Lecture 10:
(Textual) Question Answering Architectures, Attention and Transformers
Mid-quarter feedback survey

Thanks to the many of you (!) who have filled it in!

If you haven’t yet, today is a good time to do it 😊
Lecture Plan

Lecture 10: (Textual) Question Answering

1. History/The SQuAD dataset (review)
2. The Stanford Attentive Reader model
3. BiDAF
4. Recent, more advanced architectures
5. Open-domain Question Answering: DrQA
6. Attention revisited; motivating transformers; ELMo and BERT preview
7. Training/dev/test data
8. Getting your neural network to train
1. Turn-of-the Millennium Full NLP QA:
[architecture of LCC (Harabagiu/Moldovan) QA system, circa 2003]
Complex systems but they did work fairly well on “factoid” questions
Question: Which team won Super Bowl 50?

Passage

Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California.
SQuAD 2.0 No Answer Example

Genghis Khan united the Mongol and Turkic tribes of the steppes and became Great Khan in 1206. He and his successors expanded the Mongol empire across Asia. Under the reign of Genghis' third son, Ögedei Khan, the Mongols destroyed the weakened Jin dynasty in 1234, conquering most of northern China. Ögedei offered his nephew Kublai a position in Xingzhou, Hebei. Kublai was unable to read Chinese but had several Han Chinese teachers attached to him since his early years by his mother Sorghaghtani. He sought the counsel of Chinese Buddhist and Confucian advisers. Möngke Khan succeeded Ögedei's son, Gyük, as Great Khan in 1251. He

**When did Genghis Khan kill Great Khan?**

*Gold Answers*: <No Answer>

*Prediction*: 1234 [from Microsoft nlnet]
2. Stanford Attentive Reader

[Chen, Bolton, & Manning 2016]
[Chen, Fisch, Weston & Bordes 2017] DrQA
[Chen 2018]

- Demonstrated a minimal, highly successful architecture for reading comprehension and question answering
- Became known as the Stanford Attentive Reader
The Stanford Attentive Reader

Which team won Super Bowl 50?
Who did Genghis Khan unite before he began conquering the rest of Eurasia?

He courted power by uniting many of the nomadic tribes in Northern Asia. After founding the Mongol Empire and being proclaimed "Genghis Khan", he started the Mongol invasions that resulted in the conquest of most of Eurasia. These included raids or invasions of the Qara Khitai, Caucasus, Khwarezmid Empire, Western Xia and Jin dynasties. These campaigns were often accompanied by wholesale massacres of the civilian populations – especially in the Khwarezmian and Xia controlled lands. By the end of his life, the Mongol Empire occupied a substantial portion of Central Asia and China.
Who did **Genghis Khan** unite before he began **conquering** the rest of **Eurasia**?

**He** came to power by **uniting** many of the nomadic tribes of Northeast Asia. **After** founding the Mongol Empire and being proclaimed "**Genghis Khan**", he started the Mongol invasions that resulted in the **conquest** of most of **Eurasia**. These included raids or invasions of the **Qara Khitai**, Caucasus, **Khwarezmid Empire**, Western Xia and Jin dynasties. These campaigns were often accompanied by wholesale massacres of the civilian populations – especially in the Khwarezmian and Xia controlled lands. By the end of his life, the Mongol Empire occupied a substantial portion of Central Asia and China.
### SQuAD 1.1 Results (single model, c. Feb 2017)

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic regression</td>
<td>51.0</td>
</tr>
<tr>
<td>Fine-Grained Gating (Carnegie Mellon U)</td>
<td>73.3</td>
</tr>
<tr>
<td>Match-LSTM (Singapore Management U)</td>
<td>73.7</td>
</tr>
<tr>
<td>DCN (Salesforce)</td>
<td>75.9</td>
</tr>
<tr>
<td>BiDAF (UW &amp; Allen Institute)</td>
<td>77.3</td>
</tr>
<tr>
<td>Multi-Perspective Matching (IBM)</td>
<td>78.7</td>
</tr>
<tr>
<td>ReasoNet (MSR Redmond)</td>
<td>79.4</td>
</tr>
<tr>
<td>DrQA (Chen et al. 2017)</td>
<td>79.4</td>
</tr>
<tr>
<td>r-net (MSR Asia) [Wang et al., ACL 2017]</td>
<td>79.7</td>
</tr>
<tr>
<td>Human performance</td>
<td>91.2</td>
</tr>
</tbody>
</table>
NewsQA dataset consists of 100,000 question-answer pairs from CNN news articles. For other datasets like WikiQA the span is the entire sentence containing the answer (Yang et al., 2015); the task of choosing a sentence rather than a smaller answer span is sometimes called the sentence selection task.

These reading comprehension datasets are used both as a reading comprehension task in themselves, and as a training set and evaluation set for the sentence extraction component of open question answering algorithms.

Basic Reading Comprehension Algorithm.

Neural algorithms for reading comprehension are given a question \( q \) of \( l \) tokens \( q_1, \ldots, q_l \) and a passage \( p \) of \( m \) tokens \( p_1, \ldots, p_m \). Their goal is to compute, for each token \( p_i \) the probability \( p_{\text{start}}(i) \) that \( p_i \) is the start of the answer span, and the probability \( p_{\text{end}}(i) \) that \( p_i \) is the end of the answer span.

Fig. 23.8 shows the architecture of the Document Reader component of the DrQA system of Chen et al. (2017). Like most such systems, DrQA builds an embedding for the question, builds an embedding for each token in the passage, computes a similarity function between the question and each passage word in context, and then uses the question-passage similarity scores to decide where the answer span starts and ends.

When did Beyonce release Dangerously in Love?

Passage

Let's consider the algorithm in detail, following closely the description in Chen et al. (2017). The question is represented by a single embedding \( q \), which is a weighted sum of representations for each question word \( q_i \). It is computed by passing the series of embeddings \( P(E(q_1)), \ldots, E(q_l) \) of question words through an RNN (such as a bi-LSTM shown in Fig. 23.8). The resulting hidden representations \( \{q_1, \ldots, q_l\} \) are combined by a weighted sum

\[
q = \sum_j b_j q_j \tag{23.9}
\]

Training objective:

\[
\mathcal{L} = - \sum \log P^{(\text{start})}(a_{\text{start}}) - \sum \log P^{(\text{end})}(a_{\text{end}})
\]
Stanford Attentive Reader++

$$q = \sum_j b_j q_j$$

For learned $w$, $b_j = \frac{\exp(w \cdot q_j)}{\sum_{j'} \exp(w \cdot q_{j'})}$

Deep 3 layer BiLSTM is better!
Stanford Attentive Reader++

- \( \mathbf{p}_i \): Vector representation of each token in passage
  Made from concatenation of
  - Word embedding (GloVe 300d)
  - Linguistic features: POS & NER tags, one-hot encoded
  - Term frequency (unigram probability)
  - Exact match: whether the word appears in the question
    - 3 binary features: exact, uncased, lemma
  - Aligned question embedding ("car" vs "vehicle")

\[
\mathcal{f}_{\text{align}}(\mathbf{p}_i) = \sum_j a_{i,j} \mathbf{E}(q_j)
\]

\[
q_{i,j} = \frac{\exp(\alpha(\mathbf{E}(p_i)) \cdot \alpha(\mathbf{E}(q_j)))}{\sum_{j'} \exp(\alpha(\mathbf{E}(p_i)) \cdot \alpha(\mathbf{E}(q_{j'}))))}
\]

Where \( \alpha \) is a simple one layer FFNN
A big win for neural models

<table>
<thead>
<tr>
<th>Feature Classifier</th>
<th>Ours</th>
<th>SoTA (Google AI)</th>
<th>Human</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>51</td>
<td>91.8</td>
<td>91.2</td>
</tr>
<tr>
<td></td>
<td>+28.4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What do these neural models do?

Correctness (%)

- **Easy**: 13% - NN: 100, Categorical Feature Classifier: 100
- **Paraphrasing**: 41% - NN: 95, Categorical Feature Classifier: 78
- **Partial**: 19% - NN: 90, Categorical Feature Classifier: 74
- **MultiSent**: 2% - NN: 50, Categorical Feature Classifier: 50
3. **BiDAF**: Bi-Directional Attention Flow for Machine Comprehension

(Seo, Kembhavi, Farhadi, Hajishirzi, ICLR 2017)
BiDAF – Roughly the CS224N DFP baseline

- There are variants of and improvements to the BiDAF architecture over the years, but the central idea is **the Attention Flow layer**
- **Idea**: attention should flow both ways – from the context to the question and from the question to the context
- Make similarity matrix (with $w$ of dimension $6d$):
  \[
  S_{i,j} = w_{\text{sim}}^T [c_i; q_j; c_i \odot q_j] \in \mathbb{R}
  \]
- Context-to-Question (C2Q) attention:
  (which query words are most relevant to each context word)
  \[
  \alpha^i = \text{softmax}(S_{i,:}) \in \mathbb{R}^M \quad \forall i \in \{1, \ldots, N\}
  \]
  \[
  a_i = \sum_{j=1}^{M} \alpha^i_j q_j \in \mathbb{R}^{2h} \quad \forall i \in \{1, \ldots, N\}
  \]
BiDAF

- **Attention Flow Idea**: attention should flow both ways – from the context to the question and from the question to the context

- **Question-to-Context (Q2C) attention**: (the weighted sum of the most important words in the context with respect to the query – slight asymmetry through max)

\[
    m_i = \max_j S_{ij} \in \mathbb{R} \quad \forall i \in \{1, \ldots, N\}
\]

\[
    \beta = \text{softmax}(m) \in \mathbb{R}^N
\]

\[
    c' = \sum_{i=1}^{N} \beta_i c_i \in \mathbb{R}^{2h}
\]

- For each passage position, output of BiDAF layer is:

\[
    b_i = [c_i; a_i; c_i \circ a_i; c_i \circ c'] \in \mathbb{R}^{8h} \quad \forall i \in \{1, \ldots, N\}
\]
BiDAF

• There is then a “modelling” layer:
  • Another deep (2-layer) BiLSTM over the passage
• And answer span selection is more complex:
  • Start: Pass output of BiDAF and modelling layer concatenated to a dense FF layer and then a softmax
  • End: Put output of modelling layer M through another BiLSTM to give $M_2$ and then concatenate with BiDAF layer and again put through dense FF layer and a softmax
    • Editorial: Seems very complex, but it does seem like you should do a bit more than Stanford Attentive Reader, e.g., conditioning end also on start
4. Recent, more advanced architectures

Most of the question answering work in 2016–2018 employed progressively more complex architectures with a multitude of variants of attention – often yielding good task gains
Dynamic Coattention Networks for Question Answering (Caiming Xiong, Victor Zhong, Richard Socher ICLR 2017)

- Flaw: Questions have input-independent representations
- Interdependence needed for a comprehensive QA model

**Flaw: Questions have input-independent representations**

**Interdependence needed for a comprehensive QA model**
Coattention Encoder

Figure 2: Coattention encoder. The affinity matrix $L$ is not shown here. We instead directly show the normalized attention weights $A_D$ and $A_Q$. We similarly compute the summaries $C_{QA}$ of the question in light of each word of the document. Similar to Cui et al. (2016), we also compute the summaries $C_{AQ}$ of the previous attention contexts in light of each word of the document. These two operations can be done in parallel, as is shown in Eq. 3. One possible interpretation for the operation $C_{AQ}$ is the mapping of question encoding into space of document encodings.

$$C_D = \text{concat}(Q; C_{AQ})^2 \mathbb{R}^2_{\ell} \times (m+1).$$

We define $C_D$, a co-dependent representation of the question and document, as the coattention context. We use the notation $[a; b]$ for concatenating the vectors horizontally.

The last step is the fusion of temporal information to the coattention context via a bidirectional LSTM:

$$u_t = \text{Bi-LSTM}(u_{t-1}, u_{t+1}); c_D^t \mathbb{R}^2_{\ell}.$$
Coattention layer

• Coattention layer again provides a two-way attention between the context and the question

• However, coattention involves a second-level attention computation:
  • attending over representations that are themselves attention outputs

• We use the C2Q attention distributions $\alpha_i$ to take weighted sums of the Q2C attention outputs $b_j$. This gives us second-level attention outputs $s_i$:

$$s_i = \sum_{j=1}^{M+1} \alpha^i_j b_j \in \mathbb{R}^l \quad \forall i \in \{1, \ldots, N\}$$
Co-attention: Results on SQUAD Competition

<table>
<thead>
<tr>
<th>Model</th>
<th>Dev EM</th>
<th>Dev F1</th>
<th>Test EM</th>
<th>Test F1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ensemble</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCN (Ours)</td>
<td>70.3</td>
<td>79.4</td>
<td>71.2</td>
<td>80.4</td>
</tr>
<tr>
<td>Microsoft Research Asia *</td>
<td>—</td>
<td>—</td>
<td>69.4</td>
<td>78.3</td>
</tr>
<tr>
<td>Allen Institute *</td>
<td>69.2</td>
<td>77.8</td>
<td>69.9</td>
<td>78.1</td>
</tr>
<tr>
<td>Singapore Management University *</td>
<td>67.6</td>
<td>76.8</td>
<td>67.9</td>
<td>77.0</td>
</tr>
<tr>
<td>Google NYC *</td>
<td>68.2</td>
<td>76.7</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td><strong>Single model</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCN (Ours)</td>
<td>65.4</td>
<td>75.6</td>
<td>66.2</td>
<td>75.9</td>
</tr>
<tr>
<td>Microsoft Research Asia *</td>
<td>65.9</td>
<td>75.2</td>
<td>65.5</td>
<td>75.0</td>
</tr>
<tr>
<td>Google NYC *</td>
<td>66.4</td>
<td>74.9</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Singapore Management University *</td>
<td>—</td>
<td>—</td>
<td>64.7</td>
<td>73.7</td>
</tr>
<tr>
<td>Carnegie Mellon University *</td>
<td>—</td>
<td>—</td>
<td>62.5</td>
<td>73.3</td>
</tr>
<tr>
<td>Dynamic Chunk Reader (Yu et al., 2016)</td>
<td>62.5</td>
<td>71.2</td>
<td>62.5</td>
<td>71.0</td>
</tr>
<tr>
<td>Match-LSTM (Wang &amp; Jiang, 2016)</td>
<td>59.1</td>
<td>70.0</td>
<td>59.5</td>
<td>70.3</td>
</tr>
<tr>
<td>Baseline (Rajpurkar et al., 2016)</td>
<td>40.0</td>
<td>51.0</td>
<td>40.4</td>
<td>51.0</td>
</tr>
<tr>
<td>Human (Rajpurkar et al., 2016)</td>
<td>81.4</td>
<td>91.0</td>
<td>82.3</td>
<td>91.2</td>
</tr>
</tbody>
</table>

Results are at time of ICLR submission
See [https://rajpurkar.github.io/SQuAD-explorer/](https://rajpurkar.github.io/SQuAD-explorer/) for latest results
FusionNet (Huang, Zhu, Shen, Chen 2017)

Attention functions

MLP (Additive) form:

\[ S_{ij} = s^T \tanh(W_1 c_i + W_2 q_j) \]

Space: \(O(mnk)\), \(W\) is \(k \times d\)

Bilinear (Product) form:

\[ S_{ij} = c_i^T W q_j \]
\[ S_{ij} = c_i^T U^T V q_j \]
\[ S_{ij} = c_i^T W^T D W q_j \]

1. Smaller space
2. Non-linearity

\[ S_{ij} = \text{Relu}(c_i^T W^T) \text{DRelu}(W q_j) \]
FusionNet tries to combine many forms of attention

Fully-aware Fusion Network

Fully-aware Self-boosted Fusion

Context Understanding

Question Understanding

When Context is long, Self-boosted Fusion can be used

Multi-level Fusion

Understanding

High-level Concept

Low-level Concept

Input Word

Word-level Fusion

Context

Question

7-Level History-of-Word Attention

3-Level History-of-Word Attention
Multi-level inter-attention

\( \{ m_{i}^{(k),C} \}_{i=1}^{m} = \text{Attn}(\{ \text{HoW}_{i}^{C} \}_{i=1}^{m}, \{ \text{HoW}_{i}^{Q} \}_{i=1}^{n}, \{ h_{i}^{Q,k} \}_{i=1}^{n}), 1 \leq k \leq K + 1 \)

\[
\text{HoW}_{i}^{C} = [\text{GloVe}(w_{i}^{C}); \text{BERT}_{w_{i}^{C}}; h_{i}^{C,1}; \ldots; h_{i}^{C,k}],
\]

\[
\text{HoW}_{i}^{Q} = [\text{GloVe}(w_{i}^{Q}); \text{BERT}_{w_{i}^{Q}}; h_{i}^{Q,1}; \ldots; h_{i}^{Q,k}].
\]

\[
S_{ij} = (\text{HoW}_{i}^{A})^{T}U^{T}V(\text{HoW}_{j}^{B})
\]

After multi-level inter-attention, use RNN, self-attention and another RNN to obtain the final representation of context: \( \{ u_{i}^{C} \} \)
Recent, more advanced architectures

- Most of the question answering work in 2016–2018 employed progressively more complex architectures with a multitude of variants of attention – often yielding good task gains

<table>
<thead>
<tr>
<th>Architectures</th>
<th>(1)</th>
<th>(2)</th>
<th>(2’)</th>
<th>(3)</th>
<th>(3’)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Match-LSTM (Wang and Jiang, 2016)</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>DCN (Xiong et al., 2017)</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>FastQA (Weissenborn et al., 2017)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>FastQAExt (Weissenborn et al., 2017)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>BiDAF (Seo et al., 2017)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>RaSoR (Lee et al., 2016)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>DrQA (Chen et al., 2017)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>MPCM (Wang et al., 2016)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Mnemonic Reader (Hu et al., 2017)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>R-net (Wang et al., 2017b)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
SQuAD limitations

• SQuAD has a number of key limitations:
  • Only span-based answers (no yes/no, counting, implicit why)
  • Questions were constructed looking at the passages
    • Not genuine information needs
    • Generally greater lexical and syntactic matching between questions and answer span than you get IRL
  • Barely any multi-fact/sentence inference beyond coreference

• Nevertheless, it is a well-targeted, well-structured, clean dataset
  • It has been the most used and competed on QA dataset
  • It has also been a useful starting point for building systems in industry (though in-domain data always really helps!)
  • And we’re using it (SQuAD 2.0)
5. Open-domain Question Answering


Q: How many of Warsaw's inhabitants spoke Polish in 1933?

833,500
For 70–86% of questions, the answer segment appears in the top 5 articles
DrQA Demo

Hello! Please ask a question.

What is question answering?

a computer science discipline within the fields of information retrieval and natural language processing

Who was the winning pitcher in the 1956 World Series?

Don Larsen

What is the answer to life, the universe, and everything?

42
General questions

Combined with Web search, DrQA can answer 57.5% of trivia questions correctly

Q: The Dodecanese Campaign of WWII that was an attempt by the Allied forces to capture islands in the Aegean Sea was the inspiration for which acclaimed 1961 commando film?
A: The Guns of Navarone

Q: American Callan Pinckney’s eonymously named system became a best-selling (1980s-2000s) book/video franchise in what genre?
A: Fitness
6. LSTMs, attention, and transformers intro
## SQuAD v1.1 leaderboard, 2019-02-07

<table>
<thead>
<tr>
<th>Rank</th>
<th>Model</th>
<th>EM</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Human Performance</td>
<td>82.304</td>
<td>91.221</td>
</tr>
<tr>
<td></td>
<td><em>Stanford University</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>(Rajpurkar et al. '16)</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>BERT (ensemble)</td>
<td>87.433</td>
<td>93.160</td>
</tr>
<tr>
<td></td>
<td><em>Google AI Language</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>BERT (single model)</td>
<td>85.083</td>
<td>91.835</td>
</tr>
<tr>
<td></td>
<td><em>Google AI Language</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>nlnet (ensemble)</td>
<td>85.356</td>
<td>91.202</td>
</tr>
<tr>
<td></td>
<td><em>Microsoft Research Asia</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>nlnet (ensemble)</td>
<td>85.954</td>
<td>91.677</td>
</tr>
<tr>
<td></td>
<td><em>Microsoft Research Asia</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>QANet (ensemble)</td>
<td>84.454</td>
<td>90.490</td>
</tr>
<tr>
<td></td>
<td><em>Google Brain &amp; CMU</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>r-net (ensemble)</td>
<td>84.003</td>
<td>90.147</td>
</tr>
<tr>
<td></td>
<td><em>Microsoft Research Asia</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>QANet (ensemble)</td>
<td>83.877</td>
<td>89.737</td>
</tr>
<tr>
<td></td>
<td><em>Google Brain &amp; CMU</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>nlnet (single model)</td>
<td>83.468</td>
<td>90.133</td>
</tr>
<tr>
<td></td>
<td><em>Microsoft Research Asia</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>nlnet (ensemble)</td>
<td>85.954</td>
<td>91.677</td>
</tr>
<tr>
<td></td>
<td><em>Microsoft Research Asia</em></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Note: The leaderboard is as of 2019-02-07. Updated scores and rankings may vary.*
Gated Recurrent Units, again

*Intuitively, what happens with RNNs?*

1. Measure the influence of the past on the future

\[
\frac{\partial \log p(x_{t+n} | x < t+n)}{\partial h_t} = \frac{\partial \log p(x_{t+n} | x < t+n)}{\partial g} \frac{\partial g}{\partial h_{t+n}} \frac{\partial h_{t+n}}{\partial h_{t+n-1}} \cdots \frac{\partial h_{t+1}}{\partial h_t}
\]

2. How does the perturbation at \( t \) affect \( p(x_{t+n} | x < t+n) \)?

---

\( x_t \)

\( \epsilon \)

\( p(\text{the}) \)

\( p(\text{cat} | \ldots) \)

\( p(\text{is} | \ldots) \)

\( p(\text{eating} | \ldots) \)
Gated Recurrent Units: LSTM & GRU

• The signal and error must propagate through all the intermediate nodes:

• Perhaps we can create shortcut connections.
Gated Recurrent Unit

- Perhaps we can create *adaptive* shortcut connections.
- Let the net prune unnecessary connections *adaptively*.

\[ f(h_{t-1}, x_t) = u_t \odot \tilde{h}_t + (1 - u_t) \odot h_{t-1} \]

- **Candidate Update** \( \tilde{h}_t = \tanh(W [x_t] + U (r_t \odot h_{t-1}) + b) \)
- **Reset gate** \( r_t = \sigma(W_r [x_t] + U_r h_{t-1} + b_r) \)
- **Update gate** \( u_t = \sigma(W_u [x_t] + U_u h_{t-1} + b_u) \)

\( \odot \): element-wise multiplication
Gated Recurrent Unit

tanh-RNN ....

Execution

1. Read the whole register $h$

2. Update the whole register

$$h \leftarrow \tanh(W[x] + Uh + b)$$
Gated Recurrent Unit

GRU ...

Execution

1. Select a readable subset $r$
2. Read the subset $r \odot h$
3. Select a writable subset $u$
4. Update the subset
   
   $$h \leftarrow u \odot \tilde{h} + (1 - u_t) \odot h$$

Gated recurrent units are much more realistic for computation!
Gated Recurrent Units: LSTM & GRU

Two most widely used gated recurrent units: GRU and LSTM

Gated Recurrent Unit
[Cho et al., EMNLP2014; Chung, Gulcehre, Cho, Bengio, DLUFL2014]

\[ h_t = u_t \odot \tilde{h}_t + (1 - u_t) \odot h_{t-1} \]
\[ \tilde{h}_t = \tanh(W \left[ x_t \right] + U (r_t \odot h_{t-1}) + b) \]
\[ u_t = \sigma(W_u \left[ x_t \right] + U_u h_{t-1} + b_u) \]
\[ r_t = \sigma(W_r \left[ x_t \right] + U_r h_{t-1} + b_r) \]

Long Short-Term Memory
[Hochreiter & Schmidhuber, NC1999; Gers, Thesis2001]

\[ h_t = o_t \odot \tanh(c_t) \]
\[ c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \]
\[ \tilde{c}_t = \tanh(W_c \left[ x_t \right] + U_c h_{t-1} + b_c) \]
\[ o_t = \sigma(W_o \left[ x_t \right] + U_o h_{t-1} + b_o) \]
\[ i_t = \sigma(W_i \left[ x_t \right] + U_i h_{t-1} + b_i) \]
\[ f_t = \sigma(W_f \left[ x_t \right] + U_f h_{t-1} + b_f) \]
**Attention Mechanism**

- A second solution: random access memory
  - Retrieve past info as needed (but usually average)
  - Usually do content-similarity based addressing
  - Other things like positional are occasionally tried

Started in computer vision! [Larochelle & Hinton, 2010], [Denil, Bazzani, Larochelle, Freitas, 2012]
Became famous in NMT/NLM
ELMo and BERT preview

Contextual word representations
Using language model-like objectives

The transformer architecture used in BERT is sort of attention on steroids.

Look at SDNet as an example of how to use BERT as submodule: https://arxiv.org/abs/1812.03593

(Vaswani et al, 2017)
The Motivation for Transformers

- We want parallelization but RNNs are inherently sequential

![LSTM Diagram]

- Despite LSTMs, RNNs generally need attention mechanism to deal with long range dependencies – path length between states grows with distance otherwise
- But if attention gives us access to any state... maybe we can just use attention and don’t need the RNN? 😐
- And then NLP can have deep models ... and solve our vision envy
Transformer (Vaswani et al. 2017) “Attention is all you need”

- **Non-recurrent** sequence (or sequence-to-sequence) model
- A **deep** model with a sequence of **attention**-based transformer blocks
- Depth allows a certain amount of lateral information transfer in understanding sentences, in slightly unclear ways
- Final cost/error function is standard cross-entropy error on top of a softmax classifier

Initially built for NMT
Transformer block

Each block has two “sublayers”

1. Multihead attention
2. 2-layer feed-forward NNet (with ReLU)

Each of these two steps also has:
Residual (short-circuit) connection
LayerNorm (scale to mean 0, var 1; Ba et al. 2016)
Multi-head (self) attention

With simple self-attention: Only one way for a word to interact with others

Solution: Multi-head attention

Map input into \( h = 12 \) many lower dimensional spaces via \( W_h \) matrices

Then apply attention, then concatenate outputs and pipe through linear layer

\[
\text{Multihead}(x_i^{(t)}) = \text{Concat}(\text{head}_j)W^O
\]

\[
\text{head}_j = \text{Attention}(x_i^{(t)}W_j^Q, x_i^{(t)}W_j^K, x_i^{(t)}W_j^V)
\]

So attention is like bilinear: \( x_i^{(t)}(W_j^Q(W_j^K)^T)x_i^{(l)} \)
Encoder Input

Actual word representations are word pieces (byte pair encoding)
  • Topic of next week

Also added is a **positional encoding** so same words at different locations have different overall representations:

\[
P E_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})
\]

\[
P E_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})
\]
BERT (Bidirectional Encoder Representations from Transformers): Pre-training of Deep Bidirectional Transformers for Language Understanding, which is then fine-tuned for a particular task

Pre-training uses a cloze task formulation where 15% of words are masked out and predicted:

`store ↑ gallon ↑`

the man went to the [MASK] to buy a [MASK] of milk
Transformer (Vaswani et al. 2017)
BERT (Devlin et al. 2018)
7. Pots of data

- Many publicly available datasets are released with a train/dev/test structure. *We're all on the honor system to do test-set runs only when development is complete.*
- Splits like this presuppose a fairly large dataset.
- If there is no dev set or you want a separate tune set, then you create one by splitting the training data, though you have to weigh its size/usefulness against the reduction in train-set size.
- Having a fixed test set ensures that all systems are assessed against the same gold data. This is generally good, but it is problematic where the test set turns out to have unusual properties that distort progress on the task.
Training models and pots of data

• When training, models **overfit** to what you are training on
  • The model correctly describes what happened to occur in particular data you trained on, but the patterns are not general enough patterns to be likely to apply to new data
• The way to monitor and avoid problematic overfitting is using **independent** validation and test sets ...
Training models and pots of data

• You build (estimate/train) a model on a training set.
• Often, you then set further hyperparameters on another, independent set of data, the tuning set
  • The tuning set is the training set for the hyperparameters!
• You measure progress as you go on a dev set (development test set or validation set)
  • If you do that a lot you overfit to the dev set so it can be good to have a second dev set, the dev2 set
• Only at the end, you evaluate and present final numbers on a test set
  • Use the final test set extremely few times ... ideally only once
Training models and pots of data

• The **train**, **tune**, **dev**, and **test** sets need to be completely distinct

• It is invalid to test on material you have trained on
  • You will get a falsely good performance. We usually overfit on train

• You need an independent tuning set
  • The hyperparameters won’t be set right if tune is same as train

• If you keep running on the same evaluation set, you begin to overfit to that evaluation set
  • Effectively you are “training” on the evaluation set ... you are learning things that do and don’t work on that particular eval set and using the info

• To get a valid measure of system performance you need another untrained on, **independent** test set ... hence dev2 and final test
8. Getting your neural network to train

• Start with a positive attitude!
  • **Neural networks want to learn!**
    • If the network isn’t learning, you’re doing something to prevent it from learning successfully

• Realize the grim reality:
  • **There are lots of things that can cause neural nets to not learn at all or to not learn very well**
    • Finding and fixing them ("debugging and tuning") can often take more time than implementing your model

• It’s hard to work out what these things are
  • But experience, experimental care, and rules of thumb help!
Models are sensitive to learning rates

- From Andrej Karpathy, CS231n course notes
Models are sensitive to initialization

- From Michael Nielsen
Training a gated RNN

1. Use an LSTM or GRU: *it makes your life so much simpler!*
2. Initialize recurrent matrices to be orthogonal
3. Initialize other matrices with a sensible (**small**) scale
4. Initialize forget gate bias to 1: *default to remembering*
5. Use adaptive learning rate algorithms: *Adam, AdaDelta, …*
6. Clip the norm of the gradient: *1–5 seems to be a reasonable threshold when used together with Adam or AdaDelta.*
7. Either only dropout vertically or look into using Bayesian Dropout (Gal and Gahramani – not natively in PyTorch)
8. *Be patient! Optimization takes time*

[Saxe et al., ICLR2014; Ba, Kingma, ICLR2015; Zeiler, arXiv2012; Pascanu et al., ICML2013]
Experimental strategy

- Work incrementally!
- Start with a very simple model and get it to work!
  - It’s hard to fix a complex but broken model
- Add bells and whistles one-by-one and get the model working with each of them (or abandon them)

- Initially run on a tiny amount of data
  - You will see bugs much more easily on a tiny dataset
  - Something like 4–8 examples is good
  - Often synthetic data is useful for this
  - Make sure you can get 100% on this data
    - Otherwise your model is definitely either not powerful enough or it is broken
Experimental strategy

• Run your model on a large dataset
  • It should still score close to 100% on the training data after optimization
  • Otherwise, you probably want to consider a more powerful model
  • Overfitting to training data is not something to be scared of when doing deep learning
    • These models are usually good at generalizing because of the way distributed representations share statistical strength regardless of overfitting to training data
• But, still, you now want good generalization performance:
  • Regularize your model until it doesn’t overfit on dev data
    • Strategies like L2 regularization can be useful
    • But normally generous dropout is the secret to success
Details matter!

- Look at your data, collect summary statistics
- Look at your model’s outputs, do error analysis
- Tuning hyperparameters is really important to almost all of the successes of NNets
Good luck with your projects!