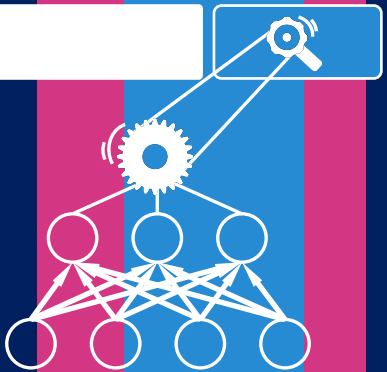
Neural Networks for Information Retrieval

nn4ir



Neural Networks for Information Retrieval

Tom Kenter[1] Alexey Borisov[1,2] Christophe Van Gysel[1] Mostafa Dehghani[1]
Maarten de Rijke[1] Bhaskar Mitra[3,4]

[1]University of Amsterdam

[2]Yandex

[3]University College London

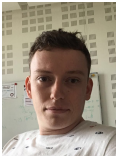
[4]Microsoft

NN4IR@SIGIR2017

Who are we?

Tom Kenter
U. Amsterdam

Alexey Borisov
Yandex & U. Amsterdam



Christophe Van Gysel
U. Amsterdam

Mostafa Dehghani
U. Amsterdam

Maarten de Rijke
U. Amsterdam

Bhaskar Mitra
Microsoft & UCL

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 1
3. Semantic matching 2

Afternoon

4. Learning to rank
5. Modeling user behavior
6. Generating responses
7. Outlook
8. Wrap up

Materials

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

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

Lecture notes: B. Mitra and N. Craswell. An introduction to neural information retrieval. Foundations and Trends in Information Retrieval, 2017, under review. `http://bit.ly/neuralir-intro`

Survey: K.D. Onal et al. Neural information retrieval: At the end of the early years. Information Retrieval Journal, 2017, under review.

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

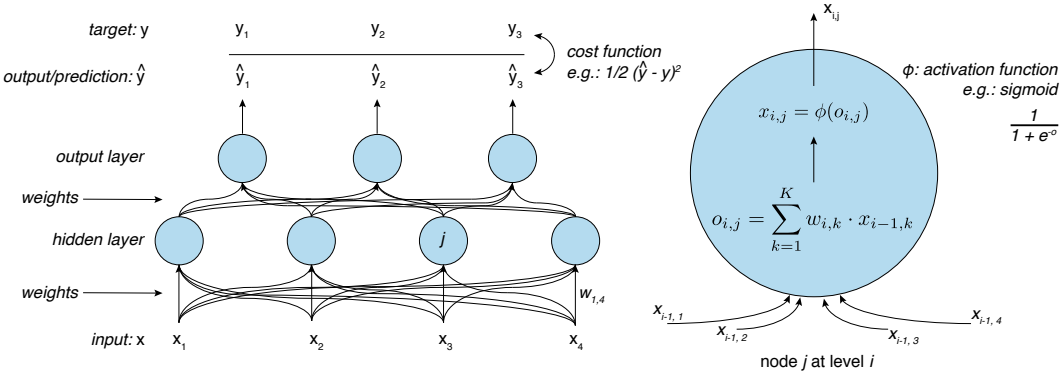
Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

Outline

Morning program
- Preliminaries
  - Feedforward neural network
  - Back propagation
  - Distributed representations
  - Recurrent neural networks
  - Sequence-to-sequence models
  - Convolutional neural networks
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

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

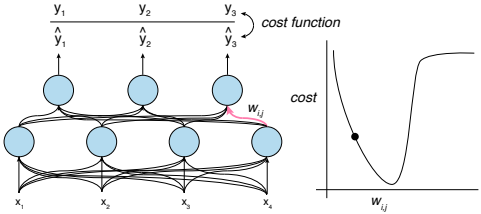


Outline

Morning program
- Preliminaries
  - Feedforward neural network
  - Back propagation
  - Distributed representations
  - Recurrent neural networks
  - Sequence-to-sequence models
  - Convolutional neural networks
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

Back propagation



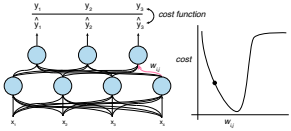
until convergence:
- do a forward pass
- compute the cost/error
- adjust weights ← how??

Adjust every weight $w_{i,j}$ by:

$$\Delta w_{i,j} = -\alpha \frac{\partial cost}{\partial w_{i,j}}$$

α is the learning rate.

Back propagation



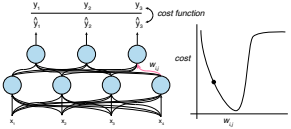
$$cost(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$

$$\hat{y_j} = x_{i,j} = \phi(o_{i,j})$$

$$\Delta w_{i,j} = -\alpha \frac{\partial cost}{\partial w_{i,j}}$$

$$= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial w_{i,j}} \quad \leftarrow \text{chain rule}$$

Back propagation



$$cost(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$

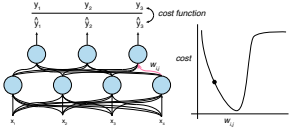
$$\hat{y_j} = x_{i,j} = \phi(o_{i,j}), \text{e.g. } \sigma(o_{i,j})$$

$$x_{i,j} = \sigma(o) = \frac{1}{1 + e^{-o}}$$

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

$$\begin{aligned}
\Delta w_{i,j} &= -\alpha \ \frac{\partial cost}{\partial w_{i,j}} \\
&= -\alpha \ \frac{\partial cost}{\partial x_{i,j}} \ \frac{\partial x_{i,j}}{\partial w_{i,j}} \\
&= -\alpha \ \frac{\partial cost}{\partial x_{i,j}} \ \frac{\partial x_{i,j}}{\partial o_{i,j}} \ \frac{\partial o_{i,j}}{\partial w_{i,j}} \quad \leftarrow \text{chain rule}
\end{aligned}$$

Back propagation



$$cost(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$

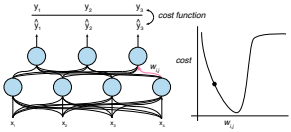
$$\hat{y}_j = x_{i,j} = \phi(o_{i,j}), \text{e.g. } \sigma(o_{i,j})$$

$$x_{i,j} = \sigma(o) = \frac{1}{1 + e^{-o}}$$

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

$$\begin{aligned}
\Delta w_{i,j} &= -\alpha \frac{\partial cost}{\partial w_{i,j}} \\
&= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial w_{i,j}} \\
&= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial o_{i,j}} \frac{\partial o_{i,j}}{\partial w_{i,j}}
\end{aligned}$$

Back propagation



$$cost(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$

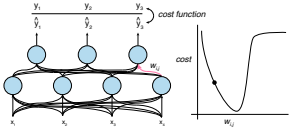
$$\hat{y}_j = x_{i,j} = \phi(o_{i,j}), \text{e.g. } \sigma(o_{i,j})$$

$$x_{i,j} = \sigma(o) = \frac{1}{1 + e^{-o}}$$

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

$$\begin{aligned}
\Delta w_{i,j} &= -\alpha \frac{\partial cost}{\partial w_{i,j}} \\
&= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial w_{i,j}} \\
&= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial o_{i,j}} \frac{\partial o_{i,j}}{\partial w_{i,j}} \\
&= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial o_{i,j}} x_{i-1,j}
\end{aligned}$$

Back propagation



$$cost(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$

$$\hat{y}_j = x_{i,j} = \phi(o_{i,j}), \text{e.g. } \sigma(o_{i,j})$$

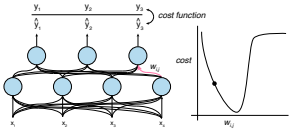
$$x_{i,j} = \sigma(o) = \frac{1}{1 + e^{-o}}$$

$$\sigma'(o) = \sigma(o)(1 - \sigma(o))$$

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

$$\begin{aligned}
\Delta w_{i,j} &= -\alpha \frac{\partial cost}{\partial w_{i,j}} \\
&= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial w_{i,j}} \\
&= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial o_{i,j}} \frac{\partial o_{i,j}}{\partial w_{i,j}} \\
&= -\alpha \frac{\partial cost}{\partial x_{i,j}} x_{i,j}(1 - x_{i,j}) \; x_{i-1,j}
\end{aligned}$$

Back propagation



$$cost(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$

$$\hat{y}_j = x_{i,j} = \phi(o_{i,j}), \text{e.g. } \sigma(o_{i,j})$$

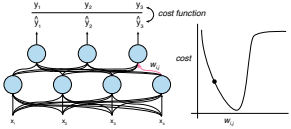
$$x_{i,j} = \sigma(o) = \frac{1}{1 + e^{-o}}$$

$$\sigma'(o) = \sigma(o)(1 - \sigma(o))$$

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

$$\begin{aligned}
\Delta w_{i,j} &= -\alpha \frac{\partial cost}{\partial w_{i,j}} \\
&= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial w_{i,j}} \\
&= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial o_{i,j}} \frac{\partial o_{i,j}}{\partial w_{i,j}} \\
&= -\alpha \; y_j - x_{i,j} \;\; x_{i,j}(1 - x_{i,j}) \;\; x_{i-1,j}
\end{aligned}$$

Back propagation



$$cost(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$

$$\hat{y}_j = x_{i,j} = \phi(o_{i,j}), \text{e.g. } \sigma(o_{i,j})$$

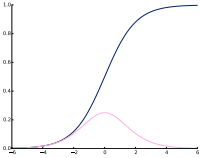
$$x_{i,j} = \sigma(o) = \frac{1}{1 + e^{-o}}$$

$$\sigma'(o) = \sigma(o)(1 - \sigma(o))$$

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

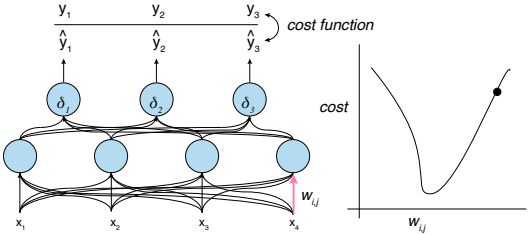
$$\begin{aligned}
\Delta w_{i,j} &= -\alpha \ \frac{\partial cost}{\partial w_{i,j}} \\
&= -\alpha \ \frac{\partial cost}{\partial x_{i,j}} \ \frac{\partial x_{i,j}}{\partial w_{i,j}} \\
&= -\alpha \ \frac{\partial cost}{\partial x_{i,j}} \ \frac{\partial x_{i,j}}{\partial o_{i,j}} \ \frac{\partial o_{i,j}}{\partial w_{i,j}} \\
&= -\alpha \ \ y_j - x_{i,j} \ \ x_{i,j}(1 - x_{i,j}) \ \ x_{i-1,j} \\
&= \text{l.rate} \ \ cost \ \ activation \ \ input
\end{aligned}$$

$\sigma(o)$ $\sigma'(o)$



Back propagation

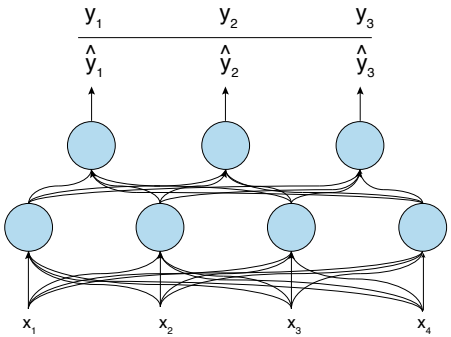
$$
\begin{aligned}
\Delta w_{i,j} &= -\alpha \ \frac{\partial cost}{\partial w_{i,j}} \\
&= -\alpha \ \frac{\partial cost}{\partial x_{i,j}} \ \frac{\partial x_{i,j}}{\partial o_{i,j}} \ \frac{\partial o_{i,j}}{\partial w_{i,j}} \\
&= \text{l.rate} \ \ cost \ \ activation \ \ input \\
&= -\alpha \ \frac{\partial cost}{\partial x_{i,j}} \ \frac{\partial x_{i,j}}{\partial o_{i,j}} \ \frac{\partial o_{i,j}}{\partial w_{i,j}} \\
&= -\alpha \ \ \delta \ \ x_{i-1,j}
\end{aligned}
$$



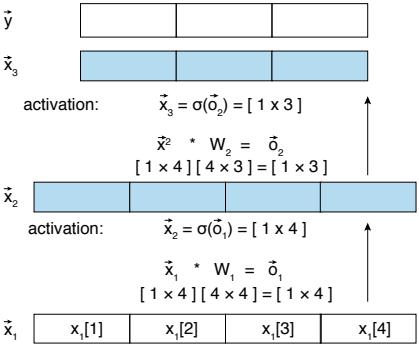
$$\delta_{output} = (y_j - x_{i,j}) \ \ x_{i,j}(1 - x_{i,j}) \quad \leftarrow \text{previous slide}$$

$$\delta_{hidden} = \left(\sum_{n \in nodes} \delta_n w_{n,j} \right) \ \ x_{i,j}(1 - x_{i,j})$$

Network representation



$\leftrightarrow$



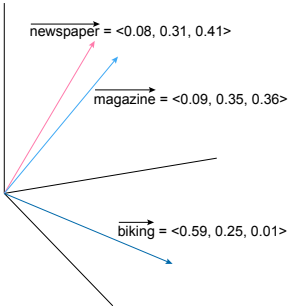
Outline

Morning program
- Preliminaries
  - Feedforward neural network
  - Back propagation
  - Distributed representations
  - Recurrent neural networks
  - Sequence-to-sequence models
  - Convolutional neural networks
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

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:

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

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

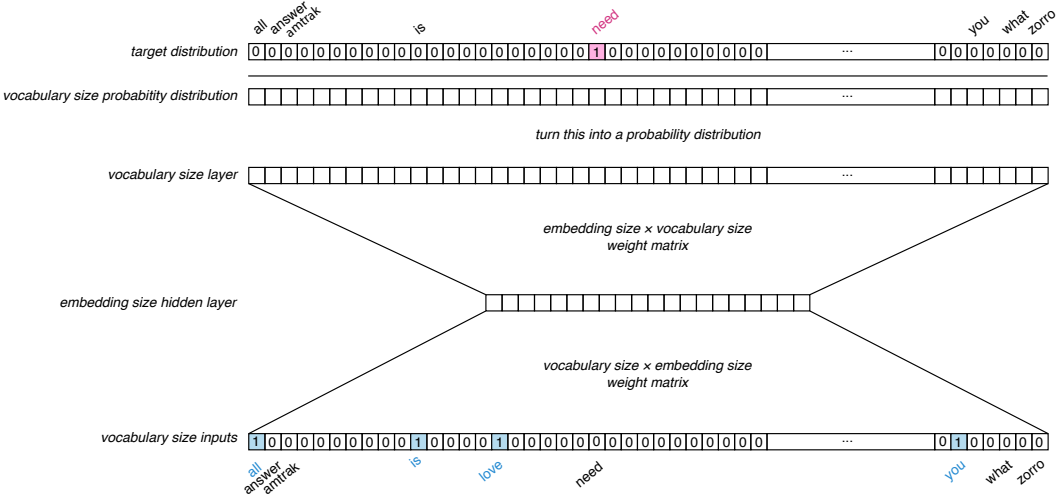
Distributed representations

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

$$\overrightarrow{all} \quad \overrightarrow{you} \quad \overrightarrow{need} \quad \overrightarrow{is} \quad \overrightarrow{love}$$

Embedding methods



Probability distributions

$$\text{softmax} = \text{normalize the logits}$$

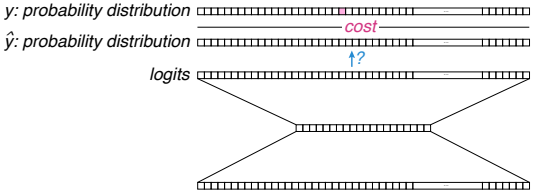
$$= \frac{e^{logits[i]}}{\sum_{j=1}^{|logits|} e^{logits[j]}}$$

$$cost = \text{cross entropy loss}$$

$$= -\sum_{x} p(x) \log \hat{p}(x)$$

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

$$= -\sum_{i} y_i \log \hat{y}_i$$



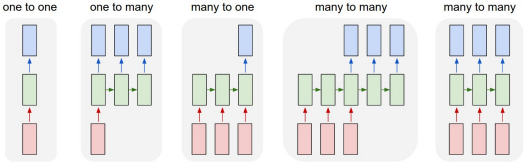
Outline

Morning program
- Preliminaries
  - Feedforward neural network
  - Back propagation
  - Distributed representations
  - Recurrent neural networks
  - Sequence-to-sequence models
  - Convolutional neural networks
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

Recurrent neural networks

- Lots of information is sequential and requires a memory for successful processing
- Sequences as input, sequences as output

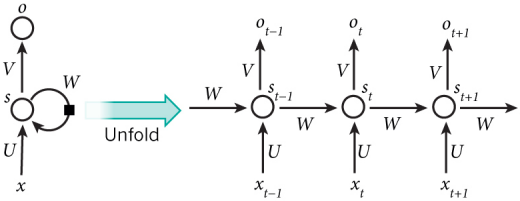


- Recurrent neural networks (RNNs) are called recurrent because they perform same task for every element of sequence, with output dependent on previous computations
- RNNs have memory that captures information about what has been computed so far
- RNNs can make use of information in arbitrarily long sequences – in practice they limited to looking back only few steps

Image credits: `http://karpathy.github.io/assets/rnn/diags.jpeg`

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 previous hidden state and input at the current step: $s_t = f(Ux_t + Ws_{t-1})$; f usually nonlinearity, e.g., $\tanh$ or ReLU; s_{-1} typically initialized to all zeroes
  - o_t output at step t. E.g.,, if we want to predict next word in sentence, a vector of probabilities across vocabulary: $o_t = \mathrm{softmax}(Vs_t)$

Image credits: Nature

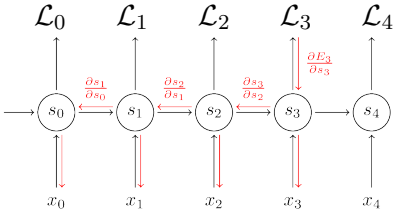
Language modeling using RNNs

- Language model allows us to predict probability of observing sentence (in a given dataset) as: $P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1})$
- In RNN, set $o_t = x_{t+1}$: we want output at step t to be actual next word
- Input x a sequence of words; each x_t is a single word; we represent each word as a one-hot vector of size `vocabulary_size`
- Initialize parameters U, V, W to small random values around 0

Language modeling using RNNs

- Language model allows us to predict probability of observing sentence (in a given dataset) as: $P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1})$
- In RNN, set $o_t = x_{t+1}$: we want output at step t to be actual next word
- Input x a sequence of words; each x_t is a single word; we represent each word as a one-hot vector of size `vocabulary_size`
- Initialize parameters U, V, W to small random values around 0
- Cross-entropy loss as loss function
- For N training examples (words in text) and C classes (the size of our vocabulary), loss with respect to predictions o and true labels y is: $\mathcal{L}(y, o) = -\frac{1}{N} \sum_{n \in N} y_n \log o_n$
- 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:

$$\frac{\partial \mathcal{L}_3}{\partial W} = \frac{\partial \mathcal{L}_3}{\partial o_3}\frac{\partial o_3}{\partial s_3}\frac{\partial s_3}{\partial W} \quad = \sum_{k=0}^{3} \frac{\partial \mathcal{L}_3}{\partial o_3}\frac{\partial o_3}{\partial s_3}\frac{\partial s_3}{\partial s_k}\frac{\partial s_k}{\partial W}$$

- More generally: $\frac{\partial \mathcal{L}}{\partial s_t} = \frac{\partial \mathcal{L}}{\partial s_m} \cdot \underbrace{\frac{\partial s_m}{\partial s_{m-1}}}_{<1} \cdot \underbrace{\frac{\partial s_{m-1}}{\partial s_{m-2}}}_{<1} \cdot \ldots \cdot \underbrace{\frac{\partial s_{t+1}}{\partial s_t}}_{<1} \Rightarrow \ll 1$
- Gradient contributions from far away steps become zero: state at those steps doesn't contribute to what you are learning

Image credits: `http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-grad`

Long Short Term Memory [Hochreiter and Schmidhuber, 1997]

LSTMs designed to combat vanishing gradients through gating mechanism

- How LSTM calculates hidden state s_t

$$
\begin{aligned}
i &= \sigma(x_t U^i + s_{t-1} W^i) \\
f &= \sigma(x_t U^f + s_{t-1} W^f) \\
o &= \sigma(x_t U^o + s_{t-1} W^o) \\
g &= \tanh(x_t U^g + s_{t-1} W^g) \\
c_t &= c_{t-1} \circ f + g \circ i \\
s_t &= \tanh(c_t) \circ o
\end{aligned}
$$

($\circ$ is elementwise multiplication)

- RNN computes hidden state as $s_t = \tanh(U x_t + W s_{t-1})$ – an LSTM unit does exact same thing

Long Short Term Memory [Hochreiter and Schmidhuber, 1997]

LSTMs designed to combat vanishing gradients through gating mechanism

- How LSTM calculates hidden state s_t

$$
\begin{aligned}
i &= \sigma(x_t U^i + s_{t-1} W^i) \\
f &= \sigma(x_t U^f + s_{t-1} W^f) \\
o &= \sigma(x_t U^o + s_{t-1} W^o) \\
g &= \tanh(x_t U^g + s_{t-1} W^g) \\
c_t &= c_{t-1} \circ f + g \circ i \\
s_t &= \tanh(c_t) \circ o
\end{aligned}
$$

($\circ$ is elementwise multiplication)

- RNN computes hidden state as $s_t = \tanh(U x_t + W s_{t-1})$ – an LSTM unit does exact same thing

- i, f, o: input, forget and output gates
- Gates optionally let information through: composed out of sigmoid neural net layer and pointwise multiplication operation
- g is a candidate hidden state computed based on current input and previous hidden state
- c_t is internal memory of LSTM unit: combines previous memory c_{t-1} multiplied by forget gate, and newly computed hidden state g, multiplied by input gate

Long Short Term Memory [Hochreiter and Schmidhuber, 1997]

LSTMs designed to combat vanishing gradients through gating mechanism

- How LSTM calculates hidden state s_t

$$
\begin{aligned}
i &= \sigma(x_t U^i + s_{t-1} W^i) \\
f &= \sigma(x_t U^f + s_{t-1} W^f) \\
o &= \sigma(x_t U^o + s_{t-1} W^o) \\
g &= \tanh(x_t U^g + s_{t-1} W^g) \\
c_t &= c_{t-1} \circ f + g \circ i \\
s_t &= \tanh(c_t) \circ o
\end{aligned}
$$

($\circ$ is elementwise multiplication)

- RNN computes hidden state as $s_t = \tanh(U x_t + W s_{t-1})$ – an LSTM unit does exact same thing
- ...
- Compute output hidden state s_t by multiplying memory with output gate
- Plain RNNs a special case of LSTMs:
  - Fix input gate to all 1's
  - Fix forget gate to all 0's (always forget the previous memory)
  - Fix output gate to all 1's (expose the whole memory)
  - Additional $\tanh$ squashes output
- Gating mechanism allows LSTMs to model long-term dependencies
- Learn parameters for gates, to learn how memory should behave

Gated Recurrent Units

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

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

- 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

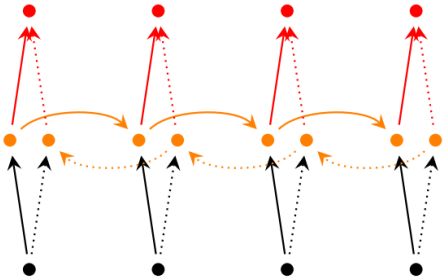


Image credits: `http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/`

Attention

- Attention mechanisms come from visual analysis — suppose you want to locate a specific object in a large image; like people, focus on specific areas
- First applied to text and NLP in 2015
- Basic mechanism behind every attention mechanism
  1. Read operator: read a "patch" from the input
  2. Glimpse sensor: extract information from "patch"
  3. Locator: predict the next location of read operator
  4. RNN: combine the previous and current responses from glimpse sensor

Assume we are at time step t. From $t-1$ we get next location to which we should pay attention (produced by locator). Move sensor there and extract information, which the RNN combines with previous outputs. After several iterations, we produce final response, e.g., classification or label.

Attention

Hard attention

- Read operator: fixed size, but there may be several of them
- Glimpse sensor can be any NN
- Locator predicts x and y location of sensor (images) or words ahead/previous (text)

Not differentiable, so Reinforcement Learning is used

Soft attention

- Read operator: it only has fixed aspect ratio; it can zoom into the image and blur if needed
- Glimpse sensor can be any NN
- Locator predict more than x and y parameter (like sigma for blur .etc)

Differentiable

Outline

Morning program
- Preliminaries
  - Feedforward neural network
  - Back propagation
  - Distributed representations
  - Recurrent neural networks
  - Sequence-to-sequence models
  - Convolutional neural networks
- Text matching I
- Text matching II

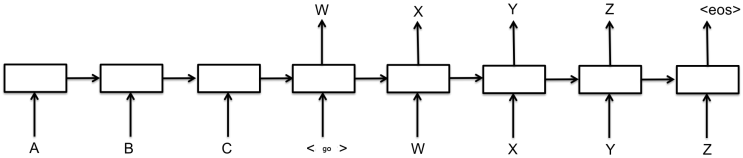
Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

Sequence-to-sequence models

Increasingly important: not just retrieval but also generation

- Snippets, summaries, small screen versions of search results, spoken results, chatbots, conversational interfaces, . . . , but also 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: `https://www.tensorflow.org/tutorials/seq2seq`

Sequence-to-sequence models

- seq2seq models build on top of language models
  - Encoder step: a model converts input sequence into a fixed representation
  - Decoder step: a language model is trained on both the output sequence (e.g., translated sentence) as well as fixed representation from the encoder
  - Since decoder model sees encoded representation of input sequence as well as output sequence, it can make more intelligent predictions about future words based on current word

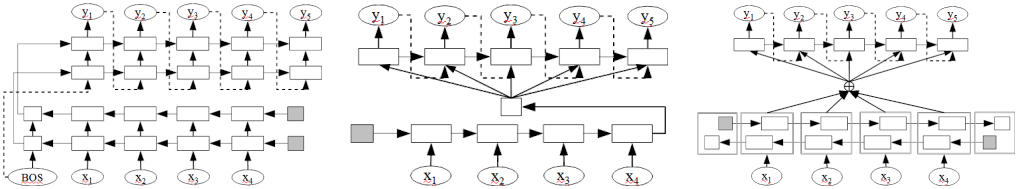


Image credits: [Sutskever et al., 2014; Cho et al., 2014; Bahdanou et al., 2014]

Sequence-to-sequence models

Used for a "traditional information retrieval task"

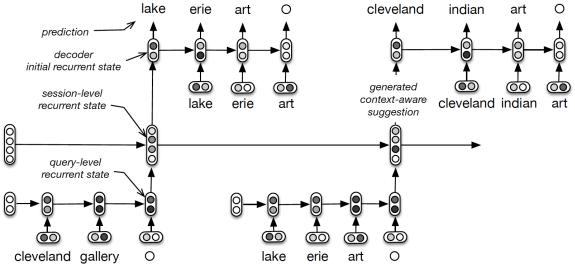


Figure 3: The hierarchical recurrent encoder-decoder (HRED) for query suggestion. Each arrow is a non-linear transformation. The user types *cleveland gallery* → *lake erie art*. During training, the model encodes *cleveland gallery*, updates the session-level recurrent state and maximize the probability of seeing the following query *lake erie art*. The process is repeated for all queries in the session. During testing, a contextual suggestion is generated by encoding the previous queries, by updating the session-level recurrent states accordingly and by sampling a new query from the last obtained session-level recurrent state. In the example, the generated contextual suggestion is *cleveland indian art*.

Image credits: Sordoni et al. [2015a]

Outline

Morning program
- Preliminaries
  - Feedforward neural network
  - Back propagation
  - Distributed representations
  - Recurrent neural networks
  - Sequence-to-sequence models
  - Convolutional neural networks
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

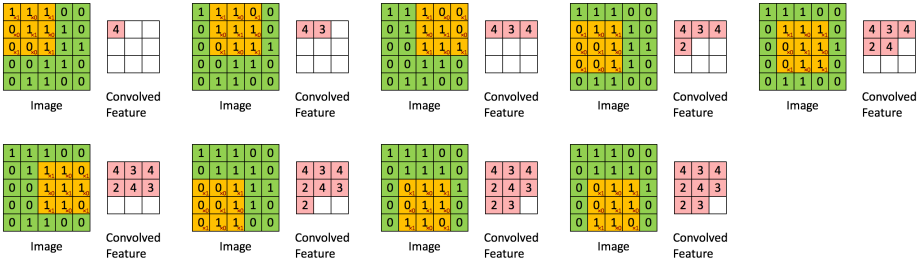
Convolutional neural networks

Major breakthroughs in image classification – at core of many computer visions systems

Some initial applications of CNNs to problems in text and information retrieval

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

Example: convolution with 3×3 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

Visual examples of CNNs

Averaging each pixel with neighboring values blurs image:

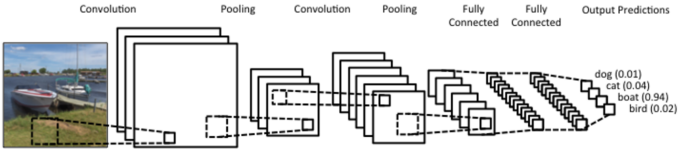
0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

Taking difference between pixel and its neighbors detects edges:

	0	1	0	
	1	-4	1	
	0	1	0	

Image credits: `https://docs.gimp.org/en/plug-in-convmatrix.html`

Convolutional neural networks



- Use convolutions over input layer to compute output
- Yields local connections: each region of input connected to a neuron in output
- Each layer applies different filters and combines results
- Pooling (subsampling) layers
- During training, CNN learns values of filters

- Image classification a CNN may learn to detect edges from raw pixels in first layer
- Then use edges to detect simple shapes in second layer
- Then use shapes to detect higher-level features, such as facial shapes in higher layers
- Last layer is then a classifier that uses high-level features

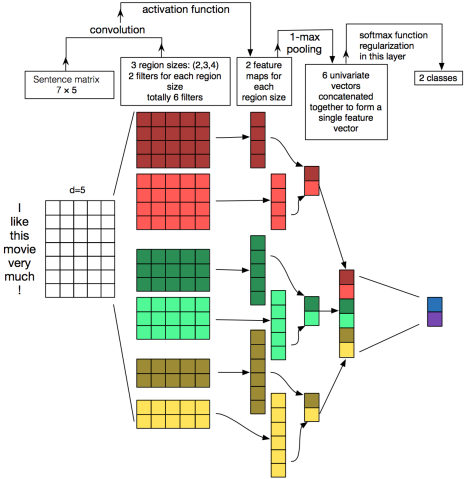
CNNs in text

Basic intution

- Instead of image pixels, input to most NLP tasks are sentences or documents represented as a matrix. Each row of matrix corresponds to one token, typically a word, but could be a character. That is, each row is vector that represents word.
- Typically, these vectors are word embeddings (low-dimensional representations) like word2vec or GloVe, but they could also be one-hot vectors that index the word into a vocabulary.
- For a 10 word sentence using a 100-dimensional embedding we would have a 10×100 matrix as our input.
- That's our "image"
- Typically use filters that slide over full rows of the matrix (words): the "width" of our filters is usually the same as the width of the input matrix. The height, or region size, may vary, but sliding windows over 2-5 words at a time is typical.

CNNs in text

Example architecture (Zhang and Wallace, 2015; Sentence classification)



CNNs in text

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
- Gao et al. Modeling Interestingness with Deep Neural Networks. EMNLP 2014; Shen et al. A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. CIKM 2014.

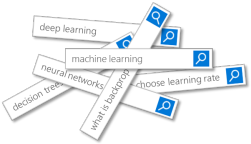
Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

Supervised text matching

Traditional IR data consists of search queries and document collection



search queries

document corpus

Ground truth can be based on explicit human judgments or implicit user behaviour data (e.g., clickthrough rate)

user interaction / click data

human annotated labels

Lexical vs. Semantic matching

Query: united states president

The President of the United States of America (POTUS) is the elected head of state and head of government of the United States. The president leads the executive branch of the federal government and is the commander in chief of the United States Armed Forces. Barack Hussein Obama II (born August 4, 1961) is an American politician who is the 44th and current President of the United States. He is the first African American to hold the office and the first president born outside the continental United States.

The President of the United States of America (POTUS) is the elected head of state and head of government of the United States. The president leads the executive branch of the federal government and is the commander in chief of the United States Armed Forces. Barack Hussein Obama II (born August 4, 1961) is an American politician who is the 44th and current President of the United States. He is the first African American to hold the office and the first president born outside the continental United States.

Traditional IR models estimate relevance based on lexical matches of query terms in document

Representation learning based models garner evidence of relevance from all document terms based on semantic matches with query

Both lexical and semantic matching are important and can be modelled with **neural networks**

Outline

Morning program
- Preliminaries
- Text matching I
  - Semantic matching
  - Lexical matching
  - Lexical and Semantic Duet
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

Semantic matching

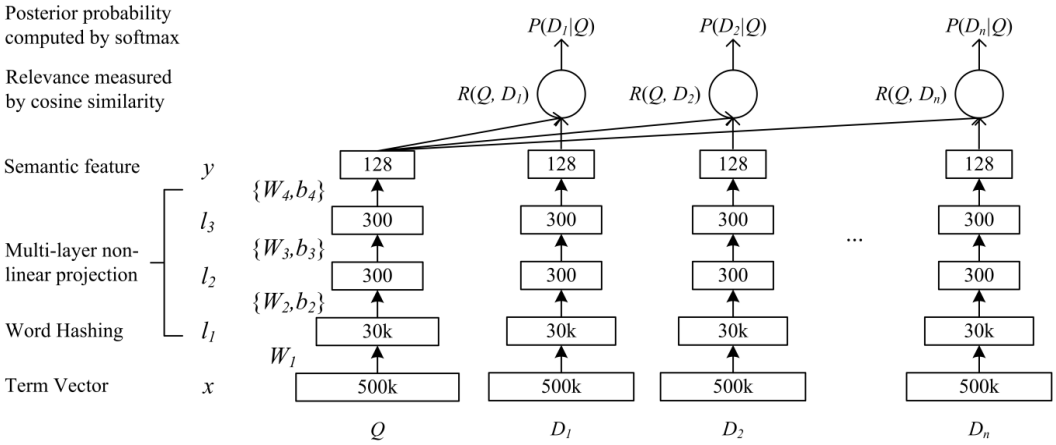
Pros

- Ability to match synonyms and related words
- Robustness to spelling variations
  ($\approx 10\%$ of search queries contain spelling errors)
- Helps in cases where lexical matching fails

Cons

- More computationally expensive than lexical matching

Deep Structured Semantic Model (DSSM) [Huang et al., 2013]

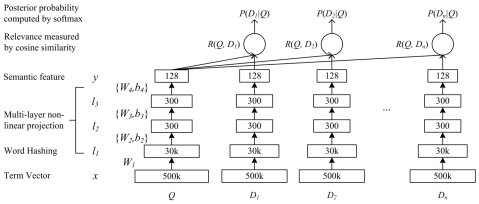


Deep Structured Semantic Model (DSSM) [Huang et al., 2013]

DSSM - Siamese Network

1. Represent query and document as vectors $\mathbf{q}$ and $\mathbf{d}$ in a latent vector space

2. Estimate the matching degree between $\mathbf{q}$ and $\mathbf{d}$ using cosine similarity



Deep Structured Semantic Model (DSSM) [Huang et al., 2013]

We learn to represent queries and documents in the latent vector space by forcing the vector representations (i) for relevant query-document pairs (q, d^+) to be close in the latent vector space (i.e., $\cos(\mathbf{q}, \mathbf{d}^+) \rightarrow \max$); and (ii) for irrelevant query-document pairs $(\mathbf{q}, \mathbf{d}^-)$ to be far in the latent vector space (i.e., $\cos(\mathbf{q}, \mathbf{d}^-) \rightarrow \min$)

DSSM - Word hashing

How to represent text (e.g., Shinjuku Gyoen)?

1. Bag of Words (BoW) [large vocabulary (500000 words)]

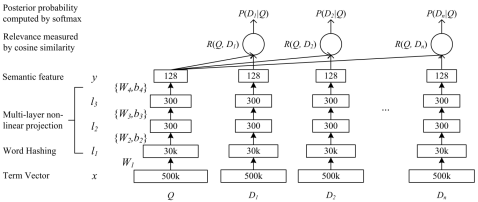
   { 0, . . . , 0 (apple), 0, . . . , 0, 1 (gyoen), 0, . . . , 0, 1 (shinjuku), 0, . . . , 0 }

2. Bag of Letter Trigrams (BoLT) [small vocabulary (30621 letter 3-grams)]

   { 0, . . . , 0 (abc), 0, . . . , 1 (_gy), 0, . . . , 0, 1 (_sh), 0, . . . , 0, 1 (en_), 0, . . . , 0, 1 (gyo), 0, . . . , 0, 1 (hin), 0, . . . , 0, 1 (inj), 0, . . . , 0, 1 (juk), 0, . . . , 0, 1 (ku_), 0, . . . , 0, 1 (oen), 0, . . . , 0, 1 (shi), 0, . . . , 0, 1 (uku), 0, . . . , 0, 1 (yoe), 0 }

DSSM - Architecture

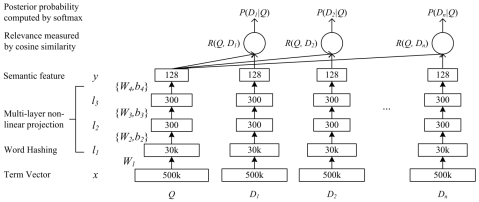
$$\mathbf{x} = \text{BoW}(\text{text})$$
$$\mathbf{l}_1 = \text{WordHashing}(\mathbf{x})$$
$$\mathbf{l}_2 = \tanh(W_2\mathbf{l}_1 + b_2)$$
$$\mathbf{l}_3 = \tanh(W_3\mathbf{l}_2 + b_3)$$
$$\mathbf{l}_4 = \tanh(W_4\mathbf{l}_3 + b_4)$$



DSSM - Training objective

Likelihood

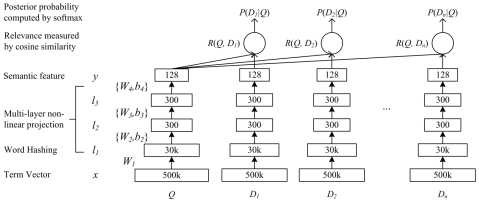
$$\prod_{(q,d^+)\in \text{DATA}} P(d^+ \mid q) \to \max$$



$$P(d^+ \mid q) = \frac{e^{\gamma \cos(\mathbf{q},\mathbf{d}^+)}}{\sum_{d\in D} e^{\gamma \cos(\mathbf{q},\mathbf{d})}} \approx \frac{e^{\gamma \cos(\mathbf{q},\mathbf{d}^+)}}{\sum_{d\in D^+\cup D^-} e^{\gamma \cos(\mathbf{q},\mathbf{d})}}$$

DSSM - Results

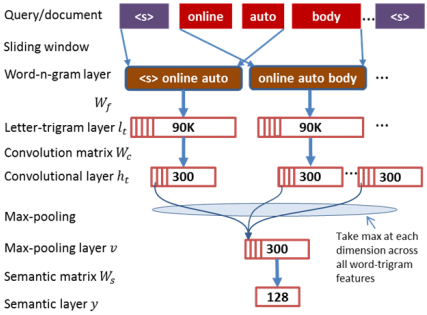
	NDCG		
Model	**@1**	**@3**	**@10**
TF-IDF	0.319	0.382	0.462
BM25	0.308	0.373	0.455
WTM	0.332	0.400	0.478
LSA	0.298	0.372	0.455
PLSA	0.295	0.371	0.456
DAE	0.310	0.377	0.459
BLTM	0.337	0.403	0.480
DPM	0.329	0.401	0.479
DSSM	0.362	0.425	0.498



CLSM

1. Embeds N-grams similar to DSSM
2. Aggregates phrase embeddings by max-pooling

	NDCG		
Model	**@1**	**@3**	**@10**
BM25	0.305	0.328	0.388
DSSM	0.320	0.355	0.431
CLSM	0.342	0.374	0.447

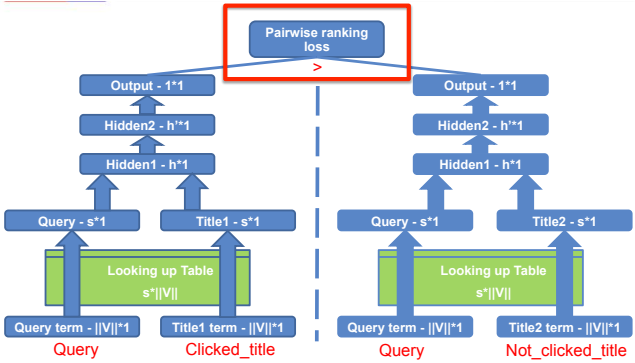


A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval [Shen et al., 2014].

In industry

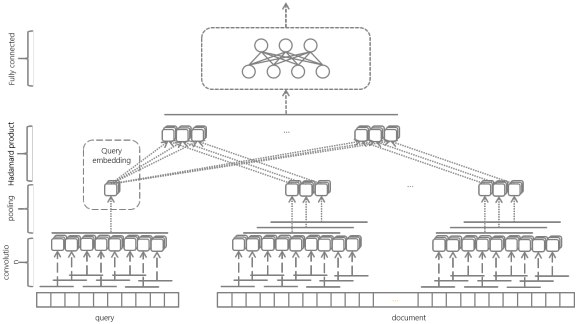
Baidu's DNN model

- Around 30% of total 2013, 2014 relevance improvement
- Use 10B clicks for training (more than 100M parameters)

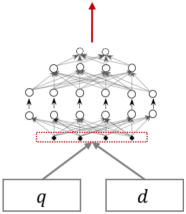


Semantic matching for long text

Semantic matching can also be applied to long text retrieval but requires large scale training data to learn meaningful representations of text



Mitra et al. [2017] train on large manually labelled data from Bing



Dehghani et al. [2017] train on pseudo labels (e.g., BM25)

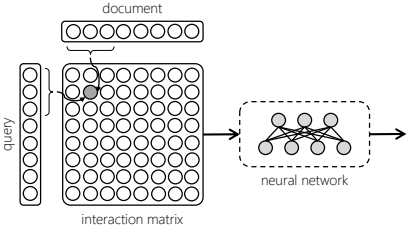
Interaction matrix based approaches

Alternative to Siamese networks

Interaction matrix X, where $x_{i,j}$ is obtained by comparing the the ith word in source sentence with j^{th} word in target sentence

Comparisons can be both lexical or semantic

E.g., Hu et al. [2014], Mitra et al. [2017], Pang et al. [2016]



Outline

Morning program
- Preliminaries
- Text matching I
  - Semantic matching
  - Lexical matching
  - Lexical and Semantic Duet
- Text matching II

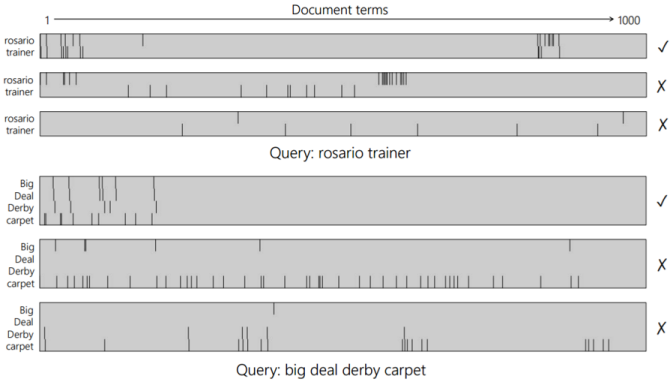
Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

Lexical matching

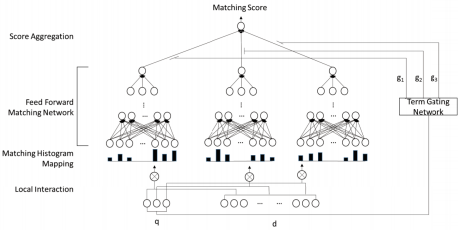
Query: "rosario trainer"

The rare term "rosario" may have never been seen during training and unlikely to have meaningful representation

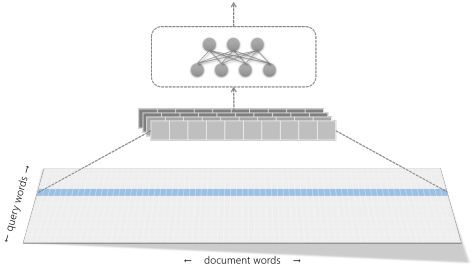
But the patterns of lexical matches of rare terms in document may be very informative for estimating relevance



Lexical matching



Guo et al. [2016] train a DNN model using features derived from frequency histograms of query term matches in document



Mitra et al. [2017] convolve over the binary interaction matrix to learn interesting patterns of lexical term matches

Outline

Morning program
- Preliminaries
- Text matching I
  - Semantic matching
  - Lexical matching
  - Lexical and Semantic Duet
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

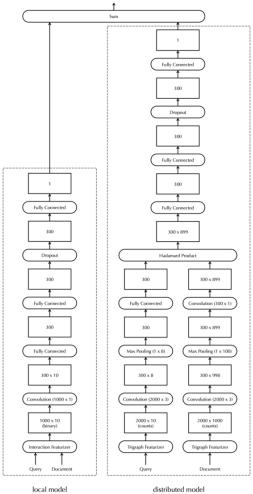
Duet

Jointly train two sub-networks focused on lexical and semantic matching [Mitra et al., 2017, Nanni et al., 2017]

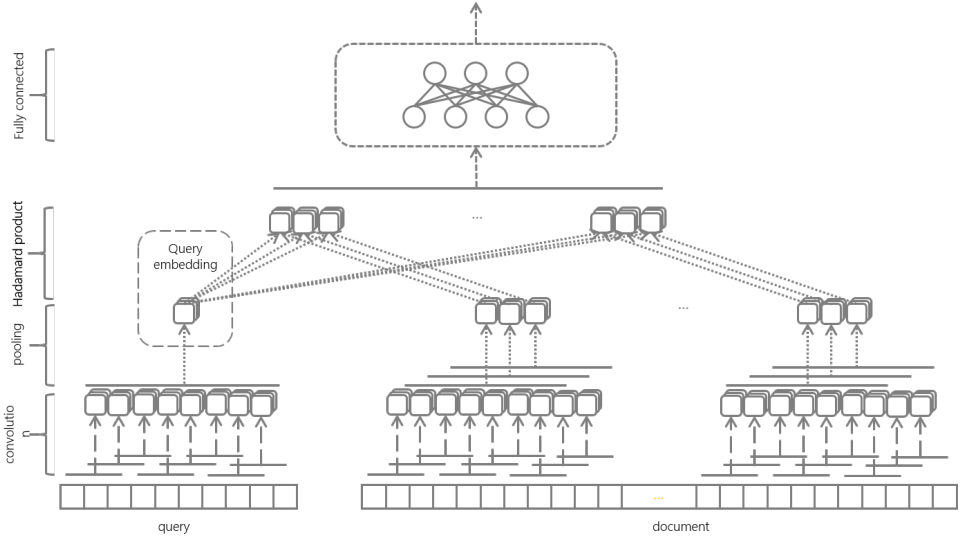
Training sample: q, d^+, d_1, d_2, d_3, d_4

$$p(d^+|q) = \frac{e^{\mathsf{ndrm}(q,d^+)}}{\sum_{d \in D^-} e^{\mathsf{ndrm}(q,d)}} \tag{1}$$

Implementation on GitHub: https://github.com/bmitra-msft/NDRM/blob/master/notebooks/Duet.ipynb



Distributed model



Results

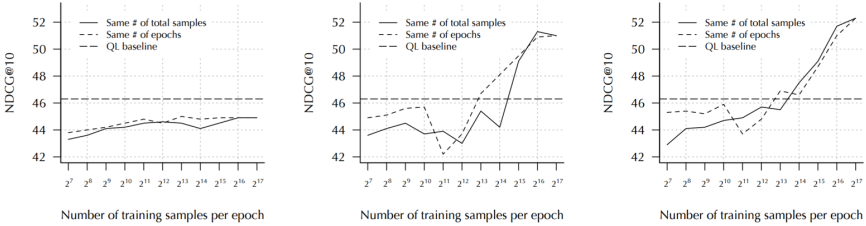
(a) weighted

	NDCG@1	NDCG@10
Non-neural baselines		
LSA	22.4	44.2
BM25	24.2	45.5
DM	24.7	46.2
QL	24.6	46.3
Neural baselines		
DRMM	24.3	45.2
DSSM	25.8	48.2
CDSSM	*27.3*	48.2
DESM	25.4	*48.3*
Our models		
Local model	24.6	45.1
Distributed model	28.6	50.5
Duet model	**32.2**	**53.0**

(b) unweighted

	NDCG@1	NDCG@10
Non-neural baselines		
LSA	31.9	62.7
BM25	34.9	63.3
DM	35.0	63.4
QL	34.9	63.4
Neural baselines		
DRMM	*35.6*	*65.1*
DSSM	34.3	64.4
CDSSM	34.3	64.0
DESM	35.0	64.7
Our models		
Local model	35.0	64.4
Distributed model	35.2	64.9
Duet model	**37.8**	**66.4**

Duet



(a) Local model (b) Distributed model (c) Duet model

The biggest impact of training data size is on the performance of the representation learning sub-model

Important: if you want to learn effective representations for semantic matching you need large scale training data!

Duet

Term importance

Local model

Only query terms have an impact

Earlier occurrences have bigger impact

Query: united states president

The President of the United States of America (POTUS) is the elected head of state and head of government of the United States. The president leads the executive branch of the federal government and is the commander in chief of the United States Armed Forces. Barack Hussein Obama II (born August 4, 1961) is an American politician who is the 44th and current President of the United States. He is the first African American to hold the office and the first president born outside the continental United States.

Visualizing impact of dropping terms on model score

Duet

Term importance

Distributed model

Non-query terms (e.g., *Obama* and *federal*) has positive impact on score

Common words like 'the' and 'of' probably good indicators of well-formedness of content

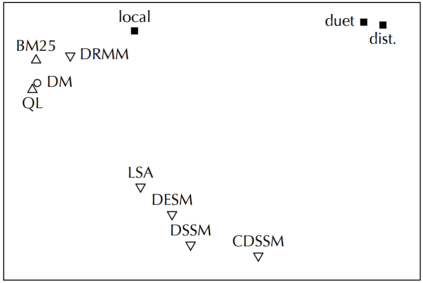
Query: united states president

The President of the United States of America (POTUS) is the elected head of state and head of government of the United States. The president leads the executive branch of the federal government and is the commander in chief of the United States Armed Forces. Barack Hussein Obama II (born August 4, 1961) is an American politician who is the 44th and current President of the United States. He is the first African American to hold the office and the first president born outside the continental United States.

Visualizing impact of dropping terms on model score

Duet

If we classify models by query level performance there is a clear clustering of lexical and semantic matching models



Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II
  - Unsupervised semantic matching with pre-training
  - Semi-supervised semantic matching
  - Obtaining pseudo relevance
  - Training neural networks using pseudo relevance
  - Learning unsupervised representations from scratch
  - Toolkits

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

Unsupervised semantic matching with pre-training

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.

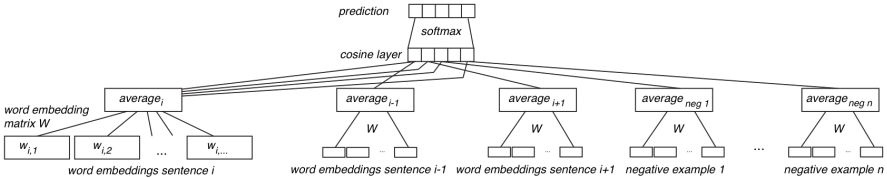
- The most common choice of context: The set of words that co-occur in a context window.
- Context-counting VS. Context-predicting [Baroni et al., 2014]

From Word Embedding to Query/Document Embedding

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

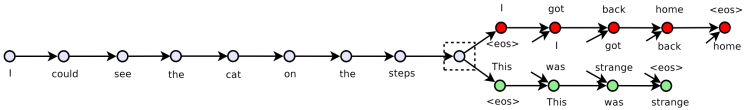
Bag of embedded words: sum or average of word vectors.

- 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].



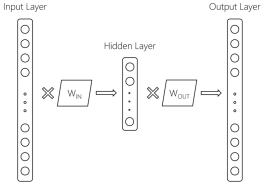
From Word Embedding to Query/Document Embedding

- Skip-Thought Vectors
  - Conceptually similar to distributional semantics: a units 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)

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

yale			**seahawks**			**eminem**		
IN-IN	OUT-OUT	IN-OUT	IN-IN	OUT-OUT	IN-OUT	IN-IN	OUT-OUT	IN-OUT
yale	yale	yale	seahawks	seahawks	seahawks	eminem	eminem	eminem
harvard	uconn	faculty	49ers	broncos	highlights	rihanna	rihanna	rap
nyu	harvard	alumni	broncos	49ers	jerseys	ludacris	dre	featuring
cornell	tulane	orientation	packers	nfl	tshirts	kanye	kanye	tracklist
tulane	nyu	haven	nfl	packers	seattle	beyonce	beyonce	diss
tufts	tufts	graduate	steelers	steelers	hats	2pac	tupac	performs

- 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).

Pre-trained word embedding 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 embedding for document retrieval and ranking

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

$$p(d|q) \sim p(q|d)p(d)$$

$$p(q|d) = \prod_{t_q \in q} p(t_q|d)$$

$$p(t_q|d) = \sum_{t_d \in d} p(t_q|t_d)p(t_d|d)$$

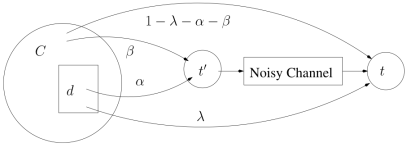
  - Uses the similarity between term embeddings as a measure for term-term translation probability $p(t_q|t_d)$.

$$p(t_q|t_d) = \frac{\cos(\vec{v}_{t_q}, \vec{v}_{t_d})}{\sum_{t \in V} \cos(\vec{v}_t, \vec{v}_{t_d})}$$

Pre-trained word embedding for document retrieval and ranking

GLM [Ganguly et al., 2015]: Generalize 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'.



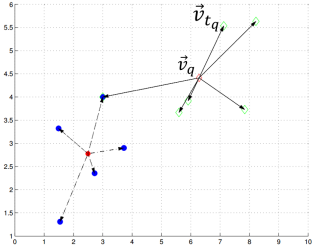
$$p(t_q|d) = \lambda p(t_q|d) + \alpha \sum_{t_d \in d} p(t_q, t_d|d) p(t_d) + \beta \sum_{t' \in N_t} p(t_q, t'|C) p(t') + 1 - \lambda - \alpha - \beta) p(t_q|C)$$

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

$$p(t', t|d) = \frac{\mathsf{sim}(\vec{v}_{t'}, \vec{v}_t).\mathsf{tf}(t', d)}{\sum_{t_1 \in d} \sum_{t_2 \in d} \mathsf{sim}(\vec{v}_{t_1}, \vec{v}_{t_2}).|d|}$$

Pre-trained word embedding 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 embedding 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

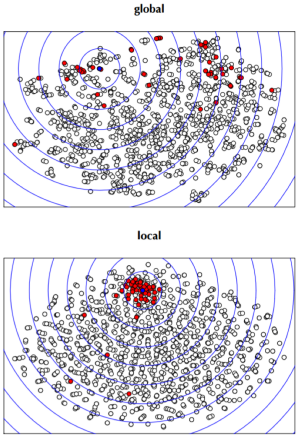
- Embedding-based Query Expansion [Zamani and Croft, 2016a]
  Main goal: Estimating a better language model for the query using embeddings.
  - Two models with different assumptions:
    - Conditional independence of query terms.
    - Query-independent term similarities.
    Different calculation of the probability of expansion terms given the query.
  - Choosing top-k probable terms as expansion terms.
- Embedding-based Relevance Model:
  Main goal: Semantic similarity in addition to term matching for PRF.

$$
\begin{aligned}
P(t|\theta_F) &= \sum_{d \in F} p(t, q, d) \\
&= \sum_{t \in V} p(q|t, d) p(t|d) p(d)
\end{aligned}
$$

$$
p(q|t, d) = \beta p_{\text{tm}}(q|t, d) + (1 - \beta) p_{\text{sm}}(q|t, d)
$$

Pre-trained word embedding for query expanssion

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
- Text matching I
- Text matching II
  - Unsupervised semantic matching with pre-training
  - Semi-supervised semantic matching
  - Obtaining pseudo relevance
  - Training neural networks using pseudo relevance
  - Learning unsupervised representations from scratch
  - Toolkits

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

Semi-supervised semantic matching

Using unsupervised pre-trained word embeddings, we have vector space of words that we have to put to good use to create query and document representations.

However, in information retrieval, there is the concept of pseudo relevance that gives us a supervised signal that was obtained from unsupervised data collections.

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II
  - Unsupervised semantic matching with pre-training
  - Semi-supervised semantic matching
  - Obtaining pseudo relevance
  - Training neural networks using pseudo relevance
  - Learning unsupervised representations from scratch
  - Toolkits

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

Pseudo test/training collections

Given a source of pseudo relevance, we can build pseudo training or test collections. We can

- use the pseudo training collections to train a model and then test on a non-pseudo test collection, or
- use the pseudo test collections to verify models in a domain where human judgments are lacking or incomplete.

History of pseudo test collections

Problems in the simulation of bibliographic retrieval systems [Tague et al., 1980]

"If tests are carried out with large operational systems, there are difficulties in experimentally controlling and modifying the variables [of bibliographic retrieval systems]. [...] An alternative approach [...] is computer simulation."

Use simulation to investigate the **complexity** (data structures) and **effectiveness** (query/document representation) of retrieval systems.

How to determine query/document relevance?

Synthesize a separate set of relevant documents for a query [Tague et al., 1980] or sample judgments for every query and all documents from a probabilistic model [Tague and Nelson, 1981].

Modern pseudo test collections for evaluating effectiveness (1/2)

Research focused on validating pseudo relevance with non-pseudo judgments.

Web search [Beitzel et al., 2003]

Find sets of pseudo-relevant documents using the Open Directory Project. Queries are editor-entered document titles (document with exact title is relevant) and category names (leaf-level documents are relevant).

Known-item search [Azzopardi et al., 2007]

Compare manual query/judgments with pseudo query/judgments using a Kolmogorov-Smirnov (KG) test on multi-lingual documents from government websites.

Desktop search [Kim and Croft, 2009]

Building upon Azzopardi et al. [2007], construct a pseudo test collection for enterprise search and verify its validity using a KG test.

Modern pseudo test collections for evaluating effectiveness (2/2)

Archive search [Huurnink et al., 2010a,b]

Generate queries and judgments using the strategy of Azzopardi et al. [2007] and validate using transactions logs of an audiovisual archive.

Product search [Van Gysel et al., 2016]

Construct queries from product category hierarchies and estimate product relevance by category membership [Beitzel et al., 2003], grounded in observations from e-commerce research. Pseudo test collections are then used to evaluate unsupervised neural representation learning algorithms (see Slide 104).

From testing to training using pseudo relevance

At some point, pseudo relevance began to be used to train retrieval functions. Learning To Rank models (see later) trained using pseudo relevance outperform non-supervised retrieval functions (e.g., BM25) on TREC collections.

Web search using anchor texts [Asadi et al., 2011]

Construct a pseudo relevance collection from anchor texts in a web corpus and use it to train Learning To Rank (LTR) models. LTR models trained using pseudo relevance outperform BM25 on TREC collections.

Microblog search using hashtags [Berendsen et al., 2013]

Tweets with a hashtag are relevant to the topic covered by the hashtag. Queries are constructed by sampling terms from tweets that discriminate the relevant set from the non-relevant set.

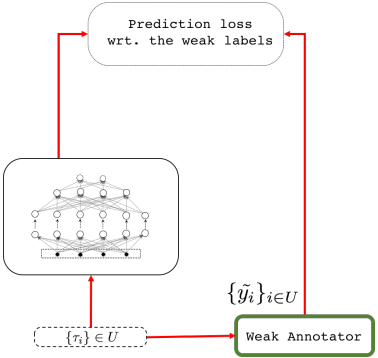
Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II
  - Unsupervised semantic matching with pre-training
  - Semi-supervised semantic matching
  - Obtaining pseudo relevance
  - Training neural networks using pseudo relevance
  - Learning unsupervised representations from scratch
  - Toolkits

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

Training neural networks using pseudo relevance

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

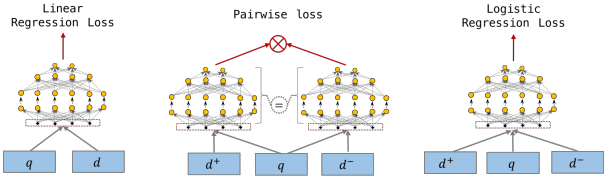


Main idea: Annotating a large amount of unlabeled data using a weak annotator (Pseudo-Labeling) and design a model which can be trained on weak supervision signal.

- Function approximation. (re-inventing BM25?)
- Beating BM25 using BM25!

Training neural networks using pseudo relevance

- Employed three different architectures: Score, Rank, and RankProb



- Employed Three different feeding paradigms:
  - Dense: $\psi(q,d) = [N||avg(l_d)_D||l_d||\{df(t_i)||tf(t_i,d)\}_{1\leq i\leq k}]$
  - Sparse: $\psi(q,d) = [tfv_c||tfv_q||tfv_d]$
  - Embedding-based: $\psi(q,d) = [\odot_{i=1}^{|q|}(\mathcal{E}(t_i^q),\mathcal{W}(t_i^q))|| \odot_{i=1}^{|d|} (\mathcal{E}(t_i^d),\mathcal{W}(t_i^d))],$

Training neural networks using pseudo relevance

Lesson Learned:

- Define an objective which enables your model to go beyond the imperfection of the weakly annotated data. (ranking instead of calibrated scoring)
- Let the network decide about the representation. Feeding the network with featurized input kills the model creativity!
- If you get enough data, you can learn embedding which is better fitted to your task by updating them just based on the objective of the downstream task.
- You can compensate the lack of enough training data by pertaining your network on weakly annotated data.

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
- Text matching I
- Text matching II
  - Unsupervised semantic matching with pre-training
  - Semi-supervised semantic matching
  - Obtaining pseudo relevance
  - Training neural networks using pseudo relevance
  - Learning unsupervised representations from scratch
  - Toolkits

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

Learning unsupervised representations from scratch

- Pseudo relevance judgments allow the training of supervised models in the absence of human judgments or implicit relevance signals (e.g., clicks).
- Introduces a dependence on relevance models, hypertext, query lists, ...
- Unsupervised retrieval models (e.g., BM25, language models) operate without pseudo relevance.

Can we learn a model of relevance in the absence of any relevance judgments?

LSI, pLSI and LDA

History of latent document representations

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

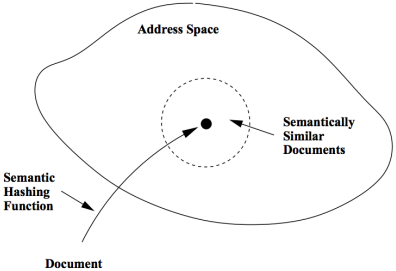
- Latent Semantic Indexing [Deerwester et al., 1990],
- Probabilistic Latent Semantic Indexing [Hofmann, 1999], and
- Latent Dirichlet Allocation [Blei et al., 2003].

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

Semantic Hashing

Salakhutdinov and Hinton [2009] propose Semantic Hashing for document similarity.

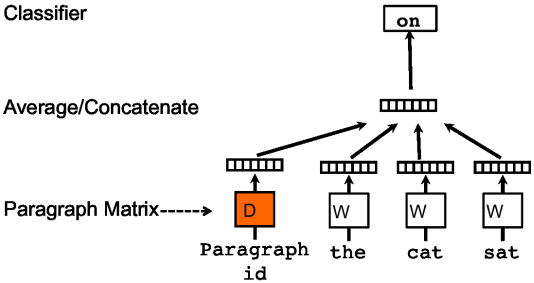
- Auto-encoder trained on frequency vectors.
- Documents are mapped to memory addresses in such a way that semantically similar documents are located at nearby bit addresses.
- Documents similar to a query document can then be found by accessing addresses that differ by only a few bits from the query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



Schematic representation of Semantic Hashing. Taken from Salakhutdinov and Hinton [2009].

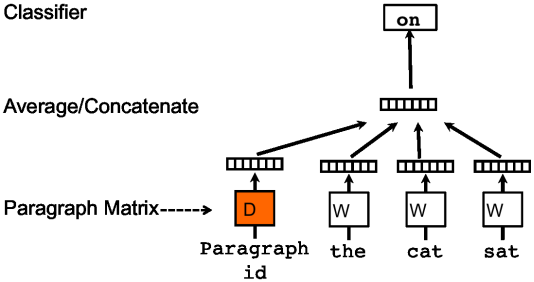
Distributed Representations of Documents [Le and Mikolov, 2014]

- 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].

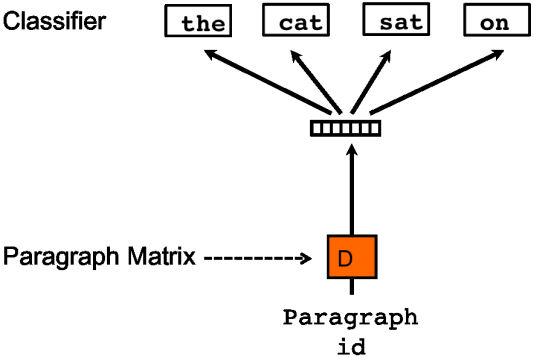


Overview of the Distributed Memory document vector model. Taken from Le and Mikolov [2014].

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].

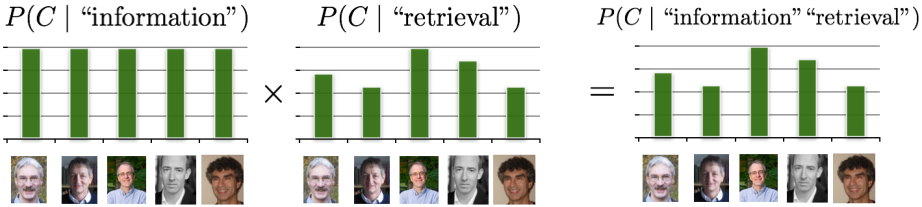
Semantic Expertise Retrieval [Van Gysel et al., 2016]

- 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 predicts the correct expert.

Taken from slides of Van Gysel et al. [2016].

Semantic Expertise Retrieval [Van Gysel et al., 2016] (cont'd)

- 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 predicts the correct expert.



Taken from slides of Van Gysel et al. [2016].

Regularities in Text-based Entity Vector Spaces [Van Gysel et al., 2017c]

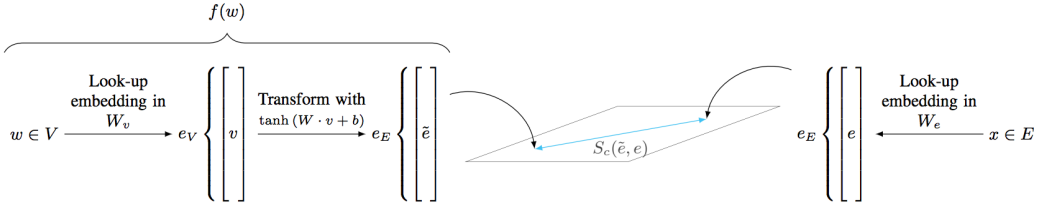
To what extent do entity representation models, trained only on text, encode structural regularities of the entity's domain?

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]

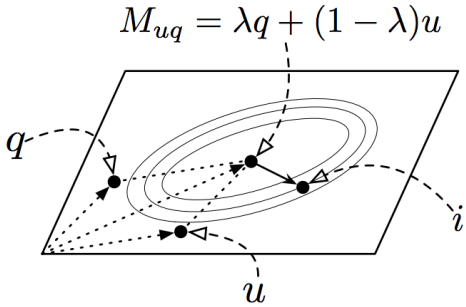
- Learn representations of e-commerce products and query terms for product search.
- Tackles learning objective scalability limitations from previous work.
- Useful as a semantic feature within a Learning To Rank model in addition to a lexical matching signal.



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].

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II
  - Unsupervised semantic matching with pre-training
  - Semi-supervised semantic matching
  - Obtaining pseudo relevance
  - Training neural networks using pseudo relevance
  - Learning unsupervised representations from scratch
  - Toolkits

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

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., 2017b]
- cuNVSM : `http://www.github.com/cvangysel/cuNVSM` [Van Gysel et al., 2017a]
- HEM : `https://ciir.cs.umass.edu/downloads/HEM` [Ai et al., 2017]

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
  - Overview & basics
  - Refresher of cross-entropy
  - Pointwise loss
  - Pairwise loss
  - Listwise loss
  - Different levels of supervision
  - Toolkits
- Modeling user behavior
- Generating responses
- Wrap up

Learning to rank (L2R)

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]

L2R 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 discuss supervised (offline) L2R models first, but briefly introduce online L2R later.

Approaches

Liu [2009] categorizes different L2R 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 L2R models employ hand-crafted features that encode IR insights

They can often be categorized as:

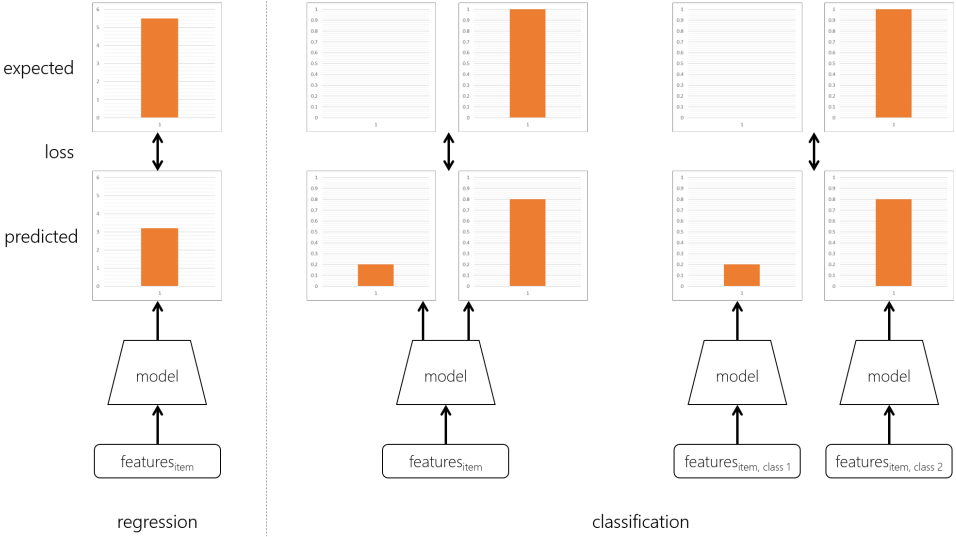
- Query-independent or static features (e.g., incoming link count and document length)
- Query-dependent or dynamic features (e.g., BM25)
- Query-level features (e.g., query length)

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
  - Overview & basics
  - Refresher of cross-entropy
  - Pointwise loss
  - Pairwise loss
  - Listwise loss
  - Different levels of supervision
  - Toolkits
- Modeling user behavior
- Generating responses
- Wrap up

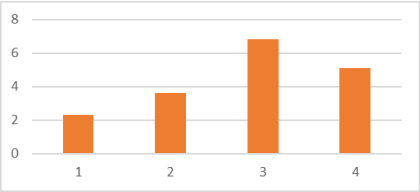
A quick refresher - Neural models for different tasks

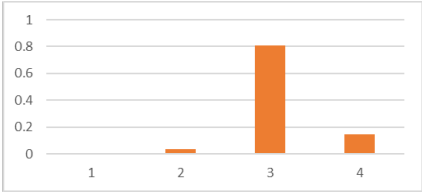


A quick refresher - What is the Softmax function?

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

$$p(z_i) = \frac{e^{\gamma z_i}}{\sum_{z \in Z} e^{\gamma z}} \qquad (\gamma \text{ is a constant}) \tag{2}$$

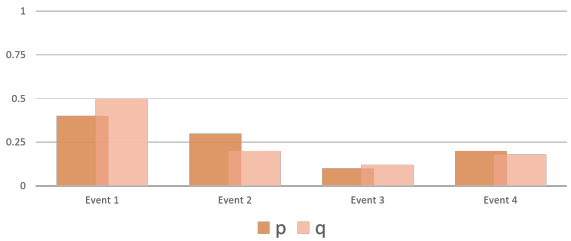




A quick refresher - What is Cross Entropy?

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

$$CE(p, q) = -\sum_i p_i \log(q_i) \tag{3}$$



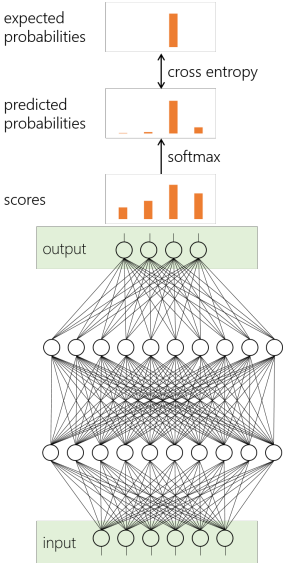
If $p_{correct} = 1$ and $p_i = 0$ for all other values of i then,

$$CE(p, q) = -\log(q_{correct}) \tag{4}$$

A quick refresher - What is the Cross Entropy with Softmax loss?

Cross entropy with softmax is a popular loss function for classification

$$\mathcal{L}_{\text{CE}} = -log\left(\frac{e^{\gamma z_{correct}}}{\sum_{z \in Z} e^{\gamma z}}\right) \tag{5}$$



Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
  - Overview & basics
  - Refresher of cross-entropy
  - Pointwise loss
  - Pairwise loss
  - Listwise loss
  - Different levels of supervision
  - Toolkits
- Modeling user behavior
- Generating responses
- Wrap up

Pointwise objectives

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,

$$\mathcal{L}_{Squared} = \|y_{q,d} - f(\vec{x}_{q,d})\|^2 \tag{6}$$

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],

$$\mathcal{L}_{\text{CE}}(q, d, y_{q,d}) = -log\Big(p(y_{q,d}|q, d)\Big) = -log\Big(\frac{e^{\gamma \cdot s_{y_{q,d}}}}{\sum_{y \in Y} e^{\gamma \cdot s_y}}\Big) \tag{7}$$

where, $s_{y_{q,d}}$ is the model's score for label $y_{q,d}$

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
  - Overview & basics
  - Refresher of cross-entropy
  - Pointwise loss
  - Pairwise loss
  - Listwise loss
  - Different levels of supervision
  - Toolkits
- Modeling user behavior
- Generating responses
- Wrap up

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 followingform [Chen et al., 2009],

$$\mathcal{L}_{pairwise} = \phi(s_i - s_j) \tag{8}$$

where, ϕ 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—popular choice for training neural L2R models and also 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,

$$\mathcal{L}_{RankNet} = -\bar{p}_{ij} \log(p_{ij}) - \bar{p}_{ji} \log(p_{ji}) \tag{9}$$

$$= -\log(p_{ij}) \tag{10}$$

$$= log(1 + e^{-\gamma(s_i - s_j)}) \tag{11}$$

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,

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

The cross entropy loss is then given by,

$$\mathcal{L}_{\mathsf{CE}}(q, d^+, D) = -log\Big(p(d^+|q)\Big) \tag{13}$$

$$= -log\Big(\frac{e^{\gamma \cdot s\left(q, d^+\right)}}{\sum_{d \in D} e^{\gamma \cdot s(q,d)}}\Big) \tag{14}$$

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—L2R models typically consider few negative candidates [Huang et al., 2013, Mitra et al., 2017, Shen et al., 2014]
- Large body of work in NLP to deal with similar issue that may be relevant to future L2R models
  - E.g., hierarchical softmax [Goodman, 2001, Mnih and Hinton, 2009, Morin and Bengio, 2005], 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], and BlackOut [Ji et al., 2015]

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
  - Overview & basics
  - Refresher of cross-entropy
  - Pointwise loss
  - Pairwise loss
  - Listwise loss
  - Different levels of supervision
  - Toolkits
- Modeling user behavior
- Generating responses
- Wrap up

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 higer 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 dont 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

$$\lambda_{LambdaRank} = \lambda_{RankNet} \cdot |\Delta NDCG| \tag{15}$$

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:

$$p(\pi|s) = \frac{\phi(s_2)}{\phi(s_1) + \phi(s_2) + \phi(s_3) + \phi(s_4)} \cdot \frac{\phi(s_1)}{\phi(s_1) + \phi(s_3) + \phi(s_4)} \cdot \frac{\phi(s_4)}{\phi(s_3) + \phi(s_4)} \tag{16}$$

where, π is a particular permutation and ϕ 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
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
  - Overview & basics
  - Refresher of cross-entropy
  - Pointwise loss
  - Pairwise loss
  - Listwise loss
  - Different levels of supervision
  - Toolkits
- Modeling user behavior
- Generating responses
- Wrap up

Training under different levels of supervision

Data requirements for training an off-line L2R system

Query/document pairs that encode an ideal ranking given a particular query.

Ideal ranking? Relevance, preference, importance [Liu, 2009], novelty & diversity [Clarke et al., 2008].

What about personalization? Triples of user, query and document.

Related to evaluation. Pairs also used to compute popular off-line evaluation measures.

Graded or binary. "documents may be relevant to a different degree" [Järvelin and Kekäläinen, 2000]

Absolute or relative? Zheng et al. [2007]

How to satisfy data-hungry models?

There are different ways to obtain query/document pairs.

Most expensive

1. Human judgments
2. Explicit user feedback
3. Implicit user feedback
4. Pseudo relevance

Least expensive

Human judgments

Human judges determine the relevance of a document for a given query.

How to determine candidate query/document pairs?

- Obtaining human judgments is expensive.
- List of queries: sample of incoming traffic or manually curated.
- Use an existing rankers to obtain rankings and pool the outputs [Sparck Jones and van Rijsbergen, 1976].
- Trade-off between number of queries (shallow) and judgments (deep) [Yilmaz and Robertson, 2009].

Explicit user feedback

When presenting results to the user, ask the user to explicitly judge the documents.

Unfortunately, users are only rarely willing to give explicit feedback [Joachims et al., 1997].

Extracting pairs from click-through data (training)

Extract implicit judgments from search engine interactions by users.

- Assumption: user clicks $\Rightarrow$ relevance (or, preference).
- Virtually unlimited data at very low cost, but interpretation is more difficult.
- Presentation bias: users are more likely to click higher-ranked links.
- How to deal with presentation bias? Joachims [2003] suggest to interleave different rankers and record preference.
- Chains of queries (i.e., search sessions) can be identified within logs and more fine-grained user preference can be extracted [Radlinski and Joachims, 2005].

Extracting pairs from click-through data (evaluation)

Clicks can also be used to evaluate different rankers.

- Radlinski et al. [2008] discuss how absolute metrics (e.g., abandonment rate) do not reliable reflect retrieval quality. However, relative metrics gathered using interleaving methods, do reflect retrieval quality.
- Carterette and Jones [2008] propose a method to predict relevance score of unjudged documents. Allows for comparisons across time and datasets.

Side-track: Online LTR

As mentioned earlier, we focus mostly on offline LTR. Besides an active learning set-up, where models are re-trained frequently, neural models have not yet conquered the online paradigm.

See the SIGIR'16 tutorial of Grotov and de Rijke [2016] for an overview.

Pseudo relevance judgments

Pseudo relevance collections (discussed first on Slide 96) can also be used to train LTR systems.

Web search Asadi et al. [2011] construct a pseudo relevance collection from anchor texts in a web corpus. LTR trained using pseudo relevance outperform non-supervised retrieval functions (e.g., BM25) on TREC collections.

Microblog search Berendsen et al. [2013] use hashtags as a topical relevance signal. Queries are constructed by sampling terms from tweets.

Personalized product search Ai et al. [2017] synthesize purchase behavior from Amazon user reviews. Queries and relevance are constructed according to the human-curated Amazon product categories [Van Gysel et al., 2016]. They learn vector space representations for query terms, users and products.

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
  - Overview & basics
  - Refresher of cross-entropy
  - Pointwise loss
  - Pairwise loss
  - Listwise loss
  - Different levels of supervision
  - Toolkits
- Modeling user behavior
- Generating responses
- Wrap up

Toolkits for off-line learning to rank

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

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

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)

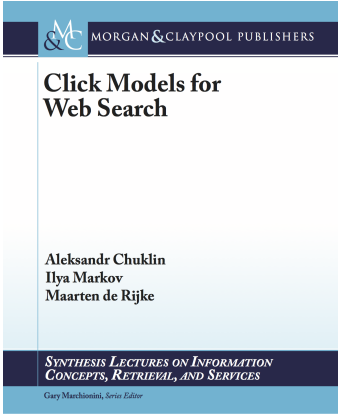
CLICK MODELS account for biases in clicks

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

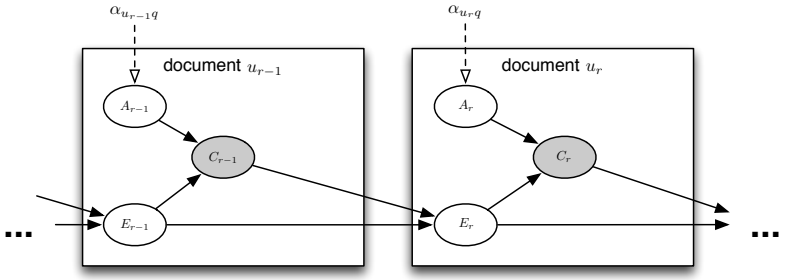


Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
  - Click behavior in web search
  - Time aspects of user behavior in web search
  - Web search vs. sponsored search
  - Future work
- Generating responses
- Wrap up

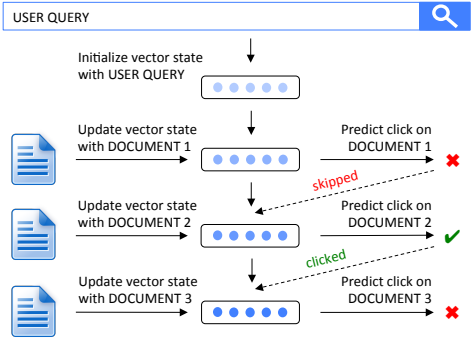
Traditional click models



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

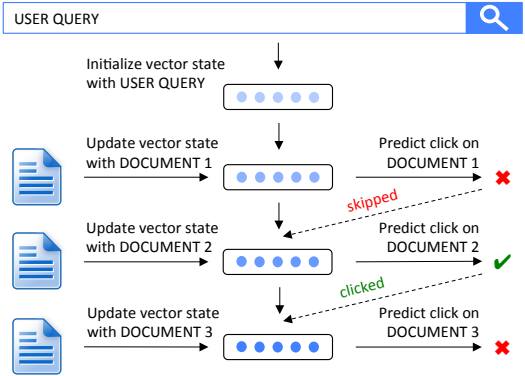
Neural click modeling framework



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

Learns patterns of user behavior directly from click-through data

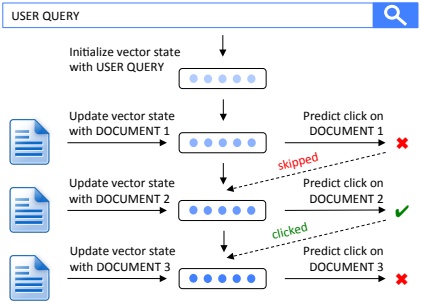
Distributed representations $(\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2, \ldots)$



We model user browsing behavior as a sequence of vector states $(\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2, \ldots)$ that describes the information consumed by the user as it evolves during a query session.

Mappings I, U and Function F

$$\mathbf{s}_0 = \mathcal{I}(q)$$
$$\mathbf{s}_{r+1} = \mathcal{U}(\mathbf{s}_r, i_r, d_{r+1})$$



$$P(C_{r+1} = 1 \mid q, i_1, \ldots, i_r, d_1, \ldots, d_{r+1}) = \mathcal{F}(\mathbf{s}_{r+1})$$

q — user query

d_r — document at rank r

i_r — user interaction with document at rank r

Neural click modeling framework → 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 $\mathcal{I}$, $\mathcal{U}$ and $\mathcal{F}$

$\mathcal{I}$ Feed-forward neural network
$\mathcal{U}$ Recurrent neural network (RNN, LSTM)
$\mathcal{F}$ 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 dataset[1]
($146,278,823$ 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

Click model	Perplexity	Log-likelihood
DBN	1.3510	-0.2824
DCM	1.3627	-0.3613
CCM	1.3692	-0.3560
UBM	1.3431	-0.2646
$\text{NCM}_{\text{QD}}^{\text{RNN}}$	1.3379	-0.2564
$\text{NCM}_{\text{QD}}^{\text{LSTM}}$	1.3362	-0.2547
$\text{NCM}_{\text{QD+Q}}^{\text{LSTM}}$	1.3355	-0.2545
$\text{NCM}_{\text{QD+Q+D}}^{\text{LSTM}}$	1.3318	-0.2526

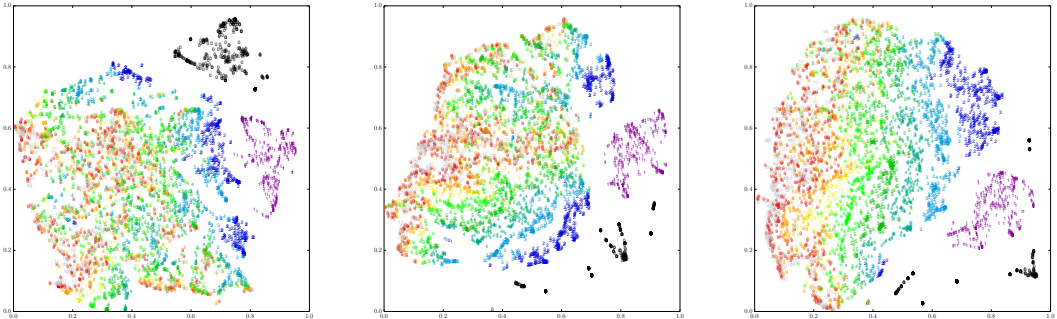
Differences between all pairs of click models are statistically significant $(p < 0.001)$

Results on relevance prediction task

Click model	NDCG @1	NDCG @3	NDCG @5	NDCG @10
DBN	0.717	0.725	0.764	0.833
DCM	0.736	0.746	0.780	0.844
CCM	0.741	0.752	0.785	0.846
UBM	0.724	0.737	0.773	0.838
$\text{NCM}_{\text{QD}}^{\text{RNN}}$	0.762	0.759	0.791	0.851
$\text{NCM}_{\text{QD}}^{\text{LSTM}}$	0.756	0.759	0.789	0.850
$\text{NCM}_{\text{QD+Q}}^{\text{LSTM}}$	0.775	0.773	0.799	0.857
$\text{NCM}_{\text{QD+Q+D}}^{\text{LSTM}}$	0.755	0.755	0.787	0.847

Improvements of $\text{NCM}_{\text{QD}}^{\text{RNN}}$, $\text{NCM}_{\text{QD}}^{\text{LSTM}}$ and $\text{NCM}_{\text{QD+Q}}^{\text{LSTM}}$ over baseline click models are statistically significant ($p < 0.05$)

Analysis



Learns regularities in user browsing behavior that

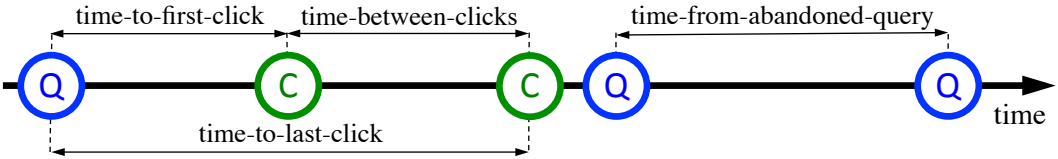
1. have been manually encoded in existing click models, such as **ranks** and **distances to previous clicks** (large clusters on t-SNE projections of vector states $\mathbf{s}_r$)
2. can not be manually encoded in traditional click models (small clusters on t-SNE projections of vector states $\mathbf{s}_r$)

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
  - Click behavior in web search
  - Time aspects of user behavior in web search
  - Web search vs. sponsored search
  - Future work
- Generating responses
- Wrap up

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

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

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

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

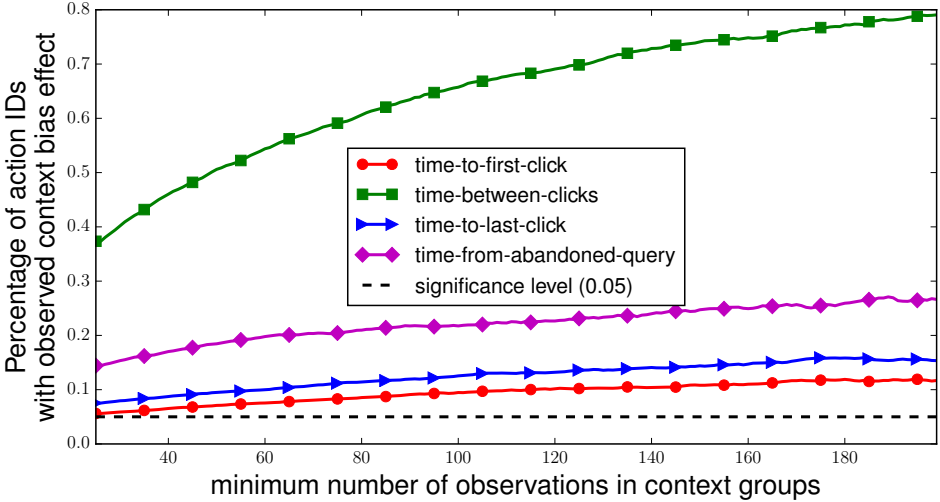
Uncertainty: $\frac{1}{2}(3 + 600)$ vs. $\frac{1}{7}(30 + 28 + 45 + 23 + 100 + 23 + 58)$

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

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

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



Context-aware time modeling (naive)

$$Time(action, context) \sim Gamma(\boldsymbol{k}(act, ctx), \boldsymbol{\theta}(act, ctx))$$

Context-aware time modeling

$$
\begin{aligned}
Time(action, context) \sim Gamma(& \\
& \boldsymbol{a}_k(ctx) \cdot \boldsymbol{k}(act) + \boldsymbol{b}_k(ctx), \\
& \boldsymbol{a}_\theta(ctx) \cdot \boldsymbol{\theta}(act) + \boldsymbol{b}_\theta(ctx))
\end{aligned}
$$

Parameter estimation

$$
\begin{aligned}
Time(action, context) \sim Gamma(& \\
& \boldsymbol{a}_k(ctx) \cdot \boldsymbol{k}(act) + \boldsymbol{b}_k(ctx), \\
& \boldsymbol{a}_\theta(ctx) \cdot \boldsymbol{\theta}(act) + \boldsymbol{b}_\theta(ctx))
\end{aligned}
$$

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
  $\Longrightarrow$ neural networks

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

Dataset

3 months of log data from Yandex search engine

Time between actions	Max time	# Observations
Time-to-first-click	1 min	$30,747,733$
Time-between-clicks	5 min	$6,317,834$
Time-to-last-click	5 min	$30,446,973$
Time-from-abandoned-query	1 min	$11,523,351$

Evaluation tasks

Task1. Predict time between clicks

- Log-likelihood
- Root mean squared error (MSE)

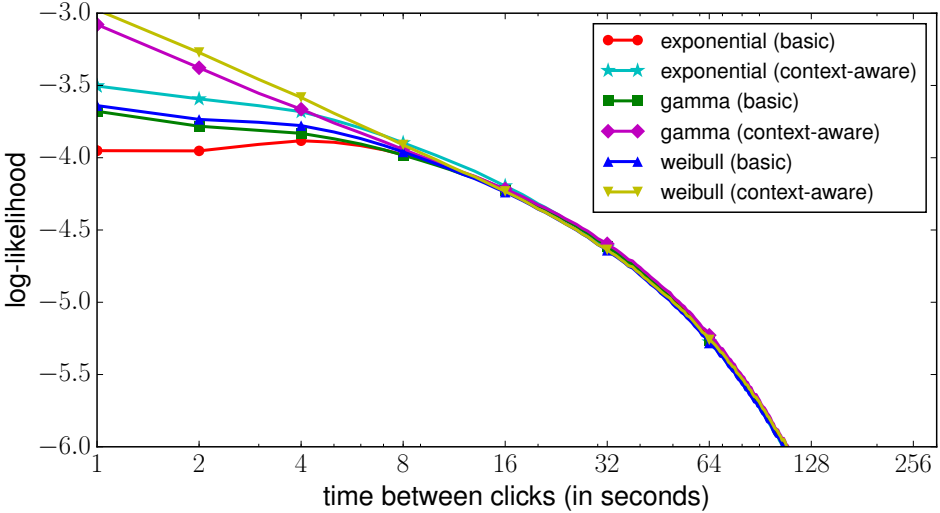
Task2. Rank results based on time between clicks

- nDCG@$\{1, 3, 5, 10\}$

Task 1. Predicting time

Time model	Distribution	Log-likelihood	RMSE
Basic	exponential	-4.9219	60.73
	gamma	-4.9105	60.76
	Weibull	-4.9077	60.76
Context-aware	exponential	-4.8787	58.93
	gamma	-4.8556	58.98
	Weibull	-4.8504	58.94

Results on time prediction task (time-between-clicks)



Task 2. Ranking results

Time model	Distribution	NDCG @1	NDCG @3	NDCG @5	NDCG @10
Average	—	0.651	0.693	0.728	0.812
Context-aware	exponential	0.668	0.710	0.743	0.820
	gamma	0.675	0.715	0.748	0.822
	Weibull	0.671	0.709	0.745	0.821

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
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
  - Click behavior in web search
  - Time aspects of user behavior in web search
  - Web search vs. sponsored search
  - Future work
- Generating responses
- Wrap up

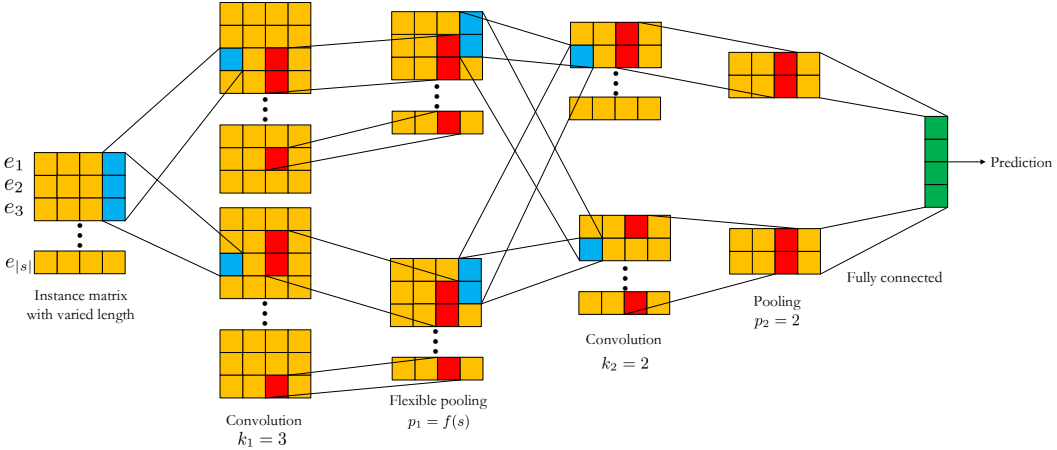
Web search vs. sponsored search

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]

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
  - Click behavior in web search
  - Time aspects of user behavior in web search
  - Web search vs. sponsored search
  - Future work
- Generating responses
- Wrap up

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

By 2027 (50th anniversary of SIGIR), 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
- Text matching I
- Text matching II

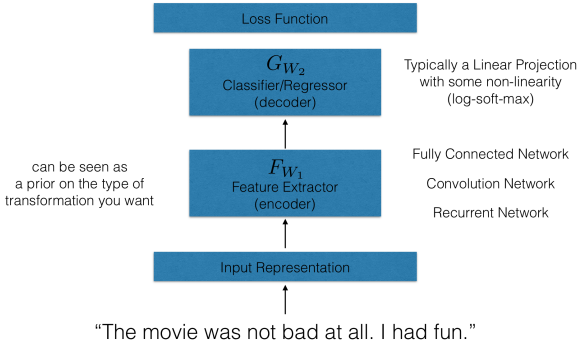
Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
  - One-shot dialogues
  - Open-ended dialogues (chit-chat)
  - Goal-oriented dialogues
  - Resources
- Wrap up

General Formulation of Typical DL Models



The general basic formulation:

- Learnable parametric function
- Inputs: generally considered I.I.D.
- Outputs: classification or regression

Question: Is this enough?

Example Scenario

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

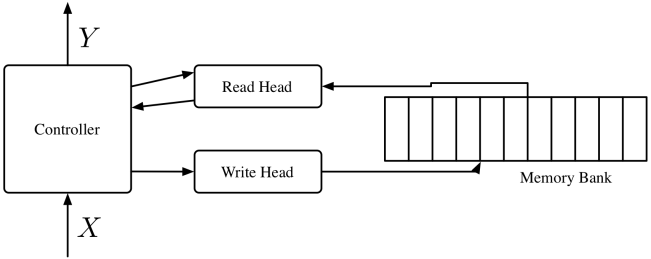
- Where is the milk now? A: office
- Where is Joe? A: bathroom
- Where was Joe before the office? A: kitchen

What is Required?

- The model needs to remember the context
- Given an input, the model needs to know where to look for in the context
- It needs to know what to look for in the context
- It needs to know how to reason using this context
- It needs to handle the potentially changing the context
- A Possible Solution:
  - Hidden states of RNNs have memory: Run an RNN on the context/story/KB and get its representation. Then use the representation to map question to answers/response.
    - will not scale as RNN don't have ability to capture long term dependency: Vanishing gradient, The state of memory is not large enough to capture rich information

Neural Networks with Memory

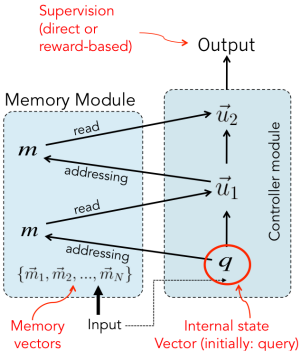
- Memory Networks
  - Fully Supervised MemNNs
  - End2End MemNNs
  - Key-Value MemNNs
- Neural Turing Machines
- Stack/List/Queue Augmented RNNs



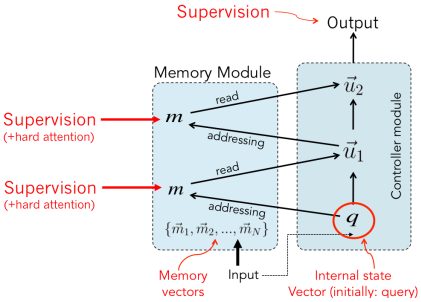
Memory Networks

Class of models which combine large memory with learning component which can read and write to it.

- Step 1: controller converts incoming data to internal feature representation (I)
- Step 2: write head updates the memories and writes the data into memory (G)
- Step 3: given the external input, the read head reads the memory and fetches relevant data (O)
- Step 4: controller combines the external data with memory contents returned by read head to generate output (O, R)
- Inference Given the question, picks the memory which scores the highest and uses the selected memory and the question to generate the answer.



Fully Supervised Memory Networks [Weston et al., 2015]



- Current supporting facts are supplied as supervision signal for learning the memory addressing stage:
  - It is essentially like hard attention except that you already know where to attend!

Fully Supervised Memory Networks

Context:

John was in the bathroom.

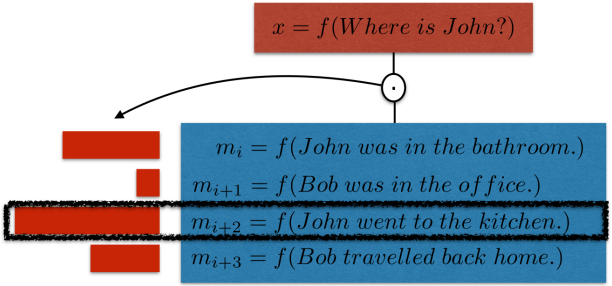
Bob was in the office.

John went to the kitchen. $\Longrightarrow$ Supporting Fact

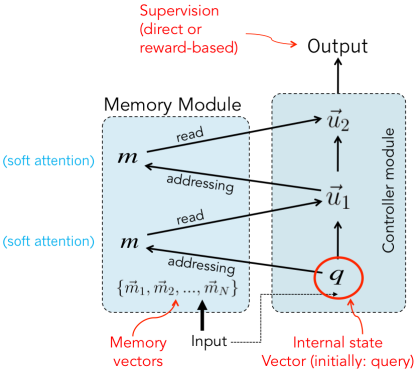
Bob traveled back home.

Question Answer Pair:

Where is John? A: **kitchen**

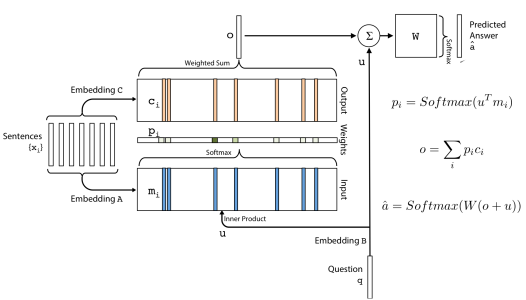


End2End Memory Networks [Sukhbaatar et al., 2015]



- No current supporting fact supplied.
- Learns which parts of the memory are relevant.
- This is achieved by reading using soft attention as opposed to hard.
- Performs multiple lookups to refine its guess about memory relevance.
- Only needs supervision at the final output.

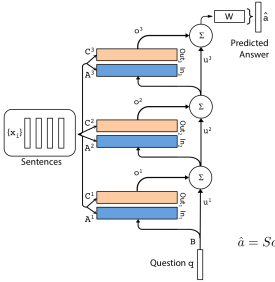
End2End Memory Networks (Multiple Hops)



$$p_i = Softmax(u^T m_i)$$

$$o = \sum_i p_i c_i$$

$$\hat{a} = Softmax(W(o+u))$$



$${}^k p_i = Softmax({}^k u^T \cdot {}^k m_i)$$

$${}^k o = \sum_i {}^k p_i {}^k c_i$$

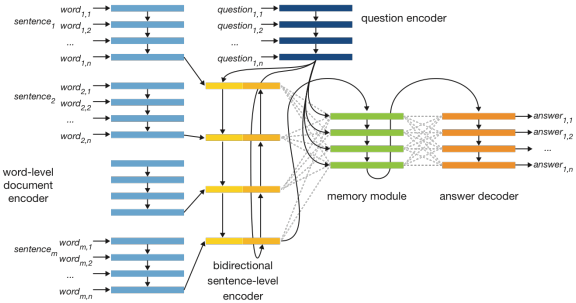
$${}^{k+1} u = {}^k u + {}^k o$$

$$\hat{a} = Softmax(W \cdot {}^{k+1} u) = Softmax(W({}^k o + {}^k u))$$

- 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 style reading at words or characters

Attentive Memory Networks [Kenter and de Rijke, 2017]

- Proposed Model: An end-to-end trainable memory networks with a hierarchical input encoder.



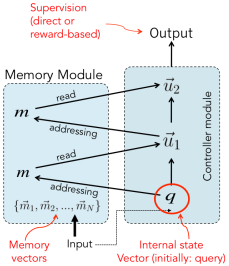
- Framing the task of conversational search as a general machine reading task.

Key-Value Memory Networks [Miller et al., 2016]

$m_i = (k_i, v_i)$

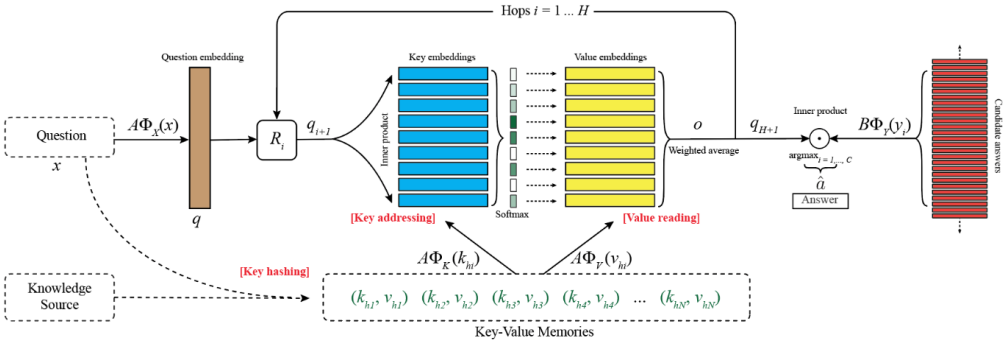
Address: score k_i vs. q

Read: return v_i



- Facts are stored in a key value structured memory.
- Memory is designed so that the model learns to use keys to address relevant memories with respect to the question.
- Structure allows the model to encode prior knowledge for the considered task.
- Structure also allows to leverage possibly complex transforms between key and value.

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.

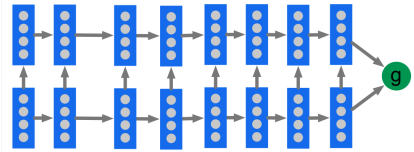
Machine Reading [Hermann et al., 2015]

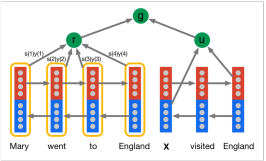
Cloze Style Question Answering:
Teaching a machine to understand language. To read a comprehension and answer questions pertaining to it. However, the questions should be such that they cannot be answered using external world knowledge.

Original Version	Anonymised Version
Context	
The BBC producer allegedly struck by Jeremy Clarkson will not press charges against the "Top Gear" host, his lawyer said Friday. Clarkson, who hosted one of the most-watched television shows in the world, was dropped by the BBC Wednesday after an internal investigation by the British broadcaster found he had subjected producer Oisin Tymon "to an unprovoked physical and verbal attack." ...	the *ent381* producer allegedly struck by *ent212* will not press charges against the " *ent153* " host , his lawyer said friday . *ent212* , who hosted one of the most - watched television shows in the world , was dropped by the *ent381* wednesday after an internal investigation by the *ent180* broadcaster found he had subjected producer *ent193* " to an unprovoked physical and verbal attack . " ...
Query	
Producer **X** will not press charges against Jeremy Clarkson, his lawyer says.	producer **X** will not press charges against *ent212* , his lawyer says .
Answer	
Oisin Tymon	*ent193*

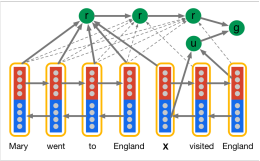
Teaching Machine to Read and Comprehend

General Structure: A two layer Deep LSTM Reader with the question encoded before the document:





(a) Attentive Reader.



(b) Impatient Reader.

Document Attentive Reading:

by *ent423* , *ent261* correspondent updated 9:49 pm et , thu march 19 , 2015 (*ent261*) a *ent114* was killed in a parachute accident in *ent45* , *ent85* , near *ent312* , a *ent119* official told *ent261* on wednesday . he was identified thursday as special warfare operator 3rd class *ent23* , 29 , of *ent187* , *ent265* . `` *ent23* distinguished himself consistently throughout his career . he was the epitome of the quiet professional in all facets of his life , and he leaves an inspiring legacy of natural tenacity and focused

. . .

ent119 identifies deceased sailor as **X** , who leaves behind a wife

by *ent270* , *ent223* updated 9:35 am et , mon march 2 , 2015 (*ent223*) *ent63* went familial for fall at its fashion show in *ent231* on sunday , dedicating its collection to `` mamma " with nary a pair of `` mom jeans " in sight . *ent164* and *ent21* , who are behind the *ent196* brand , sent models down the runway in decidedly feminine dresses and skirts adorned with roses , lace and even embroidered doodles by the designers ' own nieces and nephews . many of the looks featured saccharine needlework phrases like `` i love you ,

. . .

X dedicated their fall fashion show to moms

WikiReading [Hewlett et al., 2016]

Task:

To predict textual values from the structured knowledge base Wikidata by reading the text of the corresponding Wikipedia articles.

	Categorization		Extraction	
Document	Folkart Towers are twin skyscrapers in the Bayrakli district of the Turkish city of Izmir. Reaching a structural height of 200 m (656 ft) above ground level, they are the tallest ...	Angeles blancos is a Mexican telenovela produced by Carlos Sotomayor for Televisa in 1990. Jacqueline Andere, Rogelio Guerra and Alfonso Iturralde star as the main ...	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**, ...	Breaking Bad is an American crime drama television series created and produced by Vince Gilligan. The show originally aired on the AMC network for five seasons, from **January 20, 2008**, to ...
Property	country	original language of work	located next to body of water	start time
Answer	Turkey	Spanish	Atlantic Ocean, Arctic Ocean, Pacific Ocean	20 January 2008

- Categorical: properties, such as instance of, gender and country, require selecting between a relatively small number of possible answers.
- Relational: properties, such as date of birth, parent, and capital, typically require extracting rare or totally unique answers from the document.

WikiReading

- **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
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
  - One-shot dialogues
  - Open-ended dialogues (chit-chat)
  - Goal-oriented dialogues
  - Resources
- Wrap up

Dialogue systems

Dialogues/conversational agents/chat bots

Open-ended dialogues

- ELIZA
- Twitterbots
- Alexa/Google home/Siri/Cortana

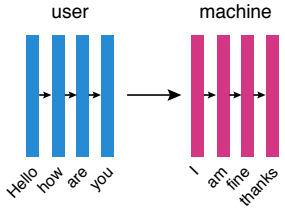
Goal-oriented dialogues

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

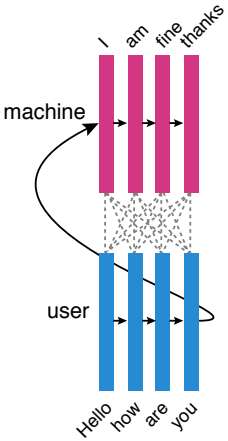
Is this IR?

Dialogue systems

Chit-chat bots



Straightforward seq-to-seq [Vinyals and Le, 2015].
([Sordoni et al., 2015b] 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

Human: *what is your job?*
Machine: *i'm a lawyer.*
Human: *what do you do?*
Machine: *i'm a doctor.*
Example from [Vinyals and Le, 2015]

Evaluation

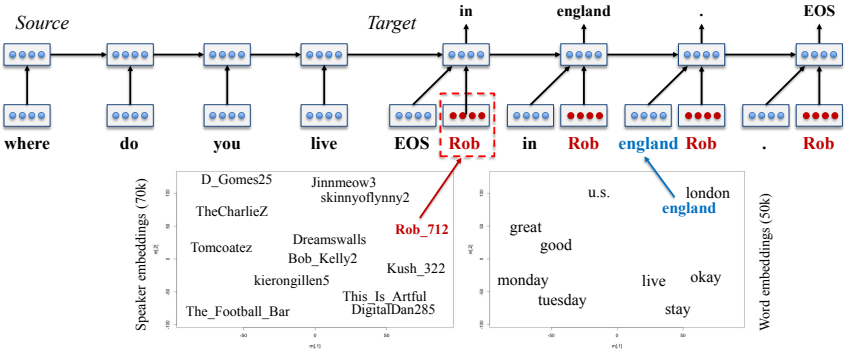
- Perplexity?
- BLUE/METEOR?
- Nice overview of *How NOT To Evaluate Your Dialogue System* [Liu et al., 2016].
- Open problem....

Dialogue systems

4 solutions

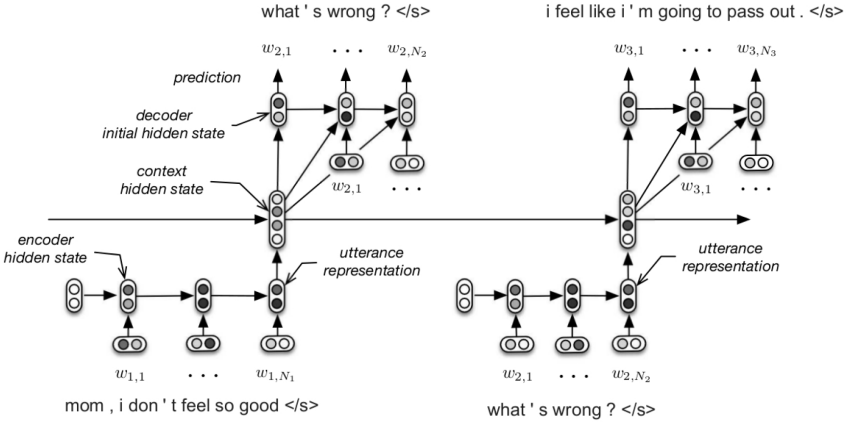
- More consistency by maintaining a persona
- More consistency in dialogue with hierarchical network
- Less generic responses with different optimization function
- More natural responses with GANs

Persona-based



A persona-based model for generating responses [Li et al., 2016].

Dialogue systems



Hierarchical seq-to-seq [Serban et al., 2016]. Main evaluation metric: perplexity.

Dialogue systems

Avoid generic responses

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

$$C_{LL} = \arg\max_{u_t} log\, p(u_t|\textit{context}) = \arg\max_{u_t} log\, p(u_t|u_0 \dots u_{t-1})$$

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

$$\begin{aligned} C_{MMI} &= \arg\max_{u_t} log \frac{p(u_t, \textit{context})}{p(u_t)p(\textit{context})} \\ &= \textit{[derivation, next page ...]} \\ &= \arg\max_{u_t} (1-\lambda) log\, p(u_t|\textit{context}) + \lambda log\, p(\textit{context}|u_t) \end{aligned}$$

Dialogue systems

Bayes rule

$$log\, p(u_t|context) = log \frac{p(context|u_t)p(u_t)}{p(context)}$$

$$log\, p(u_t|context) = log\, p(context|u_t) + log\, p(u_t) - log\, p(context)$$

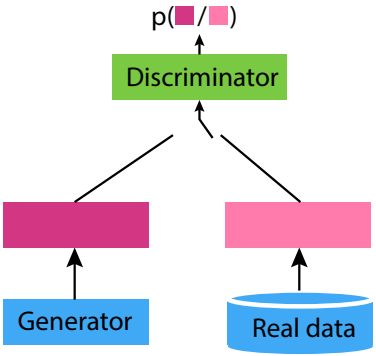
$$log\, p(u_t) = log\, p(u_t|context) - log\, p(context|u_t) + log\, p(context)$$

$$\begin{aligned}
C_{MMI} &= \arg\max_{u_t} log \frac{p(u_t, context)}{p(u_t)p(context)} = \arg\max_{u_t} log \frac{p(u_t|context)p(context)}{p(u_t)p(context)} \\
&= \arg\max_{u_t} log \frac{p(u_t|context)}{p(u_t)} \\
&= \arg\max_{u_t} log\, p(u_t|context) - log\, p(u_t) \leftarrow \text{weird, minus language model score} \\
&= \arg\max_{u_t} log\, p(u_t|context) - \lambda log\, p(u_t) \leftarrow \text{Introduce lambda} \\
&= \arg\max_{u_t} log\, p(u_t|context) - \lambda(log\, p(u_t|context) - log\, p(context|u_t) + log\, p(context)) \\
&= \arg\max_{u_t} (1-\lambda) log\, p(u_t|context) + \lambda log\, p(context|u_t)
\end{aligned}$$

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

Generative adversarial network for dialogues

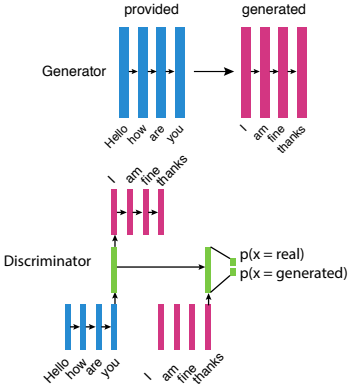
- Discriminator network
  - Classifier: real or generated utterance
- Generator network
  - Generate a realistic utterance



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.

Code available at `https://github.com/jiweil/Neural-Dialogue-Generation`

Dialogue systems

Open-ended dialogue systems

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

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
  - One-shot dialogues
  - Open-ended dialogues (chit-chat)
  - Goal-oriented dialogues
  - Resources
- Wrap up

Goal-oriented

Idea

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

Challenges

- Training data
- Keeping track of dialogues
- 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* $\rightarrow$ *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.

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
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
  - One-shot dialogues
  - Open-ended dialogues (chit-chat)
  - Goal-oriented dialogues
  - Resources
- Wrap up

Resources

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/

- WikiReading (Google Research)

https://github.com/google-research-datasets/wiki-reading

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up
  - Maarten
  - Alexey
  - Bhaskar
  - Christophe
  - Mostafa
  - Tom

Maarten

IR

Keep asking how NNs help us get closer to "getting the right information to the right people in the right way"

Away from supervision

Train deep models based naturally occurring "pseudo labels" or interactions

Autonomy

Focus on solutions that are increasingly autonomous—with that comes the need for transparency: models that are able to offer explanations of their decisions

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up
  - Maarten
  - Alexey
  - Bhaskar
  - Christophe
  - Mostafa
  - Tom

Alexey

NLP vs. IR

A lot of progress on short text matching was done in NLP community.
Should IR community be focused on long texts?

Predict → Understand → Simulate

Neural networks lead to state-of-the-art results on prediction tasks.
We understand some of the models' internal workings. But not all.
How well can we simulate user behavior remains an open question.

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up
  - Maarten
  - Alexey
  - Bhaskar
  - Christophe
  - Mostafa
  - Tom

Bhaskar's notes

Library or Librarian?

When you learn representations of text - you are encoding real-world knowledge into your model (e.g., "us president" is related to "obama"). How much knowledge should your model encode? Is your model a *library* of knowledge, or a good *librarian* capable of working with incomplete information? What's the best trade-off?

Hammer or Lens?

Neural IR research shouldn't be all about trying shiniest new neural models for IR. If you use DNNs as a *hammer*, every problem will look like a nail. These new models should also be a *lens* through which we grow our understanding of IR fundamentals.

Neu-IR'17 workshop

For more discussions, please come to the workshop on Friday.

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up
  - Maarten
  - Alexey
  - Bhaskar
  - Christophe
  - Mostafa
  - Tom

Christophe's thoughts

Unsupervised learning as a first-class citizen
Learning complicated models from query/relevance pairs is great, but let's try to figure out how well we can do without them.

Occam's razor
Do we really need those 10,000 extra parameters or 10 hidden layers?

Is the hype real? — or will the bubble burst?
We've been doing fine without deep learning for a long time. Skepticism is healthy, but only as long as it doesn't kill innovation.

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up
  - Maarten
  - Alexey
  - Bhaskar
  - Christophe
  - Mostafa
  - Tom

Mostafa

Remember the past and think about tomorrow

We have a really nice set of matured solutions and deep understanding of problems for many IR tasks. Ignoring everything we have is probably not a good idea. Let's keep in touch with what we achieved so far while developing new models.

Reproducibility

IR community is always proud of being concerned about reproducibility. It is actually getting more important with DNNs to keep the position of this concern in our mind.

Do the math!

Getting good results would be awesome if your model is also theoretically justified.

Outline

Morning program
- Preliminaries
- Text matching I
- Text matching II

Afternoon program
- Learning to rank
- Modeling user behavior
- Generating responses
- Wrap up
  - Maarten
  - Alexey
  - Bhaskar
  - Christophe
  - Mostafa
  - Tom

Tom

What have the new methods brought us

- human-computer natural language communication

Biggest current challenges

- training material
- consistency
- long-term dependencies
- evaluation
  - naturalness?
  - tone of voice?
  - on-topic-ness?

New directions

- deep reinforcement learning
- new twists
  - Attention is all you need [Vaswani et al., 2017]?
- GANs
- Your next idea...?!?

References I

Qingyao Ai, Liu Yang, Jiafeng Guo, and W. Bruce Croft. 2016a. Analysis of the Paragraph Vector Model for Information Retrieval. In *ICTIR*. ACM, 133–142.

Qingyao Ai, Liu Yang, Jiafeng Guo, and W Bruce Croft. 2016b. Improving language estimation with the paragraph vector model for ad-hoc retrieval. In *SIGIR*. ACM, 869–872.

Qingyao Ai, Yongfeng Zhang, Keping Bi, Xu Chen, and Bruce W. Croft. 2017. Learning a Hierarchical Embedding Model for Personalized Product Search. In *SIGIR*.

Nima Asadi, Donald Metzler, Tamer Elsayed, and Jimmy Lin. 2011. Pseudo test collections for learning web search ranking functions. In *SIGIR*. ACM, 1073–1082.

Leif Azzopardi, Maarten de Rijke, and Krisztian Balog. 2007. Building simulated queries for known-item topics: An analysis using six European languages. In *SIGIR*. ACM.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors.. In *ACL (1)*. 238–247.

Steven M Beitzel, Eric C Jensen, Abdur Chowdhury, and David Grossman. 2003. Using titles and category names from editor-driven taxonomies for automatic evaluation. In *CIKM*. ACM, 17–23.

Yoshua Bengio and Jean-Sébastien Senécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks* 19, 4 (2008), 713–722.

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

Richard Berendsen, Manos Tsagkias, Wouter Weerkamp, and Maarten de Rijke. 2013. Pseudo test collections for training and tuning microblog rankers. In *SIGIR*. ACM, 53–62.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *JMLR* 3 (2003), 993–1022.

Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. *ICLR* (2017).

Alexey Borisov, Ilya Markov, Maarten de Rijke, and Pavel Serdyukov. 2016. A neural click model for web search. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 531–541.

Chris Burges. 2015. RankNet: A ranking retrospective. (2015). `https://www.microsoft.com/en-us/research/blog/ranknet-a-ranking-retrospective/` Accessed July 16, 2017.

References II

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*. ACM, 89–96.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.

Christopher JC Burges, Robert Ragno, and Quoc Viet Le. 2006. Learning to rank with nonsmooth cost functions. In *NIPS*, Vol. 6. 193–200.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*. ACM, 129–136.

Gabriele Capannini, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Nicola Tonellotto. 2016. Quality versus efficiency in document scoring with learning-to-rank models. *IPM* 52, 6 (2016), 1161–1177.

Ben Carterette and Rosie Jones. 2008. Evaluating search engines by modeling the relationship between relevance and clicks. In *NIPS*. 217–224.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *KDD*. ACM, 785–794.

Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhi-Ming Ma, and Hang Li. 2009. Ranking measures and loss functions in learning to rank. In *Advances in Neural Information Processing Systems*. 315–323.

Charles LA Clarke, Maheedhar Kolla, Gordon V Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and diversity in information retrieval evaluation. In *SIGIR*. ACM, 659–666.

David Cossock and Tong Zhang. 2006. Subset ranking using regression. In *COLT*, Vol. 6. Springer, 605–619.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41, 6 (1990), 391–407.

Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W Bruce Croft. 2017. Neural Ranking Models with Weak Supervision. In *SIGIR*.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards End-to-End Reinforcement Learning of Dialogue Agents for Information Access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*.

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

John Rupert Firth. 1957. *Papers in Linguistics 1934-1951*. Oxford University Press.

References III

Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences. *Journal of machine learning research* 4, Nov (2003), 933–969.

Norbert Fuhr. 1989. Optimum polynomial retrieval functions based on the probability ranking principle. *ACM Transactions on Information Systems (TOIS)* 7, 3 (1989), 183–204.

Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth JF Jones. 2015. Word embedding based generalized language model for information retrieval. In *SIGIR*. ACM, 795–798.

Yasser Ganjisaffar, Rich Caruana, and Cristina Lopes. 2011. Bagging Gradient-Boosted Trees for High Precision, Low Variance Ranking Models. In *SIGIR*. ACM, 85–94.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

Joshua Goodman. 2001. Classes for fast maximum entropy training. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, Vol. 1. IEEE, 561–564.

Artem Grotov and Maarten de Rijke. 2016. Online Learning to Rank for Information Retrieval: SIGIR 2016 Tutorial. In *SIGIR*. ACM, 1215–1218.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. ACM, 55–64.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models.. In *AISTATS*, Vol. 1. 6.

Zellig S Harris. 1954. Distributional structure. *Word* 10, 2-3 (1954), 146–162.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 2000. Large margin rank boundaries for ordinal regression. (2000).

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*.

S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *SIGIR*. ACM, 50–57.

References IV

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*. 2042–2050.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, 2333–2338.

Bouke Huurnink, Katja Hofmann, and Maarten de Rijke. 2010a. Simulating searches from transaction logs. *Simulation of Interaction* (2010), 21.

Bouke Huurnink, Katja Hofmann, Maarten De Rijke, and Marc Bron. 2010b. Validating query simulators: An experiment using commercial searches and purchases. In *CLEF*. Springer, 40–51.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2016. Image-to-Image Translation with Conditional Adversarial Networks. *arxiv* (2016).

Rolf Jagerman, Julia Kiseleva, and Maarten de Rijke. 2017. Modeling Label Ambiguity for Neural List-Wise Learning to Rank. In *Neu-IR SIGIR Workshop*.

Kalervo Järvelin and Jaana Kekäläinen. 2000. IR Evaluation Methods for Retrieving Highly Relevant Documents. In *SIGIR*. ACM, 41–48.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On Using Very Large Target Vocabulary for Neural Machine Translation. *arXiv preprint arXiv:1412.2007* (2014).

Shihao Ji, SVN Vishwanathan, Nadathur Satish, Michael J Anderson, and Pradeep Dubey. 2015. Blackout: Speeding up recurrent neural network language models with very large vocabularies. *arXiv preprint arXiv:1511.06909* (2015).

Thorsten Joachims. 2003. Evaluating Retrieval Performance Using Clickthrough Data. In *Text Mining*. 79–96.

Thorsten Joachims. 2006. Training linear SVMs in linear time. In *KDD*. ACM, 217–226.

Thorsten Joachims, Dayne Freitag, and Tom Mitchell. 1997. Webwatcher: A tour guide for the world wide web. In *IJCAI*. 770–777.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410* (2016).

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese cbow: Optimizing word embeddings for sentence representations. *arXiv preprint arXiv:1606.04640* (2016).

Tom Kenter and Maarten de Rijke. 2017. Attentive Memory Networks: Efficient Machine Reading for Conversational Search. In *The First International Workshop on Conversational Approaches to Information Retrieval (CAIR'17)*.

References V

Jinyoung Kim and W. Bruce Croft. 2009. Retrieval Experiments Using Pseudo-desktop Collections. In *CIKM*. ACM, 1297–1306.

Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *ICML*. 1188–1196.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. In *NAACL-HLT 2016*. 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P. Spithourakis, Jianfeng Gao, and William B. Dolan. 2016. A Persona-Based Neural Conversation Model. In *ACL 2016*.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547* (2017).

Ping Li, Qiang Wu, and Christopher J Burges. 2008. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in neural information processing systems*. 897–904.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *EMNLP 2016* (2016).

Qiang Liu, Feng Yu, Shu Wu, and Liang Wang. 2015. A convolutional click prediction model. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 1743–1746.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *SIGDIAL 2015*. 285–294.

R Duncan Luce. 2005. *Individual choice behavior: A theoretical analysis*. Courier Corporation.

S. MacAvaney, K. Hui, and A. Yates. 2017. An Approach for Weakly-Supervised Deep Information Retrieval. arXiv 1707.00189. (2017).

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. 3111–3119.

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*.

Bhaskar Mitra. 2015. Exploring session context using distributed representations of queries and reformulations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 3–12.

References VI

Bhaskar Mitra and Nick Craswell. 2017. An Introduction to Neural Information R. *Foundations and Trends® in Information Retrieval (to appear)* (2017).

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to Match Using Local and Distributed Representations of Text for Web Search. 1291–1299.

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*. 1081–1088.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426* (2012).

Frederic Morin and Yoshua Bengio. 2005. Hierarchical Probabilistic Neural Network Language Model.. In *Aistats*, Vol. 5. Citeseer, 246–252.

Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. 2016. Improving Document Ranking with Dual Word Embeddings.

Federico Nanni, Bhaskar Mitra, Matt Magnusson, and Laura Dietz. 2017. Benchmark for Complex Answer Retrieval. ACM.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text Matching as Image Recognition.. In *AAAI*. 2793–2799.

Filip Radlinski and Thorsten Joachims. 2005. Query chains: learning to rank from implicit feedback. In *KDD*. ACM, 239–248.

Filip Radlinski, Madhu Kurup, and Thorsten Joachims. 2008. How does clickthrough data reflect retrieval quality?. In *CIKM*. ACM, 43–52.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, 45–50. `http://is.muni.cz/publication/884893/en`.

Dwaipayan Roy, Debjyoti Paul, Mandar Mitra, and Utpal Garain. 2016. Using Word Embeddings for Automatic Query Expansion. *arXiv preprint arXiv:1606.07608* (2016).

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *Int. J. Approximate Reasoning* 50, 7 (2009), 969–978.

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

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models.. In *AAAI*. 3776–3784.

Pararth Shah, Dilek Hakkani-Tür, and Larry Heck. 2016. Interactive reinforcement learning for task-oriented dialogue management. In *NIPS 2016 Deep Learning for Action and Interaction Workshop*.

References VII

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL 2016*.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 101–110.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob G. Simonson, and Jian-Yun Nie. 2015a. A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion. In *CIKM 2015*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Meg Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015b. A Neural Network Approach to Context-Sensitive Generation of Conversational Responses. In *Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies (NAACL-HLT)*.

K. Sparck Jones and C.J. van Rijsbergen. 1976. *Report on the need for and provision of an 'ideal' information retrieval test collection*. Technical Report. Computer Laboratory, Cambridge University.

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

Jean Tague, Michael Nelson, and Harry Wu. 1980. Problems in the simulation of bibliographic retrieval systems. In *SIGIR*. Butterworth & Co., 236–255.

Jean Tague and Michael J Nelson. 1981. Simulation of user judgments in bibliographic retrieval systems. In *ACM SIGIR Forum*, Vol. 16. ACM, 66–71.

Paul Thomas, Daniel McDuff, Mary Czerwinski, and Nick Craswell. 2017. MISC: A data set of information-seeking conversations. In *The First International Workshop on Conversational Approaches to Information Retrieval (CAIR'17)*.

Jörg Tiedemann. 2009. News from OPUS-A collection of multilingual parallel corpora with tools and interfaces. In *Recent advances in natural language processing*, Vol. 5. 237–248.

Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2016. Learning Latent Vector Spaces for Product Search. In *CIKM*. ACM, 165–174.

Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2017a. Neural Vector Spaces for Unsupervised Information Retrieval. *arXiv preprint arXiv:1708.02702* (2017).

Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2017b. Semantic Entity Retrieval Toolkit. In *Neu-IR SIGIR Workshop*.

Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2017c. Structural Regularities in Expert Vector Spaces. In *ICTIR*. ACM.

References VIII

Christophe Van Gysel, Maarten de Rijke, and Marcel Worring. 2016. Unsupervised, Efficient and Semantic Expertise Retrieval. In *WWW*. ACM, 1069–1079.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. *arXiv* (2017).

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

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *ICML Deep Learning Workshop 2015* (2015).

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

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *ICLR*.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*. ACM, 1192–1199.

Emine Yilmaz and Stephen Robertson. 2009. Deep Versus Shallow Judgments in Learning to Rank. In *SIGIR*. ACM, 662–663.

Hamed Zamani and W. Bruce Croft. 2016a. Embedding-based Query Language Models. In *ICTIR*. ACM, 147–156.

Hamed Zamani and W. Bruce Croft. 2016b. Estimating Embedding Vectors for Queries. In *ICTIR*. ACM, 123–132.

Hamed Zamani and W Bruce Croft. 2017. Relevance-based Word Embedding. In *SIGIR*.

Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. 2014. Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks.. In *AAAI*. 1369–1375.

Guoqing Zheng and Jamie Callan. 2015. Learning to reweight terms with distributed representations. In *SIGIR*. ACM, 575–584.

Zhaohui Zheng, Keke Chen, Gordon Sun, and Hongyuan Zha. 2007. A regression framework for learning ranking functions using relative relevance judgments. In *SIGIR*. ACM, 287–294.

Guido Zuccon, Bevan Koopman, Peter Bruza, and Leif Azzopardi. 2015. Integrating and Evaluating Neural Word Embeddings in Information Retrieval. In *20th Australasian Document Computing Symposium*. ACM, Article 12, 8 pages.

Notation

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

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.