Neural Network Approaches to Representation Learning for NLP

Navid Rekabsaz
Idiap Research Institute
Agenda

- Brief Intro to Deep Learning
  - Neural Networks

- Word Representation Learning
  - Neural word representation
  - Word2vec with Negative Sampling
  - Bias in word representation learning

---Break---

- Recurrent Neural Networks
- Attention Networks
- Document Classification with DL
Agenda

- **Brief Intro to Deep Learning**
  - Neural Networks
- **Word Representation Learning**
  - Neural word representation
  - word2vec with Negative Sampling
  - Bias in word representation learning

--- Break ---

- Recurrent Neural Networks
- Attention Networks
- Document Classification with DL
Recap on Linear Algebra

- Scalar \( a \)
- Vector \( \vec{b} \)
- Matrix \( W \)
- Tensor: generalization to higher dimensions
- Dot product
  - \( \vec{a} \cdot \vec{b}^T = c \)
    - dimensions: \( 1 \times d \cdot d \times 1 = 1 \)
  - \( \vec{a} \cdot W = \vec{c} \)
    - dimensions: \( 1 \times d \cdot d \times e = 1 \times e \)
  - \( A \cdot B = C \)
    - dimensions: \( l \times m \cdot m \times n = l \times n \)
- Element-wise Multiplication
  - \( \vec{a} \odot \vec{b} = \vec{c} \)
Neural Networks

- Neural Networks are non-linear functions with many parameters
  \[ \hat{y} = f(\hat{x}) \]
- They consist of several simple non-linear operations
- Normally, the objective is to maximize likelihood, namely
  \[ p(y|x, \theta) \]
- Generally optimized using Stochastic Gradient Descent (SGD)
Neural Networks – Training with SGD (simplified)

Initialize parameters

Loop over training data (or minibatches)
1. Do forward pass: given input $\vec{x}$ predict output $\hat{y}$
2. Calculate loss function by comparing $\hat{y}$ with labels $y$
3. Do backpropagation: calculate the gradient of each parameter in regard to the loss function
4. Update parameters in the direction of gradient
5. Exit if some stopping criteria are met
Neural Networks – Non-linearities

- **Sigmoid**
  - Projects input to value between 0 to 1 → becomes like a probability value

- **ReLU (Rectified Linear Units)**
  - Suggested for deep architectures to prevent vanishing gradient

- **Tanh**
Softmax turns a vector to a probability distribution
- The vector values become in the range of 0 to 1 and sum of all the values is equal 1

\[
\text{softmax}(\vec{v})_i = \frac{e^{v_i}}{\sum_{k=1}^{d} e^{v_k}}
\]

Normally applied to the output layer and provide a probability distribution over output classes

For example, given four classes:
\[
\hat{y} = [2, 3, 5, 6] \quad \text{softmax}(\hat{y}) = [0.01, 0.03, 0.26, 0.70]
\]
Deep Learning models the overall function as a composition of functions (layers)

- With several algorithmic and architectural innovations
  - dropout, LSTM, Convolutional Networks, Attention, GANs, etc.

- Backed by large datasets, large-scale computational resources, and enthusiasm from academia and industry!

Agenda

- Brief Intro to Deep Learning
  - Neural Networks
- Word Representation Learning
  - Neural word representation
  - word2vec with Negative Sampling
  - Bias in word representation learning

---Break---

- Recurrent Neural Networks
- Attention Networks
- Document Classification with DL
Vector Representation (Recall)

- Computation starts with representation of entities
- An entity is represented with a vector of $d$ dimensions
- The dimensions usually reflects features, related to an entity
- When vector representations are dense, they are often referred to as embedding e.g. word embedding

$\vec{v} \quad x_0 \quad x_1 \quad x_2 \quad \ldots \quad x_d$
Word Representation Learning

Corpus → Word Embedding Model → Words

$w_1$, $w_2$, …, $w_n$ → $d$
Vector representations of words projected in two-dimensional space
Intuition for Computational Semantics

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

J. R. Firth, A synopsis of linguistic theory 1930–1955 (1957)
Tesgüino

- alcoholic
- beverage
- fermented
- out of corn
- Mexico
- bottle
- drink
- sacred
Ale
Tesgüino ←→ Ale

Algorithmic intuition:
Two words are related when they share many context words.
Word-Context Matrix (Recall)

- Number of times a word \( c \) appears in the context of the word \( w \) in a corpus

sugar, a sliced lemon, a tablespoonful of their enjoyment. Cautiously she sampled her first well suited to programming on the digital for the purpose of gathering data and apricot pineapple computer. information preserve or jam, a pinch each of, and another fruit whose taste she likened In finding the optimal R-stage policy from necessary for the study authorized in the

<table>
<thead>
<tr>
<th>( c_1 )</th>
<th>( c_2 )</th>
<th>( c_3 )</th>
<th>( c_4 )</th>
<th>( c_5 )</th>
<th>( c_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aardvark</td>
<td>computer</td>
<td>data</td>
<td>pinch</td>
<td>result</td>
<td>sugar</td>
</tr>
<tr>
<td>( w_1 ) apricot</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( w_2 ) pineapple</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( w_3 ) digital</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( w_4 ) information</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

- Our first word vector representation!!
Words Semantic Relations (Recall)

<table>
<thead>
<tr>
<th></th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_4$</th>
<th>$c_5$</th>
<th>$c_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aardvark</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>computer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pinch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>result</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sugar</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$w_1$ apricot  0 0 0 1 0 1

$w_2$ pineapple 0 0 0 1 0 1

$w_3$ digital 0 2 1 0 1 0

$w_4$ information 0 1 6 0 4 0

- **Co-occurrence relation**
  - Words that appear near each other in the language
  - Like *(drink and beer)* or *(drink and wine)*
  - Measured by counting the co-occurrences

- **Similarity relation**
  - Words that appear in similar contexts
  - Like *(beer and wine)* or *(knowledge and wisdom)*
  - Measured by similarity metrics between the vectors

$$\text{similarity}(\text{digital, information}) = \cosine(\hat{v}_{\text{digital}}, \hat{v}_{\text{information}})$$
Sparse vs. Dense Vectors (Recall)

- Such word representations are highly **sparse**
  - Number of dimensions is the same as the number of words in the corpus $n \sim [10000–500000]$
  - Many zeros in the matrix as many words don’t co-occur
    - Normally $\sim 98\%$ sparsity

- **Dense** representations $\rightarrow$ Embeddings
  - Number of dimensions usually between $d \sim [10–1000]$

- **Why dense vectors?**
  - More efficient for storing and load
  - More suitable for machine learning algorithms as features
  - Generalize better by removing noise for unseen data
Recipe for creating (dense) word embedding with neural networks

1. Design a neural network architecture!
2. Loop over training data \((w, c)\)
   a. Set word \(w\) as input and context word \(c\) as output
   b. Calculate the output of network, namely
      The probability of observing context word \(c\) given word \(w\)
      \[
P(c|w)
      \]
   c. Optimize the network to maximize the likelihood probability
3. Repeat

Details come next!
Prepare Training Samples

Window size of 2

Source Text

The quick brown fox jumps over the lazy dog.

Training Samples

(quick, the)
(quick, brown)
(quick, fox)

(brown, the)
(brown, quick)
(brown, fox)
(brown, jumps)

(fox, quick)
(fox, brown)
(fox, jumps)
(fox, over)
Neural Word Embedding Architecture

Train sample: (Tesgüino, drink)

Input Layer
(One-hot encoder)

Output Layer
(Softmax)

$p(\text{drink}|\text{Tesgüino})$

Forward pass

Backpropagation

$\begin{align*}
\text{Words matrix} & : A_{n \times d} \\
\text{Context Words matrix} & : B_{d \times n}
\end{align*}$

https://web.stanford.edu/~jurafsky/slp3/
- Train sample: *(Tesgüino, drink)*
- Update vectors to maximize \( p(\text{drink}|\text{Tesgüino}) \)
Neural Word Embedding - Summary

- Output value is equal to: $\vec{a}_{\text{Tesgüino}} \cdot \vec{b}_{\text{drink}}$

- Output layer is normalized with Softmax

$$p(\text{drink}|\text{Tesgüino}) = \frac{\exp(\vec{a}_{\text{Tesgüino}} \cdot \vec{b}_{\text{drink}})}{\sum_{v \in \mathbb{V}} \exp(\vec{a}_{\text{Tesgüino}} \cdot \vec{b}_v)}$$

$\mathbb{V}$ is the set of vocabularies

Sorry! Denominator is too expensive!

- Loss function is the Negative Log Likelihood (NLL) over all training samples $\mathcal{T}$

$$L = -\frac{1}{T} \sum_{1}^{T} \log p(c|w)$$
word2vec (SkipGram) with Negative Sampling

- word2vec an **efficient** and **effective** algorithm

- Instead of \( p(c|w) \), word2vec measures \( p(y = 1|w, c) \): the probability of **genuine co-occurrence** of \((w, c)\)
  \[
p(y = 1|w, c) = \sigma(\hat{a}_w \cdot \hat{b}_c)
\]

- When two words \((w, c)\) appear in the training data, it is counted as a **positive sample**

- word2vec algorithm tries to distinguish between the co-occurrence probability of a **positive sample** from any **negative sample**

- To do it, word2vec draws \( k \) **negative samples** by randomly sampling from the words distribution → why randomly?
The objective function
- increases the probability for the positive sample \((w, c)\)
- decreases the probability for the \(k\) negative samples \((w, \tilde{c})\)

Loss function:

\[
L = -\frac{1}{T} \sum_{t=1}^{T} \left[ \log p(y = 1|w, c) - \sum_{i=1}^{k} \log p(y = 1|w, \tilde{c}) \right]
\]

\(k \sim 2-10\)

Training Samples

Negative Samples
Train sample: (Tesgüino, drink)
- Train sample: \((\text{Tesgüino, drink})\)
- Sample \(k\) negative context words
- Train sample: \((Teso\üino, \text{drink})\)
- Sample \(K\) negative context words
- Update vectors to
  - Maximize \(p(y = 1|Teso\üino, \text{drink})\)
  - Minimize \(p(y = 1|Teso\üino, \check{c})\)
Discussion about Bias in Data

- A word embedding model captures intrinsic patterns of the given text corpus.

- If the data contains (ethical) bias, the algorithm also encodes the bias in the embedding vectors.

- Such bias can be propagated from word embedding to end-user NLP applications.
Bias in Machine Translation

Elaheh Raisi @elaheh_raisi · Oct 3
Bias in google translate from Persian to English 😞 (Persian uses the gender-neutral pronoun)

same gender-neutral pronoun
Gender Bias in Wikipedia

- The bias of 350 occupations to female/male in the word2vec model, created on English Wikipedia