context, rather than using the representation of a word.

- Obtains fixed-dimensional representation by max-pooling over query/document; followed by feed-forward network.
Duet [Mitra et al., 2017]

- Model has an interaction-based and a representation-based component.
  
  - Interaction-based component consist of a indicator matrix showing where query terms occur in document; followed by convolution layers.
  
  - Representation-based component is similar to DSSM/ARC-I, but uses a feed-forward network to compute the similarity signal rather than cosine similarity.
  
- Both are combined at the end using a linear combination of the scores.
DeepRank [Pang et al., 2017]

- Focus only on exact term occurrences in document.
- Compute interaction between query and window surrounding term occurrence.
- RNN or CNN then combines per-window features (query representation, context representations and interaction between query/document term) into matching score.
Outline

Morning program
  Preliminaries
  Semantic matching
    Using pre-trained unsupervised representations for semantic matching
    Learning unsupervised representations for semantic matching
    Learning to match models
    Learning to match using pseudo relevance
  Toolkits
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Beyond supervised signals: semi-supervised learning

The architectures we presented for learning to match all require labels. Typically these labels are obtained from domain experts.

However, in information retrieval, there is the concept of pseudo relevance that gives us a supervised signal that was obtained from unsupervised data collections.
Given a source of pseudo relevance, we can build pseudo collections for training retrieval models [Asadi et al., 2011, Berendsen et al., 2013].

Sources of pseudo-relevance
Typically given by external knowledge about retrieval domain, such as hyperlinks, query logs, social tags, ...
Training neural networks using pseudo relevance

Training a neural ranker using weak supervision [Dehghani et al., 2017].

Main idea: Annotating a large amount of unlabeled data using a weak annotator (Pseudo-Labeling) and design a model which can be trained on weak supervision signal.

▶ Function approximation. (re-inventing BM25?)
▶ Beating BM25 using BM25!
Training neural networks using pseudo relevance

Generating weak supervision training data for training neural IR model [MacAvaney et al., 2017].

- Using a news corpus with article headlines acting as pseudo-queries and article content as pseudo-documents.

- Problems:
  - Hard-Negative
  - Mismatched-Interaction: (example: “When Bird Flies In”, a sports article about basketball player Larry Bird)

- Solutions:
  - Ranking filter:
    - top pseudo-documents are considered as negative samples.
    - only pseudo-queries that are able to retrieve their pseudo-relevant documents are used as positive samples.
  - Interaction filter:
    - building interaction embeddings for each pair.
    - filtering out based on similarity to the template query-document pairs.
Query expansion using neural word embeddings based on pseudo relevance

Locally trained word embeddings [Diaz et al., 2016]

- Performing topic-specific training, on a set of topic specific documents that are collected based on their relevance to a query.

Relevance-based Word Embedding [Zamani and Croft, 2017].

- Relevance is not necessarily equal to semantically or syntactically similarity:
  - “united state” as expansion terms for “Indian American museum”.

- **Main idea**: Defining the “context”
  Using the relevance model distribution for the given query to define the context. So the objective is to predict the words observed in the documents relevant to a particular information need.

- The neural network will be constraint by the given weights from RM3 to learn word embeddings.
Outline

Morning program
  Preliminaries
  Semantic matching
    Using pre-trained unsupervised representations for semantic matching
    Learning unsupervised representations for semantic matching
    Learning to match models
    Learning to match using pseudo relevance
  Toolkits
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Document & entity representation learning toolkits

- **gensim**: https://github.com/RaRe-Technologies/gensim [Řehůřek and Sojka, 2010]
- **SERT**: http://www.github.com/cvangysel/SERT [Van Gysel et al., 2017a]
- **cuNVSM**: http://www.github.com/cvangysel/cuNVSM [Van Gysel et al., 2018]
- **HEM**: https://ciir.cs.umass.edu/downloads/HEM [Ai et al., 2017]
- **MatchZoo**: https://github.com/faneshion/MatchZoo [Fan et al., 2017]
- **K-NRM**: https://github.com/AdeDZY/K-NRM [Xiong et al., 2017b]
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Outline

Morning program
Preliminaries
Semantic matching
Learning to rank
  Overview & basics
  Quick refreshers
  Pointwise loss
  Pairwise loss
  Listwise loss
  Toolkits
Entities

Afternoon program
Modeling user behavior
Generating responses
Recommender systems
Industry insights
Q & A
Learning to rank (LTR)

Definition
"... the task to automatically construct a ranking model using training data, such that the model can sort new objects according to their degrees of relevance, preference, or importance." - Liu [2009]

LTR models represent a rankable item—e.g., a document—given some context—e.g., a user-issued query—as a numerical vector $\vec{x} \in \mathbb{R}^n$.

The ranking model $f : \vec{x} \rightarrow \mathbb{R}$ is trained to map the vector to a real-valued score such that relevant items are scored higher.

We only discuss offline LTR models here—see Grotov and de Rijke [2016] for an overview of online LTR.
Three training objectives

Liu [2009] categorizes different LTR approaches based on training objectives:

- **Pointwise approach**: relevance label $y_{q,d}$ is a number—derived from binary or graded human judgments or implicit user feedback (e.g., CTR). Typically, a regression or classification model is trained to predict $y_{q,d}$ given $\vec{x}_{q,d}$.

- **Pairwise approach**: pairwise preference between documents for a query ($d_i \succ_q d_j$) as label. Reduces to binary classification to predict more relevant document.

- **Listwise approach**: directly optimize for rank-based metric, such as NDCG—difficult because these metrics are often not differentiable w.r.t. model parameters.
Features

Traditional LTR models employ hand-crafted features that encode IR insights

They can often be categorized as:

- **Query-independent** or **static** features (e.g., incoming link count and document length)
- **Query-dependent** or **dynamic** features (e.g., BM25)
- **Query-level** features (e.g., query length)
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
    Overview & basics
    Quick refreshers
    Pointwise loss
    Pairwise loss
    Listwise loss
    Toolkits

Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
A quick refresher - Neural models for different tasks

- **Expected**
- **Loss**
- **Predicted**

- **Model**
  - **Features**
    - Item
  - **Regression**

- **Model**
  - **Features**
    - Item
  - **Classification**
    - Item
    - Class 1
    - Class 2
A quick refresher - What is the Softmax function?

In neural classification models, the softmax function is popularly used to normalize the neural network output scores across all the classes.

\[ p(z_i) = \frac{e^{\gamma z_i}}{\sum_{z \in Z} e^{\gamma z}} \]  

(\( \gamma \) is a constant)
A quick refresher - What is Cross Entropy?

The cross entropy between two probability distributions $p$ and $q$ over a discrete set of events is given by,

$$CE(p, q) = - \sum_i p_i \log(q_i)$$  \hspace{1cm} (2)

If $p_{\text{correct}} = 1$ and $p_i = 0$ for all other values of $i$ then,

$$CE(p, q) = - \log(q_{\text{correct}})$$  \hspace{1cm} (3)
A quick refresher - What is the Cross Entropy with Softmax loss?

Cross entropy with softmax is a popular loss function for classification

\[ \mathcal{L}_{CE} = -\log \left( \frac{e^{\gamma z_{\text{correct}}}}{\sum_{z \in Z} e^{\gamma z}} \right) \] (4)
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
    Overview & basics
    Quick refreshers
    Pointwise loss
    Pairwise loss
    Listwise loss
    Toolkits

Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Regression-based or classification-based approaches are popular

**Regression loss**

Given \( \langle q, d \rangle \) predict the value of \( y_{q,d} \)

E.g., square loss for binary or categorical labels,

\[
\mathcal{L}_{Squared} = \| y_{q,d} - f(\vec{x}_{q,d}) \|^2 \tag{5}
\]

where, \( y_{q,d} \) is the one-hot representation [Fuhr, 1989] or the actual value [Cossock and Zhang, 2006] of the label
Pointwise objectives

Regression-based or classification-based approaches are popular

**Classification loss**

Given \( \langle q, d \rangle \) predict the class \( y_{q,d} \)

E.g., **Cross-Entropy with Softmax** over categorical labels \( Y \) [Li et al., 2008],

\[
\mathcal{L}_{CE}(q, d, y_{q,d}) = - \log \left( p(y_{q,d} | q, d) \right) 
= - \log \left( \frac{e^{\gamma \cdot s_{y_{q,d}}}}{\sum_{y \in Y} e^{\gamma \cdot s_y}} \right) 
\]

where, \( s_{y_{q,d}} \) is the model’s score for label \( y_{q,d} \)
Outline

Morning program
  Preliminaries
  Semantic matching
  **Learning to rank**
    - Overview & basics
    - Quick refreshers
    - Pointwise loss
    - Pairwise loss
    - Listwise loss
  Toolkits

Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Pairwise objectives

Pairwise loss minimizes the average number of inversions in ranking—i.e., $d_i \succ_q d_j$ but $d_j$ is ranked higher than $d_i$.

Given $\langle q, d_i, d_j \rangle$, predict the more relevant document.

For $\langle q, d_i \rangle$ and $\langle q, d_j \rangle$,

- Feature vectors: $\vec{x}_i$ and $\vec{x}_j$
- Model scores: $s_i = f(\vec{x}_i)$ and $s_j = f(\vec{x}_j)$

Pairwise loss generally has the following form [Chen et al., 2009],

$$\mathcal{L}_{\text{pairwise}} = \phi(s_i - s_j)$$  \hspace{1cm} (8)

where, $\phi$ can be,

- Hinge function $\phi(z) = \max(0, 1 - z)$ [Herbrich et al., 2000]
- Exponential function $\phi(z) = e^{-z}$ [Freund et al., 2003]
- Logistic function $\phi(z) = \log(1 + e^{-z})$ [Burges et al., 2005]
- etc.
RankNet

RankNet [Burges et al., 2005] is a pairwise loss function—an industry favourite [Burges, 2015]

Predicted probabilities:
$$p_{ij} = p(s_i > s_j) \equiv \frac{e^{\gamma \cdot s_i}}{e^{\gamma \cdot s_i} + e^{\gamma \cdot s_j}} = \frac{1}{1 + e^{-\gamma(s_i - s_j)}}$$
and
$$p_{ji} \equiv \frac{1}{1 + e^{-\gamma(s_j - s_i)}}$$

Desired probabilities: $$\bar{p}_{ij} = 1$$ and $$\bar{p}_{ji} = 0$$

Computing cross-entropy between $$\bar{p}$$ and $$p$$,

$$\mathcal{L}_{RankNet} = -\bar{p}_{ij} \log(p_{ij}) - \bar{p}_{ji} \log(p_{ji})$$

$$= -\log(p_{ij})$$

$$= \log(1 + e^{-\gamma(s_i - s_j)})$$
Cross Entropy (CE) with Softmax over documents

An alternative loss function assumes a single relevant document $d^+$ and compares it against the full collection $D$

Probability of retrieving $d^+$ for $q$ is given by the softmax function,

$$p(d^+ | q) = \frac{e^{\gamma \cdot s(q,d^+)}}{\sum_{d \in D} e^{\gamma \cdot s(q,d)}} \quad (12)$$

The cross entropy loss is then given by,

$$\mathcal{L}_{CE}(q, d^+, D) = - \log \left( p(d^+ | q) \right) \quad (13)$$

$$= - \log \left( \frac{e^{\gamma \cdot s(q,d^+)}}{\sum_{d \in D} e^{\gamma \cdot s(q,d)}} \right) \quad (14)$$
Notes on Cross Entropy (CE) loss

- If we consider only a pair of relevant and non-relevant documents in the denominator, CE reduces to RankNet
- Computing the denominator is prohibitively expensive—large body of work in NLP on this that may be relevant to future LTR models
  - Hierarchical softmax
  - Sampling based approaches
- In IR, LTR models typically consider few negative candidates [Huang et al., 2013, Mitra et al., 2017, Shen et al., 2014]
Hierarchical Softmax

Avoid computing $p(d^+|q)$, group candidates $D$ into set of classes $C$, then predict correct class $c^+$ given $q$ followed by predicting $d^+$ given $\langle c^+, q \rangle$ [Goodman, 2001]

$$p(d^+|q) = p(d^+|c^+, q) \cdot p(c^+|q)$$ (15)

Computational cost is a function of $|C| + |c^+| << |D|$

Employ hierarchy of classes [Mnih and Hinton, 2009, Morin and Bengio, 2005]

Hierarchy based on similarity between candidates [Brown et al., 1992, Le et al., 2011, Mikolov et al., 2013], or frequency binning [Mikolov et al., 2011]
Sampling based approaches

Alternative to computing exact softmax, estimate it using sampling based approaches

\[
\mathcal{L}_{CE}(q, d^+, D) = -\log \left( \frac{e^{\gamma \cdot s(q, d^+)}}{\sum_{d \in D} e^{\gamma \cdot s(q, d)}} \right) = -\gamma \cdot s(q, d^+) + \log \sum_{d \in D} e^{\gamma \cdot s(q, d)}
\]  

(16)

Importance sampling [Bengio and Senécal, 2008, Bengio et al., 2003, Jean et al., 2014, Jozefowicz et al., 2016], Noise Contrastive Estimation [Gutmann and Hyvärinen, 2010, Mnih and Teh, 2012, Vaswani et al., 2013], negative sampling [Mikolov et al., 2013], BlackOut [Ji et al., 2015], and others have been proposed

See [Mitra and Craswell, 2017] for detailed discussion
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
    Overview & basics
    Quick refreshers
    Pointwise loss
    Pairwise loss
    Listwise loss
  Toolkits

Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Listwise

Blue: relevant  Gray: non-relevant

NDCG and ERR higher for left but pairwise errors less for right

Due to strong position-based discounting in IR measures, errors at higher ranks are much more problematic than at lower ranks

But listwise metrics are non-continuous and non-differentiable

[Burges, 2010]
LambdaRank

Key observations:

▶ To train a model we don't need the costs themselves, only the gradients (of the costs w.r.t model scores)
▶ It is desired that the gradient be bigger for pairs of documents that produces a bigger impact in NDCG by swapping positions

**LambdaRank** [Burges et al., 2006]
Multiply actual gradients with the change in NDCG by swapping the rank positions of the two documents

$$\lambda_{\text{LambdaRank}} = \lambda_{\text{RankNet}} \cdot |\Delta NDCG|$$ (17)
ListNet and ListMLE

According to the Luce model [Luce, 2005], given four items \( \{d_1, d_2, d_3, d_4\} \) the probability of observing a particular rank-order, say \( [d_2, d_1, d_4, d_3] \), is given by:

\[
p(\pi | s) = \frac{\phi(s_2)}{\phi(s_1) + \phi(s_2) + \phi(s_3) + \phi(s_4)} \cdot \frac{\phi(s_1)}{\phi(s_1) + \phi(s_3) + \phi(s_4)} \cdot \frac{\phi(s_4)}{\phi(s_3) + \phi(s_4)}
\]  

(18)

where, \( \pi \) is a particular permutation and \( \phi \) is a transformation (e.g., linear, exponential, or sigmoid) over the score \( s_i \) corresponding to item \( d_i \).
ListNet and ListMLE

**ListNet** [Cao et al., 2007]
Compute the probability distribution over all possible permutations based on model score and ground-truth labels. The loss is then given by the K-L divergence between these two distributions.

This is computationally very costly, computing permutations of only the top-K items makes it slightly less prohibitive.

**ListMLE** [Xia et al., 2008]
Compute the probability of the ideal permutation based on the ground truth. However, with categorical labels more than one permutation is possible which makes this difficult.
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
    Overview & basics
    Quick refreshers
    Pointwise loss
    Pairwise loss
    Listwise loss
    Toolkits

Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Toolkits for off-line learning to rank

- **RankLib**: [https://sourceforge.net/p/lemur/wiki/RankLib](https://sourceforge.net/p/lemur/wiki/RankLib)
- **shoelace**: [https://github.com/rjagerman/shoelace](https://github.com/rjagerman/shoelace) [Jagerman et al., 2017]
- **QuickRank**: [http://quickrank.isti.cnr.it](http://quickrank.isti.cnr.it) [Capannini et al., 2016]
- **RankPy**: [https://bitbucket.org/tunystom/rankpy](https://bitbucket.org/tunystom/rankpy)
- **pyltr**: [https://github.com/jma127/pyltr](https://github.com/jma127/pyltr)
- **jforests**: [https://github.com/yasserg/jforests](https://github.com/yasserg/jforests) [Ganjisaffar et al., 2011]
- **XGBoost**: [https://github.com/dmlc/xgboost](https://github.com/dmlc/xgboost) [Chen and Guestrin, 2016]
- **SVMRank**: [https://www.cs.cornell.edu/people/tj/svm_light](https://www.cs.cornell.edu/people/tj/svm_light) [Joachims, 2006]
- **sofia-ml**: [https://code.google.com/archive/p/sofia-ml](https://code.google.com/archive/p/sofia-ml) [Sculley, 2009]
- **pysofia**: [https://pypi.python.org/pypi/pysofia](https://pypi.python.org/pypi/pysofia)
Outline

Morning program
- Preliminaries
- Semantic matching
- Learning to rank
- Entities

Afternoon program
- Modeling user behavior
- Generating responses
- Recommender systems
- Industry insights
- Q & A
Entities are polysemic

“Finding entities” has multiple meanings.

Entities can be

- nodes in knowledge graphs,
- mentions in unstructured texts or queries,
- retrievable items characterized by texts.
Outline

Morning program
- Preliminaries
- Semantic matching
- Learning to rank

Entities
- Knowledge graph embeddings
- Entity mentions in unstructured text
- Entity finding

Afternoon program
- Modeling user behavior
- Generating responses
- Recommender systems
- Industry insights
- Q & A
Knowledge graphs

Beyoncé Knowles

Kelly Rowland

Michelle Williams

Destiny’s Child

1997

2005

Triples

(beyoncé_knowles, member_of, destinys_child)
(kelly_rowland, member_of, destinys_child)
(michelle_williams, member_of, destinys_child)
(destinys_child, start_date, 1997)
(destinys_child, end_date, 2005)

Nice overview on using knowledge bases in IR: [Dietz et al., 2017]
Knowlegde graphs

Tasks

- Link prediction
  Predict the missing h or t for a triple (h, r, t)
  Rank entities by score. Metrics:
  - Mean rank of correct entity
  - Hits@10

- Triple classification
  Predict if (h, r, t) is correct.
  Metric: accuracy.

- Relation fact extraction from free text
  Use knowledge base as weak supervision for extracting new triples.
  Suppose some IE system gives us (steve_jobs, ‘‘was the initiator of’’, apple), then we want to predict the founder_of relation.

Datatsets

- WordNet
  (car, hyponym, vehicle)

- Freebase/DBPedia
  (steve_jobs, founder_of, apple)
Knowledge graphs

Knowledge graph embeddings

- TransE [Bordes et al., 2013]
- TransH [Wang et al., 2014]
- TransR [Lin et al., 2015]
TransE

“Translation intuition”

For a triple \((h, l, t)\): \(\vec{h} + \vec{l} \approx \vec{t}\).
“Translation intuition”
For a triple \((h, l, t)\): \(\vec{h} + \vec{l} \approx \vec{t}\).

\[
\mathcal{L} = \sum_{(h, l, t) \in S}^\text{positive examples} \left( \sum_{(h', l, t') \in S'_{(h, l, t)}}^\text{negative examples} \left[ \gamma + d(h + l, t) - d(h' + l, t') \right]^+ \right)
\]

[In Bordes et al., 2013]
“Translation intuition”
For a triple \((h, l, t)\): \(\vec{h} + \vec{l} \approx \vec{t}\).

How about:

- one-to-many relations?
- many-to-many relations?
- many-to-one relations?
TransH

(a) TransE

(b) TransH

[Wang et al., 2014]
TransH

\[ f_r(h, t) = \left\| (h - w_r^T h w_r) + d_r - (t - w_r^T t w_r) \right\|_2^2 \]

\[ \mathcal{L} = \sum_{(h, r, t) \in \Delta} \left( \sum_{(h', r', t') \in \Delta'_{(h, r, t)}} [f_r(h, t) + \gamma - f_{r'}(h', t')] + \right) \]

[Entities] [Wang et al., 2014]
TransH

Constraints

\[ \forall e \in E, \|e\|_2 \leq 1, //\text{scale} \]

\[ \forall r \in R, \|w_r^T d_r \|/\|d_r\|_2 \leq \epsilon, //\text{orthogonal} \]

\[ \forall r \in R, \|w_r\|_2 = 1, //\text{unit normal vector} \]

\[
\mathcal{L} = \sum_{(h,r,t) \in \Delta} \sum_{(h',r',t') \in \Delta'} \left[ f_r(h, t) + \gamma - f_{r'}(h', t') \right]_+ + C \left\{ \sum_{e \in E} \left[ \|e\|_2^2 - 1 \right]_+ + \sum_{r \in R} \left[ \frac{(w_r^T d_r)^2}{\|d_r\|_2^2} - \epsilon^2 \right]_+ \right\}, \tag{4}
\]

[Wang et al., 2014]
Use different embedding spaces for entities and relations

- 1 entity space
- multiple relation spaces
- perform translation in appropriate relation space

[Lin et al., 2015]
TransR

[Lin et al., 2015]
TransR

Entities: $R^d$
Relations: $R^k$

$M_r =$ projection matrix: $k \times d$

$h_r = hM_r, \quad t_r = tM_r$

$f_r(h, t) = \|h_r + r - t_r\|^2_2$

Constraints:
$\|h\|_2 \leq 1$
$\|r\|_2 \leq 1$
$\|t\|_2 \leq 1$
$\|tM_r\|_2 \leq 1$
$\|hM_r\|_2 \leq 1$

$L = \sum_{(h,r,t) \in S} \sum_{(h',r,t') \in S'} \max(0, f_r(h, t) + \gamma - f_r(h', t'))$

[Lin et al., 2015]
Challenges

- How about time?
  E.g., some relations hold from a certain date, until a certain date.
- New entities/relationships
- Evaluation
  Link prediction? Relation classification? Is this fair? As in, is this even possible in all cases (for a human without any world knowledge)?
Resources: toolkits + knowledge bases

Source Code

KB2E: https://github.com/thunlp/KB2E [Lin et al., 2015]
TransE: https://everest.hds.utc.fr/

Knowledge Graphs

- Google Knowledge Graph
google.com/insidesearch/features/search/knowledge.html
- Freebase
google.com/freebase.com
- GeneOntology
geneontology.org
- WikiLinks
code.google.com/p/wiki-links
Outline

Morning program
- Preliminaries
- Semantic matching
- Learning to rank

Entities
- Knowledge graph embeddings
  - Entity mentions in unstructured text
  - Entity finding

Afternoon program
- Modeling user behavior
- Generating responses
- Recommender systems
- Industry insights
- Q & A
Entity mentions

**Recognition**  Detect mentions within unstructured text (e.g., query).

**Linking**  Link mentions to knowledge graph entities.

**Utilization**  Use mentions to improve search.
Named entity recognition

EU rejects German call to boycott British lamb.

Task

vanilla RNN
Named entity recognition

- A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning [Collobert and Weston, 2008]
- Natural Language Processing (Almost) from Scratch [Collobert et al., 2011]

Learning a single model to solve multiple NLP tasks. Taken from [Collobert and Weston, 2008].

Feed-forward language model architecture for different NLP tasks. Taken from [Collobert and Weston, 2008].
Named entity recognition

BI-LSTM-CRF model

EU rejects German call

[Huang et al., 2015]
Entity disambiguation

- Learn representations for documents and entities.
- Optimize a distribution of candidate entities given a document using (a) cross entropy or (b) pairwise loss.

Learn initial document representation in unsupervised pre-training stage. Taken from [He et al., 2013].

Learn similarity between document and entity representations using supervision. Taken from [He et al., 2013].
Learn representations for the context, the mention, the entity (using surface words) and the entity class. Uses pre-trained word2vec embeddings. Taken from [Sun et al., 2015].
Encode Wikipedia descriptions, linked mentions in Wikipedia and fine-grained entity types. All representations are optimized jointly. Taken from [Gupta et al., 2017].
A single mention phrase refers to various entities. Multi-Prototype Mention Embedding model that learns multiple sense embeddings for each mention by jointly modeling words from textual contexts and entities derived from a KB. Taken from [Cao et al., 2017].
Improving search using linked entities

Attention-based ranking model for word-entity duet. Learn a similarity between query and document entities. Resulting model can be used to obtain ranking signal. Taken from [Xiong et al., 2017a].
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  **Entities**
    - Knowledge graph embeddings
    - Entity mentions in unstructured text
    - Entity finding

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Entity finding

Task definition
Rank entities satisfying a topic described by a few query terms.

Not just document search — (a) topics do not typically correspond to entity names, (b) average textual description much longer than typical document.

Different instantiations of the task within varying domains:

- Enterprise search: expert finding [Balog et al., 2006, 2012] (few entities, abundance of text per entity)
- E-commerce: product ranking [Rowley, 2000] (noisy text, customer preferences)
Semantic Expertise Retrieval [Van Gysel et al., 2016]

- Expert finding is a particular entity retrieval task where there is a lot of text.
- Learn representations of words and entities such that n-grams extracted from a document predict the correct expert.

Taken from slides of Van Gysel et al. [2016].
Expert finding is a particular entity retrieval task where there is a lot of text. Learn representations of words and entities such that n-grams extracted from a document predict the correct expert.

\[ P(C \mid \text{“information”}) \times P(C \mid \text{“retrieval”}) = P(C \mid \text{“information” “retrieval”}) \]

Taken from slides of Van Gysel et al. [2016].
To what extent do entity representation models, trained only on text, encode structural regularities of the entity’s domain?

**Goal**: give insight into learned entity representations.

- Clusterings of experts correlate somewhat with groups that exist in the real world.
- Some representation methods encode co-authorship information into their vector space.
- Rank within organizations is learned (e.g., Professors > PhD students) as senior people typically have more published works.
Latent Semantic Entities [Van Gysel et al., 2016]

- Learn representations of e-commerce products and query terms for product search.
- Tackles learning objective scalability limitations from previous work.
- Useful as a semantic feature within a Learning To Rank model in addition to a lexical matching signal.

Taken from slides of Van Gysel et al. [2016].
Personalized Product Search [Ai et al., 2017]

- Learn representations of e-commerce products, query terms, and users for personalized e-commerce search.
- Mixes supervised (relevance triples of query, user and product) and unsupervised (language modeling) objectives.
- The query is represented as an interpolation of query term and user representations.

Personalized product search in a latent space with query $\vec{q}$, user $\vec{u}$ and product item $\vec{i}$. Taken from Ai et al. [2017].
Resources: toolkits

SERT: http://www.github.com/cvangysel/SERT [Van Gysel et al., 2017a]
HEM: https://ciir.cs.umass.edu/downloads/HEM [Ai et al., 2017]
Resources: further reading on entities/KGs

For more information, see the tutorial on “Utilizing Knowledge Graphs in Text-centric Information Retrieval” [Dietz et al., 2017] presented at last year’s WSDM.

https://github.com/laura-dietz/tutorial-utilizing-kg
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Understanding user behavior is the key

The ability to accurately predict the behavior of a particular user allows search engines to construct optimal result pages.
User behavioral signals

**Actions**
(e.g., click, first/last click, long click, satisfied click, repeated click)

**Times between actions**
(e.g., time between clicks, time to first/last click)
Interpretation is difficult

**Biases in user behavior** — (statistically significant) differences between probability distributions of user behavioral signal observed in different contexts

Clicks are biased towards:
- higher ranked results (**position bias**)
- visually salient results (**attention bias**)
- previously unseen results (**novelty bias**)

Click dwell times are biased

Times to first/last/satisfied clicks are biased
Applications of user behavior models

- Understand users
- Simulate users
- Features for ranking
- Evaluate search
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
    Click behavior in web search
    Time aspects of user behavior in web search
    Web search vs. sponsored search
    Take aways and future work
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Traditional click models

Graphical representation of the cascade click model.

**Pros:** Based on the probabilistic graphical model (PGM) framework

**Cons:** Structure of the underlying PGM has to be set manually
Neural click modeling framework

A neural click model for web search [Borisov et al., 2016].

Learns patterns of user behavior directly from click-through data
We model user browsing behavior as a sequence of vector states \((s_0, s_1, s_2, \ldots)\) that describes the information consumed by the user as it evolves during a query session.
Mappings I, U and Function F

\[ s_0 = \mathcal{I}(q) \]
\[ s_{r+1} = \mathcal{U}(s_r, i_r, d_{r+1}) \]

\[ P(C_{r+1} = 1 \mid q, i_1, \ldots, i_r, d_1, \ldots, d_r) = F(s_{r+1}) \]

- \( q \) — user query
- \( d_r \) — document at rank \( r \)
- \( i_r \) — user interaction with document at rank \( r \)
Neural click modeling framework $\rightarrow$ NCM$_{\{\text{RNN, LSTM}\}}$$_{\{\text{QD, QD+Q, QD+Q+D}\}}$

**Representations of** $q$, $d_r$ and $i_r$

Use three sets: QD, QD+Q, QD+Q+D

**Parameterization of** $I$, $U$ and $F$

$I$ Feed-forward neural network

$U$ Recurrent neural network (RNN, LSTM)

$F$ Feed-forward neural network

(with one output unit and the sigmoid activation function)

**Training**

Stochastic gradient descent

(with AdaDelta update rules and gradient clipping)
Experimental setup

Dataset

Yandex Relevance Prediction dataset¹
(146, 278, 823 query sessions)

Tasks and evaluation metrics

Click prediction task (log-likelihood, perplexity)
Relevance prediction task (NDCG)

Baselines

Dynamic Bayesian network (DBN), Dependent click model (DCM)
Click chain model (CCM), User browsing model (UBM)
## Results on click prediction task

<table>
<thead>
<tr>
<th>Click model</th>
<th>Perplexity</th>
<th>Log-likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBN</td>
<td>1.3510</td>
<td>−0.2824</td>
</tr>
<tr>
<td>DCM</td>
<td>1.3627</td>
<td>−0.3613</td>
</tr>
<tr>
<td>CCM</td>
<td>1.3692</td>
<td>−0.3560</td>
</tr>
<tr>
<td>UBM</td>
<td>1.3431</td>
<td>−0.2646</td>
</tr>
<tr>
<td>NCM\textsuperscript{RNN} \textsubscript{QD}</td>
<td>1.3379</td>
<td>−0.2564</td>
</tr>
<tr>
<td>NCM\textsuperscript{LSTM} \textsubscript{QD}</td>
<td>1.3362</td>
<td>−0.2547</td>
</tr>
<tr>
<td>NCM\textsuperscript{LSTM} \textsubscript{QD+Q}</td>
<td>1.3355</td>
<td>−0.2545</td>
</tr>
<tr>
<td>NCM\textsuperscript{LSTM} \textsubscript{QD+Q+D}</td>
<td>1.3318</td>
<td>−0.2526</td>
</tr>
</tbody>
</table>

Differences between all pairs of click models are statistically significant ($p < 0.001$)
### Results on relevance prediction task

<table>
<thead>
<tr>
<th>Click model</th>
<th>NDCG @1</th>
<th>NDCG @3</th>
<th>NDCG @5</th>
<th>NDCG @10</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBN</td>
<td>0.717</td>
<td>0.725</td>
<td>0.764</td>
<td>0.833</td>
</tr>
<tr>
<td>DCM</td>
<td>0.736</td>
<td>0.746</td>
<td>0.780</td>
<td>0.844</td>
</tr>
<tr>
<td>CCM</td>
<td>0.741</td>
<td>0.752</td>
<td>0.785</td>
<td>0.846</td>
</tr>
<tr>
<td>UBM</td>
<td>0.724</td>
<td>0.737</td>
<td>0.773</td>
<td>0.838</td>
</tr>
<tr>
<td>NCM&lt;sub&gt;RNN&lt;/sub&gt;&lt;sup&gt;QD&lt;/sup&gt;</td>
<td>0.762</td>
<td>0.759</td>
<td>0.791</td>
<td>0.851</td>
</tr>
<tr>
<td>NCM&lt;sub&gt;LSTM&lt;/sub&gt;&lt;sup&gt;QD&lt;/sup&gt;</td>
<td>0.756</td>
<td>0.759</td>
<td>0.789</td>
<td>0.850</td>
</tr>
<tr>
<td>NCM&lt;sub&gt;LSTM&lt;/sub&gt;&lt;sup&gt;QD+Q&lt;/sup&gt;</td>
<td>0.775</td>
<td>0.773</td>
<td>0.799</td>
<td>0.857</td>
</tr>
<tr>
<td>NCM&lt;sub&gt;LSTM&lt;/sub&gt;&lt;sup&gt;QD+Q+D&lt;/sup&gt;</td>
<td>0.755</td>
<td>0.755</td>
<td>0.787</td>
<td>0.847</td>
</tr>
</tbody>
</table>

Improvements of NCM<sub>RNN</sub><sup>QD</sup>, NCM<sub>LSTM</sub><sup>QD</sup> and NCM<sub>LSTM</sub><sup>QD+Q</sup> over baseline click models are statistically significant ($p < 0.05$).
Analysis

Learns regularities in user browsing behavior that

1. have been manually encoded in existing click models, such as ranks and distances to previous clicks (large clusters on t-SNE projections of vector states $s_r$)
2. can not be manually encoded in traditional click models (small clusters on t-SNE projections of vector states $s_r$)
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
    Click behavior in web search
    Time aspects of user behavior in web search
    Web search vs. sponsored search
    Take aways and future work

Generating responses
Recommender systems
Industry insights
Q & A
Times between user actions

- **time-to-first-click** (reflects quality of result presentation)
- **time-between-clicks** (proxy for click dwell time)
- **time-to-last-click** (reflects quality of search engine results)
- **time-from-abandoned-query** (reflects quality of search engine results in query sessions with no clicks)
How to interpret times between user actions

Average

\[ \hat{t} = \frac{1}{N} \sum_{i=1}^{N} \tau_i \]

Uncertainty: \[ \frac{1}{2} (3 + 600) \] vs. \[ \frac{1}{7} (30 + 28 + 45 + 23 + 100 + 23 + 58) \]
How to interpret times between user actions

Average

$$\hat{t} = \frac{1}{N} \sum_{i=1}^{N} \tau_i$$

Uncertainty: $\frac{1}{2}(3 + 600)$ vs. $\frac{1}{7}(30 + 28 + 45 + 23 + 100 + 23 + 58)$

Fit distribution (e.g., exponential, gamma, Weibull)

$$\hat{\theta} = \arg \max_{\theta} \prod_{i=1}^{N} f(\tau_i \mid \theta)$$

Context bias: $f_{\text{high expectation}}(\tau = 15 \mid \theta_1)$ vs. $f_{\text{low expectation}}(\tau = 10 \mid \theta_2)$
Detected context bias effect

Percentage of action IDs with observed context bias effect

- time-to-first-click
- time-between-clicks
- time-to-last-click
- time-from-abandoned-query

Significance level (0.05)

Minimum number of observations in context groups

40 60 80 100 120 140 160 180
Context-aware time modeling (naive)

\[ \text{Time}(\text{action, context}) \sim \text{Gamma}(k(\text{act, ctx}), \theta(\text{act, ctx})) \]
Context-aware time modeling

\[ \text{Time}(\text{action, context}) \sim \text{Gamma}( \alpha_{k}(\text{ctx}) \cdot k(\text{act}) + b_{k}(\text{ctx}), \\alpha_{\theta}(\text{ctx}) \cdot \theta(\text{act}) + b_{\theta}(\text{ctx})) \]
Parameter estimation

\[
\text{Time}(\text{action, context}) \sim \text{Gamma}( \\
\quad a_k(\text{ctx}) \cdot k(\text{act}) + b_k(\text{ctx}), \\
\quad a_{\theta}(\text{ctx}) \cdot \theta(\text{act}) + b_{\theta}(\text{ctx}))
\]

1. Fix context-independent parameters
2. Optimize context-dependent parameters using neural networks
3. Fix context-dependent parameters
4. Optimize context-independent using gradient descent
5. Repeat until convergence
Parameter estimation

- We do not know the form of context-dependent parameters
  $\rightarrow$ neural networks

- We know the form of context-independent parameters (Gamma distribution)
  $\rightarrow$ direct optimization
### Dataset

3 months of log data from Yandex search engine

<table>
<thead>
<tr>
<th>Time between actions</th>
<th>Max time</th>
<th># Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-to-first-click</td>
<td>1 min</td>
<td>30,747,733</td>
</tr>
<tr>
<td>Time-between-clicks</td>
<td>5 min</td>
<td>6,317,834</td>
</tr>
<tr>
<td>Time-to-last-click</td>
<td>5 min</td>
<td>30,446,973</td>
</tr>
<tr>
<td>Time-from-abandoned-query</td>
<td>1 min</td>
<td>11,523,351</td>
</tr>
</tbody>
</table>
Evaluation tasks

**Task1.** Predict time between clicks
   - Log-likelihood
   - Root mean squared error (MSE)

**Task2.** Rank results based on time between clicks
   - nDCG@\{1, 3, 5, 10\}
## Task 1. Predicting time

<table>
<thead>
<tr>
<th>Time model</th>
<th>Distribution</th>
<th>Log-likelihood</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>exponential</td>
<td>−4.9219</td>
<td>60.73</td>
</tr>
<tr>
<td></td>
<td>gamma</td>
<td>−4.9105</td>
<td>60.76</td>
</tr>
<tr>
<td></td>
<td>Weibull</td>
<td>−4.9077</td>
<td>60.76</td>
</tr>
<tr>
<td>Context-aware</td>
<td>exponential</td>
<td>−4.8787</td>
<td>58.93</td>
</tr>
<tr>
<td></td>
<td>gamma</td>
<td>−4.8556</td>
<td>58.98</td>
</tr>
<tr>
<td></td>
<td>Weibull</td>
<td>−4.8504</td>
<td>58.94</td>
</tr>
</tbody>
</table>
Results on time prediction task (time-between-clicks)

- Modeling user behavior

![Graph showing log-likelihood across different time intervals for various distributions: exponential (basic), exponential (context-aware), gamma (basic), gamma (context-aware), weibull (basic), and weibull (context-aware). The x-axis represents time between clicks (in seconds) ranging from 1 to 256, while the y-axis represents log-likelihood ranging from -6.0 to -3.0. The graph shows how each distribution performs across different time intervals.]
## Task 2. Ranking results

<table>
<thead>
<tr>
<th>Time model</th>
<th>Distribution</th>
<th>NDCG @1</th>
<th>NDCG @3</th>
<th>NDCG @5</th>
<th>NDCG @10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>—</td>
<td>0.651</td>
<td>0.693</td>
<td>0.728</td>
<td>0.812</td>
</tr>
<tr>
<td>Context-aware</td>
<td>exponential</td>
<td>0.668</td>
<td>0.710</td>
<td>0.743</td>
<td>0.820</td>
</tr>
<tr>
<td></td>
<td>gamma</td>
<td>0.675</td>
<td>0.715</td>
<td>0.748</td>
<td>0.822</td>
</tr>
<tr>
<td></td>
<td>Weibull</td>
<td>0.671</td>
<td>0.709</td>
<td>0.745</td>
<td>0.821</td>
</tr>
</tbody>
</table>
Summary

- Remove context bias from time between actions
- Predict user search interactions better (Task 1)
- Use the context-independent component for better document ranking (Task 2)
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
    Click behavior in web search
    Time aspects of user behavior in web search
    Web search vs. sponsored search
    Take aways and future work

Generating responses
Recommender systems
Industry insights
Q & A
Web search vs. sponsored search

In **web search** we work (mostly) with query sessions and search sessions.

In **sponsored search** we need to consider longer user histories.

**Recurrent Neural Networks (RNNs)** can be used not only to account for biases, but also to infer user interests and behavioral patterns from very long sequences of user actions.

Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks [Zhang et al., 2014]
Web search vs. sponsored search

A Convolutional Click Prediction Model [Liu et al., 2015]

Modeling user behavior
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
    Click behavior in web search
    Time aspects of user behavior in web search
    Web search vs. sponsored search
    Take aways and future work

Generating responses
Recommender systems
Industry insights
Q & A
Take aways and future work

**Neural Networks** — an alternative to probabilistic graphical models (PGMs) that allows to learn patterns of user behavior directly from the data

**Understanding and modelling user behavior with PGMs** — is a mature field

We expect many ideas to be transfered from PGM to neural framework

**Future user behavior models**
- will learn patterns of user behavior directly from the data
- will take very long user history into account
- will extract signals from images, videos, user voice and background sounds
- will improve our understanding of humankind
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Outline

Morning program
- Preliminaries
- Semantic matching
- Learning to rank
- Entities

Afternoon program
- Modeling user behavior
- Generating responses
  - One-shot dialogues
  - Open-ended dialogues (chit-chat)
  - Goal-oriented dialogues
  - Alternatives to RNNs
- Resources
- Recommender systems
- Industry insights
- Q & A
Generating responses

Tasks

- Question Answering
- Summarization
- Query Suggestion
- Reading Comprehension / Wiki Reading
- Dialogue Systems
  - Goal-Oriented
  - Chit-Chat
Example Scenario for machine reading task

Sandra went to the kitchen. Fred went to the kitchen. Sandra picked up the milk. Sandra traveled to the office. Sandra left the milk. Sandra went to the bathroom.

- Where is the milk now? A: office
- Where is Sandra? A: bathroom
- Where was Sandra before the office? A: kitchen
Example Scenario for machine reading task

Sandra went to the kitchen. Fred went to the kitchen. Sandra picked up the milk. Sandra traveled to the office. Sandra left the milk. Sandra went to the bathroom.

- Where is the milk now? A: office
- Where is Sandra? A: bathroom
- Where was Sandra before the office? A: kitchen

I’ll be going to Los Angeles shortly. I want to book a flight. I am leaving from Amsterdam. I want the return flight to be early morning. I don't have any extra luggage. I wouldn’t mind extra leg room.

- What does the user want? A: Book a flight
- Where is the user flying from? A: Amsterdam
- Where is the user going to? A: Los Angeles
What is Required?

- The model needs to remember the context
- It needs to know what to look for in the context
- Given an input, the model needs to know where to look in the context
- It needs to know how to reason using this context
- It needs to handle changes in the context

A Possible Solution:

- Hidden states of RNNs have memory: Run an RNN on the and get its representation to map question to answers/response.

This will not scale as RNN states don’t have ability to capture long term dependency: vanishing gradients, limited state size.
Teaching Machine to Read and Comprehend

(a) Attentive Reader.

(b) Impatient Reader.

[Hermann et al., 2015]
Neural Networks with Memory

- Memory Networks
  - End2End MemNNs
  - Key-Value MemNNs
- Neural Turing Machines
- Stack/List/Queue Augmented RNNs
End2End Memory Networks [Sukhbaatar et al., 2015]

Generating responses

\[ p_i = \text{Softmax}(u^T m_i) \]

\[ o = \sum_i p_i c_i \]

\[ \hat{a} = \text{Softmax}(W(o + u)) \]
End2End Memory Networks [Sukhbaatar et al., 2015]

- Share the input and output embeddings or not
- What to store in memories: individual words, word windows, full sentences
- How to represent the memories? Bag-of-words? RNN reading of words? Characters?
Attentive Memory Networks [Kenter and de Rijke, 2017]

Framing the task of conversational search as a general machine reading task.
Key-Value Memory Networks

Example:
for a KB triple [subject, relation, object], Key could be [subject,relation] and value could be [object] or vice versa.

[Miller et al., 2016]
**WikiReading [Hewlett et al., 2016, Kenter et al., 2018]**

Task is based on Wikipedia data (datasets available in English, Turkish and Russian).

<table>
<thead>
<tr>
<th>Document</th>
<th>Categorization</th>
<th>Extraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Folkart Towers are twin skyscrapers in the Bayraklı district of the Turkish city of İzmir. Reaching a structural height of 200 m (656 ft) above ground level, they are the tallest . . .</td>
<td>Angeles blancos is a Mexican telenovela produced by Carlos So-tomayor for Televisa in 1990. Jacqueline Andere, Rogelio Guerra and Alfonso Iturralde star as the main . . .</td>
<td>Canada is a country in the northern part of North America. Its ten provinces and three territories extend from the Atlantic to the Pacific and northward into the Arctic Ocean, . . .</td>
</tr>
<tr>
<td>Property</td>
<td>country</td>
<td>original language of work</td>
</tr>
<tr>
<td>Turkey</td>
<td>Spanish</td>
<td>Atlantic Ocean, Arctic Ocean, Pacific Ocean</td>
</tr>
</tbody>
</table>

- Categorical: relatively small number of possible answer (e.g.: instance of, gender, country).
- Relational: rare or totally unique answers (e.g.: date of birth, parent, capital).
Answer Classification: Encoding document and question, using softmax classifier to assign probability to each of to-50k answers (limited answer vocab).
- Sparse BoW Baseline, Averaged Embeddings, Paragraph Vector, LSTM Reader, Attentive Reader, Memory Network.
- Generally models with RNN and attention work better, especially at relational properties.

Answer Extraction (labeling/pointing) For each word in the document, compute the probability that it is part of the answer.
- Regardless of the vocabulary so the answer requires being mentioned in the document.
- RNN Labeler: shows a complementary set of strengths, performing better on relational properties than categorical ones.

Sequence to Sequence Encoding query and document and decoding the answer as sequences of words or characters.
- Basic seq2seq, Placeholder seq2seq, Basic Character seq2seq.
- Unifies the classification and extraction in one model: Greater degree of balance between relational and categorical properties.
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
    One-shot dialogues
    Open-ended dialogues (chit-chat)
    Goal-oriented dialogues
    Alternatives to RNNs
    Resources

Recommender systems
Industry insights
Q & A
Dialogues/conversational agents/chat bots

Open-ended dialogues
- ELIZA
- Twitterbots
- Alexa/Google home/Siri/Cortana

Goal-oriented dialogues
- Restaurant finding
- Hotel reservations
- Set an alarm clock
- Order a pizza
- Play music
- Alexa/Google home/Siri/Cortana

Is this IR?
Dialogue systems

Chit-chat bots

Straightforward seq-to-seq [Vinyals and Le, 2015].
([Sordoni et al., 2015] is a precursor, but no RNN-to-RNN, and no LSTM).

Same idea, but with attention [Shang et al., 2015]
Dialogue systems

Limitations
- ‘Wrong’ optimization criterion
- Generic responses
- No way to incorporate world knowledge
- No model of conversation
  - Inconsistency
  - No memory of what was said earlier on

Evaluation
- Perplexity?
- BLUE/METEOR?
- Nice overview of *How NOT To Evaluate Your Dialogue System* [Liu et al., 2016].
- Open problem....

Example from [Vinyals and Le, 2015]

Human: what is your job?
Machine: i'm a lawyer.
Human: what do you do?
Machine: i'm a doctor.
Generating responses

Dialogue systems

3 solutions

- More consistency in dialogue with hierarchical network
- Less generic responses with different optimization function
- More natural responses with GANs
Hierarchical seq-to-seq [Serban et al., 2016]. Main evaluation metric: perplexity.
Generating responses

Dialogue systems

Avoid generic responses

Usually: optimize log likelihood of predicted utterance, given previous context:

\[
C_{LL} = \arg \max_{u_t} \log p(u_t | \text{context}) = \arg \max_{u_t} \log p(u_t | u_0 \ldots u_{t-1})
\]

To avoid repetitive/boring answer (I don't know), use maximum mutual information between previous context and predicted utterance [Li et al., 2015].

\[
C_{MMI} = \arg \max_{u_t} \log \frac{p(u_t, \text{context})}{p(u_t)p(\text{context})}
\]

\[
= [\text{derivation, next page \ldots}]
\]

\[
= \arg \max_{u_t} (1 - \lambda) \log p(u_t | \text{context}) + \lambda \log p(\text{context}|u_t)
\]
Bayes rule

\[
\log p(u_t | \text{context}) = \log \frac{p(\text{context} | u_t) p(u_t)}{p(\text{context})}
\]

\[
\log p(u_t | \text{context}) = \log p(\text{context} | u_t) + \log p(u_t) - \log p(\text{context})
\]

\[
\log p(u_t) = \log p(u_t | \text{context}) - \log p(\text{context} | u_t) + \log p(\text{context})
\]

\[
C_{\text{MMI}} = \arg \max_{u_t} \log \frac{p(u_t, \text{context})}{p(u_t)p(\text{context})} = \arg \max_{u_t} \log \frac{p(u_t | \text{context}) p(\text{context})}{p(u_t) p(\text{context})}
\]

\[
= \arg \max_{u_t} \log \frac{p(u_t | \text{context})}{p(u_t)}
\]

\[
= \arg \max_{u_t} \log p(u_t | \text{context}) - \log p(u_t) \quad \leftarrow \text{Weird, minus language model score.}
\]

\[
= \arg \max_{u_t} \log p(u_t | \text{context}) - \lambda \log p(u_t) \quad \leftarrow \text{Introduce } \lambda. \text{ Crucial step! Without this it wouldn’t work.}
\]

\[
= \arg \max_{u_t} \log p(u_t | \text{context}) - \lambda (\log p(u_t | \text{context}) - \log p(\text{context} | u_t) + \log p(\text{context}))
\]

\[
= \arg \max_{u_t} (1 - \lambda) \log p(u_t | \text{context}) + \lambda \log p(\text{context} | u_t)
\]

(More is needed to get it to work. See [Li et al., 2015] for more details.)
Generative adversarial network for dialogues

- Discriminator network
  - Classifier: real or generated utterance
- Generator network
  - Generate a realistic utterance

Original GAN paper [Goodfellow et al., 2014].
Conditional GANs, e.g. [Isola et al., 2016].
Generative adversarial network for dialogues

- Discriminator network
  - Classifier: real or generated utterance
- Generator network
  - Generate a realistic utterance

See [Li et al., 2017] for more details.

Dialogue systems

Open-ended dialogue systems

- Very cool, current problem
- Very hard
- Many problems
  - Training data
  - Evaluation
  - Consistency
  - Persona
  - ...
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
    One-shot dialogues
    Open-ended dialogues (chit-chat)
    Goal-oriented dialogues
  Alternatives to RNNs
  Resources
  Recommender systems
  Industry insights
  Q & A
Goal-oriented

Idea
- Closed domain
  - Restaurant reservations
  - Finding movies
- Have a dialogue system find out what the user wants

Challenges
- Training data
- Keeping track of dialogue history
- Handling of out-of-domain words or requests
- Going beyond task-specific slot filling
- Intermingling live API calls, chit chat, information requests, etc.
- Evaluation
  - Solve the task
  - Naturalness
  - Tone of voice
  - Speed
  - Error recovery
Goal-oriented as seq2seq

Memory network [Bordes and Weston, 2017]

- Simulated dataset
- Finite set of things the bot can say
  - Because of the way the dataset is constructed
- Memory networks
- Training: next utterance prediction
- Evaluation
  - response-level
  - dialogue-level

Restaurant Knowledge Base, i.e., a table. Queried by API calls.
Each row = restaurant:
- cuisine (10 choices, e.g., French, Thai)
- location (10 choices, e.g., London, Tokyo)
- price range (cheap, moderate or expensive)
- rating (from 1 to 8)

For words of relevant entity types
- add a trainable entity vector
Goal-oriented as reinforcement learning

A typical reinforcement learning system:

- States $S$
- Actions $A$
- State transition function: $T : S, A \rightarrow S$
- Reward function: $R : S, A, S \rightarrow \mathbb{R}$
- Policy: $\pi : S \rightarrow A$

A RL system needs an environment to interact with (e.g., real users).

Typically [Shah et al., 2016]:

- States: agents interpretation of the environment: distribution over user intents, dialogue acts and slots and their values
  - intent(buy_ticket)
  - inform(destination=Atlanta)
  - ...

- Actions: possible communications, and are usually designed as a combination of dialogue act tags, slots and possibly slot values
  - request(departure_date)
  - ...

- ...
Goal-oriented as reinforcement learning

Restaurant finding [Wen et al., 2017]:

- Neural belief tracking: distribution over a possible values of a set of slots
- Delexicalisation: swap slot-values for generic token (e.g. Chinese, Indian, Italian → FOOD_TYPE)

Movie finding [Dhingra et al., 2017]:

- Simulated user
- Soft attention over database
- Neural belief tracking:
  - Multinomial distribution for every column over possible column values
  - RNN, input is dialogue so far, output softmax over possible column values

Reward based on finding the right KB entry.
Generating responses

Goal-oriented

Goal-oriented models

- Currently works primarily in very small domains
- How about multiple speakers?
- Not clear what kind of architecture is best
- Reinforcement learning might be the way to go (?)
- Open research area...
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
    One-shot dialogues
    Open-ended dialogues (chit-chat)
    Goal-oriented dialogues
    Alternatives to RNNs
  Resources
  Recommender systems
  Industry insights
  Q & A
Alternatives to RNNs

RNNs are:

- Well-studied
- Robust and tried and trusted method for sequence tasks

However, RNNs have several drawbacks:

- Take time to train
- Expensive to unroll for many steps
- Not too good at catching long-term dependencies

Can we do better?

- WaveNet
- ByteNet
- Transformer
Alternatives to RNNs: WaveNet

WaveNet is originally introduced for a text-to-speech task (i.e. generating realistic audio waves).

We try to model:

\[
p(x) = \prod_{t=1}^{T} p(x_t|x_1, \ldots, x_{t-1}).
\]

- Stack of convolutional layers. No pooling layers.
- Output of the model has the same time dimensionality as the input.
- Output is a categorical distribution over the next value \(x_t\) with a softmax layer and it is optimized to maximize the log-likelihood of the data w.r.t. the parameters.

Based on the idea of dilated causal convolutions.

[van den Oord et al., 2016]
Alternatives to RNNs: WaveNet

Causal convolutions

[van den Oord et al., 2016]
Alternatives to RNNs: WaveNet

Dilated causal convolutions

“At training time, the conditional predictions for all timesteps can be made in parallel because all timesteps of ground truth $x$ are known. When generating with the model, the predictions are sequential: after each sample is predicted, it is fed back into the network to predict the next sample.”

[van den Oord et al., 2016]
Alternatives to RNNs: ByteNet

[Kalchbrenner et al., 2016]
Alternatives to RNNs: Transformer

- Positional encoding added to the input embeddings
- Key-value attention
- Multi-head self-attention
- The encoder attends over its own states
- The decoder alters between
  - attending over its own inputs/states
  - attending over encoder states at the same level

[Vaswani et al., 2017]
Alternatives to RNNs: Transformer
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
    One-shot dialogues
    Open-ended dialogues (chit-chat)
    Goal-oriented dialogues
    Alternatives to RNNs
  Resources
  Recommender systems
  Industry insights
  Q & A
Resources: datasets

Open-ended dialogue
- Opensubtitles [Tiedemann, 2009]
- Twitter: http://research.microsoft.com/convo/
- Weibo: http://www.noahlab.com.hk/topics/ShortTextConversation
- Ubuntu Dialogue Corpus [Lowe et al., 2015]
- Switchboard https://web.stanford.edu/~jurafsky/ws97/
- Coarse Discourse (Google Research) https://research.googleblog.com/2017/05/coarse-discourse-dataset-for.html

Goal-oriented dialogues
- MISC: A data set of information-seeking conversations [Thomas et al., 2017]
- Maluuba Frames http://datasets.maluuba.com/frames
- Loqui Human-Human Dialogue Corpus https://academiccommons.columbia.edu/catalog/ac:176612
- bAbi (Facebook Research) https://research.fb.com/downloads/babi/

Machine reading
- bAbi QA (Facebook Research) https://research.fb.com/downloads/babi/
- QA Corpus [Hermann et al., 2015] https://github.com/deepmind/rc-data/
- WikiReading (Google Research) https://github.com/google-research-datasets/wiki-reading
### Generating responses

**Resources: source code**

- **End-to-end memory network**
  
  [https://github.com/facebook/MemNN](https://github.com/facebook/MemNN)

- **Attentive Memory Networks**
  
  [https://bitbucket.org/TomKenter/attentive-memory-networks-code](https://bitbucket.org/TomKenter/attentive-memory-networks-code)

- **Hierarchical NN [Serban et al., 2016]**
  

- **GAN for dialogues**
  

- **RL for dialogue agents [Dhingra et al., 2017]**
  
  [https://github.com/MiuLab/KB-InfoBot](https://github.com/MiuLab/KB-InfoBot)

- **Transformer network**
  
  [https://github.com/tensorflow/tensor2tensor](https://github.com/tensorflow/tensor2tensor)
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
    Items and Users
    Matrix factorization
    Matrix factorization as a network
    Side information
    Richer models
    Other tasks
    Wrap-up

Industry insights
Q & A
Recommender systems – The task

- Build a model that estimates how a user will like an item.
- A typical recommendation setup has
  - matrix with users and items
  - plus ratings of users for items reflecting past/known preferences and tries to predict future preferences
- This is not about rating prediction

[Karatzoglou and Hidasi, Deep Learning for Recommender Systems, RecSys ‘17, 2017]
Approaches to recommender systems

- Collaborative filtering
  - Based on analyzing users' behavior and preferences such as ratings given to movies or books
- Content-based filtering
  - Based on matching the descriptions of items and users' profiles
  - Users' profiles are typically constructed using their previous purchases/ratings, their submitted queries to search engines and so on
- A hybrid approach
Cold start problem

- **User** cold-start problem – generate recommendations for a new user / a user for whom very few preferences are known
- **Item** cold-start problem – recommendation items that are new / for which very users have shared ratings or preferences

- Cold items/users
- Warm items/users
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
    Items and Users
    Matrix factorization
    Matrix factorization as a network
    Side information
    Richer models
    Other tasks
    Wrap-up

Industry insights
Q & A
Matrix factorization

- The recommender system’s work horse

\[ \min_{u,v} X_{i,j} \in \mathbb{R} \left( r_{i,j} u^T i v^T j \right) + \left( \| u_i \|^2 + \| v_j \|^2 \right) \]

\[ R \]

\[ U \]

\[ V \]
Recommender systems

Matrix factorization

- Discover the latent features underlying the interactions between users and items
- Don’t rely on imputation to fill in missing ratings and make matrix dense
- Instead, model observed ratings directly, avoid overfitting through a regularized model
- Minimize the regularized squared error on the set of known ratings:

\[
\min_{u,v} \sum_{i,j \in R} (r_{i,j} - u_i^T v_j) + \lambda(\|u_i\|^2 + \|v_j\|^2)
\]

Popular methods for minimizing include **stochastic gradient descent** and **alternating least squares**
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
    Items and Users
    Matrix factorization
    Matrix factorization as a network
    Side information
    Richer models
    Other tasks
    Wrap-up

Industry insights
Q & A
A feed-forward neural network view

\[ u_i^T v_j \rightarrow \min_{u,v} \sum_{i,j \in R} (r_{ij} - u_i^T v_j) + \lambda(||u_i||^2 + ||v_j||^2) \]

[Raimon and Basilico, Deep Learning for Recommender Systems, 2017]
A deeper view
Matrix factorization vs. feed-forward network

- Two models are very similar
  - Embeddings, MSE loss, gradient-based optimization
- Feed-forward net can learn different embedding combinations than a dot product
- Capturing pairwise interactions through feed-forward net requires a huge amount of data
- This approach is not superior to properly tuned traditional matrix factorization approach
Great escape . . .

- Side information
- Richer models
- Other tasks
Outline

**Morning program**
- Preliminaries
- Semantic matching
- Learning to rank
- Entities

**Afternoon program**
- Modeling user behavior
- Generating responses
- **Recommender systems**
  - Items and Users
  - Matrix factorization
  - Matrix factorization as a network
  - **Side information**
  - Richer models
  - Other tasks
  - Wrap-up

**Industry insights**

**Q & A**
Side information for recommendation
Side information for recommendation

- Textual side information
  - Product description, reviews, etc.
  - Extraction: RNNs, one dimensional CNNs, word embeddings, paragraph vectors
  - Applications: news, products, books, publication

- Images
  - Product pictures, video thumbnails
  - Extraction: CNNs
  - Applications: fashion, video

- Music/audio
  - Extraction: CNNs and RNNs
  - Applications: music
Textual side information

- Content2vec [Nedelec et al., 2016]
- Using associated textual information for recommendations [Bansal et al., 2016]
Textual information for improving recommendations

- Task: paper recommendation
- Item representation
  - Text representation: RNN based
  - Item-specific embeddings created using MF
  - Final representation: item + text embeddings
Images in recommendation

Visual Bayesian Personalized Ranking (BPR) [He and McAuley, 2016]

- Bias terms
- MF model
- Visual part:
  - Pretrained CNN features
  - Dimension reduction through embeddings
- BPR loss
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
    Items and Users
    Matrix factorization
    Matrix factorization as a network
    Side information
    Richer models
  Other tasks
  Wrap-up

Industry insights
Q & A
Recommender systems

Alternative models

- Restricted Boltzman Machines [Salakhutdinov et al., 2007]
- Auto-encoders [Wu et al., 2016]
- Prod2vec [Grbovic et al., 2015]
- Wide + Deep models [Cheng et al., 2016]
Recommender systems

Restricted Boltzmann Machines – RBM

- Generative stochastic neural network
- Visible and hidden units connected by weights
- Activation probabilities:
  \[ p(h_j = 1|v) = \sigma(b^h_j + \sum_{i=1}^{m} w_{i,j}v_i) \]
  \[ p(v_i = 1|h) = \sigma(b^v_i + \sum_{j=1}^{n} w_{i,j}h_j) \]
- Training
  - Set visible units based on data, sample hidden units, then sample visible units
  - Modify weights to approach the configuration of visible units to the data
- In recommendation:
  - Visible units: ratings on the movie
  - Vector of length 5 (for each rating value) in each unit
  - Units corresponding to users who not rated the movie are ignored
Auto-encoders

- One hidden layer
- Same number of input and output units
- Try to reconstruct the input on the output
- Hidden layer: compressed representation of the data

Constraining the model: improve generalization
- Sparse auto-encoders: activation of units are limited
- Denoising auto-encoders: corrupt the input
Auto-encoders for recommendation

Reconstruct corrupted user interaction vectors [Wu et al., 2016]

- Collaborative Denoising Auto-Encoder (CDAE)
- The link between nodes are associated with different weights
- The links with red color are user specific
- Other weights are shared across all the users
Recommender systems

Prod2vec and Item2vec

- Prod2vec and item2vec: Item-item co-occurrence factorization
- User2vec: User-user co-occurrence factorization
- The two approaches can be combined [Liang et al., 2016]
Wide + Deep models

- Combination of two models
- Deep neural network
  - On embedded item features
  - In charge of generalization
- Linear model
  - On embedded item feature
  - And cross product of item features
  - In charge of memorization on binarized features
- [Cheng et al., 2016]
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
    Items and Users
    Matrix factorization
    Matrix factorization as a network
    Side information
    Richer models
  Other tasks
    Wrap-up

Industry insights
Q & A
Other tasks

- Session-based recommendation
- Contextual sequence prediction
- Time-sensitive sequence prediction
- Causality in recommendations
- Recommendation as question answering
- Deep reinforcement learning for recommendations
Session-based recommendation

- Treat recommendations as a sequence classification problem
- Input: a sequence of user actions (purchases/ratings of items)
- Output: next action
- Disjoint sessions (instead of consistent user history)
Recommender systems

**GRU4Rec**

Network structure [Hidasi et al., 2016]
- Input: one hot encoded item ID
- Output: scores over all items
- Goal: predicting the next item in the session

Adapting GRU to session-based recommendations
- Session-parallel mini-batching: to handle sessions of (very) different length and lots of short sessions
- Sampling on the output: to handle lots of items (inputs, outputs)
GRU4Rec

Session-parallel mini-batches
  ▶ Mini-batch is defined over sessions

Output sampling
  ▶ Computing scores for all items (100K 1M) in every step is slow
  ▶ One positive item (target) + several samples
  ▶ Fast solution: scores on mini-batch targets
  ▶ Items of the other mini-batches are negative samples for the current mini-batch
Contextual sequence prediction

- Input: sequence of contextual user actions, plus current context
- Output: probability of next action
- E.g. “Given all the actions a user has taken so far, what’s the most likely video they’re going to play right now?” [Beutel et al., 2018]
Time-sensitive sequence prediction

- Recommendations are actions at a moment in time
  - Proper modeling of time and system dynamics is critical
- Experiment on a Netflix internal dataset
  - Context:
    - Discrete time – Day-of-week: Sunday, Monday, ... Hour-of-day
    - Continuous time (Timestamp)
  - Predict next play (temporal split data)

[Raimon and Basilico, Deep Learning for Recommender Systems, 2017]
And now, for some speculative tasks in the recommender systems space

Answers not clear, but good potential for follow-up research
Recommender systems

Causality in recommendations – [Schnabel et al., 2016]

- Virtually all data for training recommender systems is subject to selection biases
  - In movie recommendation, users typically watch and rate movies they like, rarely movies they do not like
- View recommendation from causal inference perspective – exposing user to item is intervention analogous to exposing patient to treatment in medical study
- Propensity-weighted MF method – propensities act as weights on loss terms

\[
\min_u, v \sum_{i,j \in R} \frac{1}{P_{i,j}}(r_{i,j} - u_i^T v_j) + \lambda(\|u_i\|^2 + \|v_j\|^2)
\]

- Performance: MF vs. propensity weighted MF

(As \( \alpha \to 0 \), data is increasingly missing not at random, only revealing top rated items.)

- How to incorporate this in neural networks for recommender systems?
Recommendations as question answering – [Dodge et al., 2015]

- Conversational recommendation agent
  - (1) question-answering (QA), (2) recommendation, (3) mix of recommendation and QA and (4) general dialog about the topic (chit-chat)

Task 2: Recommendation
Some movies I like are Heat, Kids, Fight Club, Shaun of the Dead, The Avengers, Skyfall, and Jurassic Park. Can you suggest something else I might like? Ocean’s Eleven

Task 3: QA + Recommendation Dialog
I loved Billy Madison, My Neighbor Totoro, Blades of Glory, Bio-Dome, Clue, and Happy Gilmore. I’m looking for a Music movie. School of Rock
What else is that about? Music, Musical, Jack Black, school, teacher, Richard Linklater, rock, guitar
I like rock and roll movies more. Do you know anything else? Little Richard
Tombstone, Legends of the Fall, Braveheart, The Net, Outbreak, and French Kiss are films I really liked. I’m looking for a Fantasy movie. Jumanji
Who directed that? Joe Johnston
I like Tim Burton movies more. Do you know anything else? Big Fish

- Memory network jointly trained on all (four) tasks performs best
- Incorporate short and long term memory and can use local context and knowledge bases of facts
- Performance on QA needs a real boost
- Performance degraded rather than improved when training on all four tasks at once
Recommender systems

Deep reinforcement learning for recommendations – [Zhao et al., 2017]

- MDP-based formulations of recommender systems go back to early 2000s
- Use of reinforcement learning has two advantages
  1. can continuously update strategies during interactions
  2. are able to learn strategy that maximizes the long-term cumulative reward from users
- List-wise recommendation framework, which can be applied in scenarios with large and dynamic item spaces
- Uses Actor-Critic network
- Integrate multiple orders – positional order, temporal order
- Needs proper evaluation in live environment
Outline

Morning program
Preliminaries
Semantic matching
Learning to rank
Entities

Afternoon program
Modeling user behavior
Generating responses
Recommender systems
  Items and Users
  Matrix factorization
  Matrix factorization as a network
  Side information
  Richer models
  Other tasks
  Wrap-up

Industry insights
Q & A
Wrap-up

▶ Current directions
  ▶ Item/user embedding
  ▶ Deep collaborative filtering
  ▶ Feature extraction from content
  ▶ Session- and context-based recommendation
  ▶ Fairness, accuracy, confidentiality, and transparency (FACT)

▶ Deep learning can work well for recommendations

▶ Matrix factorization and deep learning are very similar in classic recommendation setup

▶ Lots of areas to explore
Recommender systems

Resources

RankSys : https://github.com/RankSys/RankSys
LibRec : https://www.librec.net
LensKit : http://lenskit.org
LibMF : https://www.csie.ntu.edu.tw/%7Ecjlin/libmf/
proNet-core : https://github.com/cnclabs/proNet-core
rival : http://rival.recommenders.net
TensorRec : https://github.com/jfkirk/tensorrec
Outline

Morning program
  Preliminaries
  Semantic matching
  Learning to rank
  Entities

Afternoon program
  Modeling user behavior
  Generating responses
  Recommender systems
  Industry insights
  Q & A
Deep Learning in industry

- Companies have endless amounts of data!
  Or do they?

- Performance
  Is .9 accuracy/$F_1$/etc. good enough?
  No? Would 0.95 be?

- Business logic/constraints
  - *Your model is doing great in general, but not in case X, Y and Z.*
    - *Can you keep it exactly as it is now, and fix just these cases?*

- Explicit domain knowledge
  E.g.: recommending product X for user Y is not applicable, as it is not available where user Y lives.
Deep Learning in industry

- Hybrid Code Networks
  Combining RNNs with domain-specific knowledge
- Smart Reply
  Automated response suggestion for email
Hybrid Code Networks

Task
Dialogue system. User can converse with a system that can interact with APIs.

Combining RNNs with domain-specific knowledge

- Incorporate business logic by including modules in the system that can be programmed
- Explicitly condition actions on external knowledge

[Williams et al., 2017]
Hybrid Code Networks

Trapezoids refer to programmatic code provided by the software developer. Shaded boxes are trainable components.

[Williams et al., 2017]
Automated response suggestion for email
Use an RNN to generate responses for any given input message.

Additional constraints

- **Response quality**
  Ensure that the individual response options are always high quality in language and content.

- **Utility**
  Select multiple options to show a user so as to maximize the likelihood that one is chosen.

- **Scalability**
  Process millions of messages per day while remaining within the latency requirements.

- **Privacy**
  Develop this system without ever inspecting the data except aggregate statistics.

[Kannan et al., 2016]
Response selection

- Construct a set of allowed responses $R$.
- Organise the elements of $R$ into a trie.
- Conduct a left-to-right beam search, and only retain hypotheses that appear in the trie.

Complexity: $O(\text{beam size } \times \text{response length})$.

Utility/diversity

Goal: present user with diverse responses

Instead of “No”, “No, thanks”, and “Thanks!” , we’d rather produce “No, thanks”, “Yes, please”, “Let me come back to it”.

- Manually label a couple of messages per response intent.
- Use a state-of-the-art label propagation algorithm to label all other messages in $R$. 

[Kannan et al., 2016]
What do we learn?

- Deep learning component is a (small) part of a much larger system.
- Getting the right training data can be hard.
- The machine learned part is guided/corrected/prevented from predicting undesired output.
Neural IR at Bing

Long history of neural IR models at Bing/Microsoft

- RankNet/LambdaRank [Burges et al., 2005, 2006]
- ListNet/ListMLE [Cao et al., 2007, Xia et al., 2008]
- DSSM/CDSSM [Huang et al., 2013, Shen et al., 2014]
- Recent representation learning models for long text [Mitra et al., 2017, Zamani et al., 2018]

NN and GBDT are both popularly used across many teams
Neural IR at Bing

Beyond Web search, heavy use of deep learning systems for

- Speech recognition [Xiong et al., 2017c]
- Conversational models (e.g., Cortana & Zo)
- Machine translation [Hassan et al., 2018]
- Machine reading [Wang et al., 2017] and emerging Office Intelligence scenarios (e.g., [Van Gysel et al., 2017])
- And others...
Neural IR at Bing

Some of the unique challenges and considerations:

► Supervision
  ▶ Large (explicitly/implicitly) labeled datasets are available for training deep models in Web search
  ▶ Not available for many multi-tenant enterprise scenarios due to privacy and scalability considerations—distance supervision and other approaches may be necessary

► Infrastructure investments
  ▶ GPU and other machine resources for experimentation; serving infrastructure investments for running deep models in production
  ▶ Neural model based features vs. rethinking the stack with neural models as first class citizens

► Model reuse: across tenants and different services
Outline

Morning program
Preliminaries
Semantic matching
Learning to rank
Entities

Afternoon program
Modeling user behavior
Generating responses
Recommender systems
Industry insights
Q & A


Yoshua Bengio, Jean-Sébastien Senécal, and others. 2003. Quick Training of Probabilistic Neural Nets by Importance Sampling,. In AISTATS.


Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and H Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In WSDM.


Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query expansion with locally-trained word embeddings. In ACL.


References IV


References V


Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In NIPS.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. WIKIREADING: A Novel Large-scale Language Understanding Task over Wikipedia. In ACL.


References VI


References VII


Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion.. In AAAI. 2181–2187.


Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. In EMNLP.


References VIII


D. Sculley. 2009. Large scale learning to rank. In NIPS Workshop on Advances in Ranking.
References IX


Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In ACL.


Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-To-End Memory Networks. In NIPS.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling Mention, Context and Entity with Neural Networks for Entity Disambiguation. In IJCAI.


References


Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with Large-Scale Neural Language Models Improves Translation.. In EMNLP. 1387–1392.


Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In SIGIR. ACM, 363–372.


Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes.. In AAAI. 1112–1119.


References XI


<table>
<thead>
<tr>
<th>Meaning</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single query</td>
<td>$q$</td>
</tr>
<tr>
<td>Single document</td>
<td>$d$</td>
</tr>
<tr>
<td>Set of queries</td>
<td>$Q$</td>
</tr>
<tr>
<td>Collection of documents</td>
<td>$D$</td>
</tr>
<tr>
<td>Term in query $q$</td>
<td>$t_q$</td>
</tr>
<tr>
<td>Term in document $d$</td>
<td>$t_d$</td>
</tr>
<tr>
<td>Full vocabulary of all terms</td>
<td>$T$</td>
</tr>
<tr>
<td>Set of ranked results retrieved for query $q$</td>
<td>$R_q$</td>
</tr>
<tr>
<td>Result tuple (document $d$ at rank $i$)</td>
<td>$\langle i, d \rangle$, where $\langle i, d \rangle \in R_q$</td>
</tr>
<tr>
<td>Relevance label of document $d$ for query $q$</td>
<td>$rel_q(d)$</td>
</tr>
<tr>
<td>$d_i$ is more relevant than $d_j$ for query $q$</td>
<td>$rel_q(d_i) &gt; rel_q(d_j)$, or $d_i \succ_q d_j$</td>
</tr>
<tr>
<td>Frequency of term $t$ in document $d$</td>
<td>$tf(t, d)$</td>
</tr>
<tr>
<td>Number of documents that contain term $t$</td>
<td>$df(t)$</td>
</tr>
<tr>
<td>Vector representation of text $z$</td>
<td>$\vec{z}$</td>
</tr>
<tr>
<td>Probability function for an event $\mathcal{E}$</td>
<td>$p(\mathcal{E})$</td>
</tr>
<tr>
<td>$\mathbb{R}$</td>
<td>The set of real numbers</td>
</tr>
</tbody>
</table>

We adopt some neural network related notation from [Goodfellow et al., 2016] and IR related notation from [Mitra and Craswell, 2017]
Acknowledgments

All content represents the opinion of the author(s), which is not necessarily shared or endorsed by their employers and/or sponsors.