Web-Mining Agents

Word Semantics and Latent Relational Structures

Prof. Dr. Ralf Möller
Universität zu Lübeck
Institut für Informationssysteme

Tanya Braun (Übungen)
Word-Word Associations in Document Retrieval

Recap

- **LSI**: Documents as vectors, dimension reduction
- **Topic Modeling**
  - Topic = Word distribution
  - From LDA-Model: \( P(Z \mid w) \)
  - Assumption: *Bag of words* model
    (independence, *naïve Bayes*, unigram distribution)

**Words are not independent** of each other

- Word similarity measures
- Extend query with similar words automatically
- Extend query with most frequent followers/predecessors
- Insert words in anticipated gaps in a string query

Need to represent *word semantics*
Approaches for Representing Word Semantics

Distributional Semantics (Count)
- Used since the 90’s
- Sparse word-context PMI/PPMI matrix
- Decomposed with SVD

Word Embeddings (Predict)
- Inspired by deep learning
  - word2vec (Mikolov et al., 2013)
  - GloVe (Pennington et al., 2014)

Underlying Theory: The Distributional Hypothesis (Harris, ’54; Firth, ’57)
“Similar words occur in similar contexts”

https://www.tensorflow.org/tutorials/word2vec
https://nlp.stanford.edu/projects/glove/
Point(wise) Mutual Information: PMI

- Measure of association used in information theory and statistics
  
  \[ \text{pmi}(x; y) \equiv \log \frac{p(x, y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)} \]

- Positive PMI: \( \text{PPMI}(x, y) = \max( \text{pmi}(x, y), 0 ) \)
  
- Quantifies the discrepancy between the probability of their coincidence given their joint distribution and their individual distributions, assuming independence
  
- Finding collocations and associations between words
  
- Countings of occurrences and co-occurrences of words in a text corpus can be used to approximate the probabilities \( p(x) \) or \( p(y) \) and \( p(x,y) \) respectively
PMI – Example

Counts of pairs of words getting the **most and the least PMI scores** in the first 50 millions of words in Wikipedia (dump of October 2015)

- Filtering by 1,000 or more co-occurrences.
- The frequency of each count can be obtained by dividing its value by 50,000,952. (Note: natural log is used to calculate the PMI values in this example, instead of log base 2)

<table>
<thead>
<tr>
<th>word 1</th>
<th>word 2</th>
<th>count word 1</th>
<th>count word 2</th>
<th>count of co-occurrences</th>
<th>PMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>puerto</td>
<td>rico</td>
<td>1938</td>
<td>1311</td>
<td>1159</td>
<td>10.0349081703</td>
</tr>
<tr>
<td>hong</td>
<td>kong</td>
<td>2438</td>
<td>2694</td>
<td>2205</td>
<td>9.72831972408</td>
</tr>
<tr>
<td>los</td>
<td>angeles</td>
<td>3501</td>
<td>2808</td>
<td>2791</td>
<td>9.56067615065</td>
</tr>
<tr>
<td>carbon</td>
<td>dioxide</td>
<td>4265</td>
<td>1353</td>
<td>1032</td>
<td>9.09852946116</td>
</tr>
<tr>
<td>prize</td>
<td>laureate</td>
<td>5131</td>
<td>1676</td>
<td>1210</td>
<td>8.85870710982</td>
</tr>
<tr>
<td>san</td>
<td>francisco</td>
<td>5237</td>
<td>2477</td>
<td>1779</td>
<td>8.83305176711</td>
</tr>
<tr>
<td>nobel</td>
<td>prize</td>
<td>4098</td>
<td>5131</td>
<td>2498</td>
<td>8.68948811416</td>
</tr>
<tr>
<td>ice</td>
<td>hockey</td>
<td>5607</td>
<td>3002</td>
<td>1933</td>
<td>8.6555759741</td>
</tr>
<tr>
<td>star</td>
<td>trek</td>
<td>8264</td>
<td>1594</td>
<td>1489</td>
<td>8.63974676575</td>
</tr>
<tr>
<td>car</td>
<td>driver</td>
<td>5578</td>
<td>2749</td>
<td>1384</td>
<td>8.41470768304</td>
</tr>
<tr>
<td>it</td>
<td>the</td>
<td>283891</td>
<td>3293296</td>
<td>3347</td>
<td>-1.72037278119</td>
</tr>
<tr>
<td>are</td>
<td>of</td>
<td>234458</td>
<td>1761436</td>
<td>1019</td>
<td>-2.09254205335</td>
</tr>
<tr>
<td>this</td>
<td>the</td>
<td>199882</td>
<td>3293296</td>
<td>1211</td>
<td>-2.38612756961</td>
</tr>
<tr>
<td>is</td>
<td>of</td>
<td>565679</td>
<td>1761436</td>
<td>1562</td>
<td>-2.54614706831</td>
</tr>
<tr>
<td>and</td>
<td>of</td>
<td>1375396</td>
<td>1761436</td>
<td>2949</td>
<td>-2.79911817902</td>
</tr>
<tr>
<td>a</td>
<td>and</td>
<td>984442</td>
<td>1375396</td>
<td>1457</td>
<td>-2.92239510038</td>
</tr>
<tr>
<td>in</td>
<td>and</td>
<td>1187652</td>
<td>1375396</td>
<td>1537</td>
<td>-3.05660070757</td>
</tr>
<tr>
<td>to</td>
<td>and</td>
<td>1025659</td>
<td>1375396</td>
<td>1286</td>
<td>-3.08825363041</td>
</tr>
<tr>
<td>to</td>
<td>in</td>
<td>1025659</td>
<td>1187652</td>
<td>1066</td>
<td>-3.1291348956</td>
</tr>
<tr>
<td>of</td>
<td>and</td>
<td>1761436</td>
<td>1375396</td>
<td>1190</td>
<td>-3.70663100173</td>
</tr>
</tbody>
</table>
The Contributions of Word Embeddings

Novel Algorithms

(objective + training method)

• Skip Grams + Negative Sampling
• CBOW + Hierarchical Softmax
• Noise Contrastive Estimation
• GloVe
• …

New Hyperparameters

(preprocessing, smoothing, etc.)

• Subsampling
• Dynamic Context Windows
• Context Distribution Smoothing
• Adding Context Vectors
• …

What’s really improving performance?
Embedding Approaches

- Represent each word with a low-dimensional vector
- Word similarity = vector similarity
- Key idea: Predict surrounding words of every word
- Faster and can easily incorporate a new sentence/document or add a word to the vocabulary
Represent the meaning of word – word2vec

- 2 basic network models:
  - Continuous Bag of Word (CBOw): use a window of word to predict the middle word
  - Skip-gram (SG): use a word to predict the surrounding ones in window.
Word2vec – Continuous Bag of Word

• E.g. “The cat sat on floor”
  – Window size = 2
Input layer

Index of cat in vocabulary

cat

one-hot vector

Hidden layer

Output layer

one-hot vector

sat

one-hot vector
Input layer

\[ W_{V \times N} \]

\[ W'_{N \times V} \]

Hidden layer

Output layer

\text{cat}

\text{on}

\text{sat}

V-dim

N-dim

\text{N will be the size of word vector}

\text{We must learn } W \text{ and } W'
\[ W_{V \times N}^T x_{\text{cat}} = v_{\text{cat}} \]

\[ V = v_{\text{cat}} + v_{\text{on}} \]

\[ \hat{v} = \frac{v_{\text{cat}} + v_{\text{on}}}{2} \]
Input layer

\[ W_{V \times N}^{T} \times x_{on} = v_{on} \]

\[ \times \]

Output layer

\[ x_{on} = v_{on} \]

Hidden layer

\[ \hat{v} = \frac{v_{cat} + v_{on}}{2} \]

\[ W_{V \times N}^{T} \times x_{cat} = v_{cat} \]

\[ W_{V \times N} \times x_{on} = v_{on} \]

\[ v_{cat} \]

\[ v_{on} \]

V-dim

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ x_{on} \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ x_{cat} \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ V_{dim} \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ V_{dim} \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ V_{dim} \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ 0 \]

\[ \ldots \]
\[ W_{V \times N} \]

\[ \hat{y} = \text{softmax}(z) \]

\[ W'_{N \times V} \times \hat{\nu} = z \]

N will be the size of word vector

\[ \hat{\nu} \]
A logistic function or logistic curve is a common "S" shape (sigmoid curve), with equation:

\[ f(x) = \frac{L}{1 + e^{-k(x-x_0)}} \]

where

- \( e \) = the natural logarithm base (also known as Euler's number),
- \( x_0 \) = the \( x \)-value of the sigmoid's midpoint,
- \( L \) = the curve's maximum value, and
- \( k \) = the steepness of the curve.\footnote{1}

Standard logistic sigmoid function i.e. \( L = 1, k = 1, x_0 = 0 \)
The **softmax function**, or **normalized exponential function**, is a generalization of the **logistic function** that "squashes" a $K$-dimensional vector $\mathbf{z}$ of arbitrary real values to a $K$-dimensional vector $\sigma(\mathbf{z})$ of real values in the range $[0, 1]$ that add up to 1. The function is given by

$$\sigma : \mathbb{R}^K \rightarrow [0, 1]^K$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$ 

In **probability theory**, the output of the softmax function can be used to represent a **categorical distribution** – that is, a **probability distribution** over $K$ different possible outcomes.
We would prefer $\hat{y}$ close to $\hat{y}_{sat}$

$\hat{y} = \text{softmax}(z)$

$W'_{N \times V} \times \hat{v} = z$

$N$ will be the size of word vector
We can consider either $W$ or $W'$ as the word’s representation. Or even take the average.
Some interesting results

Word Analogies

Test for linear relationships, examined by Mikolov et al. (2014)

\[ a:b :: c:? \]

\[ d = \arg \max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||} \]

man:woman :: king:?

+ king [ 0.30 0.70 ]
- man [ 0.20 0.20 ]
+ woman [ 0.60 0.30 ]

queen [ 0.70 0.80 ]
Word analogies
What is word2vec?

- word2vec is not a single algorithm
- It is a software package for representing words as vectors, containing:
  - Two distinct models
    - CBoW
    - Skip-Gram (SG)
  - Various training methods
    - Negative Sampling (NS)
    - Hierarchical Softmax
  - A rich preprocessing pipeline
    - Dynamic Context Windows
    - Subsampling
    - Deleting Rare Words
Skip-Grams with Negative Sampling (SGNS)

Marco saw a furry little wampimuk hiding in the tree.
Skip-Grams with Negative Sampling (SGNS)

Marco saw a furry little *wampimuk* hiding in the tree.
Marco saw a furry little wampimuk hiding in the tree.

words | contexts
-------|---------
wampimuk | furry
wampimuk | little
wampimuk | hiding
wampimuk | in
... | ...

\( D \text{ (data)} \)
Skip-Grams with Negative Sampling (SGNS)

- SGNS finds a vector $\vec{w}$ for each word $w$ in our vocabulary $V_W$
- Each such vector has $d$ latent dimensions (e.g. $d = 100$)
- Effectively, it learns a matrix $W$ whose rows represent $V_W$
- **Key point:** it also learns a similar auxiliary matrix $C$ of context vectors
- In fact, each word has two embeddings

```
W:
\text{wampimuk} = (-3.1, 4.15, 9.2, -6.5, ...)
\neq
C:\n\text{wampimuk} = (-5.6, 2.95, 1.4, -1.3, ...)
```

“word2vec Explained…”
Goldberg & Levy, arXiv 2014
Skip-Grams with Negative Sampling (SGNS)
Skip-Grams with Negative Sampling (SGNS)

- Maximize: $\sigma(\vec{w} \cdot \vec{c})$
  - $c$ was observed with $w$

<table>
<thead>
<tr>
<th>words</th>
<th>contexts</th>
</tr>
</thead>
<tbody>
<tr>
<td>wampimuk</td>
<td>furry</td>
</tr>
<tr>
<td>wampimuk</td>
<td>little</td>
</tr>
<tr>
<td>wampimuk</td>
<td>hiding</td>
</tr>
<tr>
<td>wampimuk</td>
<td>in</td>
</tr>
</tbody>
</table>

“word2vec Explained…”
Goldberg & Levy, arXiv 2014
Skip-Grams with Negative Sampling (SGNS)

- **Maximize:** \( \sigma(\vec{w} \cdot \vec{c}) \)
  - \( c \) was *observed* with \( w \)

<table>
<thead>
<tr>
<th>words</th>
<th>contexts</th>
</tr>
</thead>
<tbody>
<tr>
<td>wampimuk</td>
<td>furry</td>
</tr>
<tr>
<td>wampimuk</td>
<td>little</td>
</tr>
<tr>
<td>wampimuk</td>
<td>hiding</td>
</tr>
<tr>
<td>wampimuk</td>
<td>in</td>
</tr>
</tbody>
</table>

- **Minimize:** \( \sigma(\vec{w} \cdot \vec{c}') \)
  - \( c' \) was *hallucinated* with \( w \)

<table>
<thead>
<tr>
<th>words</th>
<th>contexts</th>
</tr>
</thead>
<tbody>
<tr>
<td>wampimuk</td>
<td>Australia</td>
</tr>
<tr>
<td>wampimuk</td>
<td>cyber</td>
</tr>
<tr>
<td>wampimuk</td>
<td>the</td>
</tr>
<tr>
<td>wampimuk</td>
<td>1985</td>
</tr>
</tbody>
</table>

“word2vec Explained…”
Goldberg & Levy, arXiv 2014
Skip-Grams with Negative Sampling (SGNS)

- “Negative Sampling”
- SGNS samples $k$ contexts $c'$ at random as negative examples
- “Random” = unigram distribution

$$P(c) = \frac{\#c}{|D|}$$

- **Spoiler:** Changing this distribution has a significant effect
What is SGNS learning?

- Take SGNS’s embedding matrices ($W$ and $C$)

“Neural Word Embeddings as Implicit Matrix Factorization”
Levy & Goldberg, NIPS 2014
What is SGNS learning?

- Take SGNS’s embedding matrices ($W$ and $C$)
- Multiply them
- What do you get?

"Neural Word Embeddings as Implicit Matrix Factorization"
Levy & Goldberg, NIPS 2014
What is SGNS learning?

- A $V_W \times V_C$ matrix
- Each cell describes the relation between a specific word-context pair

$$\vec{W} \cdot \vec{C} = ?$$
What is SGNS learning?

- We proved that for large enough $d$ and enough iterations

![Diagram](image_url)

“Neural Word Embeddings as Implicit Matrix Factorization”
Levy & Goldberg, NIPS 2014
What is SGNS learning?

• Levy&Goldberg [2014] proved that for large enough $d$ and enough iterations …
• … one obtains the word-context PMI matrix

\[
W \quad \begin{array}{c}
\text{d} \\
V_W
\end{array} \quad C \quad \begin{array}{c}
\text{d} \\
V_C
\end{array} \quad = \quad M_{PMI}^d \quad \begin{array}{c}
\text{d} \\
V_W
\end{array} \quad \begin{array}{c}
\text{d} \\
V_C
\end{array}
\]
What is SGNS learning?

• Levy&Goldberg [2014] **proved** that for large enough $d$ and enough iterations …
• … one obtains the word-context PMI matrix …
• shifted by a global constant

\[
Opt(w \cdot \hat{c}) = PMI(w, c) - \log k
\]
What is SGNS learning?

• SGNS is doing something very similar to the older approaches

• SGNS factorizes the traditional word-context PMI matrix

• So does SVD!

• GloVe factorizes a similar word-context matrix
But embeddings are still better, right?

- Plenty of evidence that embeddings outperform traditional methods
  - “Don’t Count, Predict!” (Baroni et al., ACL 2014)
  - GloVe (Pennington et al., EMNLP 2014)

- How does this fit with our story?
The Big Impact of “Small” Hyperparameters

• word2vec & GloVe are more than just algorithms…

• Introduce new hyperparameters

• May seem minor, but make a big difference in practice
New Hyperparameters

• **Preprocessing** *(word2vec)*
  – Dynamic Context Windows
  – Subsampling
  – Deleting Rare Words

• **Postprocessing** *(GloVe)*
  – Adding Context Vectors

• **Association Metric** *(SGNS)*
  – Shifted PMI
  – Context Distribution Smoothing
Dynamic Context Windows

Marco saw a furry little wampimuk hiding in the tree.
Dynamic Context Windows

saw a furry little wampimuk hiding in the tree
Dynamic Context Windows

Marco saw a furry little wampimuk hiding in the tree.

<table>
<thead>
<tr>
<th>Model</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word2vec</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>GloVe</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Aggressive</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

The Word-Space Model *(Sahlgren, 2006)*
Adding Context Vectors

- SGNS creates word vectors $\vec{w}$
- SGNS creates auxiliary context vectors $\vec{c}$
  - So do GloVe and SVD
Adding Context Vectors

- SGNS creates word vectors $\vec{w}$
- SGNS creates auxiliary context vectors $\vec{c}$
  - So do GloVe and SVD

- Instead of just $\vec{w}$
- Represent a word as: $\vec{w} + \vec{c}$

- Introduced by Pennington et al. (2014)
- Only applied to GloVe
Context Distribution Smoothing

• SGNS samples $c' \sim P$ to form negative $(w, c')$ examples

• Our analysis assumes $P$ is the unigram distribution

\[ P(c) = \frac{\#c}{\sum_{c' \in V_c} \#c'} \]
Context Distribution Smoothing

• SGNS samples $c' \sim P$ to form negative $(w, c')$ examples

• Our analysis assumes $P$ is the unigram distribution

• In practice, it’s a smoothed unigram distribution

$$P^{0.75}(c) = \frac{(\#c)^{0.75}}{\sum_{c' \in V_C}(\#c')^{0.75}}$$

• This little change makes a big difference
Context Distribution Smoothing

• We can **adapt** context distribution smoothing to PMI!

• Replace $P(c)$ with $P^{0.75}(c)$:

\[
PMI^{0.75}(w, c) = \log \frac{P(w, c)}{P(w) \cdot P^{0.75}(c)}
\]

• Consistently improves **PMI** on **every task**

• **Always use Context Distribution Smoothing!**
Represent the meaning of **sentence/text**

- Paragraph vector (2014, Quoc Le, Mikolov)
  - Extend word2vec to text level
  - Also two models: add paragraph vector as the input
Don’t Count, Predict! [Baroni et al., 2014]

• “word2vec is better than count-based methods”

• **Hyperparameter settings** account for most of the reported gaps

• Embeddings do **not** really outperform count-based methods

• No unique conclusion available
Latent Relational Structures

Processing natural language data:
✓ Tokenization/Sentence Splitting
✓ Part-of-speech (POS) tagging
  • Phrase chunking
  • Named entity recognition
  • Coreference resolution
  • Semantic role labeling
Phrase Chunking

- Identifies phrase-level constituents in sentences
  
  [NP Boris] [ADVP regretfully] [VP told] [NP his wife] [SBAR that] [NP their child] [VP could not attend] [NP night school] [PP without] [NP permission].

- Useful for filtering: identify e.g. only noun phrases, or only verb phrases

- Used as source of features, e.g. distance, (abstracts away determiners, adjectives, for example), sequence,…
  - More efficient to compute than full syntactic parse
  - Applications in e.g. Information Extraction – getting (simple) information about concepts of interest from text documents

- Hand-crafted chunkers (regular expressions/finite automata)
- HMM/CRF-based chunk parsers derived from training data
Named Entity Recognition

- Identifies and classifies strings of characters representing proper nouns
  
  - [PER Neil A. Armstrong], the 38-year-old civilian commander, radioed to earth and the mission control room here: “[LOC Houston], [ORG Tranquility] Base here; the Eagle has landed.”

- Useful for filtering documents
  - “I need to find news articles about organizations in which Bill Gates might be involved…”

- Disambiguate tokens: “Chicago” (team) vs. “Chicago” (city)

- Source of abstract features
  - E.g. “Verbs that appear with entities that are Organizations”
  - E.g. “Documents that have a high proportion of Organizations”
Named Entity Recognition: Definition

- NE involves identification of proper names in texts, and classification into a set of predefined categories of interest
  - Three universally accepted categories: person, location and organisation
  - Other common tasks: recognition of date/time expressions, measures (percent, money, weight etc), email addresses etc.
  - Other domain-specific entities: names of drugs, medical conditions, names of ships, bibliographic references etc

- NER ist not easy
Named Entity Classification

- Category definitions are intuitively quite clear, but there are many grey areas.
- Many of these grey areas are caused by metonymy. Person vs. Artefact: “The ham sandwich wants his bill.” vs “Bring me a ham sandwich.” Organisation vs. Location: “England won the World Cup” vs. “The World Cup took place in England”. Company vs. Artefact: “shares in MTV” vs. “watching MTV” Location vs. Organisation: “she met him at Heathrow” vs. “the Heathrow authorities”
Basic Problems in NE

• Variation of NEs – e.g. John Smith, Mr Smith, John.

• Ambiguity of NE types
  – John Smith (company vs. person)
  – May (person vs. month)
  – Washington (person vs. location)
  – 1945 (date vs. time)

• Ambiguity with common words, e.g. “may”
More complex problems in NER

• Issues of style, structure, domain, genre etc.
  – Punctuation, spelling, spacing, formatting, ….all have an impact

Dept. of Computing and Maths
Manchester Metropolitan University
Manchester
United Kingdom

> Tell me more about Leonardo
> Da Vinci
List Lookup Approach

- System that recognises only entities stored in its lists (gazetteers).
- Advantages - Simple, fast, language independent, easy to retarget
- Disadvantages – collection and maintenance of lists, cannot deal with name variants, cannot resolve ambiguity
Shallow Parsing Approach

- Internal evidence – names often have internal structure. These components can be either stored or guessed.

**location:**

CapWord + {City, Forest, Center}

  e.g. *Sherwood Forest*

Cap Word + {Street, Boulevard, Avenue, Crescent, Road}

  e.g. *Portobello Street*
Shallow Parsing Approach

• External evidence - names are often used in very predictive local contexts

**Location:**

“to the” COMPASS “of” CapWord

  e.g. *to the south of Loitokitok*

“based in” CapWord

  e.g. *based in Loitokitok*

CapWord “is a” (ADJ)? GeoWord

  e.g. *Loitokitok is a friendly city*
Difficulties in Shallow Parsing Approach

- **Ambiguously capitalised words** (first word in sentence)
  [All American Bank] vs. All [State Police]
- **Semantic ambiguity**
  “John F. Kennedy” = airport (location)
  “Philip Morris” = organisation
- **Structural ambiguity**
  [Cable and Wireless] vs. [Microsoft] and [Dell]
  [Center for Computational Linguistics] vs. message from [City Hospital] for [John Smith].

“Introduction to Named Entity Recognition”, University of Sheffield
Coreference

- Identify all phrases that refer to each entity of interest – i.e., group mentions of concepts

- [Neil A. Armstrong], [the 38-year-old civilian commander], radioed to [earth]. [He] said the famous words, “[the Eagle] has landed.”

- The Named Entity Recognizer only gets us part-way…

- …if we ask, “what actions did Neil Armstrong perform?”, we will miss many instances (e.g. “He said…”)

- Coreference resolver **abstracts over different ways of referring to the same person**
  - Useful in feature extraction, information extraction
Semantic Role Labeling (SRL)

Input Text:
A car bomb that exploded outside the U.S. military base in Beniji killed 11 Iraqi citizens.

Result: Complete!

- SRL reveals relations and arguments in the sentence (where relations are expressed as verbs)
- Cannot abstract over variability of expressing the relations – e.g. kill vs. murder vs. slay…
Why is SRL Important – Applications

• Question Answering
  – Q: When was Napoleon defeated?
  – Look for: \([\text{PATIENT Napoleon}] \ [\text{PRED defeat-synset}] \ [\text{ARGM-TMP *ANS*}]\)

• Machine Translation
  English (SVO)  
  \([\text{AGENT The little boy}] \ [\text{PRED kicked}] \ [\text{THEME the red ball}] \ [\text{ARGM-MNR hard}]\)

  Farsi (SOV)  
  \([\text{AGENT pesar koocholo}] \ [\text{THEME toop germezi}] \ [\text{ARGM-MNR moqtam}] \ [\text{PRED zaad-e}]\)

• Document Summarization
  – Predicates and Heads of Roles summarize content

• Information Extraction
  – SRL can be used to construct useful rules for IE
Some History

• Minsky 74, Fillmore 1976: *Frames* describe events or situations
  – Multiple *participants*, “props”, and “*conceptual roles*”
  – E.g., agent, instrument, target, time, …

• Levin 1993: *verb class* defined by sets of frames (meaning-preserving alternations) a verb appears in
  – \{*break*, *shatter*, ..\}: Glass X’s easily; John Xed the glass, …
  – *Cut* is different: The window broke; *The window cut.*

• *FrameNet*, late ’90s: based on Levin’s work: large corpus of sentences annotated with *frames*

• *PropBank*
FrameNet [Fillmore et al. 01]

Lexical units (LUs): Words that evoke the frame (usually verbs)

Frame elements (FEs): The involved semantic roles

Frame: Hit_target (hit, pick off, shoot)

Agent
Target
Instrument
Manner
Means
Place
Purpose
Subregion
Time

[Agent Kristina] hit [Target Scott] [Instrument with a baseball] [Time yesterday].
Proposition Bank (PropBank) [Palmer et al. 05]

• Transfer sentences to propositions
  – Kristina hit Scott → hit(Kristina,Scott)

• Penn TreeBank → PropBank
  – Add a semantic layer on Penn TreeBank
  – Define a set of semantic roles for each verb
  – Each verb’s roles are numbered

  …[A0 the company] to … offer [A1 a 15% to 20% stake] [A2 to the public]
  …[A0 Sotheby’s] … offered [A2 the Dorrance heirs] [A1 a money-back guarantee]
  …[A1 an amendment] offered [A0 by Rep. Peter DeFazio] …
  …[A2 Subcontractors] will be offered [A1 a settlement] …
Latent Relational Structures

Input Text:
A car bomb that exploded outside the U.S. military base in Beniji killed 11 Iraqi citizens.

Result: Complete!

- SRL reveals relations and arguments in the sentence (where relations are expressed as verbs)
- Cannot abstract over variability of expressing the relations – e.g. kill vs. murder vs. slay…
Collective Learning on Multi-Relational Data

- Julie: Sister In-Law To Steve
- Julie: Listens To Rock Music
- Bob: Married To Julie
- Bob: Listens To Rock Music
- Bob: Drives BMW
- Bob: Colleague Of Jim
- BMW: Drives
- Fido: Has Pet
- Jim: Works For IBM
- IBM: Works For
Towards Latent Relational Structures

- Modelling binary relations as a tensor: Two modes of a tensor refer to the entities, one mode to the relations.
- The entries of the tensor are 1 when a relation between two entities exists and 0 otherwise.
- We use the RDF formalism to model relations as (subject, predicate, object) triples.
Motivation

Why Tensors?

- **Modelling simplicity**: Multiple binary relations can be expressed straightforwardly as a three-way tensor.
- **No structure learning**: Not necessary to have information about independent variables, knowledge bases, etc. or to infer it from data.
- **Expected performance**: Relational domains are high-dimensional and sparse, a setting where factorization methods have shown very good results.
Tensor Factorization with Rescal

- RESCAL takes the inherent structure of dyadic relational data into account, by employing the tensor factorization

\[ X_k \approx AR_kA^T \]

- \( A \) is a \( n \times r \) matrix, representing the global entity-latent-component space

- \( R_k \) is an asymmetric \( r \times r \) matrix that specifies the interaction of the latent components per predicate
Solving Relational Learning Tasks

- **Link Prediction**: To predict the existence of a relation between two entities, it is sufficient to look at the rank-reduced reconstruction of the appropriate slice $A R_k A^T$.

- **Collective Classification**: Can be cast as a link prediction problem by including the classes as entities and adding a class-of relation. Alternatively, standard classification algorithms could be applied to the entities' latent-component representation $A$.

- **Link-based Clustering**: Since the entities' latent-component representation is computed considering all relations, Link-based clustering can be done by clustering the entities in the latent-component space $A$. 
Compute the Factorization

To compute the factorization, we solve the optimization problem

$$\arg\min_{A, R_k} \text{loss}(A, R_k) + \text{reg}(A, R_k)$$

where \(\text{loss}\) is the loss function

$$\text{loss}(A, R_k) = \frac{1}{2} \sum_k \| X_k - AR_k A^T \|_F^2$$

and \(\text{reg}\) is the regularization term

$$\text{reg}(A, R_k) = \frac{1}{2} \lambda \left( \| A \|_F^2 + \sum_k \| R_k \|_F^2 \right)$$
Collective Learning

- Predict party membership of US (vice) presidents

![Graph showing relationships between Bill, John, Al, Lyndon, and Party X]

- Helpful to consider element-wise version of the loss function $f$

$$f(A, R_k) = \frac{1}{2} \sum_{i,j,k} (x_{ijk} - a_i^T R_k a_j)^2$$
Collective Learning

- Predict party membership of US (vice) presidents

- Helpful to consider element-wise version of the loss function $f$

$$f(A, R_k) = \frac{1}{2} \sum_{i,j,k} \left( x_{ijk} - a_i^T R_k a_j \right)^2$$
Collective Learning

- Predict party membership of US (vice) presidents

- Helpful to consider element-wise version of the loss function $f$

$$f(A, R_k) = \frac{1}{2} \sum_{i,j,k} \left( x_{ijk} - a_i^T R_k a_j \right)^2$$
Collective Learning

- Predict party membership of US (vice) presidents

- Helpful to consider element-wise version of the loss function $f$

$$f(A, R_k) = \frac{1}{2} \sum_{i,j,k} \left( x_{ijk} - a_i^T R_k b_j \right)^2$$
Collective Learning

- Predict party membership of US (vice) presidents

![Graph with nodes representing Bill, John, Al, and Lyndon, connected by party membership relationships.]

- Helpful to consider element-wise version of the loss function $f$

$$f(A, R_k) = \frac{1}{2} \sum_{i,j,k} \left( x_{ijk} - a_i^T R_k b_j \right)^2$$
Collective Learning

- Collective learning is performed via the entities’ latent-component representation
- Important aspect of the model: Entities have a unique latent-component representation, regardless of their occurrence as subjects or objects
Using Learned Relational Networks for IR

- Query answering: indirect queries requiring chains of reasoning
- KB Completion: exploits redundancy in the KB + chains to infer missing facts

**Freebase 15k benchmark**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Hits@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unstructured [Bordes et al., 2014]</td>
<td>4.5</td>
</tr>
<tr>
<td>RESCAL [Nickel et al., 2011]</td>
<td>28.4</td>
</tr>
<tr>
<td>SE [Bordes et al., 2011]</td>
<td>28.8</td>
</tr>
<tr>
<td>SME [Bordes et al., 2014]</td>
<td>31.3</td>
</tr>
<tr>
<td>LFM [Jenatton et al., 2012]</td>
<td>26.0</td>
</tr>
<tr>
<td>TransE [Bordes et al., 2013]</td>
<td>34.9</td>
</tr>
<tr>
<td>ConvNets [Shi and Zhu, 2015]</td>
<td>37.7</td>
</tr>
<tr>
<td>TransH [Wang et al., 2014b]</td>
<td>45.7</td>
</tr>
<tr>
<td>TransR [Lin et al., 2015b]</td>
<td>48.4</td>
</tr>
<tr>
<td>PTransE [Lin et al., 2015a]</td>
<td>51.8</td>
</tr>
</tbody>
</table>

Baseline method: tensor factorization
depth NN embedding
Using Learned Relational Networks for IR

TransE: find an embedding for entities and relations so that $R(X,Y)$ iff $v_Y - v_X \sim v_R$

TransE [Bordes et al., 2013]

Alternative is explicit inference rules:

$\wedge$

uncle($X,Y$) :- aunt($X,Z$), husband($Z,Y$).

William W Cohen, Machine Learning Dept. CMU