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
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
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()
list(panda.closure(hyper))
```

[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

These words will represent banking
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 \( p(w_{t+j} \mid w_t) \) the simplest first formulation is

\[
p(o \mid c) = \frac{\exp \left( u_o^T v_c \right)}{\sum_{w=1}^{W} \exp \left( u_w^T v_c \right)}
\]

- 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
Glove results

Nearest words to frog:

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus
Intrinsic word vector evaluation

- **Word Vector Analogies**
  
  \[ a:b :: c:? \]

  \[ \text{man:woman :: king:?} \]

- 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

Richard Socher
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>81.9</td>
<td>69.3</td>
<td>75.0</td>
</tr>
</tbody>
</table>

Richard Socher
Lecture 1, Slide 22
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:

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

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

Everything else is the same:

$$h_t = \sigma \left( W^{(hh)} h_{t-1} + W^{(hx)} x_t \right)$$

$$\hat{y}_t = \text{softmax} \left( W^{(S)} h_t \right)$$

$$\hat{P}(x_{t+1} = v_j \mid x_t, \ldots, x_1) = \hat{y}_{t,j}$$

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

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

$$J = -\frac{1}{T} \sum_{t=1}^{T} \sum_{j=1}^{|V|} y_{t,j} \log \hat{y}_{t,j}$$
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

\[
\begin{align*}
\hat{h}_t &= \tanh(W x_t + r_t \circ U h_{t-1}) \\
h_t &= z_t \circ h_{t-1} + (1 - z_t) \circ \hat{h}_t \\
z_t &= \sigma(W^{(z)} x_t + U^{(z)} h_{t-1}) \\
r_t &= \sigma(W^{(r)} x_t + U^{(r)} h_{t-1})
\end{align*}
\]
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 : )

@Smerity
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>PAQ8o10t</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
Current Research

Can all NLP tasks be seen as question answering problems?
QA Examples

I: Mary walked to the bathroom.  
I: Sandra went to the garden.  
I: Daniel went back to the garden.  
I: Sandra took the milk there.  
Q: Where is the milk?  
A: garden  
I: Everybody is happy.  
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: 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**

In our work, we use a gating function as our attention mechanism. It takes as input, for each pass of the input module, and a function that summarizes the episodes into a memory.

For instance, in the example in Fig. 3, we are asked to take multiple passes over the facts, focusing attention on different facts at each pass. Each pass produces an episode which returns an episode given the output of the attention mechanism and the facts from sentence 2, which makes some intuitive sense, as sentence 2 is another place John had been.

Specifically, the episodic memory module is characterized by an attention mechanism, a semantic memory module, and these episodes are then summarized into the memory. Endowing our gates. The episode is the final state of the GRU:

## Equation 4

\[
G(t) = \sigma(W_h h_t + W_c c_t + b_g)
\]

Where is the football?

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 put down the football.

Mary went to the garden.

John put down the football.
The Modules: Input

Standard GRU. The last hidden state of each sentence is accessible.
The Modules: Question

\[ q_t = GRU \left( v_t, q_{t-1} \right), \]
For datasets that mark which facts are important for a given question, such as Facebook's bAbI, described it here as its own computation to highlight the potential modularity of these subcomponents. This is equivalent to setting the memory to simply the attention mechanism's final state, but we have de-

In our work, we use a gating function as our attention mechanism. It takes as input, for each pass, a function which returns an episode given the output of the attention mechanism and the facts from the previous states. In its general form, the episodic memory module is characterized by an attention mechanism, a semantic memory module, and a question module. The state may be initialized randomly, but in practice we have found that initializing it to the first Glove vectors produces an episodic memory module that can effectively store and retrieve information.

To compute the episode for pass $t$, we employ a modified GRU over the sequence of states: $h^t_i = g^t_i \text{GRU}(s_i, h^t_{i-1}) + (1 - g^t_i)h^t_{i-1}$

Last hidden state: $m^t$
The Modules: Episodic Memory

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

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

\[ Z_i^t = W^{(2)} \tanh \left( W^{(1)} z_i^t + b^{(1)} \right) + b^{(2)} \]

\[ g_i^t = \frac{\exp(Z_i^t)}{\sum_{k=1}^{M_i} \exp(Z_k^t)} \]

• 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>
</tbody>
</table>

**Mean Accuracy (%)**

- MemNN: 93.3
- DMN: 93.6

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>
Next, we show that the additional correct classifications are than 0.1 between different numbers of passes.

A complicated architecture for them is small. The same is negation or misleading expressions. Hence the need to have sentiment contain only simple sentiment words and no ten correctly classified with 2 passes but many examples note that, especially complicated examples are more of-

directly from the input module to the answer module. We there is no episodic memory at all and outputs are passed outperform a single pass or zero passes. In the latter case, sentiment the differences are smaller. However, two passes the inputs are crucial to achieving high performance. For several of the hard reasoning tasks, multiple passes over 4.4. Quantitative Analysis of Episodic Memory Module

Table 4 shows the accuracies on a subset of bAbI tasks as how the number of passes over the input affect accuracy.

The main novelty of the DMN architecture is in its episodic memory module. Hence, we analyze how important the episodic memory module is for NLP tasks and in particular for several of the hard reasoning tasks, multiple passes over

```
<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>
```

We conclude that the ability of the episodic memory mod-

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

4.3. Sequence Tagging: Part-of-Speech Tagging

Part-of-speech tagging is traditionally modeled as a se-

sequence tagging problem: every word in a sentence is to

be classified into its part-of-speech class (see Fig. 1). We evaluate on the standard Wall Street Journal dataset (Mar-

cus et al., 1993). We use the standard splits of sections

0-18 for training, 19-21 for development and 22-24 for test

1. We compare the DMN with the results in (Søgaard, 2011).

2. We present specific examples from the experiments to illus-

3. trate that the iterative nature of the episodic memory mod-

4. We also evaluate the episodic memory module for senti-

5. mulation analysis. Given that the DMN performs well with

6. focus on during each pass of a three-iteration scan on a

7. For instance, Table 5 shows an example of what the DMN

8. enables the model to focus on relevant parts of the input.

9. Apart from a quantitative analysis, we also show qualita-

10. tively what happens to the attention during multiple passes.

4.5. Qualitative Analysis of Episodic Memory Module

An episodic memory module can take. Note that for the 0-pass DMN, the

network essential reduces to the output of the attention module.

A different maximum limit for the number of passes the episodic

memory module can take. The table shows the accuracies on a subset of bAbI tasks as how the number of passes over the input affect accuracy.

The main novelty of the DMN architecture is in its episodic memory module. Hence, we analyze how important the episodic memory module is for several of the hard reasoning tasks, multiple passes over

```
<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)
4.1 Question Answering

The Facebook bAbI dataset is a synthetic dataset meant to test a model's ability to retrieve facts and reason over them. Each task tests a different skill that a good question answering model ought to have, such as coreference resolution, deduction, and induction. Training on the bAbI dataset

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

Mean Accuracy (%) 93.3

Table 1: Test accuracies on the bAbI dataset. MemNN numbers taken from Weston et al. [18]. The DMN passes (accuracy > 95%) 18 tasks, whereas the MemNN passes 16.

4.2 Sequence Tagging: Part of Speech Tagging

Part-of-speech tagging is traditionally modeled as a sequence tagging problem: every word in a sentence is to be classified into its part-of-speech class (see Fig. 1). We evaluate on the standard Wall Street Journal dataset included in Penn-III [26]. We use the standard splits of sections 0-18 for training, 19-21 for development and 22-24 for test sets [27]. Since this is a word level tagging task, DMN memories are produced at the word -rather than sentence- level. We compare the DMN

<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><strong>97.56</strong></td>
</tr>
</tbody>
</table>

Experiments: POS Tagging

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

What kind of tree is in the background?
The VGG-19 model (from the image, we use a convolutional neural network Local region feature extraction: in Sec.

feature embedding, and the input fusion layer introduced illustrated in Fig.

The input module for VQA is composed of three parts, il-
gion equivalent to a sentence in the input module for text.
an image into small local regions and considers each re-

To apply the DMN to visual question answering, we intro-

3. DMN Input Module for VQA

LSTM model with using a ReLU layer for memory update, calcu-

Sukhbaatar et al.

updates. Following the memory update component used in

dependent weights, the GRU makes less sense for memory

each pass through the episodic memory module has inde-

different weights for each pass through the episodic mem-

The work of

Episode Memory Updates

mutual information by

cannot do. To produce the contextual vector
to update the episodic memory.

to update the episode memory

to update the episode memory state

to update the episode memory

to update the episodic memory state

Episode Memory Updates

Based on the memory component that is applied to language processing

we experience

we experience

we experience

we experience

4. Related Work

the original image 2D, some spatial information may be

bi-directional GRU over these input facts

to produce the input facts

of the textual input module described in Sec.

To solve this, we add an input fusion layer similar to that

ing or locational variance causing accuracy problems.

power is quite limited, with simple issues like object scal-

from above do not yet have global information available

Input fusion layer:

vector

to the textual feature space used by the question

with tanh activation to project the

both image features and text features, we add a linear layer

into a grid of

Figure 3.

Visual feature extraction

Feature embedding

Input fusion layer

Input Module for Images
**Accuracy: Visual Question Answering**

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)

<table>
<thead>
<tr>
<th>Method</th>
<th>VQA Test-dev</th>
<th>VQA Test-std</th>
</tr>
</thead>
<tbody>
<tr>
<td>VQA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image</td>
<td>28.1</td>
<td>3.8</td>
</tr>
<tr>
<td>Question</td>
<td>48.1</td>
<td>27.1</td>
</tr>
<tr>
<td>Q+I</td>
<td>52.6</td>
<td>37.4</td>
</tr>
<tr>
<td>LSTM Q+I</td>
<td>53.7</td>
<td>36.4</td>
</tr>
<tr>
<td>ACK</td>
<td>55.7</td>
<td>40.1</td>
</tr>
<tr>
<td>iBOWIMG</td>
<td>55.7</td>
<td>42.6</td>
</tr>
<tr>
<td>DPPnet</td>
<td>57.2</td>
<td>41.7</td>
</tr>
<tr>
<td>D-NMN</td>
<td>57.9</td>
<td>43.1</td>
</tr>
<tr>
<td>SAN</td>
<td>58.7</td>
<td>46.1</td>
</tr>
<tr>
<td>DMN+</td>
<td>60.3</td>
<td>48.3</td>
</tr>
</tbody>
</table>

*Table 2. Failed tasks (err > 5%) 1 3 -

<table>
<thead>
<tr>
<th>Test error rates of various model architectures on tasks</th>
<th>VQA Test-dev</th>
<th>VQA Test-std</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test: Accuracy: VQA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test-dev</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VQA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image</td>
<td>28.1</td>
<td>3.8</td>
</tr>
<tr>
<td>Question</td>
<td>48.1</td>
<td>27.1</td>
</tr>
<tr>
<td>Q+I</td>
<td>52.6</td>
<td>37.4</td>
</tr>
<tr>
<td>LSTM Q+I</td>
<td>53.7</td>
<td>36.4</td>
</tr>
<tr>
<td>ACK</td>
<td>55.7</td>
<td>40.1</td>
</tr>
<tr>
<td>iBOWIMG</td>
<td>55.7</td>
<td>42.6</td>
</tr>
<tr>
<td>DPPnet</td>
<td>57.2</td>
<td>41.7</td>
</tr>
<tr>
<td>D-NMN</td>
<td>57.9</td>
<td>43.1</td>
</tr>
<tr>
<td>SAN</td>
<td>58.7</td>
<td>46.1</td>
</tr>
<tr>
<td>DMN+</td>
<td>60.3</td>
<td>48.3</td>
</tr>
</tbody>
</table>

*Table 3. Test error rates of various model architectures on tasks*

- **Accuracy:** 58.9
- **Visual Question Answering**
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

Dynamic Memory Network by MetaMind

Question

What color is the building that has a clock?

Run DMN

VQA sample
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
Sequence to Sequence Learning for NLP and Speech

Quoc V. Le
Google Brain team
“AutoReply”

- 508 unread emails!!!
- Some emails just require “Yes” / “No” answers
- Let’s build “AutoReply”
• From: Ann
• Subject: Hi
• Content: Are you visiting Vietnam for the new year, Quoc?
• Probable Reply: Yes
Dataset

• Are you visiting Vietnam for the new year, Quoc? -> Yes
• Are you hanging out with us tonight? -> No
• Did you read the cool paper on ResNet? -> Yes
• ...

Preprocessing

- Are you visiting Vietnam for the new year, Quoc? -> Yes
- Are you hanging out with us tonight? -> No
- Did you read the cool paper on ResNet? -> Yes
- ...

...
Preprocessing

- Are you visiting Vietnam for the new year, Quoc? -> Yes
- Are you hanging out with us tonight? -> No
- Did you read the cool paper on ResNet? -> Yes
- ...


Are you visiting Vietnam for the new year, Quoc?

\[\{0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, \ldots, 0, 0, 1, 0, 0, 0, 0, 2\}\]

20,000 dimensions
Are you visiting Vietnam for the new year, Quoc?

\[ [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, \ldots, 0, 0, 1, 0, 0, 0, 0, 2] \]

20,000 dimensions
Are you visiting Vietnam for the new year, Quoc?

[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, ..., 0, 0, 1, 0, 0, 0, 0, 2]

20,000 dimensions
Are you visiting Vietnam for the new year, Quoc?

\[ [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, \ldots, 0, 0, 1, 0, 0, 0, 0, 0, 2] \]

20,000 dimensions
Are you visiting Vietnam for the new year, Quoc?

[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, ..., 0, 0, 1, 0, 0, 0, 2]

20,000 dimensions

Special dimension reserved for out of vocabulary words
Formulation

\[ [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, \ldots, 0, 0, 1, 0, 0, 0, 0, 2] \rightarrow 1 \]

\[ [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, \ldots, 1, 0, 0, 0, 0, 0, 0] \rightarrow 0 \]

\[ [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, \ldots, 0, 3, 0, 0, 0, 0, 1] \rightarrow 1 \]
Formulation

\[
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, \ldots, 0, 0, 1, 0, 0, 0, 0, 2] \rightarrow 1
\]

\[
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, \ldots, 1, 0, 0, 0, 0, 0, 0] \rightarrow 0
\]

\[
[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, \ldots, 0, 3, 0, 0, 0, 0, 1] \rightarrow 1
\]
Formulation

• Find $W$ such that $Wx$ approximates $y$

• Since $y$ is in \{“Yes”, “No”\}, this is a “Logistic Regression” problem

\[
\frac{\exp(w_1^T x)}{\exp(w_1^T x) + \exp(w_2^T x)}
\]
Formulation

• Find $W$ such that $Wx$ approximates $y$

• Since $y$ is in \{“Yes”, “No”\}, this is a “Logistic Regression” problem

\[
\frac{\exp(w_1^T x)}{\exp(w_1^T x) + \exp(w_2^T x)}
\]

Positive and sum up to 1
Training with stochastic gradient descent

• For iteration 1, 2, 3, ..., 1000000
  • Sample a random email $x$ and a reply
  • If reply == Yes, update $w_1$ and $w_2$ to increase
    \[
    \frac{\exp(w_1^T x)}{\exp(w_1^T x) + \exp(w_2^T x)}
    \]
  • If reply == No, update $w_1$ and $w_2$ to increase
    \[
    \frac{\exp(w_2^T x)}{\exp(w_1^T x) + \exp(w_2^T x)}
    \]
Training with stochastic gradient descent

• For iteration 1, 2, 3, …, 1000000

• Sample a random email $x$ and a reply

• If reply == Yes, update $w_1$ and $w_2$ to increase

\[
\frac{\exp(w_1^T x)}{\exp(w_1^T x) + \exp(w_2^T x)}
\]

• If reply == No, update $w_1$ and $w_2$ to increase

\[
\frac{\exp(w_2^T x)}{\exp(w_1^T x) + \exp(w_2^T x)}
\]
Training with stochastic gradient descent

• For iteration 1, 2, 3, …, 1000000

• Sample a random email $x$ and a reply

• If reply == Yes, update $w_1$ and $w_2$

  \[ w_1 = w_1 + \alpha \frac{d \log(p_1)}{d w_1} \quad w_2 = w_2 + \alpha \frac{d \log(p_1)}{d w_2} \]

• If reply == No, update $w_1$ and $w_2$

  \[ w_1 = w_1 + \alpha \frac{d \log(p_2)}{d w_1} \quad w_2 = w_2 + \alpha \frac{d \log(p_2)}{d w_2} \]
Prediction

• For any incoming email $x$

• Compute

\[
\frac{\exp(w_1^T x)}{\exp(w_1^T x) + \exp(w_2^T x)}
\]

• If $> 0.5$ -> reply = Yes

• If $\leq 0.5$ -> reply = No
Information Loss

Are you visiting Vietnam for the new year, Quoc?

[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, ..., 0, 0, 1, 0, 0, 0, 2]

20,000 dimensions

This “bag-of-words representation” does not care about the order of the words!
Recurrent Neural Network

Are you visiting Vietnam for the new year, Quoc?
Recurrent Neural Network

Are you visiting Vietnam for the new year, Quoc?
Recurrent Neural Network

are you visiting Vietnam for the new year, Quoc?
Recurrent Neural Network

\[ h_1 = \tanh(A^* h_0 + U^* v(\text{"you"})) \]

Are you visiting Vietnam for the new year, Quoc?
Recurrent Neural Network

$h_2 = \tanh(A \cdot h_1 + U \cdot v(\text{“visting”}))$

Are you visiting Vietnam for the new year, Quoc?
Recurrent Neural Network
Training RNN with stochastic gradient descent

• For iteration 1, 2, 3, ..., 1000000

• Sample a random email $x$ and a reply

• If reply == Yes, update $w_1$ and $w_2$

$$w_1 = w_1 + \alpha \frac{d \log(p_1)}{d w_1} \quad w_2 = w_2 + \alpha \frac{d \log(p_1)}{d w_2}$$

Update $U$, and $A$

$$A = A + \alpha \frac{d \log(p_1)}{d A} \quad U = U + \alpha \frac{d \log(p_1)}{d U}$$

Update all relevant $v$'s

$$v_i = v_i + \alpha \frac{d \log(p_1)}{d v_i}$$
Training RNN with stochastic gradient descent

- For iteration 1, 2, 3, …, 1000000
- Sample a random email \( x \) and a reply
- If reply == Yes, update \( w_1 \) and \( w_2 \)

\[
\begin{align*}
  w_1 &= w_1 + \alpha \frac{d \log(p_1)}{d w_1} \\
  w_2 &= w_2 + \alpha \frac{d \log(p_1)}{d w_2}
\end{align*}
\]

Update \( U \), and \( A \)

\[
\begin{align*}
  A &= A + \alpha \frac{d \log(p_1)}{d A} \\
  U &= U + \alpha \frac{d \log(p_1)}{d U}
\end{align*}
\]

Update all relevant \( v \)'s

\[
\begin{align*}
  v_i &= v_i + \alpha \frac{d \log(p_1)}{d v_i}
\end{align*}
\]
Training RNN with stochastic gradient descent

- For iteration 1, 2, 3, ..., 1000000
- Sample a random email $x$ and a reply
- If reply == Yes, update $w_1$ and $w_2$

\[
\begin{align*}
    w_1 &= w_1 + \alpha \frac{d \log(p_1)}{d w_1} \\
    w_2 &= w_2 + \alpha \frac{d \log(p_1)}{d w_2}
\end{align*}
\]

Update $U$, and $A$

\[
\begin{align*}
    A &= A + \alpha \frac{d \log(p_1)}{d A} \\
    U &= U + \alpha \frac{d \log(p_1)}{d U}
\end{align*}
\]

Update all relevant $v$'s

\[
\begin{align*}
    v_i &= v_i + \alpha \frac{d \log(p_1)}{d v_i}
\end{align*}
\]

Very hard to derive! Use autodiff :)
The big picture so far

- Bag-of-word representation
- RNN representation for variable-sized input
- Autodiff to compute the partial derivatives (TensorFlow, Theano, Torch)
- Stochastic gradient descent for training
More friendly “AutoReply”

• Are you visiting Vietnam for the new year, Quoc? -> Yes, see you soon!
• Are you hanging out with us tonight? -> No, I am too busy.
• Did you read the cool paper on ResNet? -> Yes, it’s nice!
• …
Better Formulation

- Mapping between variable-length input to variable length output
Better Formulation

- Mapping between variable-length input to variable length output
- Applications: AutoReply, Translation, Image Captioning, Summarization, Speech Transcription, Conversation, Q&A, …
Andrej Karpathy. The Unreasonable Effectiveness of Recurrent Neural Networks
Better Formulation

I’m fine thanks <end>

W W W W

B B B

<end>
Hi, how are you?

I'm fine; thanks. <end>

Number of choices = number of words in vocabulary.
Hi, how are you?

I'm fine, thanks.
Hi  how  are  you  I’m  fine  thanks  <end>

Better Formulation
Sequence to Sequence Training with SGD

• For iteration 1, 2, 3, …., 1000000

• Sample an email $x$ and a reply $y$

• Sample a random word $y(t)$ in $y$

• Update RNN encoder and decoder parameters to increase the probability of word $y(t)$ given $y(t-1), y(t-2), \ldots, y(0), x(n), x(n-1), \ldots, x(0)$ using partial derivative with respect to $W, U, A, B$, and all $v$’s
Sequence to Sequence Training with SGD

- For iteration 1, 2, 3, …., 1000000
- Sample an email \( x \) and a reply \( y \)
- Sample a random word \( y(t) \) in \( y \)
- Update RNN encoder and decoder parameters to increase the probability of word \( y(t) \) given \( y(t-1), y(t-2), \ldots, y(0), x(n), x(n-1), \ldots, x(0) \) using partial derivative with respect to \( W, U, A, B \), and all \( v \)'s

Very hard to derive! Use autodiff :)
Sequence to Sequence Prediction

- For any incoming email $x$
  - Given $x$, find word $y^{(0)}$ with highest probability using RNN
  - Given $y^{(0)}$ and $x$, find word $y^{(1)}$ with highest probability using RNN
    - ...
  - Stop when see `<end>`

“Greedy Decoding”
Sequence to Sequence Prediction

• For any incoming email $x$
  • Given $x$, find $k$ candidates for $y^{(0)}$ with highest probability using RNN
  • Given $x$, for each candidate $y^{(0)}$, find $k$ candidates for word $y^{(1)}$ with highest probability using RNN
  • ...
  • Stop when see $<end>$ on each beam
• Reply = beam with highest probability

“Beam Search Decoding”
Sequence to Sequence Prediction

“Beam Search Decoding”

Input $x$

Hi
Yes
Please

, how what sure certainly <end>
I
see come

...
Scheduled Sampling

Scheduled Sampling involves injecting noise sampled from the previous outputs.

- \( h_0 \rightarrow h_1 \rightarrow h_2 \rightarrow h_3 \rightarrow \ldots \rightarrow h_n \)
- \( g_0 \rightarrow g_1 \rightarrow g_2 \)

Here are some examples of the conversation:

- Hi
- How are you
- I'm fine, thanks

<end>
SmartReply feature in Inbox
The big picture so far

- RNN encoder and RNN decoder for sequence to sequence learning
- Use stochastic gradient descent for training
- Beam search decoding
Attention Mechanism

I'm fine thanks
I'm fine
<end>

Encoder

Decoder
Attention Mechanism

Fixed-length representation for variable-length inputs

Encoder → Decoder

I’m fine thanks <end>

<end> I’m fine thanks

Hi how are you

h₀ h₁ h₂ h₃ ... hₙ g₀ g₁ g₂
Attention Mechanism

I’m

h₀ → h₁ → h₂ → h₃ → ... → hₙ

Hi  how  are  you  <end>
Attention Mechanism

\[ h_0 \rightarrow h_1 \rightarrow h_2 \rightarrow h_3 \rightarrow \ldots \rightarrow h_n \]

Hi how are you <end>
Attention Mechanism

Hi how are you

fine

I’m
Attention Mechanism

Hi how are you

I’m

c

<end>
Attention Mechanism

Hi, how are you? I'm c a 0 a 1 a 2 a 3 h 0 h 1 h 2 h 3 ... h n <end>
Attention Mechanism

$$b_i = \frac{\exp(a_i)}{\exp(a_1) + \exp(a_2) + \ldots + \exp(a_n)}$$
Attention Mechanism

Implemented in TensorFlow seq2seq
Model Understandability with Attention Mechanism

Hola!

Hi how are you

<end>
Model Understandability with Attention Mechanism

Hola! ¿Cómo estás? Hola!
Deeper Networks work Better

b₀  b₁  b₂  b₃
a₀  a₁  a₂  a₃

Hi  how  are  you

Hola!  ¿Cómo

<end>  Hola!
Sequence to Sequence With Attention

• Currently the state-of-art in many translation tasks
  • Tip 1: Use word segments or word/character hybrid instead of just words
  • Tip 2: Gradient Clipping to prevent explosion
  • Tip 3: Use Long Short Term Memory

Other applications:
• Summarization, Image Captioning,
• Speech Transcription, Q&A
LSTMCell vs. RNNCell

RNNCell:

\[ h = \tanh(\theta \ast [\text{inputs, h}]) \]

LSTMCell:

\[ Z = \theta \ast [\text{inputs, h}] \]

\[ i, j, f, o = \text{split}(1, 4, Z) \# \text{split to four blocks} \]

\[ \text{new}_c = c \ast \text{sigmoid}(f) + \text{sigmoid}(i) \ast \tanh(j) \# \text{integral of c} \]

\[ \text{new}_h = \tanh(\text{new}_c) \ast \text{sigmoid}(o) \]
Applications

• Other applications:
  • Summarization, Image Captioning,
  • Speech Transcription, Q&A
Applications

• Other applications:
  • Summarization, Image Captioning,
  • Speech Transcription, Q&A
seq2seq for Speech

→ Hi how’s it?
seq2seq for Speech
seq2seq for Speech

MFCC
seq2seq for Speech

MFCC

$h_0 \rightarrow h_1 \rightarrow h_2 \rightarrow h_3 \rightarrow \cdots \rightarrow h_n$

$g_0 \rightarrow g_1 \rightarrow g_2 \rightarrow g_3$

<go> Hi how's it? <end>

MFCC
seq2seq for Speech

Too many input steps!

MFCC

<go> Hi how's it? <end>

<go> Hi how's it? <end>

<go> Hi how's it? <end>
seq2seq for Speech

MFCC

h₀ → h₁ → ... → hₙ

<go> → g₀ → Hi → how’s → it? → <end>

<go> → h₁ → g₁ → Hi → how’s → it? → <end>

<go> → h₂ → g₂ → Hi → how’s → it? → <end>

<go> → hₙ → gₙ → Hi → how’s → it? → <end>
seq2seq for Speech

Character output

MFCC

<go> Hi how's it? <end>

h₀ h₁ . . . hₙ

g₀ g₁ g₂ g₃
Sequence to Sequence With Attention for Speech

• Implicit language model
• “Offline” beam search decoding
• Not as good as
  • CTC (Adam Coates’ talk)
  • HMM-DNN hybrid (most widely-used speech systems)
The Big Picture

• Sequence to sequence is an “End-to-end Deep Learning” algorithm

• It’s very general, so it should work with most NLP-related tasks when you have a lot of data

• If you don’t have enough data:
  
  • Consider dividing your problem into smaller problems, and train seq2seq on each of them.
  
  • Train jointly with many other tasks

• What I present next is an active area of research
Automatic Q&A

• Reading a book and answer a question

• Seq2seq with attention: Read the book, then read the question, then revisit all pages in the book.

—> Augmented RNNs with memory (Memory Networks, Neural Turing Machines, Dynamic Memory Networks, Stack-augmented RNNs etc.)
Revisit Attention Mechanism

Encoder

Input

Controller (h)

WRITE

Memory (h₁, h₂, ..., hₙ)
Revisit Attention Mechanism

Controller (g)

Input

Output

Memory (h₁, h₂, ..., hₙ)

Decoder

READ
Differentiable Memory (Neural Turing Machines, Memory Networks, Stack-Augmented RNNs)
Differentiable Memory

\[ h_0 \xrightarrow{\text{WRITE}} h_1 \]

\[ v_0 \xleftarrow{\text{memory bank}} h_0 \]

\[ v_1 \xrightarrow{\text{memory bank}} h_1 \]
Differentiable Memory
Differentiable Memory

RNN with augmented memory
RNN with augmented operations

- **Context:** The building was constructed in 2000. ... It was destroyed in 2010. ...

- **Question:** How long did the building survive?

- **Answer:** 10 years.
Neural Programmers

<table>
<thead>
<tr>
<th>Subtraction</th>
<th>Addition</th>
</tr>
</thead>
</table>

Stack of numbers

$h_0$ $h_1$

$V_0$ $V_1$

…
The Big Picture

- Sequence to sequence is an “End-to-end Deep Learning” algorithm

- It’s very general, so it should work with most NLP-related tasks when you have a lot of data

- If you don’t have enough data:
  - Consider dividing your problem into smaller problems, and train seq2seq on each of them.
  - Train jointly with many other tasks

- RNN with memory, or operation augmentation are exciting work in progress
Additional Reading

• Chris Olah’s blog: Attention and Augmented Recurrent Neural Networks

• My own tutorials: http://ai.stanford.edu/~quocle/tutorial2.pdf

• Seq2seq in TensorFlow: https://www.tensorflow.org/versions/r0.10/tutorials/seq2seq/index.html
References

• Modeling
  • Sequence to Sequence with Neural Networks by Sutskever, Vinyals, Le. NIPS, 2014
  • Neural machine translation by jointly learning to align and translate by Bahdanau, Cho, Bengio. ICLR, 2015
  • Neural Turing Machines, by Graves, Wayne, Danihelka. arXiv, 2014
  • End-to-End Memory Networks by Sukhbaatar, Weston, Fergus. NIPS, 2015
  • Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks, by Bengio, Vinyals, Jaitly, Shazeer. NIPS, 2015
  • Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets by Joulin and Mikolov. NIPS, 2015.

• Applications
  • Show and Tell: A Neural Image Caption Generator, by Vinyals, Toshev, Bengio, Erhan. CVPR, 2015
  • Neural Conversational Model, by Vinyals and Le. ICML Workshop, 2015
  • Neural responding machine for short-text conversation by Shang, Lu, Li. ACL, 2015.
  • Attention-Based Models for Speech Recognition. Chorowski, Bahdanau, Serdyuk, Cho, Bengio. NIPS, 2015
  • Listen, Attend and Spell. Chan, Jaitly, Le, Vinyals. ICASSP, 2016