Overview of the Document Reader Question Answering

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

Good source code available!
Big idea

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. As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L"), so that the logo could prominently feature the Arabic numerals 50.

Q: Which NFL team represented the AFC at Super Bowl 50?

A: Denver Broncos
Start and End Probabilities

\[ P_{\text{start}}(i) \propto \exp \{ \vec{p}_i W_s \vec{q} \} \]  
\[ P_{\text{end}}(i) \propto \exp \{ \vec{p}_i W_e \vec{q} \} \]

1. A vector representing our question
2. Vector representing each word in the query text
3. Parameter: here’s the start/end of the answer
Start and End Probabilities

\[ P_{\text{start}}(i) \propto \exp \{ \vec{p}_i \, W_s \, \vec{q} \} \]  
\[ P_{\text{end}}(i) \propto \exp \{ \vec{p}_i \, W_e \, \vec{q} \} \]

1. A vector representing our question
2. Vector representing each word in the query text
3. Parameter: here’s the start/end of the answer
Start and End Probabilities

\[ P_{\text{start}}(i) \propto \exp \{ \tilde{p}_i W_s \tilde{q} \} \]  \hspace{1cm} (1)

\[ P_{\text{end}}(i) \propto \exp \{ \tilde{p}_i W_e \tilde{q} \} \]  \hspace{1cm} (2)

1. A vector representing our question
2. Vector representing each word in the query text
3. Parameter: here’s the start/end of the answer
Start and End Probabilities

\[ P_{\text{start}}(i) \propto \exp \{ \tilde{p}_i W_s \tilde{q} \} \]  
\[ P_{\text{end}}(i) \propto \exp \{ \tilde{p}_i W_e \tilde{q} \} \]

1. A vector representing our question
2. Vector representing each word in the query text
3. Parameter: here’s the start/end of the answer
Start and End Probabilities

\[ P_{\text{start}}(i) \propto \exp \{ \vec{p}_i W_s \vec{q} \} \]  
\[ P_{\text{end}}(i) \propto \exp \{ \vec{p}_i W_e \vec{q} \} \]  

1. A vector representing our question
2. Vector representing each word in the query text
3. Parameter: here’s the start/end of the answer
Start and End Probabilities

\[ P_{\text{start}}(i) \propto \exp \{ \vec{p}_i W_s \vec{q} \} \]  \hspace{1cm} (1)

\[ P_{\text{end}}(i) \propto \exp \{ \vec{p}_i W_e \vec{q} \} \]  \hspace{1cm} (2)

1. A vector representing our question
2. Vector representing each word in the query text
3. Parameter: here’s the start/end of the answer

This is your objective function! Will backprop into each of these parameters.
Question Encoding

\[ \hat{q} = \sum_{j} b_j \hat{q}_j \quad (3) \]

\[ b_j = \frac{\exp\left\{ \hat{w} \cdot q_j \right\}}{\sum_{j'} \exp\left\{ w \cdot q_{j'} \right\}} \quad (4) \]
Question Encoding

\[
\hat{q} = \sum_j b_j \hat{q}_j \tag{3}
\]

\[
b_j = \frac{\exp\{\hat{w} \cdot q_j\}}{\sum_{j'} \exp\{w \cdot q_{j'}\}} \tag{4}
\]

Question vector is a weighted sum
Question Encoding

\[ \tilde{q} = \sum_j b_j \tilde{q}_j \quad (3) \]

\[ b_j = \frac{\exp\{ \tilde{\mathbf{w}} \cdot \mathbf{q}_j \} \sum_{j'} \exp\{ \mathbf{w} \cdot \mathbf{q}_{j'} \}} {\sum_{j'} \exp\{ \mathbf{w} \cdot \mathbf{q}_{j'} \} } \quad (4) \]

The weight is a scalar
Question Encoding

\[ \tilde{q} = \sum_j b_j \tilde{q}_j \]  
\[ b_j = \frac{\exp \{ \tilde{w} \cdot q_j \}}{\sum_{j'} \exp \{ w \cdot q_{j'} \}} \]

A focus parameter learns how to focus on particular words in the question
Paragraph Encoding

- Word Embedding
- Exact Match
- Token Features
- Question Alignment
Paragraph Encoding

- Word Embedding
- Exact Match
- Token Features
- Question Alignment

pre-trained word embedding
Paragraph Encoding

Part of speech, NER tags, normalized term frequency
Who is the leader of the US
Donald Trump is the president of the United States

\[ a_{i,j} = \frac{\exp\{\vec{E}(p_i) \cdot E(q_j)\}}{\sum_{j'} E(p_i) \cdot E(q_{j'})} \]
Who is the leader of the US
Donald Trump is the president of the United States

$$a_{i,j} = \frac{\exp\{\tilde{E}(p_i) \cdot E(q_j)\}}{\sum_{j'} E(p_i) \cdot E(q_{j'})}$$
Paragraph Encoding

Create learned representations
Paragraph Encoding

LSTM: encode contextual effects
Paragraph Encoding

Add a backwards direction as well (bi-directional LSTM)
Paragraph Encoding

Use the concatenation of these two hidden layers as the representation of the word.
Paragraph Encoding

\[ P_{\text{start}}(i) \propto \exp\{\tilde{p}_i W_s \tilde{q}\} \]

\[ P_{\text{end}}(i) \propto \exp\{\tilde{p}_i W_e \tilde{q}\} \]
Implementation

- Trained on passages
- Backprop through all layers
- Look at code

```python
# RNN document encoder
self.doc_rnn = layers.StackedBRNN(
    input_size=doc_input_size,
    hidden_size=hidden_size,
    num_layers=doc_layers,
    dropout_rate=dropout_rnn,
    dropout_output=dropout_rnn_output,
    concat_layers=concat_rnn_layers,
    rnn_type=self.RNN_TYPES[nn.rnn_type],
    padding=nn.rnn_padding,
)

# RNN question encoder
self.question_rnn = layers.StackedBRNN(
    input_size=embedding_dim,
    hidden_size=hidden_size,
    num_layers=question_layers,
    dropout_rate=dropout_rnn,
    dropout_output=dropout_rnn_output,
    concat_layers=concat_rnn_layers,
    rnn_type=self.RNN_TYPES[nn.rnn_type],
    padding=nn.rnn_padding,
)
```

https://github.com/facebookresearch/DrQA/
More complicated models

Output Layer
Modeling Layer
Attention Flow Layer
Phrase Embed Layer
Word Embed Layer
Character Embed Layer

Dense + Softmax

LSTM + Softmax

Start

End

m₁

m₂

mₜ

LSTM

Query2Context

and

Context2Query

Attention

h₁

h₂

hₜ

LSTM

x₁

x₂

x₃

xₜ

Query

q₁

qⱼ

LSTM

u₁

uⱼ

g₁

g₂

gₜ