Deep Approaches to Semantic Matching for Text

Jun Xu
Institute of Computing Technology, Chinese Academy of Sciences
junxu@ict.ac.cn
Outline

❖ Problems with direct methods
❖ Deep matching models for text
  ❖ Composition focused methods
  ❖ Interaction focused methods
❖ Summary
Problems with direct methods

[Problem 1] *The order information of words is missing*

Bag of words assumption:

\[ \text{hot dog} = \text{dog hot} \]

However:

\[ \text{hot dog} \neq \text{dog hot} \]
The importance of the words order

- Assume that comprehension vocabulary is 100,000 words, that sentences are 20 words long, and that word order is important only within sentences.

- Then the contributions, in bits are $\log_2(100000^{20})$ and $\log_2(20!)$ respectively, which works out to over 80% of the potential information in language being in the choice of words without regard to the order in which they appear.

Problems with direct methods

[Problem 2] *Over simplified sentence representation*

"The cat sat on the yellow mat = The yellow cat sat on the mat" under bag-of-words assumption
Problems with direct methods

[Problem 3] *Heuristic matching function*

- A vector for representing the whole sentence
- Based on distance measures between two vectors
  - Cosine, Euclidean distance …

Limited information of two vectors are taken into consideration
How to design deep semantic matching models for text?
Keeping order information

- A sequence of word embeddings
  - Convert each word to its embedding (e.g., word2vec)
  - Concatenate embeddings to a sequence

![Diagram showing conversion from Bag of Word Embeddings to Sequence of Word Embeddings](image.png)
Rich sentence representation

- Hierarchical structure of sentence representation, e.g., different levels of embeddings
Powerful matching function

- Considering different levels/types of matching signals

Noodles and dumplings were famous Chinese food.

Down the ages, dumplings and noodles were popular in China.

N-gram  N-term  N-term

Learning the matching function

- Data-driven approaches to determining the parameters

Matching Score

Learning to composite

+0.8

+1.0

+0.5

+0.1

Keyword Matching Signal

N-gram Matching Signal

N-term Matching Signal

Semantic Matching Signal

Python

Hot Dot

work hard

learn

study

Python

Hot Dot

hard working

...
Outline

- Problems with direct methods
- Deep matching models for text
  - Composition focused
  - Interaction focused
- Summary
Existing deep text matching models

- **Composition focused methods**
  - [Problem 1: order] [Problem 2: structure]
  - Composite each sentence into one embedding
  - Measure the similarity between the two embeddings

- **Interaction focused methods**
  - [Problem 1: order] [Problem 3: matching function]
  - Two sentences meet before their own high-level representations mature
  - Capture complex matching patterns
Composition Focused Methods
Composition focused methods

- Step 1: Composite sentence representation $\phi(x)$
- Step 2: Matching between the representations $F(\phi(x), \phi(y))$
Composition focused methods will be discussed

- Based on DNN
  - **DSSM**: Learning Deep Structured Semantic Models for Web Search using Click-through Data (Huang et al., CIKM ’13)

- Based on CNN
  - **CDSSM**: A latent semantic model with convolutional-pooling structure for information retrieval (Shen Y et al., CIKM ’14)
  - **ARC I**: Convolutional Neural Network Architectures for Matching Natural Language Sentences (Hu et al., NIPS ’14)
  - **CNTN**: Convolutional neural tensor network architecture for community-based question Answering (Qiu et al., IJCAI ’15)

- Based on RNN
  - **LSTM-RNN**: Deep Sentence Embedding Using the Long Short Term Memory Network: Analysis and Application to Information Retrieval (Palangi et al., TASLP ’16)
Deep structured semantic model (DSSM)

DSSM input: letter-trigram

- Bag of words representation
  - “candy store”: [0 0 0 1 0 0 0 1 0 0 0 ... ]

- Letter-trigram representation
  - “#candy# #store#” → #ca | can | and | ndy | dy# | #st | sto | tor | ore | re#
  - [0 0 1 0 0 ... 0 1 0 1 ... 0 0 ... ]

- Advantages:
  - Compact representation: # words: 500K → # letter-trigram: 30K
  - Generalize to unseen words
  - Robust to noisy inputs, e.g., misspelling, inflection ...
DSSM sentence representation: DNN

Model: DNN for capturing the compositional sentence representation
DSSM matching function

- Cosine similarity between semantic vectors
  \[ S = \frac{x^T \cdot y}{|x| \cdot |y|} \]

- Training
  - A query \( q \) and a list of docs \( D = \{d^+, d_1^-, \ldots, d_k^-\} \)
  - \( d^+ \) relevant doc, \( d_1^- , \ldots, d_k^- \) irrelevant docs
  - Objective: \( P(d^+|q) = \frac{\exp(\gamma \cos(q, d^+))}{\sum_{d \in D} \exp(\gamma \cos(q, d))} \)
  - Optimizing with SGD
DSSM: short summary

- Input: sub-word units (i.e. letter-trigram) as input for scalability and generalizability
- Representation: mapping sentences to vectors (i.e. DNN): semantically similar sentences close to each other
- Matching: cosine similarity as the matching function
- Problem: bag of letter-trigrams as inputs, the order information of words ignored
Capturing the order information

❖ Input: **word sequence** rather than bag of letter-trigrams

❖ Model:

❖ **Convolutional** based methods can keep **locally order**

❖ **Recurrent** based methods can keep **long dependence relations**
CNN can model the order information

- Inspired by the cat’s visual cortex [Hubel68].

- Convolution & max pooling operations on text
RNN can model the order information

- RNNs implement dynamical systems
- RNNs can approximate arbitrary dynamical systems with arbitrary precision
- Training: back propagation through time
  \[ s(t) = f(Uw(t) + Ws(t - 1) + b) \]
- Two popularly used variations: long-short term memory (LSTM) and gated recurrent unit (GRU)
Using CNN: CDSSM

- Input: encode each word as bag of letter-trigram
- Model: the convolutional operation in CNN compacts each sequence of k words

Semantic layer: $y$
Affine projection matrix: $W_z$
Max pooling layer: $v$
Max pooling operation
Convolutional layer: $h_t$
Convolution matrix: $W_c$
Word hashing layer: $f_t$
Word hashing matrix: $W_f$
Word sequence: $x_t$

Using CNN: ARC-I / CNTN

- Input: sequence of word embeddings
  - Word embeddings from word2vec model trained on large dataset
- Model: CNN compacts each sequence of k words
Using RNN: LSTM-RNN

- Input: sequence letter trigrams
- Model: long-short term memory (LSTM)
  - The last output as the sentence representation

Matching functions

Heuristic: cosine, dot product
Learning: MLP, Neural tensor networks
Matching functions (cont’)

❖ Given the representations of two sentences: \( x \) and \( y \).

❖ Similarity between these two embeddings:
  ❖ Cosine Similarity (DSSM, CDSSM, RNN-LSTM)
    \[
    S = \frac{x^T \cdot y}{|x| \cdot |y|}
    \]
  ❖ Dot Product
    \[
    S = x^T \cdot y
    \]
  ❖ Multi-Layer Perception (ARC-I)
    \[
    S = W_2 \cdot \left( W_1 \cdot \left[ x \right]_y + b_1 \right) + b_2
    \]
Matching functions (cont’) 

- Neural Tensor Network (CNTN)

\[ S = u^T f(x^T M^{[1:r]} y + V \left[ \begin{array}{c} x \\ y \end{array} \right] + b) \]
Performance evaluation based on QA task

- Dataset: Yahoo! Answers
- Contain 60,564 (question, answer) pairs

Example:
- Q: How to get rid of memory stick error of my sony cyber shot?
- A: You might want to try to format the memory stick but what is the error message you are receiving.
## Experimental results

<table>
<thead>
<tr>
<th>Model</th>
<th>P@1</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>0.200</td>
<td>0.457</td>
</tr>
<tr>
<td>BM25</td>
<td>0.579</td>
<td>0.726</td>
</tr>
<tr>
<td>ARC-I</td>
<td>0.581</td>
<td>0.756</td>
</tr>
<tr>
<td>CNTN</td>
<td>0.626</td>
<td>0.781</td>
</tr>
<tr>
<td>LSTM-RNN</td>
<td>0.690</td>
<td>0.822</td>
</tr>
</tbody>
</table>

- Composition focused methods outperformed the baselines
- Semantic representation is important
- LSTM-RNN is the best performed method
- Modeling the order information does help
Extensions to composition focused methods

- Problem: sentence representations are too coarse to conduct exact text matching tasks
  - Experience in IR: combining topic level and word level matching signals usually achieve better performances

- Add fine-grained matching signals in composition focused methods

  - **MultiGranCNN**: An Architecture for General Matching of Text Chunks on Multiple Levels of Granularity. (Yin W, Schütze T, Hinrich. ACL2015)
  - **U-RAE**: Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection, (Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, Christopher D. Manning, NIPS2011)
# Performance evaluation on QA task

<table>
<thead>
<tr>
<th>Model</th>
<th>P@1</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Statistic</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>0.200</td>
<td>0.457</td>
</tr>
<tr>
<td>BM25</td>
<td>0.579</td>
<td>0.726</td>
</tr>
<tr>
<td><strong>Traditional</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARC-I</td>
<td>0.581</td>
<td>0.756</td>
</tr>
<tr>
<td>CNTN</td>
<td>0.626</td>
<td>0.781</td>
</tr>
<tr>
<td><strong>Composition</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSTM-RNN</td>
<td>0.690</td>
<td>0.822</td>
</tr>
<tr>
<td>uRAE</td>
<td>0.398</td>
<td>0.652</td>
</tr>
<tr>
<td><strong>Focused</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MultiGranCNN</td>
<td>0.725</td>
<td>0.840</td>
</tr>
<tr>
<td>MV-LSTM</td>
<td>0.766</td>
<td>0.869</td>
</tr>
</tbody>
</table>

- MultiGranCNN and MV-LSTM achieved the best performance
- Fine-grained matching signals are useful
Outline

- Problems with direct methods
- Deep matching models for text
  - Composition focused
  - Interaction focused
- Summary
Interaction focused methods

- Step 1: Construct basic low-level interaction signals
- Step 2: Aggregate matching patterns

![Diagram showing input, interaction, matching pattern, and similarity aggregation.](image)
Interaction focused methods will be discussed

- **ARC II**: Convolutional Neural Network Architectures for Matching Natural Language Sentences (Hu et al., NIPS’14)
- **MatchPyramid**: Text Matching as Image Recognition. (Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. AAAI 2016)
- **Match-SRNN**: Modeling the Recursive Matching Structure with Spatial RNN. (Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. IJCAI 2016)
ARC-II

- Let two sentences meet before their own high-level representations mature
- Basic interaction: phrase sum interaction matrix
- Compositional interaction: CNN to capture the local interaction structure
- Aggregation: MLP

ARC-II (cont’)

- Order preservation
  - Both the convolution and pooling have order preserving property

- However, the **word level matching signals are lost**
  - 2-D matching matrix is construct based on the embedding of the words in two N-grams
MatchPyramid

- Inspired by image recognition task
- Basic interaction: word-level matching matrix
- Compositional interaction: hierarchical convolution
- Aggregation: MLP

MatchPyramid: the matching matrix

- Basic interaction: word similarity matrix
- Strength of the word-level matching
- Positions of the matching occurs

\[ M_{ij} = w_i \otimes v_j \]

(a) Indicator  (b) Cosine

Instance 1 Instance 2 Instance 3

ARC-II
Indicator
Dot Product
MatchPyramid: the hierarchical convolution

- Compositional interaction: CNN constructs different levels of matching patterns, based on word-level matching signals.
Match-SRNN

- Spatial recurrent neural network (SRNN) for text matching
- Basic interaction: word similarity tensor
- Compositional interaction: recursive matching
- Aggregation: MLP

Match-SRNN: recursive matching structure

- Matching scores are calculated recursively (from top left to bottom right)
- All matchings between sub sentences have been utilized
  - sat $\leftrightarrow$ balls
  - The cat $\leftrightarrow$ the dog played
  - The cat $\leftrightarrow$ The dog played balls
  - The cat sat $\leftrightarrow$ The dog played

S1[1:2] $\text{The cat sat}$ on the mat.

S1[1:3] $\text{The dog played}$ balls on the floor.

S2[1:3] $\text{The dog played}$ balls on the floor.

S2[1:4] $\text{The dog played}$ balls on the floor.
Using spatial GRU (two dimensions)

Softmax function is used to select connections among these four choices softly.

\[ q^T = [h_{i-1,j}^T, h_{i,j-1}^T, h_{i-1,j-1}^T, s_{ij}^T]^T, \]
\[ r_l = \sigma(W^{(r)} q + b^{(r)}), \]
\[ r_t = \sigma(W^{(r_t)} q + b^{(r_t)}), \]
\[ r_d = \sigma(W^{(r_d)} q + b^{(r_d)}), \]
\[ r^T = [r_l^T, r_t^T, r_d^T]^T, \]
\[ z_i' = W(z_i) q + b, \]
\[ z_l' = W(z_l) q + b, \]
\[ z_t' = W(z_t) q + b, \]
\[ z_d' = W(z_d) q + b, \]
\[ [z_i, z_l, z_t, z_d] = \text{SoftmaxByRow}([z_i', z_l', z_t', z_d']), \]
\[ h_{i,j}' = \phi(Ws_{ij} + U(r \odot [h_{i,j-1}^T, h_{i-1,j}^T, h_{i-1,j-1}^T]) + b), \]
\[ h_{i,j} = z_l \odot h_{i,j-1} + z_t \odot h_{i-1,j} + z_d \odot h_{i-1,j-1} + z_i \odot h_{i,j}. \]
Connection to LCS

- Longest common sub-sequence (LCS)
- S1: A B C D E
- S2: F A C G D
- LCS: A C D

- Solving LCS with dynamic programming (DP)
  - Step function: \( c[i, j] = \max(c[i, j-1], c[i-1, j], c[i-1, j-1] + \mathbb{I}_{x_i = y_j}) \)
  - Backtrace: depends on the selection of “max” operation
Connection to LCS

- Match-SRNN can be explained with LCS
- Simplified Match-SRNN
  - Only exact word-level matching signals
  - Remove the reset gate $r$ and set hidden dimension to 1
    $$ h_{ij} = z_l \cdot h_{i,j-1} + z_t \cdot h_{i-1,j} + z_d \cdot h_{i-1,j-1} + z_i \cdot h'_{ij} $$
- Simplified Match-SRNN simulates LCS
  $$ c[i, j] = \max(c[i, j-1], c[i-1, j], c[i-1, j-1] + \mathbb{I}_{x_i=y_j}) $$
- Since that $z$ is obtained by SOFTMAX
- Backtrace by the value of $z$ in simplified Match-SRNN
Simulation

- Simulation data
  - Random sampled sequence
  - Ground truth obtained by DP
  - The label is the length of LCS

Match-SRNN simulates LCS well!
Question: “How to get rid of memory stick error of my sony cyber shot?”

Answer: “You might want to try to format the memory stick but what is the error message you are receiving.”
Performance evaluations on QA task

<table>
<thead>
<tr>
<th>Model</th>
<th>P@1</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistic traditional</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>0.200</td>
<td>0.457</td>
</tr>
<tr>
<td>BM25</td>
<td>0.579</td>
<td>0.726</td>
</tr>
<tr>
<td>Composition focused</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARC-I</td>
<td>0.581</td>
<td>0.756</td>
</tr>
<tr>
<td>CNTN</td>
<td>0.626</td>
<td>0.781</td>
</tr>
<tr>
<td>LSTM-RNN</td>
<td>0.690</td>
<td>0.822</td>
</tr>
<tr>
<td>uRAE</td>
<td>0.398</td>
<td>0.652</td>
</tr>
<tr>
<td>MultiGranCNN</td>
<td>0.725</td>
<td>0.840</td>
</tr>
<tr>
<td>MV-LSTM</td>
<td>0.766</td>
<td>0.869</td>
</tr>
<tr>
<td>Interaction focused</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeepMatch</td>
<td>0.452</td>
<td>0.679</td>
</tr>
<tr>
<td>ARC-II</td>
<td>0.591</td>
<td>0.765</td>
</tr>
<tr>
<td>MatchPyramid</td>
<td>0.764</td>
<td>0.867</td>
</tr>
<tr>
<td>Match-SRNN</td>
<td>0.790</td>
<td>0.882</td>
</tr>
</tbody>
</table>

- Interaction focused methods outperformed the composition focused ones
- Low level interaction (word level) signals are also important
- Match-SRNN performs the best
- Powerful recursive matching structure
Outline

- Problems with direct methods
- Deep matching models for text
  - Composition focused
  - Interaction focused
- Summary
Summary

- **Order of words**

- **Structure of sentence**

- **Matching function** \[ S = \frac{x^T \cdot y}{|x| \cdot |y|} \]
Summary (cont’)

- Composition focused

- Interaction focused
Challenges

❖ Data: building benchmarks
  ❖ Current: lack of large scale text matching data
  ❖ Deep learning models have a lot of parameters to learn
❖ Model: leveraging human knowledge
  ❖ Current: most models are purely data-driven
  ❖ Prior information (e.g., large scale knowledge base and other information) should be helpful
❖ Application: domain specific matching models
  ❖ Current: matching models are designed for a general goal (similarity)
  ❖ Different applications have different matching goal
  ❖ For example, in IR, relevance ≠ similarity
Thanks!