Mining Unstructured Data

8. Word Embeddings

Javier Ferrando

Based on Stanford CS224N, Speech and Language Processing book and Jay Alammar's blog
MT-UPC

- [https://mt.cs.upc.edu/](https://mt.cs.upc.edu/)
- Deep Learning in NLP
- Focus on Machine Translation
- Text-to-text
- Speech-to-text

About me

- Interpretability and analysis of NLP models
- Understand their inner workings
- Understand model predictions
Index

- Motivation
  - Word meaning
  - WordNet
  - Word embeddings

- Sparse vectors
  - One-hot encoding
  - Vectors and documents
  - tf-idf vectors
  - PPMI vectors

- Semantic similarity

- Dense word vectors
  - Word2Vec (Skip-Gram)
  - Model Training
  - Negative Sampling
  - Skip-Gram with Negative Sampling (SGNS)
  - Word2Vec (CBOW)

- Visualization
Motivation
How do we represent the meaning of a word?

Definition of meaning:

- The idea that is represented by a word, phrase, etc
- The idea that a person wants to express by using words, signs, etc
- The idea that is expressed in a work of writing, art, etc
WordNet

- lexical database of semantic relations between words
- Semantic relationships: synonyms, hypernyms ("is a" relationships)
WordNet

Drawbacks:

- Hard to keep it updated
  - Invention of new words
  - New meanings of existing words
- Synonyms of words are not context-dependent
  - ‘to get’ is synonym of ‘become’
- Human decisions (subjective)
- Human labor to create and adapt
Word embeddings

Male-Female

Verb Tense

Country-Capital
Sparse vectors
Distributional hypothesis

- Words that occur in similar contexts tend to have similar meaning (Harris, 1954)
- Never ask for the meaning of a word in isolation, but only in the context of a sentence (Frege, 1884)
- You shall know a word by the company it keeps (Firth, 1957)
Vectors and documents

- **term-document matrix**: number of times a term (row) appears in a document (column)
- Originally defined as a means of finding similar documents for the task of document information retrieval
- We can use document vectors to find other similar documents

Term-document matrix for four words (rows) in four Shakespeare plays.

The comedies have high values for the fool dimension and low values for the battle dimension.
Vectors and documents

- **term-document matrix**: number of times a term (row) appears in a document (column)
- Similar words have similar vectors because they tend to occur in similar documents

Word vectors

<table>
<thead>
<tr>
<th></th>
<th>As You Like It</th>
<th>Twelfth Night</th>
<th>Julius Caesar</th>
<th>Henry V</th>
</tr>
</thead>
<tbody>
<tr>
<td>battle</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>good</td>
<td>114</td>
<td>80</td>
<td>62</td>
<td>89</td>
</tr>
<tr>
<td>fool</td>
<td>36</td>
<td>58</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>wit</td>
<td>20</td>
<td>15</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Red boxes show that each word is represented as a row vector of length four.
Vectors and documents (issue)

- Hard to get meaningful results for frequent words (the, it...)
- ‘good’ appears frequently in different contexts

Solution:
- tf-idf
tf-idf

- Term frequency - inverse document frequency
- Used when the dimensions are documents

**Term frequency (tf)**

- Number of times a term occurs in a document
  \[ tf_{t,d} = \text{count}(t,d) \]
  \[ tf_{t,d} = \log_{10}(\text{count}(t,d) + 1) \]

**Document frequency (df)**

- Number of documents a term occurs in
- Higher weight to words that occur in few documents
  \[ \text{idf}_t = \log_{10}\left(\frac{N}{df_t}\right) \]
  \[ w_{t,d} = tf_{t,d} \times \text{idf}_t \]
### tf-idf

#### Before tf-idf

<table>
<thead>
<tr>
<th></th>
<th>As You Like It</th>
<th>Twelfth Night</th>
<th>Julius Caesar</th>
<th>Henry V</th>
</tr>
</thead>
<tbody>
<tr>
<td>battle</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>good</td>
<td>114</td>
<td>80</td>
<td>62</td>
<td>89</td>
</tr>
<tr>
<td>fool</td>
<td>36</td>
<td>58</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>wit</td>
<td>20</td>
<td>15</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

#### After tf-idf weighting

<table>
<thead>
<tr>
<th></th>
<th>As You Like It</th>
<th>Twelfth Night</th>
<th>Julius Caesar</th>
<th>Henry V</th>
</tr>
</thead>
<tbody>
<tr>
<td>battle</td>
<td>0.074</td>
<td>0</td>
<td>0.22</td>
<td>0.28</td>
</tr>
<tr>
<td>good</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>fool</td>
<td>0.019</td>
<td>0.021</td>
<td>0.0036</td>
<td>0.0083</td>
</tr>
<tr>
<td>wit</td>
<td>0.049</td>
<td>0.044</td>
<td>0.018</td>
<td>0.022</td>
</tr>
</tbody>
</table>
One-hot encoding

- Sparse vectors with one 1 and all the rest 0s: 
  - [0,0,0,1,0,0,0,0,0,0,0,0,0,0]
- Dimensionality of one-hot vectors = vocabulary size $|V|$ 
- One-hot vectors are all orthogonal to one another. We are assuming:
  - the different tokens we are encoding are all independent from each other 
  - ‘movie’ and ‘film’ are usually interchangeable, so they should not be orthogonal to each other
Vectors and documents

- **term-term matrix**: number of times a target word (row) and a context word (column) co-occur
- Co-occurrence in some context (document, or window around target word)
- This matrix is of dimensionality $|V| \times |V|$

Word vectors

Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions.
PPMI (Pointwise Mutual Information)

- Pointwise mutual information (PMI) measures how often two events $x$ and $y$ occur, compared with what we would expect if they were independent:

$$I(x, y) = \log_2 \frac{P(x, y)}{P(x)P(y)}$$

- The PMI between a target word $w$ and a context word $c$:

$$\text{PMI}(w, c) = \log_2 \frac{P(w, c)}{P(w)P(c)}$$

How often two words are observed together

How often we would expect to appear together if they occurred independently
PPMI (Pointwise Mutual Information)

- Positive Pointwise mutual information (PPMI):

\[
\text{PPMI}(w, c) = \max \left( \log_2 \frac{P(w, c)}{P(w)P(c)}, 0 \right)
\]

<table>
<thead>
<tr>
<th></th>
<th>computer</th>
<th>data</th>
<th>result</th>
<th>pie</th>
<th>sugar</th>
<th>count(w)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cherry</td>
<td>2</td>
<td>8</td>
<td>9</td>
<td>442</td>
<td>25</td>
<td>486</td>
</tr>
<tr>
<td>strawberry</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>60</td>
<td>19</td>
<td>80</td>
</tr>
<tr>
<td>digital information</td>
<td>1670</td>
<td>1683</td>
<td>85</td>
<td>5</td>
<td>4</td>
<td>3447</td>
</tr>
<tr>
<td>count(context)</td>
<td>4997</td>
<td>5673</td>
<td>473</td>
<td>512</td>
<td>61</td>
<td>11716</td>
</tr>
</tbody>
</table>

\[
P(w=\text{information}, c=\text{data}) = \frac{3982}{11716} = .3399
\]

\[
P(w=\text{information}) = \frac{7703}{11716} = .6575
\]

\[
P(c=\text{data}) = \frac{5673}{11716} = .4842
\]

\[
\text{PPMI(\text{information}, \text{data})} = \log_2(\frac{.3399}{.6575 \times .4842}) = .0944
\]
PPMI (Pointwise Mutual Information)

<table>
<thead>
<tr>
<th>p(w,context)</th>
<th>p(w)</th>
</tr>
</thead>
<tbody>
<tr>
<td>computer</td>
<td>data</td>
</tr>
<tr>
<td>cherry</td>
<td>0.002</td>
</tr>
<tr>
<td>strawberry</td>
<td>0.000</td>
</tr>
<tr>
<td>digital</td>
<td>0.1425</td>
</tr>
<tr>
<td>information</td>
<td>0.2838</td>
</tr>
</tbody>
</table>

p(context) = 0.4265, 0.4842, 0.0404, 0.0437, 0.0052

Probabilities extracted from counts

<table>
<thead>
<tr>
<th>computer</th>
<th>data</th>
<th>result</th>
<th>pie</th>
<th>sugar</th>
</tr>
</thead>
<tbody>
<tr>
<td>cherry</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4.38</td>
</tr>
<tr>
<td>strawberry</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4.10</td>
</tr>
<tr>
<td>digital</td>
<td>0.18</td>
<td>0.01</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>information</td>
<td>0.02</td>
<td>0.09</td>
<td>0.28</td>
<td>0</td>
</tr>
</tbody>
</table>

PPMI matrix
Semantic similarity
Dot product

- It will tend to be high just when the two vectors have large values in the same dimensions
- Problem: it favors long vectors

\[
dot \text{product}(v, w) = v \cdot w = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + ... + v_N w_N
\]
Cosine similarity

- Normalized dot product
- \( \cos(v,w) = 1 \): same direction
- \( \cos(v,w) = -1 \): opposite direction
- \( \cos(v,w) = 0 \): orthogonal vectors

<table>
<thead>
<tr>
<th></th>
<th>pie</th>
<th>data</th>
<th>computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>cherry</td>
<td>442</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>digital</td>
<td>5</td>
<td>1683</td>
<td>1670</td>
</tr>
<tr>
<td>information</td>
<td>5</td>
<td>3982</td>
<td>3325</td>
</tr>
</tbody>
</table>

\[
\cos(\text{cherry, information}) = \frac{442 \times 5 + 8 \times 3982 + 2 \times 3325}{\sqrt{442^2 + 8^2 + 2^2} \sqrt{5^2 + 3982^2 + 3325^2}} = .018
\]

\[
\cos(\text{digital, information}) = \frac{5 \times 5 + 1683 \times 3982 + 1670 \times 3325}{\sqrt{5^2 + 1683^2 + 1670^2} \sqrt{5^2 + 3982^2 + 3325^2}} = .996
\]
Dense word vectors
Dense Word vectors

- Low-dimensional floating-point vectors (100, 200 components)
- Geometric relationship between two word vectors reflects the semantic relationship between these words
- Synonyms are embedded into similar word vectors
Word2Vec (Skip-gram): Overview

- We have a large corpus (“body”) of text
- Every word in a fixed vocabulary is represented by a vector (static embedding)
- Go through each position $t$ in the text, which has a center word $c$ and context (“outside”) words $o$
- Use the similarity of the word vectors for $c$ and $o$ to calculate the probability of $o$ given $c$
- Keep adjusting the word vectors to maximize this probability
Word2Vec (Skip-gram): Overview

\[ P(w_{t+j} | w_t) \]

Outside context words in window of size 2
Center word at position t
Outside context words in window of size 2
Word2Vec (Skip-gram)

For each position $t = 1, ..., T$, predict context words within a window of fixed size $m$, given center word $w_t$. Data likelihood:

$$
\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m, j \neq 0} P(w_{t+j} | w_t; \theta)
$$

$\theta$ is all variables to be optimized

sometimes called a cost or loss function

The objective function $J(\theta)$ is the (average) negative log likelihood:

$$
J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log P(w_{t+j} | w_t; \theta)
$$
Word2Vec (Skip-gram)

- We want to minimize the objective function:
\[
J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{m \leq m \neq t} \log P(w_{t+j} | w_t; \theta)
\]

- **Question:** How to calculate \( P(w_{t+j} | w_t; \theta) \)?
- **Answer:** We will use two vectors per word \( w \):
  - \( \nu_w \) when \( w \) is a center word
  - \( u_w \) when \( w \) is a context word
- Then for a center word \( c \) and a context word \( o \):
\[
P(o | c) = \frac{\exp(u_o^T \nu_c)}{\sum_{w \in V} \exp(u_w^T \nu_c)}
\]
Word2Vec (Skip-gram)

\[ P(o|c) = \frac{\exp(u^T_v v_c)}{\sum_{w \in V} \exp(u^T_w v_c)} \]

1. Dot product compares similarity of \( o \) and \( c \).
   \[ u^T_v = u \cdot v = \sum_{i=1}^{n} u_i v_i \]
   Larger dot product = larger probability

2. Exponentiation makes anything positive

3. Normalize over entire vocabulary to give probability distribution

- This is an example of the softmax function \( \mathbb{R}^n \to (0,1)^n \)
  \[ \text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i \]
Model Training

- Minimize cost function with gradient descent
- Compute gradient of cost function w.r.t parameters
- Update parameters in the direction of negative gradient
Model Training

- Update equation (in matrix notation):

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$$

\[\alpha = \text{step size or learning rate}\]

- Update equation (for single parameter):

$$\theta_j^{new} = \theta_j^{old} - \alpha \frac{\partial}{\partial \theta_j^{old}} J(\theta)$$
Word2Vec (Skip-gram)

\[ P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)} \]

Computationally expensive!
Negative Sampling

- We now ask the model to predict if the word is in the same context or not (1 or 0)
- Sigmoid/Logistic function
- We need to create negative samples

Change Task from:

Untrained Model

\[ \text{Task: Predict neighbouring word} \]

\[ \text{not} \quad \rightarrow \quad \text{thou} \]

To:

Untrained Model

\[ \text{Task: Are the two words neighbours?} \]

\[ \text{not} \quad \rightarrow \quad \text{thou} \quad \rightarrow \quad 0.90 \]
Negative Sampling

- We now ask the model to predict if the word is in the same context or not (1 or 0)
- Sigmoid/Logistic function
- We need to create negative samples

**Dataset**

<table>
<thead>
<tr>
<th>input word</th>
<th>output word</th>
<th>target</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>thou</td>
<td>1</td>
</tr>
<tr>
<td>not</td>
<td>aaron</td>
<td>0</td>
</tr>
<tr>
<td>not</td>
<td>taco</td>
<td>0</td>
</tr>
<tr>
<td>not</td>
<td>shalt</td>
<td>1</td>
</tr>
<tr>
<td>not</td>
<td>mango</td>
<td>0</td>
</tr>
<tr>
<td>not</td>
<td>fangling</td>
<td>0</td>
</tr>
<tr>
<td>not</td>
<td>make</td>
<td>1</td>
</tr>
<tr>
<td>not</td>
<td>plimbus</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Skip-gram with negative sampling (SGNS)

“Thou shalt not make a machine in the likeness of a human mind”

Positive example

Central word

Negative examples: Aaron, Taco
Skip-gram with negative sampling (SGNS)

We have four words:
- the input word *not*
- the output/context words:
  - thou (the actual neighbor), aaron, and taco (the negative examples).

1- We proceed to **look up their embeddings** – for the input word, we look in the Embedding matrix. For the context words, we look in the Context matrix (even though both matrices have an embedding for every word in our vocabulary).
Skip-gram with negative sampling (SGNS)

2- Then, we take the **dot product** of the input embedding with each of the context embeddings, which indicates the similarity of the input and context embeddings.

3- Now we need a way to **turn these scores into something that looks like probabilities**. We use the sigmoid function.
Skip-gram with negative sampling (SGNS)

4- Now we calculate how much error is in the model’s prediction. To do that, we just subtract the sigmoid scores from the target labels.

<table>
<thead>
<tr>
<th>input word</th>
<th>output word</th>
<th>target</th>
<th>input • output</th>
<th>sigmoid()</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>thou</td>
<td>1</td>
<td>0.2</td>
<td>0.55</td>
<td>0.45</td>
</tr>
<tr>
<td>not</td>
<td>aaron</td>
<td>0</td>
<td>-1.11</td>
<td>0.25</td>
<td>-0.25</td>
</tr>
<tr>
<td>not</td>
<td>taco</td>
<td>0</td>
<td>0.74</td>
<td>0.68</td>
<td>-0.68</td>
</tr>
</tbody>
</table>
Skip-gram with negative sampling (SGNS)

5- We can now use this error score to adjust the embeddings of not, thou, aaron, and taco.

\[
\theta_{new} = \theta_{old} - \alpha \nabla_{\theta} J(\theta)
\]
Word2Vec (CBOW)

- CBOW model computes the probability of the central word given the context ‘outside’ words (bag-of-words)
Word2Vec (CBOW)

- CBOW model computes the probability of the central word given the context ‘outside’ words (bag-of-words)
Visualization

- Dimensionality reduction techniques: UMAP, PCA, t-SNE
Visualization

- Linear structures man-woman
Visualization

- Linear structures comparative - superlative
Analogies

- parallelogram model for solving analogy problems
- Apple is to tree as grape is to ____
- the vector from the word apple to the word tree (tree − apple) is added to the vector for grape (grape); the nearest word to that point is returned.