Deep Learning for Selected Natural Language Applications

Xiaodong He
Microsoft Research, Redmond, WA

Background for deep learning

Machine learning

Data → Statistics → Programs

Deep learning

Machine learning
First train a stack of N models each of which has one hidden layer. Each model in the stack treats the hidden variables of the previous model as data.

Then compose them into a single Deep Belief Network.

Then add outputs and train the DNN with backprop.

DNN: (Fully-Connected) Deep Neural Networks

Hinton, Deng, Yu, etc., DNN for AM in speech recognition, *IEEE SPM*, 2012
The Universal Translator ... *comes true!*

**Scientists See Promise in Deep-Learning Programs**

*John Markoff*

November 23, 2012

**Rick Rashid** in Tianjin, China, October, 25, 2012

Deep learning technology enabled speech-to-speech translation

A voice recognition program translated a speech given by Richard F. Rashid, Microsoft’s top scientist, into Mandarin Chinese.
Impact of deep learning in speech technology
After no improvement for 10+ years by the research community...
...MSR reduced error from ~23% to <13% (and under 7% for Rick Rashid’s S2S demo)!

CD-DNN-HMM


Progress of spontaneous speech recognition

little progress for 10+ yrs

MSR

Rashid Demo
Deep Convolutional NN for Images

**CNN**: local connections with weight sharing; pooling for translation invariance

Yann LeCun

LeCun et al., 1998
A Basic Module of the CNN

Pooling

Convolution

Image
Deep Convolutional NN for Images

2012-2014

A paradigm shift!

earlier

SVM

Pooling

Histogram Oriented Grads

Image

Fully connected

Convolution/pooling

Convolution/pooling

Convolution/pooling

Convolution/pooling

Fully connected

Convolution/pooling

Convolution/pooling

Convolution/pooling

Raw Image pixels
ImageNet 1K Competition


![Graph showing progress of object recognition (1k ImageNet)](image)

**Fall 2012**

Top-5 classification error rate

- LEAR-YRCE
- U. of Amsterdam
- YRCE/INRIA
- Oxford
- ISL
- Supervision

Deep CNN !!!
Univ. Toronto team

2012 - 2014
Deep learning demonstrates great success in speech and image!

Is Deep Learning, the 'holy grail' of big data? - CNBC - Video
video.cnbc.com/gallery/?video=3000192292
Aug 22, 2013
Derrick Harris, GigaOM, explains how "Deep Learning" computers are able to process and understand ...
How about natural language ...
Neural network based language model

LM: predict the next word given the past:
e.g., $p(\text{chases}|\text{the cat}) = \?$, $p(\text{says}|\text{the cat}) = \?$

Recurrent NN based language model

Mikolov, Karafiat, Burget, Cernocky, Khudanpur, “Recurrent neural network based language model.” Interspeech, 2010

Tomas Mikolov

- Large LM perplexity reduction
- Lower ASR WER improvement
- Expensive in learning
- Later turned to FFNN at Google: Word2vec, Skip-gram, etc.
- All UNSUPERVISED

Table 1: Performance of models on WSJ DEV set when increasing size of training data.

<table>
<thead>
<tr>
<th>Model</th>
<th># words</th>
<th>PPL</th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>KN5 LM</td>
<td>200K</td>
<td>336</td>
<td>16.4</td>
</tr>
<tr>
<td>KN5 LM + RNN 90/2</td>
<td>200K</td>
<td>271</td>
<td>15.4</td>
</tr>
<tr>
<td>KN5 LM</td>
<td>1M</td>
<td>287</td>
<td>15.1</td>
</tr>
<tr>
<td>KN5 LM + RNN 90/2</td>
<td>1M</td>
<td>225</td>
<td>14.0</td>
</tr>
<tr>
<td>KN5 LM</td>
<td>6.4M</td>
<td>221</td>
<td>13.5</td>
</tr>
<tr>
<td>KN5 LM + RNN 250/5</td>
<td>6.4M</td>
<td>156</td>
<td>11.7</td>
</tr>
</tbody>
</table>
Deep learning for spoken language processing

The scenarios
- Domain & intent classification
- Semantic slot filling

“Show me flights from Boston to New York today”

**Domain**: travel

**Intent**: find_flight

**Semantic slots**: City-departure, City-arrival, Date
Deep stack net for domain & intent classification:

1) A stack of a series of 3-layer perceptron modules
2) Output layer is concatenated with raw input to form input layer of the next module

"Show me flights from Boston to New York today"

Domain: travel

Output domain

Input sentence

[Tur, Deng, Hakkani-Tur, He, 2012; Deng, Tur, He, Hakkani-Tur, 2012]
Domain classification results

Table 2. Comparisons of the domain classification error rates among the boosting-based baseline system, DCN system, and K-DCN system for a domain classification task. Three types of raw features (lexical, query clicks, and name entities) and four ways of their combinations are used for the evaluation as shown in four rows of the table.

<table>
<thead>
<tr>
<th>Feature Sets</th>
<th>Baseline</th>
<th>DCN</th>
<th>K-DCN</th>
</tr>
</thead>
<tbody>
<tr>
<td>lexical features</td>
<td>10.40%</td>
<td>10.09%</td>
<td>9.52%</td>
</tr>
<tr>
<td>lexical features + Named Entities</td>
<td>9.40%</td>
<td>9.32%</td>
<td>8.88%</td>
</tr>
<tr>
<td>lexical features + Query clicks</td>
<td>8.50%</td>
<td>7.43%</td>
<td>5.94%</td>
</tr>
<tr>
<td>lexical features + Query clicks + Named Entities</td>
<td>10.10%</td>
<td>7.26%</td>
<td>5.89%</td>
</tr>
</tbody>
</table>

30% error reduction over a boosting-based baseline!

Table 3. More detailed results of K-DCN in Table 2 with Lexical+QueryClick features. Domain classification error rates (percent) on Train set, Dev set, and Test set as a function of the depth of the K-DCN.

<table>
<thead>
<tr>
<th>Depth</th>
<th>Train Err%</th>
<th>Dev Error%</th>
<th>Test Err%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9.54%</td>
<td>12.90%</td>
<td>12.20%</td>
</tr>
<tr>
<td>2</td>
<td>6.36%</td>
<td>10.50%</td>
<td>9.99%</td>
</tr>
<tr>
<td>3</td>
<td>4.12%</td>
<td>9.25%</td>
<td>8.25%</td>
</tr>
<tr>
<td>4</td>
<td>1.39%</td>
<td>7.00%</td>
<td>7.20%</td>
</tr>
<tr>
<td>5</td>
<td>0.28%</td>
<td>6.50%</td>
<td>5.94%</td>
</tr>
<tr>
<td>6</td>
<td>0.26%</td>
<td>6.45%</td>
<td>5.94%</td>
</tr>
<tr>
<td>7</td>
<td>0.26%</td>
<td>6.55%</td>
<td>6.26%</td>
</tr>
<tr>
<td>8</td>
<td>0.27%</td>
<td>6.60%</td>
<td>6.20%</td>
</tr>
</tbody>
</table>

Error keeps decreasing until up to six layers are added up
Semantic slot filling

A example in the Airline Travel Information System (ATIS) corpus

<table>
<thead>
<tr>
<th>Slots</th>
<th>show</th>
<th>flights</th>
<th>from</th>
<th>boston</th>
<th>to</th>
<th>new</th>
<th>york</th>
<th>today</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>B-dept</td>
<td>O</td>
<td>B-arr</td>
<td>I-arr</td>
<td>B-date</td>
</tr>
</tbody>
</table>

Slot filling can be viewed as a sequential tagging problem
Recurrent neural networks for slot filling

$h_t$ is the hidden layer that carries the information from time 0~t
where $x_t$: the input word, $y_t$: the output tag
$y_t = \text{SoftMax}(U \cdot h_t)$, where $h_t = \sigma(W \cdot h_{t-1} + V \cdot x_t)$

[Mesnil, He, Deng, Bengio, 2013; Yao, Zweig, Hwang, Shi, Yu, 2013]
Back-propagation through time (BPTT)

At time $t = 3$

1. Forward propagation
2. Generate output
3. Calculate error
4. Back propagation
5. Back prop. through time
Results

- Evaluated on the ATIS corpus
  - 4978 utterances for training
  - 893 utterances for testing
  - Using word feature only
  - Baseline CRF: 92.94% in F1-measure

SGD vs. minibatch training

<table>
<thead>
<tr>
<th>Model</th>
<th>Elman</th>
<th>Jordan</th>
<th>Hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stochastic GD</td>
<td>94.55±0.51</td>
<td>94.66±0.23</td>
<td>94.75±0.31</td>
</tr>
<tr>
<td>Sentence-minibatch</td>
<td>94.54±0.23</td>
<td>94.33±0.19</td>
<td>94.25±0.28</td>
</tr>
</tbody>
</table>

~25% error reduction!

Left-to-right vs. bi-directional RNN

<table>
<thead>
<tr>
<th>Model</th>
<th>Elman</th>
<th>Jordan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left-to-right</td>
<td>94.54</td>
<td>94.33</td>
</tr>
<tr>
<td>bi-direction</td>
<td>94.73</td>
<td>94.03</td>
</tr>
</tbody>
</table>

With local context window

Without local context window

<table>
<thead>
<tr>
<th>Model</th>
<th>Elman</th>
<th>Jordan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left-to-right</td>
<td>93.15</td>
<td>65.23</td>
</tr>
<tr>
<td>bi-direction</td>
<td>93.46</td>
<td>90.31</td>
</tr>
</tbody>
</table>
However, understanding human language is more challenging than that ...
Why understanding language is difficult?

Human language has great variability
similar concepts are expressed in different ways, e.g., *kitty* vs. *cat*

Human language has great ambiguity
similar expressions mean different concepts, e.g.,
*new york* vs. *new york times*

The meaning of text is usually vague and latent
e.g., no clear “supervision” signal to learn from as in speech/image recog.

Learning semantic meaning of texts is a key challenge in NLP
Semantic embedding

Project raw text into a continuous semantic space e.g., word embedding

Captures the word meaning in a semantic space

\[ f(\text{cat}) = \text{a.k.a the 1-hot word vector} \rightarrow \text{word embedding vector in the semantic space} \]

The index of “cat” in the vocabulary

\[ Dim=100 \sim 1000 \]

\[ Dim=|V|=100K \sim 100M \]

Deerwester, Dumais, Furnas, Landauer, Harshman, "Indexing by latent semantic analysis," JASIS 1990
SENNA word embedding

Scoring:

\[ \text{Score}(w_1, w_2, w_3, w_4, w_5) = U^T \sigma(W[f_1, f_2, f_3, f_4, f_5] + b) \]

Training:

\[ J = \max(0, 1 + S^- - S^+) \]

Where

\[ S^+ = \text{Score}(w_1, w_2, w_3, w_4, w_5) \]
\[ S^- = \text{Score}(w_1, w_2, w^-, w_4, w_5) \]

And

\(<w_1, w_2, w_3, w_4, w_5>\) is a valid 5-gram
\(<w_1, w_2, w^-, w_4, w_5>\) is a “negative sample” constructed by replacing the word \(w_3\) with a random word \(w^-\)

E.g., a negative example: “cat chills X a mat”

Collobert, Weston, Bottou, Karlen, Kavukcuoglu, Kuksa, “Natural Language Processing (Almost) from Scratch,” JMLR 2011
RNN-LM base word embedding

CBOU/Skip-gram Word Embeddings

Continuous Bag-of-Words

The CBOU architecture (a) on the left, and the Skip-gram architecture (b) on the right. [Mikolov et al., 2013 ICLR].
Word Embedding: revisit

• Word embedding is a neat and effective representation:

A decomposable, robust representation is preferable for large scale NL tasks

• Vocabulary of real-world big data tasks could be huge (scalability)
  >100M unique words in a modern commercial search engine log, and keeps growing

• New words, misspellings, and word fragments frequently occur (generalizability)
Build semantic embedding on top of sub-word units

Learning semantic embedding on top of sub-word units (SWU)

- Decompose a word into sub-word units
- Reduce the problem of modeling from an almost unbounded variability (word) to a bounded variability (sub-word)

\[ W \rightarrow U \times V \]

SWU encoding matrix: \(500 \times 50K\)

SWU embedding matrix: \(500 \times 100M\)

1-hot word vector

[Huang, He, Gao, Deng, Acero, Heck, 2013]
Sub-word unit

- Letters, context-dept letters, positioned-phones, context-dept phones, positioned-roots/morphs, context-dept morphs
- Multi-hashing approach to word input representation
- Random projection
From sub-word unit vector to word vector

SWU uses context-dependent letter, e.g., letter-trigram.
Learn one vector per letter-trigram (LTG), the encoding matrix is a fixed matrix
- Use the count of each LTG in the word for encoding

Example: cat → #cat# → #c-a, c-a-t, a-t-#
(w/ word boundary mark #)

\[
v(cat) = \sum_{k=1}^{K} (\alpha_{cat,k} \cdot u_k)
\]

Count of LTG(k) in the word “cat” \( u_k \): The vector of LTG(k)

Two words has the same LTG: collision rate \( \approx 0.004\% \)
Semantic embedding: from word to phrase

The semantic intent is better defined at the phrase/sentence level rather than at the word level.

The meaning of a single word is often ambiguous.

A phrase/sentence/document contains rich contextual information that could be leveraged.
Deep learning for semantic embedding

Abstract representation in the semantic space

each non-linear layer gradually extracts deeper invariance

Raw text, e.g., a sequence of words

Input 1

Text string $s$

However

• the semantic meaning of texts – to be learned – is latent
• no clear target for the model to learn
• How to do back-propagation / training?
• Fortunately, we usually know if two texts are “similar” or not.
• That’s the training signal for us!
DSSM for semantic embedding learning

Deep structured semantic model/Deep semantic similarity model (DSSM)

the DSSM learns phrase/sentence level semantic vector representation, e.g., query, document

The DSSM is built upon sub-word units for scalability and generalizability
e.g., letter-trigram, phones, roots/morphs

The DSSM is trained by an similarity-driven objective
projecting semantically similar phrases to vectors close to each other
projecting semantically different phrases to vectors far apart

The DSSM is trained using various signals, with/without human labeling effort
semantically-similar text pairs, e.g., user behavior log data, contextual text

[Huang, He, Gao, Deng, Acero, Heck, CIKM2013]
[Shen, He, Gao, Deng, Mesnil, WWW2014]
[Gao, He, Yih, Deng, ACL2014]
[Yih, He, Meek, ACL2014]
[Gao, Pantel, Gamon, He, Deng, Shen, EMNLP2014]
[Shen, He, Gao, Deng, Mesnil, CIKM2014]
[He, Gao, Deng, ICASSP2014 Tutorial]
DSSM for semantic embedding Learning

**Initialization:**
Neural networks are initialized with random weights


Semantic vector $\mathbf{v}_s$:
- Dimension $d=300$
- Matrix $W_4$
- Dimension $d=500$
- Matrix $W_3$
- Dimension $d=500$
- Matrix $W_2$
- Dimension $d=500$
- Matrix $W_1$
- Dimension $d=500$
- Bag-of-words vector $\mathbf{v}_s$: “racing car”

Letter-trigram encoding matrix (fixed):
- Dimension $d=500$
- Letter-trigram embedding matrix
- Dimension $d=500$
- Letter-trigram encoding matrix
- Dimension $d=500$

Bag-of-words vector $\mathbf{v}_t^+$:
- Dimension $d=500$
- Matrix $W_4$
- Dimension $d=500$
- Matrix $W_3$
- Dimension $d=500$
- Matrix $W_2$
- Dimension $d=500$
- Matrix $W_1$
- Dimension $d=500$
- Input word/phrase $t^+$: “formula one”

Bag-of-words vector $\mathbf{v}_t^-$:
- Dimension $d=500$
- Matrix $W_4$
- Dimension $d=500$
- Matrix $W_3$
- Dimension $d=500$
- Matrix $W_2$
- Dimension $d=500$
- Matrix $W_1$
- Dimension $d=500$
- Input word/phrase $t^-$: “racing to me”

Initialization:
Neural networks are initialized with random weights

---

Initialization:
Neural networks are initialized with random weights


Semantic vector $\mathbf{v}_s$:
- Dimension $d=300$
- Matrix $W_4$
- Dimension $d=500$
- Matrix $W_3$
- Dimension $d=500$
- Matrix $W_2$
- Dimension $d=500$
- Matrix $W_1$
- Dimension $d=500$
- Bag-of-words vector $\mathbf{v}_s$: “racing car”

Letter-trigram encoding matrix (fixed):
- Dimension $d=500$
- Letter-trigram embedding matrix
- Dimension $d=500$
- Letter-trigram encoding matrix
- Dimension $d=500$

Bag-of-words vector $\mathbf{v}_t^+$:
- Dimension $d=500$
- Matrix $W_4$
- Dimension $d=500$
- Matrix $W_3$
- Dimension $d=500$
- Matrix $W_2$
- Dimension $d=500$
- Matrix $W_1$
- Dimension $d=500$
- Input word/phrase $t^+$: “formula one”

Bag-of-words vector $\mathbf{v}_t^-$:
- Dimension $d=500$
- Matrix $W_4$
- Dimension $d=500$
- Matrix $W_3$
- Dimension $d=500$
- Matrix $W_2$
- Dimension $d=500$
- Matrix $W_1$
- Dimension $d=500$
- Input word/phrase $t^-$: “racing to me”

---

Initialization:
Neural networks are initialized with random weights


Semantic vector $\mathbf{v}_s$:
- Dimension $d=300$
- Matrix $W_4$
- Dimension $d=500$
- Matrix $W_3$
- Dimension $d=500$
- Matrix $W_2$
- Dimension $d=500$
- Matrix $W_1$
- Dimension $d=500$
- Bag-of-words vector $\mathbf{v}_s$: “racing car”

Letter-trigram encoding matrix (fixed):
- Dimension $d=500$
- Letter-trigram embedding matrix
- Dimension $d=500$
- Letter-trigram encoding matrix
- Dimension $d=500$

Bag-of-words vector $\mathbf{v}_t^+$:
- Dimension $d=500$
- Matrix $W_4$
- Dimension $d=500$
- Matrix $W_3$
- Dimension $d=500$
- Matrix $W_2$
- Dimension $d=500$
- Matrix $W_1$
- Dimension $d=500$
- Input word/phrase $t^+$: “formula one”

Bag-of-words vector $\mathbf{v}_t^-$:
- Dimension $d=500$
- Matrix $W_4$
- Dimension $d=500$
- Matrix $W_3$
- Dimension $d=500$
- Matrix $W_2$
- Dimension $d=500$
- Matrix $W_1$
- Dimension $d=500$
- Input word/phrase $t^-$: “racing to me”

---

Initialization:
Neural networks are initialized with random weights


Semantic vector $\mathbf{v}_s$:
- Dimension $d=300$
- Matrix $W_4$
- Dimension $d=500$
- Matrix $W_3$
- Dimension $d=500$
- Matrix $W_2$
- Dimension $d=500$
- Matrix $W_1$
- Dimension $d=500$
- Bag-of-words vector $\mathbf{v}_s$: “racing car”

Letter-trigram encoding matrix (fixed):
- Dimension $d=500$
- Letter-trigram embedding matrix
- Dimension $d=500$
- Letter-trigram encoding matrix
- Dimension $d=500$

Bag-of-words vector $\mathbf{v}_t^+$:
- Dimension $d=500$
- Matrix $W_4$
- Dimension $d=500$
- Matrix $W_3$
- Dimension $d=500$
- Matrix $W_2$
- Dimension $d=500$
- Matrix $W_1$
- Dimension $d=500$
- Input word/phrase $t^+$: “formula one”

Bag-of-words vector $\mathbf{v}_t^-$:
- Dimension $d=500$
- Matrix $W_4$
- Dimension $d=500$
- Matrix $W_3$
- Dimension $d=500$
- Matrix $W_2$
- Dimension $d=500$
- Matrix $W_1$
- Dimension $d=500$
- Input word/phrase $t^-$: “racing to me”

---

Initialization:
Neural networks are initialized with random weights


Semantic vector $\mathbf{v}_s$:
- Dimension $d=300$
- Matrix $W_4$
- Dimension $d=500$
- Matrix $W_3$
- Dimension $d=500$
- Matrix $W_2$
- Dimension $d=500$
- Matrix $W_1$
- Dimension $d=500$
- Bag-of-words vector $\mathbf{v}_s$: “racing car”

Letter-trigram encoding matrix (fixed):
- Dimension $d=500$
- Letter-trigram embedding matrix
- Dimension $d=500$
- Letter-trigram encoding matrix
- Dimension $d=500$

Bag-of-words vector $\mathbf{v}_t^+$:
- Dimension $d=500$
- Matrix $W_4$
- Dimension $d=500$
- Matrix $W_3$
- Dimension $d=500$
- Matrix $W_2$
- Dimension $d=500$
- Matrix $W_1$
- Dimension $d=500$
- Input word/phrase $t^+$: “formula one”

Bag-of-words vector $\mathbf{v}_t^-$:
- Dimension $d=500$
- Matrix $W_4$
- Dimension $d=500$
- Matrix $W_3$
- Dimension $d=500$
- Matrix $W_2$
- Dimension $d=500$
- Matrix $W_1$
- Dimension $d=500$
- Input word/phrase $t^-$: “racing to me”
**Training:**

Compute Cosine similarity between semantic vectors

Compute gradients

\[
\frac{\exp(\cos(v_s, v_{t^+}))}{\sum_{t'=\{t^+, t^\}} \exp(\cos(v_s, v_{t'}))} \partial W
\]

Semantic vector

\(d = 300\)

\(W_4\)

\(d = 500\)

\(W_3\)

\(d = 500\)

\(W_2\)

\(\text{dim} = 50K\)

Letter-trigram embedding matrix

Letter-trigram encoding matrix (fixed)

Bag-of-words vector

Input word/phrase

\(s: "racing\ car"\)

\(t^+: "formula\ one"\)

\(t^-: "racing\ to\ me"\)
DSSM for Semantic Embedding Learning

Runtime:

Semantic vector

Input word/phrase

Bag-of-words vector

Letter-trigram encoding matrix (fixed)

Letter-trigram embedding matrix

W_1

W_2

W_3

W_4

v_s

v_{t1}

v_{t2}

Semantic vector

Dimension: dim = 100M

W_1

d = 500

W_2

d = 500

W_3

d = 500

W_4

d = 500

v_s

v_{t1}

v_{t2}

Similar

Apart

S: "racing car"

T_1: "formula one"

T_2: "racing to me"
Training the DSSM

Data: semantically-similar text pairs
  e.g., context <-> word in word vector learning
  query <-> doc in Web Search
  source language <-> target language in Machine Translation

Objective: cosine similarity based loss
  • Web search as an example, for each query \( Q \), there is a set of documents \( D \)
  • \( D \) can be constructed via sampling
  • Each \( D \) in \( D \) has a relevance label w.r.t. \( Q \)

\[
P(D|Q) = \frac{\exp(\gamma R(Q,D))}{\sum_{D_i \in D} \exp(\gamma R(Q,D_i))}
\]
  • \( R(Q,D) \) is cosine similarity
  • \( \text{loss}(Q,D^+) = -\log P(D^+|Q) \)
  • \( D^+ \) is the user-clicked document of query \( Q \)
Convolutional DSSM: Model Architecture

Word sequence input: capture the sequential structure in the text (in stead of using bag-of-words)

Convolutional and Max-pooling layer: identify key words/concepts in Q and D

Shen, He, Gao, Deng, Mesnil, “A latent semantic model with convolutional-pooling structure for IR,” CIKM 2014
Example: search intent identification

Query as a word sequence rather than “bag of words”

**Sliding Window input:**
- n-gram phrase (n = 3)

**Convolutional Layer h:** generate word-within-context embedding

**Max Pooling Layer v:** identify key words in a query

**Semantic Layer y**
What does the model learn at the convolutional layer?

Capture the local context dependent word sense

- Learn one embedding vector for each local context-dependent word

<table>
<thead>
<tr>
<th>Word Combination</th>
<th>Cosine Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>car body shop</td>
<td>0.698</td>
</tr>
<tr>
<td>car body kits</td>
<td>0.578</td>
</tr>
<tr>
<td>auto body repair</td>
<td>0.555</td>
</tr>
<tr>
<td>wave body language</td>
<td>0.301</td>
</tr>
<tr>
<td>calculate body fat</td>
<td>0.220</td>
</tr>
<tr>
<td>forcefield body armour</td>
<td>0.165</td>
</tr>
</tbody>
</table>

The similarity between different “body” within contexts

The embedding vector of “auto body repair”
CDSSM: What happens at the max-pooling layer?

- Aggregate *local topics* to form the *global intent*
- Identify salient words/phrase at the max-pooling layer

Words that win the most active neurons at the **max-pooling layers**: 

```
auto body repair  cost calculator software
```

Usually, those are salient words containing clear intents/topics

\[
v(i) = \max_{t=1,...,T} \{ h_t(i) \}
\]
Intent matching via convolutional DSSM

- Semantic matching of query and document

Most active neurons at the max-pooling layers of the query and document nets, respectively
sarcoidosis is a disease, a symptom is excessive amount of calcium in one's urine and blood. So medicines that increase the absorbing of calcium should be avoid. While Vitamin d is closely associated to calcium absorbing.

We observed that “sarcoidosis” in the document title and “absorbs” “excessive” and “vitamin (d)” in the query have high activations at neurons 90, 66, 79, indicating that the model knows that “sarcoidosis” share similar semantic meaning with “absorbs” “excessive” “vitamin (d)”, collectively.

Most active neurons at the max-pooling layers of the query and document nets, respectively.

CDSSM: very complex semantic matching
Deep Structured Semantic Model (DSSM) for natural language applications

<table>
<thead>
<tr>
<th>Tasks</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web search</td>
<td>search query</td>
<td>web documents</td>
</tr>
<tr>
<td>Ad selection</td>
<td>search query</td>
<td>ad keywords</td>
</tr>
<tr>
<td>Entity ranking</td>
<td>mention (highlighted)</td>
<td>entities</td>
</tr>
<tr>
<td>Recommendation</td>
<td>doc in reading</td>
<td>interesting things / other docs</td>
</tr>
<tr>
<td>Machine translation</td>
<td>sentence in language a</td>
<td>translations in language b</td>
</tr>
<tr>
<td>Knowledge-base construction</td>
<td>entity</td>
<td>entity</td>
</tr>
<tr>
<td>Question answering</td>
<td>pattern / mention</td>
<td>relation / entity</td>
</tr>
<tr>
<td>Semantic reasoning</td>
<td>context</td>
<td>word</td>
</tr>
<tr>
<td>Text/Image retrieval</td>
<td>text</td>
<td>image</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DSSM: learning words’ meaning

- Learn a word’s semantic meaning by means of its neighbors (context)
- Construct context <-> word training pair for DSSM
- Similar words with similar context -> higher cosine

- Training Condition:
  - 600K vocabulary size
  - 1B words from Wikipedia
  - 300-dimentional vector

You shall know a word by the company it keeps (J. R. Firth 1957: 11)

You shall know a word by the company it keeps (J. R. Firth 1957: 11)

[Song, He, Gao, Deng, Shen, 2014]
Plotting 3K words in 2D
Plotting 3K words in 2D
Semantic reasoning (as algebra in the semantic space)

Semantic clustering: top 3 neighbors of each word

<table>
<thead>
<tr>
<th>Word</th>
<th>Neighbor 1</th>
<th>Neighbor 2</th>
<th>Neighbor 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>king</td>
<td>earl (0.77)</td>
<td>pope (0.77)</td>
<td>lord (0.74)</td>
</tr>
<tr>
<td>woman</td>
<td>person (0.79)</td>
<td>girl (0.77)</td>
<td>man (0.76)</td>
</tr>
<tr>
<td>france</td>
<td>spain (0.94)</td>
<td>italy (0.93)</td>
<td>belgium (0.88)</td>
</tr>
<tr>
<td>rome</td>
<td>constantinople (0.81)</td>
<td>paris (0.79)</td>
<td>moscow (0.77)</td>
</tr>
<tr>
<td>winter</td>
<td>summer (0.83)</td>
<td>autumn (0.79)</td>
<td>spring (0.74)</td>
</tr>
<tr>
<td>rain</td>
<td>rainfall (0.76)</td>
<td>storm (0.73)</td>
<td>wet (0.72)</td>
</tr>
<tr>
<td>car</td>
<td>truck (0.8)</td>
<td>driver (0.73)</td>
<td>motorcycle (0.72)</td>
</tr>
</tbody>
</table>

Semantic analogy:

\[ w_1 : w_2 = w_3 : ? \Rightarrow V_? = V_3 - V_1 + V_2 \]

<table>
<thead>
<tr>
<th>Equation</th>
<th>Neighbor 1</th>
<th>Neighbor 2</th>
<th>Neighbor 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>summer : rain = winter : ?</td>
<td>snow (0.79)</td>
<td>rainfall (0.73)</td>
<td>wet (0.71)</td>
</tr>
<tr>
<td>italy : rome = france : ?</td>
<td>paris (0.78)</td>
<td>constantinople (0.74)</td>
<td>egypt (0.73)</td>
</tr>
<tr>
<td>man : eye = car : ?</td>
<td>motor (0.64)</td>
<td>brake (0.58)</td>
<td>overhead (0.58)</td>
</tr>
<tr>
<td>man : woman = king : ?</td>
<td>mary (0.70)</td>
<td>prince (0.70)</td>
<td>queen (0.68)</td>
</tr>
<tr>
<td>read : book = listen : ?</td>
<td>sequel (0.65)</td>
<td>tale (0.63)</td>
<td>song (0.60)</td>
</tr>
</tbody>
</table>
DSSM: Web Search

**how to deal with stuffy nose?**

**stuffy nose treatment**

**cold home remedies**

---

<table>
<thead>
<tr>
<th>QUERY (Q)</th>
<th>Title (T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>how to deal with stuffy nose</td>
<td>best home remedies for cold and flu</td>
</tr>
<tr>
<td>stuffy nose treatment</td>
<td>best home remedies for cold and flu</td>
</tr>
<tr>
<td>cold home remedies</td>
<td>best home remedies for cold and flu</td>
</tr>
<tr>
<td>... ...</td>
<td>... ...</td>
</tr>
<tr>
<td>go israel</td>
<td>forums goisrael community</td>
</tr>
<tr>
<td>skate at wholesale at pr</td>
<td>wholesale skates southeastern skate supply</td>
</tr>
<tr>
<td>breastfeeding nursing blister baby</td>
<td>clogged milk ducts babycenter</td>
</tr>
<tr>
<td>thank you teacher song</td>
<td>lyrics for teaching educational children's music</td>
</tr>
<tr>
<td>immigration canada lacolle</td>
<td>CBSA office detailed information</td>
</tr>
</tbody>
</table>

Mine Q-D pairs from search logs (for DSSM training)  

[Gao, He, Nie, 2010]
Evaluation Methodology

- Measurement: NDCG, t-test
- Test set:
  - 12,071 English queries sampled from 1-y log
  - 5-level relevance label for each query-doc pair
- Training data for translation models:
  - 82,834,648 query-title pairs from search log
- Baselines
  - Lexicon matching models: BM25, ULM
  - Topic models
Web search results

Convolutional DSSM is the new state-of-the-art

<table>
<thead>
<tr>
<th>Models</th>
<th>NDCG@1 (%)</th>
<th>NDCG@3 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Lexical Matching Models</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BM25</td>
<td>30.5</td>
<td>32.8</td>
</tr>
<tr>
<td>Unigram LM</td>
<td>30.4 (-0.1)</td>
<td>32.7 (-0.1)</td>
</tr>
<tr>
<td><strong>Topic Models</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLSA [Hofmann 1999]</td>
<td>30.5 (+0.0)</td>
<td>33.5 (+0.7)</td>
</tr>
<tr>
<td>BLTM [Gao et al. 2011]</td>
<td>31.6 (+1.1)</td>
<td>34.4 (+1.6)</td>
</tr>
<tr>
<td><strong>Deep Structure Semantic Model</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSSM [Huang et al. 2013]</td>
<td>32.7 (+2.2)</td>
<td>36.3 (+3.5)</td>
</tr>
<tr>
<td>C-DSSM [Shen et al. 2014]</td>
<td>34.8 (+4.3)</td>
<td>37.9 (+5.1)</td>
</tr>
</tbody>
</table>

The higher the NDCG score the better, usually 1% NDCG difference is statistically significant.

DSSM outperforms shallow models by 3~4 pt NDCG!

[Shen, He, Gao, Deng, Mesnil, CIKM2014]
DSSM: Modeling interestingness

- **Contextual entity search**
  - Given a user-highlighted text span representing an entity of interest
  - Search for supplementary document for the entity

- **Automatic highlighting**
  - Given a document a user is reading
  - Discover the concepts/entities/topics that interest the user and highlight the corresponding text span

- **Document prefetching**
  - Given a document a user is reading
  - Prefetching a document that the user will be interested in next

DSSM for contextual entity ranking

<table>
<thead>
<tr>
<th>Ranker</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM25 (mention)</td>
<td>60%</td>
</tr>
<tr>
<td>Ranker (2306 features)</td>
<td>72%</td>
</tr>
<tr>
<td>DSSM (1 feature)</td>
<td>72%</td>
</tr>
<tr>
<td>Ranker+ DSSM</td>
<td>77%</td>
</tr>
</tbody>
</table>

- DSSM beats manually crafted text features
- +5 AUC gain over full ranker
• Features
  • DSM: DSSM
  • WCAT: semantic labels (page categories) assigned by editors
  • JTT: LDA-style topic models
  • NSF: non-semantic features

• **DSSM learned features outperform the thousands of features coming from manually assigned labels (WCAT)**
DSSM in Machine Translation

- Map the phrases in source/target languages into the same, language-independent semantic space

Gao, He, Yih, Deng, “Learning continuous phrase representations for translation modeling,” ACL2014
A closer look at the mapping

- Bag-of-words representation of a phrase: $\mathbf{x}$
- Map $\mathbf{x}$ to a low-dim semantic space: $\phi(\mathbf{x}): \mathbb{R}^d \rightarrow \mathbb{R}^k$
- Mapping is performed using a neural net:
  $$y = \phi(\mathbf{x}) = \tanh \left( \mathbf{w}_2^T \tanh (\mathbf{w}_1^T \mathbf{x}) \right)$$
- Translation score as similarity between feature vectors
  $$\text{score}(f, e) \equiv \text{sim}_\theta(\mathbf{x}_f, \mathbf{x}_e) = \mathbf{y}_f^T \mathbf{y}_e$$
Using the DSSM for SMT

- Define a new translation feature:
  \[ h_{M+1}(F_i, E, \Theta) = \sum_{(f, e) \in A} \text{sim}_\Theta(x_f, x_e) \]

- Integrate into the log-linear model for SMT:
  \[
P(E|F) = \frac{1}{Z(F, E)} \exp \sum_i \lambda_i h_i(F, E) \]
  \[E^* = \arg\max_E \sum_i \lambda_i h_i(F, E)\]
Loss function: $\mathcal{L}(\theta)$

- Expected BLEU based on n-best list (He and Deng 2012)
  - $\text{xBleu}(\theta) = \sum_{E \in \text{GEN}(F_i)} P(E|F_i)\text{sBleu}(E_i, E)$
  - $P(E|F_i) = \frac{\exp(\lambda^T h(F_i,E,A) + \lambda_{M+1} h_{M+1}(F_i,E,\theta))}{\sum_{E \in \text{GEN}(F_i)} \exp(\lambda^T h(F_i,E,A) + \lambda_{M+1} h_{M+1}(F_i,E,\theta))}$

- Friendly to optimizer?
  - Differentiable but non-convex

- Aiming the right target?
  - Closely related to BLEU
## Results

<table>
<thead>
<tr>
<th>Systems</th>
<th>WMT EN-FR</th>
<th>WMT DE-EN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TEST1</td>
<td>TEST2</td>
</tr>
<tr>
<td>Baseline</td>
<td>33.04</td>
<td>33.06</td>
</tr>
<tr>
<td>DSSM</td>
<td><strong>34.03</strong></td>
<td><strong>34.39</strong></td>
</tr>
<tr>
<td>Topic model</td>
<td>33.08</td>
<td>33.15</td>
</tr>
<tr>
<td>DPM</td>
<td>33.10</td>
<td>33.29</td>
</tr>
</tbody>
</table>

- DSSM: DSSM with xBleu
- Topic model: generative bilingual topic model (Gao et al. 2011)
- DPM: discriminative linear projection model (Gao et al. 2011)
Who is Justin Bieber’s sister?

Jazmyn Bieber

sibling-of(Justin-Bieber, Jazmyn-Bieber)
gender(Jazmyn-Bieber, female)

\( \lambda x. \text{sister-of}(\text{justin-bieber}, x) \)

Semantic parsing

Inference

Knowledge Base
Single-Relation Questions

• Challenge: lots of ways to ask the same question
  • “What was the date that Minnesota became a state?”
  • “Minnesota became a state on?”
  • “When was the state Minnesota created?”
  • “Minnesota's date it entered the union?”
  • “When was Minnesota established as a state?”
  • “What day did Minnesota officially become a state?”
  • …
DSSM in question answering

Question (in natural language)

When were DVD players invented?

CDSSM match in the relation space
Prob(R|P)

CDSSM match in the entity space
Prob(E1|M)

when were M invented
dvd players

Q

......

Decoding the best answer:
Ans* = argmax

P(Ans|KB, Q)

P(Ans|KB, Q) = \sum_{SP} P(Ans, SP|KB, Q)

≈ \max_{SP,Triple} P(Ans|SP, KB, Q)P(SP|Q)

≈ \max_{SP,Triple} \text{Prob}(R|P) \times \text{Prob}(E1|M)

\lambda x. \text{be-invent-in}(dvd-player, x)

Yih, He, Meek, “Semantic parsing for single-relation question answering,” ACL 2014
Experiments: Data

**Paralex dataset** [Fader et al., 2013]

- 1.8M (question, single-relation queries)
  
  \[
  \text{When were DVD players invented?} \\
  \lambda x. \text{be—invent—in} (\text{dvd—player}, x)
  \]

- 1.2M (relation pattern, relation)
  
  \[
  \text{When were X invented?} \\
  \text{be—invent—in}_2
  \]

- 160k (mention, entity)
  
  \[
  \text{Saint Patrick day} \\
  \text{st—patrick—day}
  \]
Experiments: Task – Question Answering

- Same test questions in the Paralex dataset
- 698 questions from 37 clusters

- **What language do people in Hong Kong use?**
  - be—speak—in *(english, hong—kong)*
  - be—predominant—language—in *(cantonese, hong—kong)*

- **Where do you find Mt Ararat?**
  - be—highest—mountain—in *(ararat, turkey)*
  - be—mountain—in *(ararat, armenia)*
Experiments: Results

![Graph showing precision and recall for Paralex and DSSM]
Go beyond text
DSSM for multi-modal representation learning

- Recall DSSM for text inputs: $s, t_1, t_2, t_3, \ldots$
- Now: replace text $s$ by image $s$
- Using DNN/CNN features of image
- Can rank/generate text’s given image or can rank images given text.

Text: *a parrot rides a tricycle*
Automatic image captioning at a human-level of performance

Evaluation: *How far are we from human?*

**Training:** 400K image/caption pairs as training data

**Testing:** 20K images, 5 annotators providing 5 captions per image

- Hold one human as the control system
- The other four annotations are gold reference for testing

<table>
<thead>
<tr>
<th>Entry</th>
<th>BLEU % (higher the better)</th>
<th>Human preference*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human (control)</td>
<td>19.3</td>
<td>(upper bound~50%)</td>
</tr>
<tr>
<td>Machine</td>
<td>21.1</td>
<td>17.3%</td>
</tr>
</tbody>
</table>

*given the machine’s output and a human annotation, the percentage that the judge prefers the machine’s output.
Other relevant work on deep learning in NLP

**Long short-term memory RNN (LSTM-RNN)**
- Capable to capture long-span dependency in natural language
- LSTM for MT (Sutskever, et al., “Sequence to sequence learning with neural networks,” to appear)

**Recursive NN (ReNN)**
- Model the hierarchical structure of nature language
- ReNN for parsing (Socher et al., “Parsing natural scenes and natural language with recursive neural networks”, 2011)

**Tensor product representation (TPR)**
- Efficient representation of the structure of natural language
- Smolensky & Legendre: The Harmonic Mind, From Neural Computation to Optimality-Theoretic Grammar, MIT Press, 2006
Summary

Great progress in deep learning breakthrough in speech, image, and language

Exciting advances in learning continuous semantic space
deep models effectively learn semantic representation vectors
  leads to superior performance in a range of NL tasks
learning image and text vectors in an joint semantic space
  facilitates exciting cross-modality scenarios
Look forward

Building an universal semantic space for all modalities
speech, vision, text, social graph ...

Building an universal intelligence space, too
knowledge, reasoning, ...

Acquiring intelligence from ambient signals automatically

Deep learning meets big data!
big capacity to digest big data
efficient computation even for small labs: one GPU machine,
10000 cores, learn a billion sentences in one day ...
Thank You

Q/A & discussions
References

- Gao, J., He, X., and Nie, J-Y. 2010. Clickthrough-based translation models for web search: from word models to phrase models. In CIKM.
References

- Huang, E., Socher, R., Manning, C., and Ng, A. 2012. Improving word representations via global context and multiple word prototypes, Proc. ACL.
- Li, P., Hastie, T., and Church, K., 2006. Very sparse random projections, in Proc. SIGKDD.
- Mikolov, T., Chen, K., Corrado, G., and Dean, J. 2013. Efficient estimation of word representations in vector space, Proc. ICLR.
References

- Salakhutdinov R., and Hinton, G., 2007 Semantic hashing. in Proc. SIGIR Workshop Information Retrieval and Applications of Graphical Models