CS230: Lecture 8
Word2Vec applications + Recurrent Neural Networks with Attention
Kian Katanforoosh, Andrew Ng
We will learn how to:

- **Generalize** results with word vectors
- Augment a RNN with **Attention mechanisms**

I. Word Vector Representation
   i. Training
   ii. Operations
   iii. Applications: debasing / restaurant reviews

II. Attention
   i. Machine Translation
   ii. Image Captioning

Today’s outline
How do you get the vector representation?
1. Data?

Stanford is going to beat Cal next week

<table>
<thead>
<tr>
<th>Target word (x)</th>
<th>Nearby word (y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stanford</td>
<td>is</td>
</tr>
<tr>
<td>is</td>
<td>Stanford</td>
</tr>
<tr>
<td>is</td>
<td>going</td>
</tr>
<tr>
<td>going</td>
<td>is</td>
</tr>
<tr>
<td>going</td>
<td>to</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

\[
\text{loss? } L = -\sum y \log(\hat{y})
\]

\[
\mathbf{z}^{[1]} = \mathbf{W}^{[1]} \mathbf{v} + \mathbf{b}^{[1]}
\]

\[
\begin{pmatrix}
\vdots & \vdots & \ldots & \vdots & \vdots \\
0 & 0 & \ldots & 0 & 1 \\
0 & \vdots & \vdots & \vdots & \vdots \\
0 & \vdots & \vdots & \vdots & \vdots \\
\end{pmatrix}
\]

Kian Katanforoosh, Andrew Ng
Word Vector Representation: Embedding matrix

\[
W^{[1]}v_{\text{football}} = \begin{pmatrix}
\vdots & \vdots & \ldots & \vdots & \vdots \\
e_{\text{a}} & e_{\text{abs}} & \ldots & e_{\text{zone}} & e_{\text{zoo}} \\
\vdots & \vdots & \ldots & \vdots & \vdots \\
0 & 1 & 0 & \vdots & 0
\end{pmatrix}
\begin{pmatrix}
0 \\
\vdots \\
0
\end{pmatrix}
= e_{\text{football}}
\]
Scatter plot of Word vectors

Operations on vectors

Expression | Nearest token
--- | ---
Paris - France + Italy | Rome
bigger - big + cold | colder
sushi - Japan + Germany | bratwurst
Cu - copper + gold | Au
Windows - Microsoft + Google | Android
Montreal Canadiens - Montreal + Toronto | Toronto Maple Leafs

$\text{man} - \text{king} = \text{women} - x$

$x = \text{queen}$
**Sexist bias**

\[
\text{man} - \text{king} = \text{women} - x \\
x = \text{queen}
\]

but also …

\[
\text{man} - \text{woman} = \text{computer programmer} - \text{homemaker}
\]

---

**Extreme she occupations**

1. homemaker 2. nurse 3. receptionist
4. librarian 5. socialite 6. hairdresser
7. nanny 8. bookkeeper 9. stylist
10. housekeeper 11. interior designer 12. guidance counselor

**Extreme he occupations**

1. maestro 2. skipper 3. protege
4. philosopher 5. captain 6. architect
7. financier 8. warrior 9. broadcaster
10. magician 11. figher pilot 12. boss

---

Figure 2: Analogy examples. Examples of automatically generated analogies for the pair *she-he* using the procedure described in text. For example, the first analogy is interpreted as *she:sewing :: he:carpentry* in the original w2vNEWS embedding. Each automatically generated analogy is evaluated by 10 crowd-workers are to whether or not it reflects gender stereotype. Top: Illustrative gender stereotypic analogies automatically generated from w2vNEWS, as rated by at least 5 of the 10 crowd-workers. Bottom: Illustrative generated gender-appropriate analogies.
Word Vector Representation: Application

Sentiment analysis on restaurant reviews

1. Data?
   50 reviews

2. Architecture?
   "Tonight" $\rightarrow$ $v_{\text{tonight}}$ $\rightarrow$ $W^{[1]} \times \ldots$ $\rightarrow$ $e$ $\rightarrow$ $\sigma(We+b)$ $\rightarrow$ 0.74 $>0.5$ $\rightarrow$ 1
   "was" $\rightarrow$ $v_{\text{was}}$ $\rightarrow$ $W^{[1]} \times \ldots$ $\rightarrow$ $\sigma(We+b)$ $\rightarrow$ 0.12 $0.52$
   "awesome" $\rightarrow$ $v_{\text{awesome}}$ $\rightarrow$ $W^{[1]} \times \ldots$ $\rightarrow$ $\sigma(We+b)$ $\rightarrow$ 0.23 $0.11$

3. Loss?
   $L = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$

Generalizes very well because of word vector representations

Review (x) | Label (Negative/Positive)
---|---
"Tonight was awesome" | 1
"Worst entrée ever!" | 0
.... | ....
To remember:
- In NLP, Words are often represented by “meaningful” vectors
- These vectors are trained thanks to a Neural Network
- We can do operations on these vectors
- They can be biased, depending on the dataset used to train them
- They have a great generalization power
Neural Machine Translation

Encoder

Decoder

Attention: Motivation
Attention: Motivation

Inverting input works better?!!

Encoder

Decoder

[kitchen] [the] [in] [is] [Brian] <eos>

"Brian" "est" "dans" "la" "cuisine" <eos>
Some problems:
- The encoder encodes all information in the source sentence into a fixed length vector
- While it seems that some specific parts of the source sentence are more useful to predict some parts of the output sentence
- Bad performances on long sentences

Ideas:
- We’d like to spread the information encoded from the source sentence and selectively retrieve the relevant parts at each prediction of the output sentence
- Why don’t we use every hidden state $h_j$ from the encoding part?
Attention: Motivation

\[ c_1 = \sum_{j=1}^{6} \alpha_{1,j} h_j \]

Kian Katanforoosh, Andrew Ng
Attention: Motivation

\[ c_k = \sum_{j=1}^{6} \alpha_{k,j} h_j \]

\[ h_4 = \text{contains information about the input sentence up to “is” with a stronger focus on the parts closer to “is”} \]

\[ \alpha_{2,3} = f(s_1, h_3) \]

\[ = \text{probability that the target word } y_2 (“est”) \text{ is translated from source word } x_3 (“in”) \]

\[ = \text{score that went through a softmax function} \]

\[ c_1 = \text{the expected annotation over all annotations with probabilities } \alpha_{1,j} \]

“kitchen” “the” “in” “is” “Brian” <eos>
Neural Machine Translation with Attention: Architecture

$c_i$ = the expected annotation over all annotations with probabilities $\alpha_{i,j}$

$\alpha_{i,j} = f(s_{i-1}, h_j)$
Neural Machine Translation with Attention: Architecture

\[ \argmax_i c_i \text{ is the expected annotation over all annotations with probabilities } \alpha_{i,j} \]

\[ \alpha_{i,j} = f(s_{i-1}, h_j) \]
Neural Machine Translation with Attention: Architecture

\[ \alpha_{i,j} = f(s_{i-1}, h_j) \]

c_i = the expected annotation over all annotations with probabilities \( \alpha_{i,j} \)

Kian Katanforoosh, Andrew Ng
Neural Machine Translation with Attention: Architecture

```
LSTM → LSTM → LSTM → LSTM → LSTM → LSTM → ... 
```

```
softmax  softmax  softmax  softmax  softmax  softmax  ... 
```

```
argmax  argmax  argmax  argmax  argmax  argmax  ... 
```

```
[0.11]  [0.11]  [0.11]  [0.11]  [0.11]  [0.11]  ... 
```

```
[0.02]  [0.02]  [0.02]  [0.02]  [0.02]  [0.02]  ... 
```

```
h_1  h_2  h_3  h_4  h_5  h_6  ... 
```

```
c_4  s_3  s_4  ... 
```

```
Attention mechanism 
```

```
“Brian”  “est”  “dans”  “la”  
```

```
s_1  s_2  s_3  s_4  ... 
```

Kian Katanforoosh, Andrew Ng
Neural Machine Translation with Attention: Architecture

How to train this?

same as Machine Translation but takes also derivatives with respect to attention parameters
Neural Machine Translation with Attention: Training

What are the parameters?

**Encoder**

\[ W^{[1]} = \text{EmbeddingMatrix} \]

**LSMT:**

\[
\begin{align*}
    f_i &= \sigma(W_f[h_{t-1}, x_t] + b_f) \\
    i_i &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\
    C_t &= \tanh(W_C[h_{t-1}, x_t] + b_C) \\
    C_t &= f_i \circ C_{t-1} + i_i \circ \tilde{C}_t \\
    o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\
    h_t &= o_t \circ \tanh(C_t)
\end{align*}
\]

**Attention**

\[
\begin{align*}
    c_i &= \sum_{j=1}^{T_x} \alpha_{ij} h_j \\
    \alpha_{ij} &= \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \\
    e_{ij} &= v^T_a \tanh(W_a s_{t-1} + U_a h_j)
\end{align*}
\]

**Decoder**

**LSMT:**

\[
\begin{align*}
    f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \\
    i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\
    \tilde{C}_t &= \tanh(W_C[h_{t-1}, x_t] + b_C) \\
    C_t &= f_t \circ C_{t-1} + i_t \circ \tilde{C}_t \\
    o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\
    h_t &= o_t \circ \tanh(C_t)
\end{align*}
\]

**Loss function:**

\[
L = -\sum_{\text{batch}} \left( \log P(y|x; \theta) + \lambda \sum_{i \in \text{patch}} \left( 1 - \sum_{t \in \text{output}} \alpha_{ii} \right) \right)
\]
Figure 3: Four sample alignments found by RNNsearch-50. The x-axis and y-axis of each plot correspond to the words in the source sentence (English) and the generated translation (French), respectively. Each pixel shows the weight $\alpha_{ij}$ of the annotation of the $j$-th source word for the $i$-th target word (see Eq. (6)), in grayscale (0: black, 1: white). (a) an arbitrary sentence. (b–d) three randomly selected samples among the sentences without any unknown words and of length between 10 and 20 words from the test set.
Image Captioning with Attention

Neural Machine Translation

Encoder

Decoder

Kian Katanforoosh, Andrew Ng
Image Captioning with Attention

Image Captioning

Encoder

Decoder

Kian Katanforoosh, Andrew Ng
Image Captioning with Attention

Image Captioning with no attention

Encoder

CNN

image

$h_6$

Decoder

softmax softmax softmax softmax softmax softmax softmax

“a” “bird” “flying” “over” “a” ....

argmax argmax argmax argmax argmax argmax

0.11 0.11 0.11 0.11 0.11 0.11
0.02 0.02 0.02 0.02 0.02 0.02

LSTM → LSTM → LSTM → LSTM → LSTM → LSTM → LSTM

Vinyals et al.: Show and Tell: A Neural Image Caption Generator

Kian Katanforoosh, Andrew Ng
Image Captioning with Attention

Image Captioning with attention

Encoder

CNN

\{a_1, a_2, \ldots, a_L \}

Decoder

“A” \rightarrow \text{argmax} \rightarrow \text{softmax} \rightarrow \text{LSTM}

“bird” \rightarrow \text{argmax} \rightarrow \text{softmax} \rightarrow \text{LSTM}

“is” \rightarrow \text{argmax} \rightarrow \text{softmax} \rightarrow \text{LSTM}

“flying” \rightarrow \text{argmax} \rightarrow \text{softmax} \rightarrow \text{LSTM}

average

s_1 \rightarrow \text{Attention mechanism} \rightarrow a_1 \\

s_2 \rightarrow \text{Attention mechanism} \rightarrow a_2 \\

s_3 \rightarrow \text{Attention mechanism} \rightarrow a_3 \\

s_4 \rightarrow \text{Attention mechanism} \rightarrow a_4 \\

s_5 \rightarrow \text{Attention mechanism} \rightarrow a_5 \\

s_6 \rightarrow \text{Attention mechanism} \rightarrow a_6

Kian Katanforoosh, Andrew Ng

Vinyals et al. : Show and Tell: A Neural Image Caption Generator
Kelvin Xu et al. : Show, Attend and Tell: Neural Image Caption Generation with Visual Attention