DL for NLP
From the Basics to Research

Richard Socher
Research work joint with the MetaMind-Salesforce team
Caiming Xiong, Stephen Merity, James Bradbury,
Ankit Kumar, Ozan Irsoy and others
What is Natural Language Processing?

• Natural language processing is a field at the intersection of
  – computer science
  – artificial intelligence
  – and linguistics.

• Goal: for computers to process or “understand” natural language in order to perform tasks that are useful, e.g.
  – Question Answering

• Fully understanding and representing the meaning of language (or even defining it) is an illusive goal.

• Perfect language understanding is AI-complete

Richard Socher
NLP Levels

- Phonetic/Phonological Analysis
- OCR/Tokenization
- Morphological analysis
- Syntactic analysis
- Semantic Interpretation
- Discourse Processing
Why is NLP hard?

• Complexity in representing, learning and using linguistic/situational/world/visual knowledge

• Jane hit June and then she [fell/ran].

• Ambiguity: “I made her duck”
(A tiny sample of) NLP Applications

- Applications range from simple to complex:
  - Spell checking, keyword search, finding synonyms
  - Extracting information from websites such as
    - product price, dates, location, people or company names
  - Classifying, reading level of school texts, positive/negative sentiment of longer documents
  - Machine translation
  - Question answering or automated email replies
  - Spoken dialog systems

Richard Socher
Representations for Language Tasks: Machine Translation

- Many levels of translation have been tried in the past:

- Traditional MT systems are very large complex systems

- What do you think is the interlingua for the DL approach to translation?
- Representation for all levels: Vectors!
Outline

1. Words
   Basics: Word2vec and Glove

2. Sentences (~)
   Basics: Recurrent neural networks

3. Multiple sentences
   Research:
   Dynamic memory networks
How to represent meaning?

Common answer: Use a taxonomy like WordNet that has hypernyms (is-a) relationships and synonym sets (good):

```python
from nltk.corpus import wordnet as wn
panda = wn.synset('panda.n.01')
hyper = lambda s: s.hypernyms()
lst = pandaclosure(hyper)
```

```python
[Synset('procyonid.n.01'),
 Synset('carnivore.n.01'),
 Synset('placental.n.01'),
 Synset('mammal.n.01'),
 Synset('vertebrate.n.01'),
 Synset('chordate.n.01'),
 Synset('animal.n.01'),
 Synset('organism.n.01'),
 Synset('living_thing.n.01'),
 Synset('whole.n.02'),
 Synset('object.n.01'),
 Synset('physical_entity.n.01'),
 Synset('entity.n.01')]
```

S: (adj) full, good
S: (adj) estimable, good, honorable, respectable
S: (adj) beneficial, good
S: (adj) good, just, upright
S: (adj) adept, expert, good, practiced, proficient, skillful
S: (adj) dear, good, near
S: (adj) good, right, ripe
...
S: (adv) well, good
S: (adv) thoroughly, soundly, good
S: (n) good, goodness
S: (n) commodity, trade good, good
Problems with discrete representations

• Great as resource but missing nuances, e.g. synonyms: adept, expert, good, practiced, proficient, skillful?

• Missing new words (impossible to keep up to date): wicked, badass, nifty, crack, ace, wizard, genius, ninja

• Subjective

• Requires human labor to create and adapt

• Hard to compute accurate word similarity
Instead: Use distributional similarity

You can get a lot of value by representing a word by means of its neighbors

“You shall know a word by the company it keeps”

(J. R. Firth 1957: 11)

One of the most successful ideas of modern statistical NLP

government debt problems turning into banking crises as has happened in saying that Europe needs unified banking regulation to replace the hodgepodge

_macros: These words will represent banking

---

10
Window based cooccurrence matrix

Simple example: For each word define:

• Window length 1 (more common: 5 - 10)
• Symmetric window around each word
• Example corpus:
  • I like deep learning.
  • I like NLP.
  • I enjoy flying.
Window based cooccurrence matrix

Example corpus:

- I like deep learning.
- I like NLP.
- I enjoy flying.

<table>
<thead>
<tr>
<th>counts</th>
<th>I</th>
<th>like</th>
<th>enjoy</th>
<th>deep</th>
<th>learning</th>
<th>NLP</th>
<th>flying</th>
<th>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>like</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>enjoy</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>deep</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>learning</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NLP</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>flying</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Could run SVD on this matrix
word2vec (Mikolov et al 2013)

• Instead of capturing cooccurrence counts directly
• Predict surrounding words of every word

• Faster and can easily incorporate a new sentence/document or add a word to the vocabulary
Details of Word2Vec

- Predict surrounding words in a window of length m of every word.

- Objective function: Maximize the log probability of any context word given the current center word:

\[
J(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log p(w_{t+j} | w_t)
\]

- Where represents all variables we optimize
Details of Word2Vec

- Predict surrounding words in a window of length m of every word
- For $p(w_{t+j} | w_t)$ the simplest first formulation is

$$p(o | c) = \frac{\exp (u_o^T v_c)}{\sum_{w=1}^{W} \exp (u_w^T v_c)}$$

- where o is the outside (or output) word id, c is the center word id, v and u are “center” and “outside” vectors of indices c and o
- Every word has two vectors!
- This is essentially “dynamic” logistic regression
- For improved versions + detailed derivation, see cs224d.stanford.edu
Count based vs direct prediction

LSA, HAL (Lund & Burgess), COALS (Rohde et al), Hellinger-PCA (Lebret & Collobert)

- Fast training
- Efficient usage of statistics
- Primarily used to capture word similarity
- Disproportionate importance given to large counts

- NNLM, HLBL, RNN, Skip-gram/CBOW, (Bengio et al; Collobert & Weston; Huang et al; Mnih & Hinton; Mikolov et al; Mnih & Kavukcuoglu)

- Scales with corpus size
- Inefficient usage of statistics
- Generate improved performance on other tasks
- Can capture complex patterns beyond word similarity
Combining the best of both worlds: GloVe (Pennington et al. 2014)

\[ J(\theta) = \frac{1}{2} \sum_{i,j=1}^{W} f(P_{ij})(u_i^T v_j - \log P_{ij})^2 \]

- Fast training
- Scalable to huge corpora
- Good performance even with small corpus, and small vectors

Richard Socher
Glove results

Nearest words to frog:
1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

Richard Socher
Intrinsic word vector evaluation

• Word Vector Analogies

\[ d = \arg \max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||} \]

man:woman :: king:?  

• Evaluate word vectors by how well their cosine distance after addition captures intuitive semantic and syntactic analogy questions
• Discarding the input words from the search!
• Problem: What if the information is there but not linear?
Glove Visualizations
Glove Visualizations: Superlatives
Analogy evaluation and hyperparameters

- More data is better

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ivLBL</td>
<td>100</td>
<td>1.5B</td>
<td>55.9</td>
<td>50.1</td>
<td>53.2</td>
</tr>
<tr>
<td>HPCA</td>
<td>100</td>
<td>1.6B</td>
<td>4.2</td>
<td>16.4</td>
<td>10.8</td>
</tr>
<tr>
<td>GloVe</td>
<td>100</td>
<td>1.6B</td>
<td>67.5</td>
<td>54.3</td>
<td>60.3</td>
</tr>
<tr>
<td>SG</td>
<td>300</td>
<td>1B</td>
<td>61</td>
<td>61</td>
<td>61</td>
</tr>
<tr>
<td>CBOB</td>
<td>300</td>
<td>1.6B</td>
<td>16.1</td>
<td>52.6</td>
<td>36.1</td>
</tr>
<tr>
<td>vLBL</td>
<td>300</td>
<td>1.5B</td>
<td>54.2</td>
<td>64.8</td>
<td>60.0</td>
</tr>
<tr>
<td>ivLBL</td>
<td>300</td>
<td>1.5B</td>
<td>65.2</td>
<td>63.0</td>
<td>64.0</td>
</tr>
<tr>
<td>GloVe</td>
<td>300</td>
<td>1.6B</td>
<td>80.8</td>
<td>61.5</td>
<td>70.3</td>
</tr>
<tr>
<td>SVD</td>
<td>300</td>
<td>6B</td>
<td>6.3</td>
<td>8.1</td>
<td>7.3</td>
</tr>
<tr>
<td>SVD-S</td>
<td>300</td>
<td>6B</td>
<td>36.7</td>
<td>46.6</td>
<td>42.1</td>
</tr>
<tr>
<td>SVD-L</td>
<td>300</td>
<td>6B</td>
<td>56.6</td>
<td>63.0</td>
<td>60.1</td>
</tr>
<tr>
<td>CBOB†</td>
<td>300</td>
<td>6B</td>
<td>63.6</td>
<td>67.4</td>
<td>65.7</td>
</tr>
<tr>
<td>SG†</td>
<td>300</td>
<td>6B</td>
<td>73.0</td>
<td>66.0</td>
<td>69.1</td>
</tr>
<tr>
<td>GloVe†</td>
<td>300</td>
<td>6B</td>
<td>77.4</td>
<td>67.0</td>
<td>71.7</td>
</tr>
<tr>
<td>CBOB</td>
<td>1000</td>
<td>6B</td>
<td>57.3</td>
<td>68.9</td>
<td>63.7</td>
</tr>
<tr>
<td>SG</td>
<td>1000</td>
<td>6B</td>
<td>66.1</td>
<td>65.1</td>
<td>65.6</td>
</tr>
<tr>
<td>SVD-L</td>
<td>300</td>
<td>42B</td>
<td>38.4</td>
<td>58.2</td>
<td>49.2</td>
</tr>
<tr>
<td>GloVe</td>
<td>300</td>
<td>42B</td>
<td><strong>81.9</strong></td>
<td><strong>69.3</strong></td>
<td><strong>75.0</strong></td>
</tr>
</tbody>
</table>
Section 2: Recurrent neural networks
Recurrent Neural Networks (!)

- Similar to normal neural networks
- RNNs tie the weights at each time step
- Condition the neural network on all previous words
RNN language model

Given list of word vectors: \( x_1, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_T \)

At each time step, predict the next word:

\[
\begin{align*}
    h_t &= \sigma \left( W^{(hh)} h_{t-1} + W^{(hx)} x_t \right) \\
    \hat{y}_t &= \text{softmax} \left( W^{(S)} h_t \right)
\end{align*}
\]

\[
\hat{P}(x_{t+1} = v_j \mid x_t, \ldots, x_1) = \hat{y}_{t,j}
\]
RNN language model

Same set of $W$ weights at all time steps!

Everything else is the same:

\[
\begin{align*}
    h_t &= \sigma \left( W^{(hh)} h_{t-1} + W^{(hx)} x_t \right) \\
    \hat{y}_t &= \text{softmax} \left( W^{(S)} h_t \right) \\
    \hat{P}(x_{t+1} = v_j \mid x_t, \ldots, x_1) &= \hat{y}_{t,j}
\end{align*}
\]

$h_0 \in \mathbb{R}^{D_h}$ is initialization vector for hidden layer at first step

The next word is the “class.” Performance measured in Perplexity: $2^J$ where:

\[
J = - \frac{1}{T} \sum_{t=1}^{T} \sum_{j=1}^{\lvert V \rvert} y_{t,j} \log \hat{y}_{t,j}
\]
Main RNN improvement: Better Units

• More complex hidden unit computation in recurrence!

• Gated Recurrent Units (GRU) introduced by Cho et al. 2014. Special case of an LSTM Hochreiter and Schmidhuber 1997

• Main ideas:
  • keep around memories to capture long distance dependencies
  • allow error messages to flow at different strengths depending on the inputs
GRUs

- Standard RNN computes hidden layer at next time step directly:
  \[ h_t = f \left( W^{(hh)} h_{t-1} + W^{(hx)} x_t \right) \]

- GRU first computes an update gate (another layer) based on current input word vector and hidden state
  \[ z_t = \sigma \left( W^{(z)} x_t + U^{(z)} h_{t-1} \right) \]

- Compute reset gate similarly but with different weights
  \[ r_t = \sigma \left( W^{(r)} x_t + U^{(r)} h_{t-1} \right) \]
GRUs

• Update gate
  \[ z_t = \sigma \left( W^{(z)} x_t + U^{(z)} h_{t-1} \right) \]

• Reset gate
  \[ r_t = \sigma \left( W^{(r)} x_t + U^{(r)} h_{t-1} \right) \]

• New memory content:
  \[ \tilde{h}_t = \tanh \left( W x_t + r_t \circ U h_{t-1} \right) \]
  If reset gate unit is \( \sim 0 \), then this ignores previous memory and only stores the new word information.

• Final memory at time step combines current and previous time steps:
  \[ h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t \]
Attempt at a clean illustration

\[ z_t = \sigma \left( W^{(z)} x_t + U^{(z)} h_{t-1} \right) \]
\[ r_t = \sigma \left( W^{(r)} x_t + U^{(r)} h_{t-1} \right) \]
\[ \tilde{h}_t = \tanh \left( W x_t + r_t \circ U h_{t-1} \right) \]
\[ h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t \]
GRU intuition

• If reset is close to 0, ignore previous hidden state → Allows model to drop information that is irrelevant in the future

• Update gate \( z \) controls how much of past state should matter now.
  - If \( z \) close to 1, then we can copy information in that unit through many time steps! **Less vanishing gradient**!

• Units with short-term dependencies often have reset gates very active

\[
\begin{align*}
    z_t &= \sigma \left( W^{(z)} x_t + U^{(z)} h_{t-1} \right) \\
    r_t &= \sigma \left( W^{(r)} x_t + U^{(r)} h_{t-1} \right) \\
    \tilde{h}_t &= \tanh \left( W x_t + r_t \circ U h_{t-1} \right) \\
    h_t &= z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t
\end{align*}
\]
More amazing RNN work

• In Quoc’s lecture
Basic lego blocks

• Word vectors and RNNs are the two most important concepts for deepNLP

• Congrats!

• Now we can play with these lego blocks
Problem with all models so far

• Can only predict frequently seen classes
• Example: Language modeling where classes=words

• New words occur all the time during testing

• Solution: Combine softmax with pointers to context words!
• Work by Stephen Merity et al. 2016 (Released next week : )
Pointer sentinel mixture models
Language Model Evaluation

• Perplexity: $2^J$ where:
  $$J = - \frac{1}{T} \sum_{t=1}^{T} \sum_{j=1}^{|V|} y_{t,j} \log \hat{y}_{t,j}$$
  • Lower is better

• Results with normal RNNs plus count-based models

• Mikolov 2010

<table>
<thead>
<tr>
<th>Model</th>
<th>PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>GT3</td>
<td>165.2</td>
</tr>
<tr>
<td>KN5</td>
<td>147.8</td>
</tr>
<tr>
<td>KN5+cache</td>
<td>133.1</td>
</tr>
<tr>
<td>Structured LM (Chelba)</td>
<td>148.9</td>
</tr>
<tr>
<td>Structured LM (Roark)</td>
<td>137.2</td>
</tr>
<tr>
<td>Structured LM (Filimonov)</td>
<td>127.2</td>
</tr>
<tr>
<td>Random Forest (Peng Xu)</td>
<td>131.9</td>
</tr>
<tr>
<td>PAQ8010t</td>
<td>131.1</td>
</tr>
<tr>
<td>Syntactic NN (Emami, baseline KN4 141)</td>
<td>107</td>
</tr>
<tr>
<td>8xRNN static</td>
<td>105.4</td>
</tr>
<tr>
<td>8xRNN dynamic</td>
<td>104.5</td>
</tr>
<tr>
<td>static+dynamic</td>
<td>97.4</td>
</tr>
<tr>
<td>+KN5</td>
<td>93.9</td>
</tr>
<tr>
<td>+KN5(cache)</td>
<td>90.4</td>
</tr>
<tr>
<td>+Random forest (Peng Xu)</td>
<td>87.9</td>
</tr>
<tr>
<td>+Structured LM (Filimonov)</td>
<td>87.7</td>
</tr>
</tbody>
</table>
Lots of progress in last years

From 87 perplexity with 8 RNNs ensemble plus count-based methods to 70.9 with single end-to-end trainable neural model

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zaremba et al. (2014) - LSTM (medium)</td>
<td>20M</td>
<td>86.2</td>
<td>82.7</td>
</tr>
<tr>
<td>Zaremba et al. (2014) - LSTM (large)</td>
<td>66M</td>
<td>82.2</td>
<td>78.4</td>
</tr>
<tr>
<td>Gal (2015) - Variational LSTM (medium, untied)</td>
<td>20M</td>
<td>81.9 ± 0.2</td>
<td>79.7 ± 0.1</td>
</tr>
<tr>
<td>Gal (2015) - Variational LSTM (medium, untied, MC)</td>
<td>20M</td>
<td>-</td>
<td>78.6 ± 0.1</td>
</tr>
<tr>
<td>Gal (2015) - Variational LSTM (large, untied)</td>
<td>66M</td>
<td>77.9 ± 0.3</td>
<td>75.2 ± 0.2</td>
</tr>
<tr>
<td>Gal (2015) - Variational LSTM (large, untied, MC)</td>
<td>66M</td>
<td>-</td>
<td>73.4 ± 0.0</td>
</tr>
<tr>
<td>Zilly et al. (2016) - Variational RHN</td>
<td>32M</td>
<td>72.8</td>
<td>71.3</td>
</tr>
<tr>
<td>Zoneout + Variational LSTM (medium)</td>
<td>20M</td>
<td>84.4</td>
<td>80.6</td>
</tr>
<tr>
<td>Pointer Sentinel-LSTM (medium)</td>
<td>21M</td>
<td>72.4</td>
<td>70.9</td>
</tr>
</tbody>
</table>
Section 3: Dynamic memory networks
Can all NLP tasks be seen as question answering problems?
QA Examples

I: Mary walked to the bathroom.  I: Sandra took the milk there.
I: Sandra went to the garden.  I: Mary walked to the bathroom.
I: Daniel went back to the garden.  Q: Where is the milk?
I: Sandra took the milk there.  A: garden
Q: What’s the sentiment?
A: positive

I: Jane has a baby in Dresden.
Q: What are the named entities?
A: Jane - person, Dresden - location

I: Jane has a baby in Dresden.
Q: What are the POS tags?
A: NNP VBZ DT NN IN NNP .

I: Everybody is happy.
Q: What’s the sentiment?
A: positive

I: I think this model is incredible
Q: In French?
A: Je pense que ce modèle est incroyable.
Goal

A joint model for general QA
First Major Obstacle

• For NLP no single model **architecture** with consistent state of the art results across tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>State of the art model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question answering (babi)</td>
<td>Strongly Supervised MemNN (Weston et al 2015)</td>
</tr>
<tr>
<td>Sentiment Analysis (SST)</td>
<td>Tree-LSTMs (Tai et al. 2015)</td>
</tr>
<tr>
<td>Part of speech tagging (PTB-WSJ)</td>
<td>Bi-directional LSTM-CRF (Huang et al. 2015)</td>
</tr>
</tbody>
</table>
Second Major Obstacle

• Fully joint multitask learning* is hard:
  – Usually restricted to lower layers
  – Usually helps only if tasks are related
  – Often hurts performance if tasks are not related

* meaning: same decoder/classifier and not only transfer learning
Tackling First Obstacle

Dynamic Memory Networks

An architecture for any QA task
High level idea for harder questions

- Imagine having to read an article, memorize it, then get asked various questions → Hard!
- You can't store everything in working memory
- **Optimal**: give you the input data, give you the question, allow as many glances as possible
Basic lego block: GRU (defined before)

\[ h_t = GRU(x_t, h_{t-1}) : \]

\[ z_t = \sigma \left( W^{(z)} x_t + U^{(z)} h_{t-1} + b^{(z)} \right) \]

\[ r_t = \sigma \left( W^{(r)} x_t + U^{(r)} h_{t-1} + b^{(r)} \right) \]

\[ \tilde{h}_t = \tanh \left( W x_t + r_t \circ U h_{t-1} + b^{(h)} \right) \]

\[ h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t, \]

Cho et al. 2014
Dynamic Memory Network

Semantic Memory Module
(Glove vectors)

Episodic Memory Module

<table>
<thead>
<tr>
<th>$e_1'$</th>
<th>$e_2'$</th>
<th>$e_3'$</th>
<th>$e_4'$</th>
<th>$e_5'$</th>
<th>$e_6'$</th>
<th>$e_7'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.9</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Answer module

$m'$

Question Module
$q$

Answer module

$m'$

Dynamic Memory Network

Input Module

<table>
<thead>
<tr>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
<th>$s_4$</th>
<th>$s_5$</th>
<th>$s_6$</th>
<th>$s_7$</th>
</tr>
</thead>
</table>

| Mary got the milk there. | John moved to the bedroom. | Sandra went back to the kitchen. | Mary travelled to the hallway. | John got the football there. | John went to the hallway. | Mary went to the garden. |

Answer module

$m'$

Question Module

Where is the football?
Dataset, the gates of Eq. 4 can be trained supervised with a standard cross entropy classification attention mechanism's state.

Finally, to summarize the follows:

In our work, we use a gating function as our attention mechanism. It takes as input, for each pass function which returns an episode given the output of the attention mechanism and the facts from the input module, and a function that summarizes the episodes into a memory.

In its general form, the episodic memory module is characterized by an attention mechanism, a module with this episodic component allows its attention mechanism to attend more selectively to specific facts on each pass, as it can attend to other important facts at a later pass. It also allows for this behavior is indeed seen. Note that the second iteration has wrongly placed some weight in retrieving where John was. In this example, taken from a true test question on Facebook's bAbI task, where John went to the hallway. Sentence 2, which makes some intuitive sense, as sentence 2 is another place John had been.

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?

Where is the football?
\[ q_t = GRU\left(v_t, q_{t-1}\right). \]
The Modules: Episodic Memory

\[ h^t_i = g^t_i \text{GRU}(s_i, h^t_{i-1}) + (1 - g^t_i)h^t_{i-1} \]

Last hidden state: \( m^t \)
The Modules: Episodic Memory

• Gates are activated if sentence relevant to the question or memory

\[ z^t_i = [s_i \circ q \; ; s_i \circ m^{t-1} \; ; |s_i - q| \; ; |s_i - m^{t-1}|] \]

\[ Z^t_i = W^{(2)} \tanh \left( W^{(1)} z^t_i + b^{(1)} \right) + b^{(2)} \]

\[ g^t_i = \frac{\exp(Z^t_i)}{\sum_{k=1}^{M_i} \exp(Z^t_k)} \]

• When the end of the input is reached, the relevant facts are summarized in another GRU
The Modules: Episodic Memory

- If summary is insufficient to answer the question, repeat sequence over input.
Inspiration from Neuroscience

- **Episodic memory** is the memory of autobiographical events (times, places, etc). A collection of past personal experiences that occurred at a particular time and place.

- The hippocampus, the seat of episodic memory in humans, is active during transitive inference.

- In the DMN repeated passes over the input are needed for transitive inference.
The Modules: Answer

\[ a_t = \text{GRU}([y_{t-1}, q], a_{t-1}), \quad y_t = \text{softmax}(W^{(a)} a_t) \]
Related work

- Sequence to Sequence (Sutskever et al. 2014)
- Neural Turing Machines (Graves et al. 2014)
- Teaching Machines to Read and Comprehend (Hermann et al. 2015)
- Learning to Transduce with Unbounded Memory (Grefenstette 2015)
- Structured Memory for Neural Turing Machines (Wei Zhang 2015)

- Memory Networks (Weston et al. 2015)
- End to end memory networks (Sukhbaatar et al. 2015)

→ More on these in Quoc’s lecture
Comparison to MemNets

Similarities:
• MemNets and DMNs have input, scoring, attention and response mechanisms

Differences:
• For input representations MemNets use bag of word, nonlinear or linear embeddings that explicitly encode position
• MemNets iteratively run functions for attention and response

• DMNs show that neural sequence models can be used for input representation, attention and response mechanisms
  → naturally captures position and temporality
• Enables broader range of applications
Experiments: QA on babI (1k)

<table>
<thead>
<tr>
<th>Task</th>
<th>MemNN</th>
<th>DMN</th>
<th>Task</th>
<th>MemNN</th>
<th>DMN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Single Supporting Fact</td>
<td>100</td>
<td>100</td>
<td>11: Basic Coreference</td>
<td>100</td>
<td>99.9</td>
</tr>
<tr>
<td>2: Two Supporting Facts</td>
<td>100</td>
<td>98.2</td>
<td>12: Conjunction</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>3: Three Supporting facts</td>
<td>100</td>
<td>95.2</td>
<td>13: Compound Coreference</td>
<td>100</td>
<td>99.8</td>
</tr>
<tr>
<td>4: Two Argument Relations</td>
<td>100</td>
<td>100</td>
<td>14: Time Reasoning</td>
<td>99</td>
<td>100</td>
</tr>
<tr>
<td>5: Three Argument Relations</td>
<td>98</td>
<td>99.3</td>
<td>15: Basic Deduction</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>6: Yes/No Questions</td>
<td>100</td>
<td>100</td>
<td>16: Basic Induction</td>
<td>100</td>
<td>99.4</td>
</tr>
<tr>
<td>7: Counting</td>
<td>85</td>
<td>96.9</td>
<td>17: Positional Reasoning</td>
<td>65</td>
<td>59.6</td>
</tr>
<tr>
<td>8: Lists/Sets</td>
<td>91</td>
<td>96.5</td>
<td>18: Size Reasoning</td>
<td>95</td>
<td>95.3</td>
</tr>
<tr>
<td>9: Simple Negation</td>
<td>100</td>
<td>100</td>
<td>19: Path Finding</td>
<td>36</td>
<td>34.5</td>
</tr>
<tr>
<td>10: Indefinite Knowledge</td>
<td>98</td>
<td>97.5</td>
<td>20: Agent’s Motivations</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mean Accuracy (%)</td>
<td>93.3</td>
<td>93.6</td>
</tr>
</tbody>
</table>

This still requires that relevant facts are marked during training to train the gates.
Experiments: Sentiment Analysis

- Stanford Sentiment Treebank

- Test accuracies:
- MV-RNN and RNTN: Socher et al. (2013)
- DCNN: Kalchbrenner et al. (2014)
- PVec: Le & Mikolov. (2014)
- CNN-MC: Kim (2014)
- CT-LSTM: Tai et al. (2015)

<table>
<thead>
<tr>
<th>Task</th>
<th>Binary</th>
<th>Fine-grained</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV-RNN</td>
<td>82.9</td>
<td>44.4</td>
</tr>
<tr>
<td>RNTN</td>
<td>85.4</td>
<td>45.7</td>
</tr>
<tr>
<td>DCNN</td>
<td>86.8</td>
<td>48.5</td>
</tr>
<tr>
<td>PVec</td>
<td>87.8</td>
<td>48.7</td>
</tr>
<tr>
<td>CNN-MC</td>
<td>88.1</td>
<td>47.4</td>
</tr>
<tr>
<td>DRNN</td>
<td>86.6</td>
<td>49.8</td>
</tr>
<tr>
<td>CT-LSTM</td>
<td>88.0</td>
<td>51.0</td>
</tr>
<tr>
<td>DMN</td>
<td><strong>88.6</strong></td>
<td><strong>52.1</strong></td>
</tr>
</tbody>
</table>
## Analysis of Number of Episodes

- How many attention + memory passes are needed in the episodic memory?

<table>
<thead>
<tr>
<th>Max passes</th>
<th>task 3 (three-facts)</th>
<th>task 7 (count)</th>
<th>task 8 (lists/sets)</th>
<th>sentiment (fine grain)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 pass</td>
<td>0</td>
<td>48.8</td>
<td>33.6</td>
<td>50.0</td>
</tr>
<tr>
<td>1 pass</td>
<td>0</td>
<td>48.8</td>
<td>54.0</td>
<td>51.5</td>
</tr>
<tr>
<td>2 pass</td>
<td>16.7</td>
<td>49.1</td>
<td>55.6</td>
<td><strong>52.1</strong></td>
</tr>
<tr>
<td>3 pass</td>
<td>64.7</td>
<td>83.4</td>
<td>83.4</td>
<td>50.1</td>
</tr>
<tr>
<td>5 pass</td>
<td><strong>95.2</strong></td>
<td><strong>96.9</strong></td>
<td><strong>96.5</strong></td>
<td>N/A</td>
</tr>
</tbody>
</table>
Analysis of Attention for Sentiment

- Sharper attention when 2 passes are allowed.
- Examples that are wrong with just one pass

1-iter DMN (pred: negative, ans: positive)

2-iter DMN (pred: positive, ans: positive)
Analysis of Attention for Sentiment

- Examples where full sentence context from first pass changes attention to words more relevant for final prediction.

1-iter DMN (pred: positive, ans: negative)

2-iter DMN (pred: negative, ans: negative)
Experiments: POS Tagging

- PTB WSJ, standard splits
- Episodic memory does not require multiple passes, single pass enough

<table>
<thead>
<tr>
<th>Model</th>
<th>SVMTool</th>
<th>Sogaard</th>
<th>Suzuki et al.</th>
<th>Spoustova et al.</th>
<th>SCNN</th>
<th>DMN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc (%)</td>
<td>97.15</td>
<td>97.27</td>
<td>97.40</td>
<td>97.44</td>
<td>97.50</td>
<td>97.56</td>
</tr>
</tbody>
</table>
Modularization Allows for Different Inputs

Episodic Memory

Answer

Kitchen

Episodic Memory

Answer

Palm

Input Module

John moved to the garden.
John got the apple there.
John moved to the kitchen.
Sandra picked up the milk there.
John dropped the apple.
John moved to the office.

Question

Where is the apple?

Answer

What kind of tree is in the background?
The earliest recent work with a visual and textual question answering module is that of memory networks (Sukhbaatar et al. 2015) based upon memory networks that are applied to language processing (Weston et al. 2014). For untied experiments where the episodic memory is tied to the question or previous episode memory, we propose replacing the update gates of the update gates with using a ReLU layer for memory update, calculating the new input module for images. The module splits an image into small local regions and considers each region equivalent to a sentence in the input module for text. To extract features of visual and textual question answering which have, until now, been difficult to capture.

The local regional vectors extracted to produce the input facts of the textual input module described in Sec. 3.2.2, we use the final hidden state of the attention based GRU. After each pass through the attention mechanism, we wish to update the episodic memory state by constructing contextual vector to update the episode memory. We propose replacing the update gates with using a ReLU layer for memory update, calculating the new input module for images. The module splits an image into small local regions and considers each region equivalent to a sentence in the input module for text. To extract features of visual and textual question answering which have, until now, been difficult to capture.

Visual feature extraction:

Input fusion layer:

Feature embedding:

Input Module for Images

- GRU
- GRU
- GRU
- GRU
- W
- W
- W
- CNN
- 14
- 14
- 512

The earliest recent work with a visual and textual question answering module is that of memory networks (Sukhbaatar et al. 2015) based upon memory networks that are applied to language processing (Weston et al. 2014). For untied experiments where the episodic memory is tied to the question or previous episode memory, we propose replacing the update gates with using a ReLU layer for memory update, calculating the new input module for images. The module splits an image into small local regions and considers each region equivalent to a sentence in the input module for text. To extract features of visual and textual question answering which have, until now, been difficult to capture.
VQA test-dev and test-standard:
- Antol et al. (2015)
- ACK Wu et al. (2015);
- iBOWIMG - Zhou et al. (2015);
- DPPnet - Noh et al. (2015); D-NMN - Andreas et al. (2016);
- SAN - Yang et al. (2015)

### Accuracy: Visual Question Answering

<table>
<thead>
<tr>
<th>Method</th>
<th>test-dev</th>
<th>test-std</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All</td>
<td>Y/N</td>
</tr>
<tr>
<td>VQA</td>
<td>28.1</td>
<td>64.0</td>
</tr>
<tr>
<td>Image</td>
<td>48.1</td>
<td>75.7</td>
</tr>
<tr>
<td>Question</td>
<td>52.6</td>
<td>75.6</td>
</tr>
<tr>
<td>Q+I</td>
<td>53.7</td>
<td>78.9</td>
</tr>
<tr>
<td>LSTM Q+I</td>
<td>55.7</td>
<td>79.2</td>
</tr>
<tr>
<td>ACK</td>
<td>55.7</td>
<td>76.5</td>
</tr>
<tr>
<td>iBOWIMG</td>
<td>57.2</td>
<td>80.7</td>
</tr>
<tr>
<td>DPPnet</td>
<td>57.9</td>
<td>80.5</td>
</tr>
<tr>
<td>D-NMN</td>
<td>58.7</td>
<td>79.3</td>
</tr>
<tr>
<td>SAN</td>
<td>60.3</td>
<td>80.5</td>
</tr>
</tbody>
</table>
Attention Visualization

What is the main color on the bus? Answer: blue

What type of trees are in the background? Answer: pine

How many pink flags are there? Answer: 2

Is this in the wild? Answer: no
Attention Visualization

Which man is dressed more flamboyantly?
Answer: right

Who is on both photos?
Answer: girl

What time of day was this picture taken?
Answer: night

What is the boy holding?
Answer: surfboard
Attention Visualization

What is this sculpture made out of? Answer: metal

What color are the bananas? Answer: green

What is the pattern on the cat's fur on its tail? Answer: stripes

Did the player hit the ball? Answer: yes
Live Demo
What is the girl holding?  
*tennis racket*

What is the girl doing?  
*playing tennis*

Is the girl wearing a hat?  
*yes*

What is the girl wearing?  
*shorts*

What is the color of the ground?  
*brown*

What color is the ball?  
*yellow*

What color is her skirt?  
*white*

What did the girl just hit?  
*tennis ball*
Summary

• Word vectors and RNNs are building blocks
• Most NLP tasks can be reduced to QA
• DMN accurately solves variety of QA tasks