Neural Networks for Information Retrieval

nn4ir

Neural Networks for Information Retrieval

Tom Kenter[1] Alexey Borisov[2,3] Christophe Van Gysel[2] Mostafa Dehghani[2]
Maarten de Rijke[2] Hosein Azarbonyad[2]

[1]Booking.com

[2]University of Amsterdam

[3]Yandex

NN4IR@WSDM2018

Who are we?

Tom Kenter
Booking.com

Alexey Borisov
Yandex & U. Amsterdam

Christophe Van Gysel
U. Amsterdam

Mostafa Dehghani
U. Amsterdam

Maarten de Rijke
U. Amsterdam

Hosein Azarbonyad
U. Amsterdam

Who are we?

Tom Kenter
Booking.com

Alexey Borisov
Yandex & U. Amsterdam

Christophe Van Gysel
U. Amsterdam

Mostafa Dehghani
U. Amsterdam

Maarten de Rijke
U. Amsterdam

Hosein Azarbonyad
U. Amsterdam

Aims

Describe the use of neural network based methods in information retrieval

Compare architectures

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

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

Structure of the tutorial

Morning

1. Preliminaries
2. Modeling user behavior
3. Semantic matching
4. Learning to rank

Afternoon

5. Entities
6. Generating responses
7. Recommender systems
8. Industry insights
9. Q & A

Materials

Slides available at `http://nn4ir.com/wsdm2018`

Bibliography available at `http://nn4ir.com/wsdm2018`

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

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Common language

Q1: What is Information Retrieval (IR)?

Q2: What is a Neural Network (NN)?

Common language

Q1: What is Information Retrieval (IR)?
A1: The purpose of Information Retrieval systems is to help people find the right (most useful) information in the right (most convenient) format at the right time (when they need it).

Q2: What is a Neural Network (NN)?

Common language

Q1: What is Information Retrieval (IR)?
A1: The purpose of Information Retrieval systems is to help people find the right (most useful) information in the right (most convenient) format at the right time (when they need it).

Q2: What is a Neural Network (NN)?
A2: A function $F(x;\Theta)$ with (a large number of) parameters Θ that maps an input object x (which can be text, image or arbitrary vector of features) to an output object y (class label, sequence of class labels, text, image).

Common language

Q1: What is Information Retrieval (IR)?
A1: The purpose of Information Retrieval systems is to help people find the right (most useful) information in the right (most convenient) format at the right time (when they need it).

Q2: What is a Neural Network (NN)?
A2: A function $F(x;\Theta)$ with (a large number of) parameters Θ that maps an input object x (which can be text, image or arbitrary vector of features) to an output object y (class label, sequence of class labels, text, image).

Q3: How to obtain the parameters Θ?

Q4: What kinds of functions $F(x;\Theta)$ can be used?

Common language

Q1: What is Information Retrieval (IR)?
A1: The purpose of Information Retrieval systems is to help people find the right (most useful) information in the right (most convenient) format at the right time (when they need it).

Q2: What is a Neural Network (NN)?
A2: A function $F(x;\Theta)$ with (a large number of) parameters Θ that maps an input object x (which can be text, image or arbitrary vector of features) to an output object y (class label, sequence of class labels, text, image).

Q3: How to obtain the parameters Θ?
A3: We learn/train the parameters Θ using back-propagation

Q4: What kinds of functions $F(x;\Theta)$ can be used?

Common language

Q1: What is Information Retrieval (IR)?
A1: The purpose of Information Retrieval systems is to help people find the right (most useful) information in the right (most convenient) format at the right time (when they need it).

Q2: What is a Neural Network (NN)?
A2: A function $F(x;\Theta)$ with (a large number of) parameters Θ that maps an input object x (which can be text, image or arbitrary vector of features) to an output object y (class label, sequence of class labels, text, image).

Q3: How to obtain the parameters Θ?
A3: We learn/train the parameters Θ using back-propagation

Q4: What kinds of functions $F(x;\Theta)$ can be used?
A4: There are many different classes of $F(x;\Theta)$, which are called architectures. We will talk about them throughout the tutorial.

Outline

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

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

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

Outline

Morning program

Preliminaries

Feedforward neural network

Distributed representations

Recurrent neural networks

Sequence-to-sequence models

Convolutional neural networks

Modeling user behavior

Semantic matching

Learning to rank

Afternoon program

Entities

Generating responses

Recommender systems

Industry insights

Q & A

Distributed representations

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

Cosine similarity to determine how similar two vectors are:

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

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

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

Distributed representations

Recurrent neural networks

Sequence-to-sequence models

Convolutional neural networks

Modeling user behavior

Semantic matching

Learning to rank

Afternoon program

Entities

Generating responses

Recommender systems

Industry insights

Q & A

Recurrent neural networks

- Lots of information is sequential and requires a memory for successful processing
- For example, Alice helped Bob is not the same as Bob helped Alice; and I had my car cleaned is not the same as I had cleaned my car
- Sequences of arbitrary lengths as input and 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 are 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

Outline

Morning program

Preliminaries

Feedforward neural network

Distributed representations

Recurrent neural networks

Sequence-to-sequence models

Convolutional neural networks

Modeling user behavior

Semantic matching

Learning to rank

Afternoon program

Entities

Generating responses

Recommender systems

Industry insights

Q & A

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`

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

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

Distributed representations

Recurrent neural networks

Sequence-to-sequence models

Convolutional neural networks

Modeling user behavior

Semantic matching

Learning to rank

Afternoon program

Entities

Generating responses

Recommender systems

Industry insights

Q & A

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.

Take aways

1. Information Retrieval (IR) systems help people find the right (most useful) information in the right (most convenient) format at the right time (when they need it).
2. Neural Network (NN) is a function $F(x;\Theta)$ with (a large number of) parameters Θ that maps an input object x (which can be text, image or arbitrary vector of features) to an output object y (class label, sequence of class labels, text, image).
3. There three main architectures (classes of $F(x;\Theta)$): (i) feed-forward NN (FFNN), (ii) recurrent NN (RNN), (iii) convolutional NN (CNN).
4. Embeddings are vector representations of objects in a high-dimensional space that are learned during training. In many practical applications, these vectors reflect similarities between objects that are important for solving the task.
5. Other stuff, such as seq2seq, vanishing and exploding gradients, LSTM and GRU, attention mechanism, softmax, cross-entropy.

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Understanding user behavior is the key

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

User behavioral signals

Actions

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

Times between actions

(e.g., time between clicks, time to first/last click)

Interpretation is difficult

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

Clicks are biased towards:

- higher ranked results (position bias)
- visually salient results (attention bias)
- previously unseen results (novelty bias)

CLICK 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
- Modeling user behavior
  - Click behavior in web search
  - Time aspects of user behavior in web search
  - Web search vs. sponsored search
  - Take aways and future work
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Traditional click models

Graphical representation of the cascade click model.

Pros: Based on the probabilistic graphical model (PGM) framework
Cons: Structure of the underlying PGM has to be set manually

Neural click modeling framework

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

Learns patterns of user behavior directly from click-through data

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 $\rightarrow$ NCM$^{\{\text{RNN, LSTM}\}}_{\{\text{QD, QD+Q, QD+Q+D}\}}$

Representations of q, d_r and i_r

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

Parameterization of $\mathcal{I}$, $\mathcal{U}$ 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
- Modeling user behavior
  - Click behavior in web search
  - Time aspects of user behavior in web search
  - Web search vs. sponsored search
  - Take aways and future work
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Times between user actions

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

How to interpret times between user actions

Average

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

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

		NDCG			
Time model	Distribution	@1	@3	@5	@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
- Modeling user behavior
  - Click behavior in web search
  - Time aspects of user behavior in web search
  - Web search vs. sponsored search
  - Take aways and future work
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Web search vs. sponsored search

In web search we work (mostly) with query sessions and search sessions
In sponsored search we need to consider longer user histories

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

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

Web search vs. sponsored search

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

Outline

Morning program
- Preliminaries
- Modeling user behavior
  - Click behavior in web search
  - Time aspects of user behavior in web search
  - Web search vs. sponsored search
  - Take aways and future work
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Take aways and future work

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

Understanding and modelling user behavior with PGMs — is a mature field
We expect many ideas to be transfered from PGM to neural framework

Future user behavior models

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

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Semantic matching

Definition

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

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

Outline

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

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Unsupervised semantic matching with pre-trained representations

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

How can we use word embeddings for information retrieval tasks?

Word embedding

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

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

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

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

From word embeddings to query/document embeddings

Creating representations for compound units of text (e.g., documents) from representation of lexical units (e.g., words).

From word embeddings to query/document embeddings

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

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

- Averaging the word representations of query terms has been extensively explored in different settings. [Vulić and Moens, 2015, Zamani and Croft, 2016b]
  - Effective but for small units of text, e.g. query [Mitra, 2015].
- Training word embeddings directly for the purpose of being averaged [Kenter et al., 2016].

From word embeddings to query/document embeddings

- 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)

Using similarity amongst documents, queries and terms.

Given low-dimensional representations, integrate their similarity signal within IR.

Dual Embedding Space Model (DESM) [Nalisnick et al., 2016]

Word2vec optimizes IN-OUT dot product which captures the co-occurrence statistics of words from the training corpus:

- We can gain by using these two embeddings differently

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 embeddings for document retrieval and ranking

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

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

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

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

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

- IN-OUT captures more topical notion of similarity than IN-IN and OUT-OUT.
- DESM is effective at, but only at, ranking at least somewhat relevant documents.

Pre-trained word embeddings for document retrieval and ranking

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

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

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

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

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

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

Pre-trained word embeddings for document retrieval and ranking

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

- Terms in a query are generated by sampling them independently from either the document or the collection.
- The noisy channel may transform (mutate) a term t into a term t'.

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

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

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

Pre-trained word embeddings for query term weighting

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

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

$$\vec{x}_{t_q} = \vec{v}_{t_q} - \frac{1}{|q|} \sum_{t'_q \in q} \vec{v}_{t'_q}$$

- $\vec{x}_{t_q}$ measures the semantic difference of a term to the whole query.
- Learn a model to map the feature vectors the defined target term weights.

Pre-trained word embeddings for query expansion

- Identify expansion terms using word2vec cosine similarity [Roy et al., 2016].
  - pre-retrieval:
    - Taking nearest neighbors of query terms as the expansion terms.
  - post-retrieval:
    - Using a set of pseudo-relevant documents to restrict the search domain for the candidate expansion terms.
  - pre-retrieval incremental:
    - Using an iterative process of reordering and pruning terms from the nearest neighbors list.
      - Reorder the terms in decreasing order of similarity with the previously selected term.
- Works better than having no query expansion, but does not beat non-neural query expansion methods.

Pre-trained word embedding for query expansion

- Embedding-based Query Expansion [Zamani and Croft, 2016a]
  Main goal: Estimating a better language model for the query using embeddings.
- Embedding-based Relevance Model:
  Main goal: Semantic similarity in addition to term matching for PRF.

Pre-trained word embedding for query expansion

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

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

Outline

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

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Learning unsupervised representations for semantic matching

Pre-trained word embeddings can be used to obtain

- a query/document representation through compositionality, or
- a similarity signal to integrate within IR frameworks.

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

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].

Neural Vector Spaces for Unsupervised IR [Van Gysel et al., 2017a]

- Learns query (term) and document representations directly from the document collection.
- Outperforms existing latent vector space models and provides semantic matching signal complementary to lexical retrieval models.
- Learns a notion of term specificity.
- Luhn significance: mid-frequency words are more important for retrieval than infrequent and frequent words.

Relation between query term representation L2-norm within NVSM and its collection frequency. Taken from [Van Gysel et al., 2017a].

Outline

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

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Text matching as a supervised objective

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

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

Representation-based models

Representation-based models construct a fixed-dimensional vector representation for each text separately and then perform matching within the latent space.

(C)DSSM [Huang et al., 2013, Shen et al., 2014]

- Siamese network between query and document, performed on character trigrams.
- Originally introduced for learning from implicit feedback.

ARC-I [Hu et al., 2014]

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

Interaction-based models

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

The interaction matrix is subsequently summarized into a matching score.

DRMM [Guo et al., 2016]

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

MatchPyramid [Pang et al., 2016]

- Interaction matrix between query/document terms, followed by convolutional layers.
- After convolutions, feed-forward layers determine matching score.

aNMM [Yang et al., 2016]

- Compute word interaction matrix.
- Aggregate similarities by running multiple kernels.
- Every kernel assigns a different weight to a particular similarity range.
- Similarities are aggregated to the kernel output by weighting them according to which bin they fall in.

Match-SRNN [Wan et al., 2016b]

- Word interaction layer, followed by a spatial recurrent NN.
- The RNN hidden state is updated using the current interaction coefficient, and the hidden state of the prefix.

K-NRM [Xiong et al., 2017b]

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

Hybrid models

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

These two components can occur (1) in serial or (2) in parallel.

ARC-II [Hu et al., 2014]

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

MV-LSTM [Wan et al., 2016a]

- Cascade approach where input representations for the interaction matrix are generated using a bi-directional LSTM.
- Differs from pure interaction-based approaches as the LSTM builds a representation of the context, rather than using the representation of a word.
- Obtains fixed-dimensional representation by max-pooling over query/document; followed by feed-forward network.

Duet [Mitra et al., 2017]

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

DeepRank [Pang et al., 2017]

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

Outline

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

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Beyond supervised signals: semi-supervised learning

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

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

Pseudo test/training collections

Given a source of pseudo relevance, we can build pseudo collections for training retrieval models [Asadi et al., 2011, Berendsen et al., 2013].

Sources of pseudo-relevance

Typically given by external knowledge about retrieval domain, such as hyperlinks, query logs, social tags, ...

Training neural networks using pseudo relevance

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

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

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

Training neural networks using pseudo relevance

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

- Using a news corpus with article headlines acting as pseudo-queries and article content as pseudo-documents.
- Problems:
  - **Hard-Negative**
  - **Mismatched-Interaction**: (example: "When Bird Flies In", a sports article about basketball player Larry Bird)
- Solutions:
  - **Ranking filter**:
    - top pseudo-documents are considered as negative samples.
    - only pseudo-queries that are able to retrieve their pseudo-relevant documents are used as positive samples.
  - **Interaction filter**:
    - building interaction embeddings for each pair.
    - filtering out based on similarity to the template query-document pairs.

Query expansion using neural word embeddings based on pseudo relevance

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

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

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

- Relevance is not necessarily equal to semantically or syntactically similarity:
  - "united state" as expansion terms for "Indian American museum".
- Main idea: Defining the "context"
  Using the relevance model distribution for the given query to define the context. So the objective is to predict the words observed in the documents relevant to a particular information need.
- The neural network will be constraint by the given weights from RM3 to learn word embeddings.

Outline

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

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Document & entity representation learning toolkits

gensim : `https://github.com/RaRe-Technologies/gensim` [Řehůřek and Sojka, 2010]

SERT : `http://www.github.com/cvangysel/SERT` [Van Gysel et al., 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]

MatchZoo : `https://github.com/faneshion/MatchZoo` [Fan et al., 2017]

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Outline

Morning program

- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank
  - Overview & basics
  - Pointwise loss
  - Pairwise loss
  - Listwise loss
  - Different levels of supervision
  - Some recent relevant work
  - Toolkits

Afternoon program

- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

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.

A problem of historical terminology

With the increasing interest within semantic matching models (LTM), the term learning to rank has become ambiguous.

Training data and objectives can be used to optimize models that specifically focus on text matching (see previous section).

In this tutorial, we use learning to rank to refer to signal-agnostic models. That is, models that learn to generate rankings from arbitrary matching, importance or recency signals, amongst others.

A problem of historical terminology

Semantic matching signals as input to a general-purpose ranker. Taken from [Li and Lu, 2016].

How long will this hierarchical view remain valid?

Three training objectives

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)

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

$$p(z_i) = \frac{e^{\gamma z_i}}{\sum_{z \in Z} e^{\gamma z}} \qquad (\gamma \text{ is a constant}) \tag{1}$$

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,

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

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

$$CE(p, q) = -\log(q_{correct}) \tag{3}$$

Outline

Morning program

- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank
  - Overview & basics
  - Pointwise loss
  - Pairwise loss
  - Listwise loss
  - Different levels of supervision
  - Some recent relevant work
  - Toolkits

Afternoon program

- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

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,

$$\mathcal{L}_{Squared} = \|y_{q,d} - f(\vec{x}_{q,d})\|^2 \tag{4}$$

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],

$$\mathcal{L}_{\text{CE}}(q, d, y_{q,d}) = -\log\Big(p(y_{q,d}|q, d)\Big) = -\log\Big(\frac{e^{\gamma \cdot s_{y_{q,d}}}}{\sum_{y \in Y} e^{\gamma \cdot s_y}}\Big) \tag{5}$$

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

Outline

Morning program

Preliminaries

Modeling user behavior

Semantic matching

Learning to rank

Overview & basics

Pointwise loss

Pairwise loss

Listwise loss

Different levels of supervision

Some recent relevant work

Toolkits

Afternoon program

Entities

Generating responses

Recommender systems

Industry insights

Q & A

Pairwise objectives

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

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

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

Feature vectors: $\vec{x}_i$ and $\vec{x}_j$

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

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

$$\mathcal{L}_{pairwise} = \phi(s_i - s_j) \tag{6}$$

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,

$$\mathcal{L}_{RankNet} = -\bar{p}_{ij} \log(p_{ij}) - \bar{p}_{ji} \log(p_{ji}) \tag{7}$$

$$= -\log(p_{ij}) \tag{8}$$

$$= \log(1 + e^{-\gamma(s_i - s_j)}) \tag{9}$$

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,

$$p(d^+|q) = \frac{e^{\gamma \cdot s\left(q, d^+\right)}}{\sum_{d \in D} e^{\gamma \cdot s(q,d)}} \tag{10}$$

The cross entropy loss is then given by,

$$\mathcal{L}_{\mathsf{CE}}(q, d^+, D) = -\log\left(p(d^+|q)\right) \tag{11}$$

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

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
- Modeling user behavior
- Semantic matching
- Learning to rank
  - Overview & basics
  - Pointwise loss
  - Pairwise loss
  - Listwise loss
  - Different levels of supervision
  - Some recent relevant work
  - Toolkits

Afternoon program

- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Listwise

Blue: relevant Gray: non-relevant

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

Due to strong position-based discounting in IR measures, errors at 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

$$\lambda_{LambdaRank} = \lambda_{RankNet} \cdot |\Delta NDCG| \tag{13}$$

LambdaMart

LambdaMART combines LambdaRank and MART (Multiple Additive Regression Trees).

- MART is a boosted tree model in which the output of the model is a linear combination of the outputs of a set of regression trees.
- While MART uses gradient boosted decision trees for prediction tasks, LambdaMART uses gradient boosted decision trees using a cost function derived from LambdaRank for solving a ranking task.
- LambdaMART is able to choose splits and leaf values that may decrease the utility for some queries, as long as the overall utility increases.
- On experimental datasets, LambdaMART has shown better results than LambdaRank and the original RankNet.

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:

$$p(\pi|s) = \frac{\phi(s_2)}{\phi(s_1) + \phi(s_2) + \phi(s_3) + \phi(s_4)} \cdot \frac{\phi(s_1)}{\phi(s_1) + \phi(s_3) + \phi(s_4)} \cdot \frac{\phi(s_4)}{\phi(s_3) + \phi(s_4)} \tag{14}$$

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

Modeling user behavior

Semantic matching

Learning to rank

Overview & basics

Pointwise loss

Pairwise loss

Listwise loss

Different levels of supervision

Some recent relevant work

Toolkits

Afternoon program

Entities

Generating responses

Recommender systems

Industry insights

Q & A

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.

Outline

Morning program

- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank
  - Overview & basics
  - Pointwise loss
  - Pairwise loss
  - Listwise loss
  - Different levels of supervision
  - Some recent relevant work
  - Toolkits

Afternoon program

- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

IRGAN [Wang et al., 2017]

There are two main way of thinking about modeling retrieval:

- The generative retrieval focusing on predicting relevant documents given a query
- The discriminative retrieval focusing on predicting relevancy given a query-document pair.

Main idea: a theoretical minimax game to iteratively optimize both of these models based on the idea of GAN.

Two cool, new ideas

- Learning to Rank with Query-level Semi-supervised Autoencoders [Xu et al., 2017]: Besides the reconstruction loss, they introduce extra supervision using a query-level constraint.
  - Objectives:
    - Minimizing the distance between its inputs and output (reconstruction loss)
    - Minimizing differences of the query-level retrieval performance between the inputs and the outputs.
- Alternating Pointwise and Pairwise Learning [Lei et al., 2017]
  - Try to get the best of both worlds: alternating between Pointwise and Pairwise loss over pairwise examples
  - Evaluated on personalized item ranking

Outline

Morning program

Preliminaries

Modeling user behavior

Semantic matching

Learning to rank

Overview & basics

Pointwise loss

Pairwise loss

Listwise loss

Different levels of supervision

Some recent relevant work

Toolkits

Afternoon program

Entities

Generating responses

Recommender systems

Industry insights

Q & A

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
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Entities are polysemic

"Finding entities" has multiple meanings.

Entities can be

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

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
  - Knowledge graph embeddings
  - Entity mentions in unstructured text
  - Entity finding
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Knowledge graphs

Triples

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

...
```

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

Knowlegde graphs

Tasks

- Link prediction
  Predict the missing `h` or `t` for a triple `(h, r, t)`
  Rank entities by score. Metrics:
  - Mean rank of correct entity
  - Hits@10
- Triple classification
  Predict if `(h, r, t)` is correct.
  Metric: accuracy.
- Relation fact extraction from free text
  Use knowledge base as weak supervision for extracting new triples.
  Suppose some IE system gives us `(steve_jobs, ``was the initiator of'', apple)`, then we want to predict the `founder_of` relation.

Datatsets

WordNet

`(car, hyponym, vehicle)`

Freebase/DBPedia

`(steve_jobs, founder_of, apple)`

Knowlegde graphs

Knowledge graph embeddings

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

TransE

"Translation intuition"

For a triple `(h, l, t)` : $\vec{\mathtt{h}} + \vec{\mathtt{l}} \approx \vec{\mathtt{t}}$.

TransE

"Translation intuition"

For a triple `(h, l, t)` : $\vec{\mathtt{h}} + \vec{\mathtt{l}} \approx \vec{\mathtt{t}}$.

$$\mathcal{L} = \sum_{(h,\ell,t)\in S} \sum_{(h',\ell,t')\in S'_{(h,\ell,t)}} \left[\gamma + d(\boldsymbol{h}+\boldsymbol{\ell},\boldsymbol{t}) - d(\boldsymbol{h}'+\boldsymbol{\ell},\boldsymbol{t}')\right]_+$$

positive examples

negative examples

distance function

[Bordes et al., 2013]

TransE

"Translation intuition"
For a triple `(h, l, t)` : $\vec{\mathtt{h}} + \vec{\mathtt{l}} \approx \vec{\mathtt{t}}$.

How about:

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

TransH

(a) TransE (b) TransH

[Wang et al., 2014]

TransH

(a) TransE (b) TransH

$$f_r(\mathbf{h},\mathbf{t}) = \|(\overbrace{\mathbf{h}}^{\mathbf{h}_\perp} - \mathbf{w}_r^\top \mathbf{h}\mathbf{w}_r) + \mathbf{d}_r - (\overbrace{\mathbf{t}}^{\mathbf{t}_\perp} - \mathbf{w}_r^\top \mathbf{t}\mathbf{w}_r)\|_2^2$$

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

distance function

[Wang et al., 2014]

TransH

i.e., translation vector d_r is in the hyperplane

Constraints

$$\forall e \in E, \|\mathbf{e}\|_2 \leq 1, \text{//scale}$$

$$\forall r \in R, |\mathbf{w}_r^\top \mathbf{d}_r| / \|\mathbf{d}_r\|_2 \leq \epsilon, \text{//orthogonal}$$

$$\forall r \in R, \|\mathbf{w}_r\|_2 = 1, \text{//unit normal vector}$$

(b) TransH

soft constraints

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

$$+ C \left\{ \sum_{e\in E} \left[\|\mathbf{e}\|_2^2 - 1 \right]_+ + \sum_{r\in R} \left[\frac{(\mathbf{w}_r^\top \mathbf{d}_r)^2}{\|\mathbf{d}_r\|_2^2} - \epsilon^2 \right]_+ \right\}, \quad (4)$$

[Wang et al., 2014]

TransR

Use different embedding spaces for entities and relations

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

[Lin et al., 2015]

TransR

[Lin et al., 2015]

TransR

Relations: R^d

Entities: R^k

M_r = projection matrix: k * d

$$\mathbf{h}_r = \mathbf{h}\mathbf{M}_r, \quad \mathbf{t}_r = \mathbf{t}\mathbf{M}_r$$

$$f_r(h,t) = \|\mathbf{h}_r + \mathbf{r} - \mathbf{t}_r\|_2^2$$

Constraints:

$$\|\mathbf{h}\|_2 \leq 1$$
$$\|\mathbf{r}\|_2 \leq 1$$
$$\|\mathbf{t}\|_2 \leq 1$$
$$\|\mathbf{t}\mathbf{M_r}\|_2 \leq 1$$
$$\|\mathbf{h}\mathbf{M_r}\|_2 \leq 1$$

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

[Lin et al., 2015]

Challenges

- How about time?
  E.g., some relations hold from a certain date, until a certain date.
- New entities/relationships
- Finding synonymous relationships/duplicate entities `(2005, end_date, destinys_child)` `(destinys_child, disband, 2005)` `(destinys_child, last_performance, 2005)`
- Evaluation
  Link prediction? Relation classification? Is this fair? As in, is this even possible in all cases (for a human without any world knowledge)?

Resources: toolkits + knowledge bases

Source Code

KB2E : `https://github.com/thunlp/KB2E` [Lin et al., 2015]

TransE : `https://everest.hds.utc.fr/`

Knowledge Graphs

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

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
  - Knowledge graph embeddings
  - Entity mentions in unstructured text
  - Entity finding
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Entity mentions

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

Linking Link mentions to knowledge graph entities.

Utilization Use mentions to improve search.

Named entity recognition

B-ORG	O	B-MISC	O	O	O	B-MISC	O	O
EU	rejects	German	call	to	boycott	British	lamb	.

Task

vanilla RNN

Named entity recognition

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

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

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

Named entity recognition

BI-LSTM-CRF model

[Huang et al., 2015]

Entity disambiguation

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

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

Learn similarity between document and entity representations using supervision. Taken from [He et al., 2013].

Entity linking

Learn representations for the context, the mention, the entity (using surface words) and the entity class. Uses pre-trained word2vec embeddings. Taken from [Sun et al., 2015].

Entity linking

Encode Wikipedia descriptions, linked mentions in Wikipedia and fine-grained entity types. All representations are optimized jointly. Taken from [Gupta et al., 2017].

Entity linking

A single mention phrase refers to various entities. Multi-Prototype Mention Embedding model that learns multiple sense embeddings for each mention by jointly modeling words from textual contexts and entities derived from a KB. Taken from [Cao et al., 2017].

Improving search using linked entities

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

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
  - Knowledge graph embeddings
  - Entity mentions in unstructured text
  - Entity finding
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Entity finding

Task definition

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

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

Different instantiations of the task within varying domains:

- Wikipedia: INEX Entity Ranking Track [de Vries et al., 2007, Demartini et al., 2008, 2009, 2010] (lots of text, knowledge graph, revisions)
- Enterprise search: expert finding [Balog et al., 2006, 2012] (few entities, abundance of text per entity)
- E-commerce: product ranking [Rowley, 2000] (noisy text, customer preferences)

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

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

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

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 predict 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].

Resources: toolkits

SERT : `http://www.github.com/cvangysel/SERT` [Van Gysel et al., 2017b]

HEM : `https://ciir.cs.umass.edu/downloads/HEM` [Ai et al., 2017]

Resources: further reading on entities/KGs

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

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

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

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

Tasks

- Question Answering
- Summarization
- Query Suggestion
- Reading Comprehension / Wiki Reading
- Dialogue Systems
  - Goal-Oriented
  - Chit-Chat

Example Scenario for machine reading task

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

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

Example Scenario for machine reading task

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

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

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

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

What is Required?

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

A Possible Solution:

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

This will not scale as RNN states don't have ability to capture long term dependency: vanishing gradients, limited state size.

Teaching Machine to Read and Comprehend

(a) Attentive Reader.

(b) Impatient Reader.

[Hermann et al., 2015]

Neural Networks with Memory

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

End2End Memory Networks [Sukhbaatar et al., 2015]

- Learns which parts of the memory are relevant.
- This is achieved by reading using attention.
- 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 reading of words? 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

Example:

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

[Miller et al., 2016]

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

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

	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: relatively small number of possible answer (e.g.: instance of, gender, country).
- Relational: rare or totally unique answers (e.g.: date of birth, parent,capital).

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
- Modeling user behavior
- Semantic matching
- Learning to rank

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

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

3 solutions

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

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

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

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

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

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

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

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

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

$$= \arg\max_{u_t} log\, p(u_t|context) - \lambda(log\, p(u_t|context) - log\, p(context|u_t) + log\, p(context))$$

$$= \arg\max_{u_t} (1 - \lambda) log\, p(u_t|context) + \lambda log\, p(context|u_t)$$

(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
- Modeling user behavior
- Semantic matching
- Learning to rank

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

Goal-oriented

Idea

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

Challenges

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

Goal-oriented as seq2seq

Memory network [Bordes and Weston, 2017]

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

Restaurant Knowledge Base, i.e., a table.
Queried by API calls.
Each row = restaurant:

- cuisine (10 choices, e.g., French, Thai)
- location (10 choices, e.g., London, Tokyo)
- price range (cheap, moderate or expensive)
- rating (from 1 to 8)

For words of relevant entity types

- add a trainable entity vector

Goal-oriented as reinforcement learning

A typical reinforcement learning system:

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

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

Typically [Shah et al., 2016]:

- States: agents interpretation of the environment: distribution over user intents, dialogue acts and slots and their values
  - intent(buy_ticket)
  - inform(destination=Atlanta)
  - ...
- Actions: possible communications, and are usually designed as a combination of dialogue act tags, slots and possibly slot values
  - request(departure_date)
  - ...

Goal-oriented as reinforcement learning

Restaurant finding [Wen et al., 2017]:

- Neural belief tracking: distribution over a possible values of a set of slots
- Delexicalisation: swap slot-values for generic token (e.g. *Chinese*, *Indian*, *Italian* $\rightarrow$ *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
- Modeling user behavior
- Semantic matching
- Learning to rank

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

Alternatives to RNNs

RNNs are:

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

However, RNNs have several drawbacks:

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

Can we do better?

- WaveNet
- ByteNet
- Transformer

Alternatives to RNNs: WaveNet

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

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

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

Based on the idea of dilated causal convolutions.

[van den Oord et al., 2016]

Alternatives to RNNs: WaveNet

Causal convolutions

[van den Oord et al., 2016]

Alternatives to RNNs: WaveNet

Dilated causal convolutions

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

[van den Oord et al., 2016]

Alternatives to RNNs: ByteNet

[Kalchbrenner et al., 2016]

Alternatives to RNNs: Transformer

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

[Vaswani et al., 2017]

Alternatives to RNNs: Transformer

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

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

Resources: datasets

Open-ended dialogue

- Opensubtitles [Tiedemann, 2009]
- Twitter: http://research.microsoft.com/convo/
- Weibo:

http://www.noahlab.com.hk/topics/ShortTextConversation

- Ubuntu Dialogue Corpus [Lowe et al., 2015]
- Switchboard

https://web.stanford.edu/~jurafsky/ws97/

- Coarse Discourse (Google Research)

https://research.googleblog.com/2017/05/

coarse-discourse-dataset-for.html

Goal-oriented dialogues

- MISC: A data set of information-seeking conversations [Thomas et al., 2017]
- Maluuba Frames

http://datasets.maluuba.com/Frames

- Loqui Human-Human Dialogue Corpus

https://academiccommons.columbia.edu/catalog/ac:176612

- bAbi (Facebook Research)

https://research.fb.com/downloads/babi/

Machine reading

- bAbi QA (Facebook Research)

https://research.fb.com/downloads/babi/

- QA Corpus [Hermann et al., 2015]

https://github.com/deepmind/rc-data/

- WikiReading (Google Research)

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

Resources: source code

- End-to-end memory network

  https://github.com/facebook/MemNN

- Attentive Memory Networks

  https://bitbucket.org/TomKenter/attentive-memory-networks-code

- Hierarchical NN [Serban et al., 2016]

  https://github.com/julianser/hed-dlg, https://github.com/julianser/rnn-lm

- GAN for dialogues

  https://github.com/jiweil/Neural-Dialogue-Generation

- RL for dialogue agents [Dhingra et al., 2017]

  https://github.com/MiuLab/KB-InfoBot

- Transformer network

  https://github.com/tensorflow/tensor2tensor

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
  - Items and Users
  - Matrix factorization
  - Matrix factorization as a network
  - Side information
  - Richer models
  - Other tasks
  - Wrap-up
- Industry insights
- Q & A

Recommender systems – The task

- Build a model that estimates how a user will like an item.
- A typical recommendation setup has
  - matrix with users and items
  - plus ratings of users for items reflecting past/known preferences

  and tries to predict future preferences
- This is **not** about rating prediction

2			4	5	
5		4			1
		5		2	
	1		5		4
		4		2	
4	5		1		

[Karatzoglou and Hidasi, Deep Learning for Recommender Systems, RecSys '17, 2017]

Approaches to recommender systems

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

Warm, cold

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

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
  - Items and Users
  - Matrix factorization
  - Matrix factorization as a network
  - Side information
  - Richer models
  - Other tasks
  - Wrap-up
- Industry insights
- Q & A

Matrix factorization

- The recommender system's work horse

Matrix factorization

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

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

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

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
  - Items and Users
  - Matrix factorization
  - Matrix factorization as a network
  - Side information
  - Richer models
  - Other tasks
  - Wrap-up
- Industry insights
- Q & A

A feed-forward neural network view

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

[Raimon and Basilico, Deep Learning for Recommender Systems, 2017]

A deeper view

Matrix factorization vs. feed-forward network

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

Great escape . . .

- Side information
- Richer models
- Other tasks

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
  - Items and Users
  - Matrix factorization
  - Matrix factorization as a network
  - Side information
  - Richer models
  - Other tasks
  - Wrap-up
- Industry insights
- Q & A

Side information for recommendation

(1)

(2)

(3)

(4)

Side information for recommendation

- Textual side information
  - Product description, reviews, etc.
  - Extraction: RNNs, one dimensional CNNs, word embeddings, paragraph vectors
  - Applications: news, products, books, publication
- Images
  - Product pictures, video thumbnails
  - Extraction: CNNs
  - Applications: fashion, video
- Music/audio
  - Extraction: CNNs and RNNs
  - Applications: music

Textual side information

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

Textual information for improving recommendations

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

Images in recommendation

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

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

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
  - Items and Users
  - Matrix factorization
  - Matrix factorization as a network
  - Side information
  - Richer models
  - Other tasks
  - Wrap-up
- Industry insights
- Q & A

Alternative models

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

Restricted Boltzman Machines – RBM

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

Auto-encoders

Auto-encoders

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

Constraining the model: improve generalization

- Sparse auto-encoders: activation of units are limited
- Denoising auto-encoders: corrupt the input

Auto-encoders for recommendation

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

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

Prod2vec and Item2vec

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

user2vec
(Continuous Bag of Words)

Wide + Deep models

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

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
  - Items and Users
  - Matrix factorization
  - Matrix factorization as a network
  - Side information
  - Richer models
  - Other tasks
  - Wrap-up
- Industry insights
- Q & A

Other tasks

- Session-based recommendation
- Contextual sequence prediction
- Time-sensitive sequence prediction

- Causality in recommendations
- Recommendation as question answering
- Deep reinforcement learning for recommendations

Session-based recommendation

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

GRU4Rec

Network structure [Hidasi et al., 2016]

- Input: one hot encoded item ID
- Output: scores over all items
- Goal: predicting the next item in the session

Adapting GRU to session-based recommendations

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

GRU4Rec

Session-parallel mini-batches

- Mini-batch is defined over sessions

Output sampling

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

Contextual sequence prediction

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

Time-sensitive sequence prediction

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

[Raimon and Basilico, Deep Learning for Recommender Systems, 2017]

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

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

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

- Performance: MF vs. propensity weighted MF

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

- How to incorporate this in neural networks for recommender systems?

Recommendations as question answering – [Dodge et al., 2015]

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

Task 2: Recommendation

Schindler's List, The Fugitive, Apocalypse Now, Pulp Fiction, and The Godfather are films I really liked.
Can you suggest a film? The Hunt for Red October

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

Task 3: QA + Recommendation Dialog

I loved Billy Madison, My Neighbor Totoro, Blades of Glory, Bio-Dome, Clue, and Happy Gilmore.
I'm looking for a Music movie. School of Rock
What else is that about? Music, Musical, Jack Black, school, teacher, Richard Linklater, rock, guitar
I like rock and roll movies more. Do you know anything else? Little Richard

Tombstone, Legends of the Fall, Braveheart, The Net, Outbreak, and French Kiss are films I really liked.
I'm looking for a Fantasy movie. Jumanji
Who directed that? Joe Johnston
I like Tim Burton movies more. Do you know anything else? Big Fish

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

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

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

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
  - Items and Users
  - Matrix factorization
  - Matrix factorization as a network
  - Side information
  - Richer models
  - Other tasks
  - Wrap-up
- Industry insights
- Q & A

Wrap-up

- Current directions
  - Item/user embedding
  - Deep collaborative filtering
  - Feature extraction from content
  - Session- and context-based recommendation
  - Fairness, accuracy, confidentiality, and transparency (FACT)
- Deep learning can work well for recommendations
- Matrix factorization and deep learning are very similar in classic recommendation setup
- Lots of areas to explore

Resources

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

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Deep Learning in industry

- Companies have endless amounts of data!
  Or do they?
- Performance
  Is .9 accuracy/F_1/etc. good enough?
  No? Would 0.95 be?
- Business logic/constraints
  - *Your model is doing great in general, but not in case X, Y and Z.*
    Can you keep it exactly as it is now, and fix just these cases?
- Explicit domain knowledge
  E.g.: recommending product X for user Y is not applicable, as it is not available where user Y lives.

Deep Learning in industry

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

Hybrid Code Networks

Task

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

Combining RNNs with domain-specific knowledge

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

[Williams et al., 2017]

Hybrid Code Networks

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

[Williams et al., 2017]

Hybrid Code Networks

Training of the RNN

Supervised setting

Every step: update weights, according to entropy loss on correct prediction of actions.

Reinforcement learning

At the end of the dialogue: update weights, according to:

LSTM

return of the dialog: $G = 0.95^{T-1}$

$$\mathbf{w} \leftarrow \mathbf{w} + \Big(\sum_t \nabla_{\mathbf{w}} \log \pi(a_t|\mathbf{h_t};\mathbf{w})\Big)(G - b)$$

Jacobian

estimate of the average return of the current policy, estimated on the last 100 dialogs

Smart Reply

Automated response suggestion for email

Use an RNN to generate responses for any given input message.

Additional constraints

- **Response quality**
  Ensure that the individual response options are always high quality in language and content.
- **Utility**
  Select multiple options to show a user so as to maximize the likelihood that one is chosen.
- **Scalability**
  Process millions of messages per day while remaining within the latency requirements.
- **Privacy**
  Develop this system without ever inspecting the data except aggregate statistics.

[Kannan et al., 2016]

Smart Reply

[Kannan et al., 2016]

Response selection

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

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

Utility/diversity

Goal: present user with diverse responses
Instead of "No", "No, thanks", and "Thanks!", we'd rather produce "No, thanks", "Yes, please", "Let me come back to it".

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

What do we learn?

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

Outline

Morning program
- Preliminaries
- Modeling user behavior
- Semantic matching
- Learning to rank

Afternoon program
- Entities
- Generating responses
- Recommender systems
- Industry insights
- Q & A

Q & A

Tom Kenter
Booking.com

Alexey Borisov
Yandex & U. Amsterdam

Christophe Van Gysel
U. Amsterdam

Hosein Azarbonyad
U. Amsterdam

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.

K. Balog, L. Azzopardi, and M. de Rijke. 2006. Formal models for expert finding in enterprise corpora. In *SIGIR*. 43–50.

Krisztian Balog, Yi Fang, Maarten de Rijke, Pavel Serdyukov, and Luo Si. 2012. Expertise Retrieval. *Found. & Tr. in Inform. Retr.* 6, 2-3 (2012), 127–256.

Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the GRU: Multi-task Learning for Deep Text Recommendations. In *RecSys*. 107–114.

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*. 238–247.

Yoshua Bengio and Jean-Sébastien Senécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *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.

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

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

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*. 2787–2795.

References II

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 *WWW*. 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.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *ICML*. 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.

Yixin Cao, Lifu Huang, Heng Ji, Xu Chen, and Juanzi Li. 2017. Bridge text and knowledge by learning multi-prototype entity mention embedding. In *ACL*, Vol. 1. 1623–1633.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *ICML*. 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 *NIPS*. 315–323.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *DLRS*. 7–10.

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.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*. ACM, 160–167.

References III

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR* 12, Aug (2011), 2493–2537.

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

Arjen P. de Vries, Anne-Marie Vercoustre, James A. Thom, Nick Craswell, and Mounia Lalmas. 2007. Overview of the INEX 2007 entity ranking track. In *Focused Access to XML Documents*. Springer, 245–251.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *JASIS* 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*.

Gianluca Demartini, Arjen P. de Vries, Tereza Iofciu, and Jianhan Zhu. 2008. Overview of the INEX 2008 Entity Ranking Track. In *Advances in Focused Retrieval*. Springer Berlin Heidelberg, 243–252.

Gianluca Demartini, Tereza Iofciu, and Arjen P De Vries. 2009. Overview of the INEX 2009 entity ranking track. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*. Springer, 254–264.

Gianluca Demartini, Tereza Iofciu, and Arjen P De Vries. 2010. Overview of the INEX 2009 Entity Ranking Track. In *Focused Retrieval and Evaluation*. Springer, 254–264.

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 *ACL*.

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

Laura Dietz, Alexander Kotov, and Edgar Meij. 2017. Utilizing Knowledge Graphs in Text-centric Information Retrieval. In *WSDM*. ACM, 815–816.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander H. Miller, Arthur Szlam, and Jason Weston. 2015. Evaluating Prerequisite Qualities for Learning End-to-End Dialog Systems. *CoRR* abs/1511.06931 (2015).

Yixing Fan, Liang Pang, JianPeng Hou, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2017. MatchZoo: A Toolkit for Deep Text Matching. *arXiv preprint arXiv:1707.07270* (2017).

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

References IV

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

Norbert Fuhr. 1989. Optimum polynomial retrieval functions based on the probability ranking principle. *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 *NIPS*. 2672–2680.

Joshua Goodman. 2001. Classes for fast maximum entropy training. In *ICASSP*, Vol. 1. IEEE, 561–564.

Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in Your Inbox: Product Recommendations at Scale. In *KDD*. 1809–1818.

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.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *EMNLP*. 2681–2690.

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.

Ruining He and Julian McAuley. 2016. VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback. In *AAAI*. 144–150.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013. Learning entity representation for entity disambiguation. In *ACL*, Vol. 2. 30–34.

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

References V

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

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

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. *ICLR* (2016).

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.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*. 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 *CIKM*. ACM, 2333–2338.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2016. Image-to-Image Translation with Conditional Adversarial Networks. *arXiv preprint arXiv:1611.07004* (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.

References VI

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

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural Machine Translation in Linear Time. *arXiv preprint arXiv:1610.10099* (2016).

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, László Lukács, Marina Ganea, Peter Young, and others. 2016. Smart Reply: Automated response suggestion for email. In *KDD*. ACM, 955–964.

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese CBOW: Optimizing Word Embeddings for Sentence Representations. In *ACL*. 941–951.

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)*.

Tom Kenter, Llion Jones, and Daniel Hewlett. 2018. Byte-level Machine Reading across Morphologically Varied Languages. In *AAAI*.

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

Yu Lei, Wenjie Li, Ziyu Lu, and Miao Zhao. 2017. Alternating Pointwise-Pairwise Learning for Personalized Item Ranking. In *CIKM*. ACM, 2155–2158.

Hang Li and Zhengdong Lu. 2016. Deep learning for information retrieval. In *SIGIR*. ACM, 1203–1206.

Hang Li, Jun Xu, and others. 2014. Semantic matching in search. *Foundations and Trends® in Information Retrieval* 7, 5 (2014), 343–469.

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, 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 *NIPS*. 897–904.

Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M. Blei. 2016. Factorization Meets the Item Embedding: Regularizing Matrix Factorization with Item Co-occurrence. In *RecSys '16*. 59–66.

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

References VII

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. In *EMNLP*.

Qiang Liu, Feng Yu, Shu Wu, and Liang Wang. 2015. A convolutional click prediction model. In *CIKM*. 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*. 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 *SIGIR*. ACM, 3–12.

Bhaskar Mitra and Nick Craswell. 2017. An introduction to neural information retrieval. *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 *NIPS*. 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. 246–252.

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

Thomas Nedelec, Elena Smirnova, and Flavian Vasile. 2016. Content2vec: Specializing joint representations of product images and text for the task of product recommendation. (2016).

References VIII

Kezban Dilek Onal, Ye Zhang, Ismail Sengor Altingovde, Md Mustafizur Rahman, Pinar Karagoz, Alex Braylan, Brandon Dang, Heng-Lu Chang, Henna Kim, Quinten McNamara, and others. 2017. Neural information retrieval: At the end of the early years. *Information Retrieval Journal* (2017), 1–72.

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

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. 2017. DeepRank: A New Deep Architecture for Relevance Ranking in Information Retrieval. In *CIKM*. ACM, 257–266.

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 *LREC Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, 45–50. `http://is.muni.cz/publication/884893/en`.

Jennifer Rowley. 2000. Product search in e-shopping: a review and research propositions. *Journal of consumer marketing* 17, 1 (2000), 20–35.

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.

Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann Machines for Collaborative Filtering. In *ICML*. 791–798.

Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as Treatments: Debiasing Learning and Evaluation. *CoRR* abs/1602.05352 (2016).

D. Sculley. 2009. Large scale learning to rank. In *NIPS 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 Workshop on Deep Learning for Action and Interaction*.

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

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *CIKM*. ACM, 101–110.

References IX

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*.

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 *ACL 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*.

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

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 *ACL RANLP*, Vol. 5. 237–248.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. In *Arxiv*. `https://arxiv.org/abs/1609.03499`

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.

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, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 6000–6010.

References X

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. In *ICML Workshop on Deep Learning*.

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

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016a. A Deep Architecture for Semantic Matching with Multiple Positional Sentence Representations.. In *AAAI*.

Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. 2016b. Match-SRNN: Modeling the Recursive Matching Structure with Spatial RNN. In *IJCAI*. AAAI Press, 2922–2928.

Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models. In *SIGIR*. ACM, 515–524.

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

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 Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid Code Networks: Practical and Efficient End-To-End Dialog Control With Supervised and Reinforcement Learning. In *ACL*.

Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *WSDM*. 153–162.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *ICML*. ACM, 1192–1199.

Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. 2017a. Word-Entity Duet Representations for Document Ranking. In *SIGIR*.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017b. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *SIGIR*. ACM, 55–64.

Bo Xu, Hongfei Lin, Yuan Lin, and Kan Xu. 2017. Learning to Rank with Query-level Semi-supervised Autoencoders. In *CIKM*. ACM, 2395–2398.

References XI

Liu Yang, Qingyao Ai, Jiafeng Guo, and W Bruce Croft. 2016. aNMM: Ranking short answer texts with attention-based neural matching model. In *CIKM*. ACM, 287–296.

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.

Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Dawei Yin, Yihong Zhao, and Jiliang Tang. 2017. Deep Reinforcement Learning for List-wise Recommendations. *CoRR* abs/1801.00209 (2017).

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.