NLP: Embeddings

Presenter: Anuj Pareek
What are word embeddings?

Complete vocabulary:

\[ \mathcal{V} = \{ \text{elephant}, \text{monkey}, \text{zebra} \} \]

Easy! Represent words as one-hot vectors:

\[
\begin{align*}
    w_{\text{elephant}} &= [1, 0, 0] \\
    w_{\text{monkey}} &= [0, 1, 0] \\
    w_{\text{zebra}} &= [0, 0, 1]
\end{align*}
\]

Problems! No notion of similarity/dissimilarity. They are orthogonal vectors.

No contextual information in encoding!
Motivation for word embeddings

Male-Female

Verb Tense

Country-Capital
Motivation for word embeddings

![Diagram showing relationships between words like king, queen, man, woman, and royal through word embeddings.]

- The diagram on the left illustrates the spatial representation of words in a high-dimensional space, where similar words are closer together.
- The diagram on the right uses arrows to connect words, showing how word embeddings can capture semantic relationships, such as the connection between king and man.
NLP preliminaries: Bag-of-Words model

Count the word occurrences:

<table>
<thead>
<tr>
<th>Document</th>
<th>the</th>
<th>cat</th>
<th>sat</th>
<th>in</th>
<th>hat</th>
<th>with</th>
</tr>
</thead>
<tbody>
<tr>
<td>the cat sat</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>the cat sat in the hat</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>the cat with the hat</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Sort of similar to one-hot encoding, but includes the counts, rather than binary.

A “count-based” model.
Objective: Compute continuous vector representations of words from large data sets...

...with the expectation that not only will similar words tend to be close to each other, but that words can have multiple degrees of similarity

Two models in Word2Vec:

- Continuous Bag-of-Words model
- Skip-gram model
**Word2Vec: Continuous Bag-of-Words model**

**Main idea:** Create word embeddings by learning to predict target word from context words.

**Example:**

![Diagram showing the Word2Vec model](image)
Word2Vec: Continuous Bag-of-Words model

Step 1:
Slide window over text.

“The man who passes the sentence should swing the sword.”

Step 2:
Encode input and output one-hot vectors.

Vocabulary: $|\mathcal{V}| = V$

Input vectors: $\mathbf{x}_1 \in \mathbb{R}^{1 \times V}, \mathbf{x}_2 \in \mathbb{R}^{1 \times V}, \mathbf{x}_3..., \mathbf{x}_n$

Output is single row vector: $\mathbf{y} \in \mathbb{R}^{1 \times V}$
Step 3: Setup the model:

$\mathbf{x}_1 \in \mathbb{R}^{1 \times V}$
$\mathbf{x}_2 \in \mathbb{R}^{1 \times V}$
$\mathbf{x}_3 \in \mathbb{R}^{1 \times V}$

$\mathbf{W} \in \mathbb{R}^{V \times N}$

$\mathbf{h} \in \mathbb{R}^{1 \times N}$

$\mathbf{W}' \in \mathbb{R}^{N \times V}$

$\mathbf{y} \in \mathbb{R}^{1 \times V}$

*Note:* $W'$ is not the transpose of $W$. It's just “bad notation” in Word2Vec.
Word2Vec: Continuous Bag-of-Words model

Step 4:
We don’t have \( W, h \) or \( W' \).

\( h \)-vectors are embedded word vectors in N-dimensional space.

“Force” model to make \( h \):

Maximize: \( P(\text{target word}|\text{context words}) \)

Corresponds to minimizing loss: \( J = -\log P(\text{target word}|\text{context words}) \)

Loss \( J \) minimized through gradient descent, by updating parameters \( W \) and \( W' \).
Objective: Propose a specific weighted least squares model that trains on **global** word-word co-occurrence counts and thus makes efficient use of statistics...
...produce a word vector space with meaningful substructure

Sub-structure:
GloVe: Global Vectors for Word Representation

Make word-word co-occurrence matrix $X$.

**Example** with window-size of 2:

1. I enjoy flying.
2. I like NLP.
3. I like deep learning.

The resulting counts matrix will then be:

$$X = \begin{bmatrix}
I & like & enjoy & deep & learning & NLP & flying \\
I & 0 & 2 & 1 & 0 & 0 & 0 \\
like & 2 & 0 & 0 & 1 & 0 & 1 \\
enjoy & 1 & 0 & 0 & 0 & 0 & 1 \\
deep & 0 & 1 & 0 & 0 & 1 & 0 \\
learning & 0 & 0 & 1 & 0 & 0 & 1 \\
NLP & 0 & 1 & 0 & 0 & 0 & 1 \\
flying & 0 & 0 & 1 & 0 & 0 & 0 \\
.
\end{bmatrix} \quad X \in \mathbb{R}^{V \times V}$$
GloVe: Global Vectors for Word Representation

The probability that word $j$ appear in context of word $i$ is: $P(j|i)$

Relationship between $i$ and $j$ checked with co-occurrence ratios, with probe words $k$

Example:

$i = ice, j = steam \quad k_1 = solid, \quad k_2 = gas, \quad k_3 = water, \quad k_4 = fashion$

Co-occurrence ratio able to distinguish relevant words; solid ($>>1$) and gas ($<<1$) from irrelevant words; water and fashion ($\approx 1$) and discriminate between two relevant words.
GloVe: Global Vectors for Word Representation

Train model parameters using co-occurrence ratios to create “global context” word embeddings.

General form of model F: \[ F(w_i, w_j, \tilde{w}_k) = \frac{P(i|k)}{P(j|k)} \]

Comparison to Mikolov et al.’s CBOW and Skip-gram on word analogy task:

A is to A* as B is to ____

\[ \mathbf{w}_A - \mathbf{w}_{A*} + \mathbf{w}_B = \]
Word2Vec: Skip-gram model

Main idea: Create word embeddings by learning probability distribution of context words from center word.

Example:
Word2Vec: Skip-gram model

Step 1:
Sliding window of size = 5 on training data:

"The man who passes the sentence should swing the sword." – Ned Stark

<table>
<thead>
<tr>
<th>Sliding window (size = 5)</th>
<th>Target word</th>
<th>Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>[The man who]</td>
<td>the</td>
<td>man, who</td>
</tr>
<tr>
<td>[The man who passes]</td>
<td>man</td>
<td>the, who, passes</td>
</tr>
<tr>
<td>[The man who passes the]</td>
<td>who</td>
<td>the, man, passes, the</td>
</tr>
<tr>
<td>[man who passes the sentence]</td>
<td>passes</td>
<td>man, who, the, sentence</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>[sentence should swing the sword]</td>
<td>swing</td>
<td>sentence, should, the, sword</td>
</tr>
<tr>
<td>[should swing the sword]</td>
<td>the</td>
<td>should, swing, sword</td>
</tr>
<tr>
<td>[swing the sword]</td>
<td>sword</td>
<td>swing, the</td>
</tr>
</tbody>
</table>
Word2Vec: Skip-gram model

**Step 2:**
Encode input and output pairs as one-hot vectors.

Size of our vocabulary: $|\mathcal{V}| = V$

Input and output are row vectors: $\mathbf{x} \in \mathbb{R}^{1 \times V}, \mathbf{y} \in \mathbb{R}^{1 \times V}$
Step 3:
Setup the model:

**Note:** \( W' \) is not the transpose of \( W \). It’s just “bad notation” in Word2Vec.

Diagram: Lilian Weng
Word2Vec: Skip-gram model

Step 4:

\( \mathbf{h} \)-vectors are embedded word vectors.

“Force” model to make \( \mathbf{h} \):

Maximize: \( P(\text{context words} | \text{center word}) \)

Corresponds to minimizing loss: \( J = -\log P(\text{context words} | \text{center word}) \)

Loss \( J \) minimized with gradient descent, by updating parameters \( \mathbf{W} \) and \( \mathbf{W}' \).
Cer et al., Universal Sentence Encoder (USE)

Objective: Present models for encoding sentences into embedding vectors...

...compute context aware representations of words in a sentence that take into account both the ordering and identity of all the other words.

Two models:

- **Transformer architecture model** (our focus)
- **Deep Averaging Network**
Universal Sentence Encoder (USE)

Why not separately embed words in a sentence?

**Problem 1:**
Common words increases similarity:

“It must be true” vs. “He must be taking it to the car wash”

**Problem 2:**
Swapping word order doesn’t change similarity:

“Is this garbage?” vs. “This is garbage”
USE: Transformer architecture model

Takes as input a sentence of arbitrary length, outputs 512-dimensional embedding vector.

Use **multi-task learning**!

Intuition; capture most informative features and discard noise
USE: The multiple tasks

Skip-thought:

Conversational Response Prediction:

NLP / Text Classification:

"How old are you?" (96%)
"What is your age?" (98%)
"My phone is good." (7%)
Transformer architecture uses **self-attention**.
Summary

- Convert free-text language into numeric values for NLP
- One-hot vectors are simple, but no contextual information
- Contextualized word embeddings capture similarity/dissimilarity in N dimensions.
- Setup model with proper task and objective function
- Continuous Bag-of-Words, GloVe and Skip-gram for word embeddings
- USE Transformer architecture for sentence embeddings
Thank you!

- E-mail: apareek@student.ethz.ch
- Thanks to mentor, Zhao Meng

References:


