Word Embeddings
Past, Present and Future

Piero Molino
Motivation

Word Embeddings: **hot trend** in NLP (Post-word2vec era, 2013+)

Many researchers and practitioner are **oblivious of previous work** in computer science, cognitive science and computational linguistics (Pre-word2vec era: up to 2013)

**Delays progress** due to reinventing the wheel + **many lessons to be learned**
Goal

Overview* of the **history** of the field to start **building** on **existing knowledge**

Give **some hints** on **future directions**

*Not complete overview, but a useful **starting point** for exploration*
Outline

1. Linguistic background: Structuralism
2. Distributional Semantics
3. Methods overview
4. Open issues and current trends
Terminology


They all mean (almost) the same thing

Distributional Semantic Models $\rightarrow$ Computational Linguistics literature

Word Embeddings $\rightarrow$ Neural Networks literature
Structuralism
Structuralism

“The belief that phenomena of human life are not intelligible except through their interrelations. These relations constitute a structure, and behind local variations in the surface phenomena there are constant laws of abstract culture”

Origins of Structuralism

Ferdinand de Saussure, *Cours de linguistique générale*, 1916

Published posthumous from notes of his students

Previous ideas close to structuralism:

- Wilhelm von Humboldt, *Über den Dualis*, 1828
- Wilhelm von Humboldt, *Über die Verschiedenheit des menschlichen Sprachbaues*, 1836
- Ferdinand de Saussure, *Mémoire sur le système des primitif voyelles dans les langues indo-européennes*, 1879
Structuralism and Semiotics

*Langue vs Parole*

Sign, Signifier, Signified

Different languages use different signifiers for the same signified → the choice of signifiers is arbitrary

Meaning of signs is defined by their relationships and contrasts with other signs
Meaning of signs is defined by their relationships and contrasts with other signs.
**Linguistic relationships**

**Paradigmatic**: relationship between words that can be **substituted** for each other in the same position within a given sentence.

**Syntagmatic**: relationship a word has with other words that **surround** it.

Originally de Saussure used the term 'associative', the term 'paradigmatic' was introduced by Louis Hjelmslev, *Principes de grammaire générale*, 1928.
Paradigmatic

Synonymy

- Bubbling
- Effervescent
- Sparkling

Antonymy

- Hot
- Cold

Hyponymy

- Feline
  - Tiger
  - Lion

Hypernym

- Hyponym
Syntagmatic

Collocation

- against the law
- law enforcement
- become law
- law is passed

Colligation

- VERB past time
- saved
- spent
- wasted
- ADJECTIVE time
- half
- extra
- full

sport normal
Distributionalism

American structuralist branch

Leonard Bloomfield, *Language*, 1933

Zellig Harris, *Methods in Structural Linguistics*, 1951

Zellig Harris, *Distributional Structure*, 1954

Zellig Harris, *Mathematical Structure of Language*, 1968
Philosophy of Language

"The meaning of a word is its use in the language"

- Ludwig Wittgenstein, Philosophical Investigation, 1953
Corpus Linguistics

"You shall know a word by the company it keeps"

- J.R. Firth, Papers in Linguistics, 1957
Other relevant work

Willard Van Orman Quine, *Word and Object*, 1960

Margaret Masterman, *The Nature of a Paradigm*, 1965
Distributional Semantics
Distributional Hypothesis

The degree of semantic similarity between two linguistic expressions $A$ and $B$ is a function of the similarity of the linguistic contexts in which $A$ and $B$ can appear.

First formulation by Harris, Charles, Miller, Firth or Wittgenstein?
We found a little, hairy wampimuk sleeping behind the tree.

He filled the wampimuk, passed it around and we all drank some.

– McDonald and Ramscar, 2001
We found a little, hairy wampimuk sleeping behind the tree

– McDonald and Ramscar, 2001
Distributional Semantic Model

1. Represent words through vectors recording their co-occurrence counts with context elements in a corpus

2. (Optionally) Apply a re-weighting scheme to the resulting co-occurrence matrix

3. (Optionally) Apply dimensionality reduction techniques to the co-occurrence matrix

4. Measure geometric distance of word vectors as proxy to semantic similarity / relatedness
Example

**Target:** a specific word

**Context:** noun and verbs in the same sentence

*The dog barked in the park. The owner of the dog put him on the leash since he barked.*

<table>
<thead>
<tr>
<th>word</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>bark</td>
<td>2</td>
</tr>
<tr>
<td>park</td>
<td>1</td>
</tr>
<tr>
<td>leash</td>
<td>1</td>
</tr>
<tr>
<td>owner</td>
<td>1</td>
</tr>
</tbody>
</table>
## Example

<table>
<thead>
<tr>
<th>Targets</th>
<th>leash</th>
<th>walk</th>
<th>run</th>
<th>owner</th>
<th>leg</th>
<th>bark</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>cat</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>lion</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>light</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>dark</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>car</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Example

Use **cosine similarity** as a measure of relatedness

\[
\cos \theta = \frac{x \cdot y}{\|x\|\|y\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=0}^{n} x_i^2} \sqrt{\sum_{i=0}^{n} y_i^2}}
\]
Similarity and Relatedness

Semantic similarity
words sharing salient attributes/features
- synonymy (car / automobile)
- hypernymy (car / vehicle)
- co-hyponymy (car / van / truck)

Semantic relatedness
words semantically associated without being necessarily similar
- function (car / drive)
- meronymy (car / tyre)
- location (car / road)
- attribute (car / fast)

(Budansky and Hirst, 2006)
The meaning of a word can be defined in terms of its context (properties, features)

- Other words in the same document / paragraph / sentence
- Words in the immediate neighbors
- Words along dependency paths
- Linguistic patterns
- Predicate-Argument structures
- Frames
- Hand crafted features
  First attempt in 1960s in Charles Osgood’s semantic differentials, also used in first connectionist AI approaches in the 1980s

Any process that builds a structure on sentences can be used as a source for properties
DOC1: The silhouette of the sun beyond a wide-open bay on the lake; the sun still glitters although evening has arrived in Kuhmo. It’s midsummer; the living room has its instruments and other objects in each of its corners.
Context Examples

*Wide window*

**DOC1:** The silhouette of the *sun* beyond a wide-open bay on the lake; the *sun* still glitters although evening has arrived in Kuhmo. It’s midsummer; the living room has its instruments and other objects in each of its corners.
The silhouette of the sun beyond a wide-open bay on the lake; the sun still glitters although evening has arrived in Kuhmo. It’s midsummer; the living room has its instruments and other objects in each of its corners.
The silhouette of the sun beyond a wide-open bay on the lake; the sun still glitters although evening has arrived in Kuhmo. It’s midsummer; the living room has its instruments and other objects in each of its corners.
The silhouette of the sun beyond a wide-open bay on the lake; the sun still glitters although evening has arrive in Kuhmo. It’s midsummer; the living room has its instruments and other objects in each of its corners.
DOC1: The silhouette/N of the sun beyond a wide-open bay on the lake; the sun still glitters/V although evening has arrived in Kuhmo. It’s midsummer; the living room has its instruments and other objects in each of its corners.
 CONTEXT EXAMPLES

Syntactic path coded lemmas

**DOC1**: The *silhouette/N_PPDEP* of the *sun* beyond a wide-open bay on the lake; the *sun* still *glitters/V_SUBJ* although evening has arrived in Kuhmo. It’s midsummer; the living room has its instruments and other objects in each of its corners.
Effect of Context
Neighbors of **dog** in *BNC Corpus*

<table>
<thead>
<tr>
<th>2-word window</th>
<th>30-word window</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>kennel</td>
</tr>
<tr>
<td>horse</td>
<td>puppy</td>
</tr>
<tr>
<td>fox</td>
<td>pet</td>
</tr>
<tr>
<td>pet</td>
<td>bitch</td>
</tr>
<tr>
<td>rabbit</td>
<td>terrier</td>
</tr>
<tr>
<td>pig</td>
<td>rottweiler</td>
</tr>
<tr>
<td>animal</td>
<td>canine</td>
</tr>
<tr>
<td>mongrel</td>
<td>cat</td>
</tr>
<tr>
<td>sheep</td>
<td>bark</td>
</tr>
<tr>
<td>pigeon</td>
<td>alsatian</td>
</tr>
</tbody>
</table>

More paradigmatic

More syntagmatic
# Effect of Context

Neighbors of *Turing* in *Wikipedia*

<table>
<thead>
<tr>
<th>Syntactic dependencies</th>
<th>5-word window</th>
<th>Topically related</th>
</tr>
</thead>
<tbody>
<tr>
<td>Co-hyponyms Paradigmatic</td>
<td>Pauling</td>
<td>nondeterministic</td>
</tr>
<tr>
<td></td>
<td>Hotelling</td>
<td>non-deterministic</td>
</tr>
<tr>
<td></td>
<td>Heting</td>
<td>computability</td>
</tr>
<tr>
<td></td>
<td>Lessing</td>
<td>deterministic</td>
</tr>
<tr>
<td></td>
<td>Hamming</td>
<td>finite-state</td>
</tr>
</tbody>
</table>

*Paradigmatic* and *Syntagmatic* are topically related.
Weighting Schemes

So far we used raw counts

Several other options for populating the target x context matrix are available

In most cases Positive Pointwise Mutual Information is the best choice

Kiela and Clark, A systematic study of Semantic Vector Space Parameters, 2014, is a good review

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>( w_{ij} = f_{ij} )</td>
</tr>
<tr>
<td>TF-IDF</td>
<td>( w_{ij} = \log(f_{ij}) \times \log\left(\frac{N}{n_j}\right) )</td>
</tr>
<tr>
<td>TF-ICF</td>
<td>( w_{ij} = \log(f_{ij}) \times \log\left(\frac{N}{f_{ij}}\right) )</td>
</tr>
<tr>
<td>Okapi BM25</td>
<td>( w_{ij} = \frac{f_{ij}}{0.5+1.5 \times \frac{f_{ij}}{f_{ij}+f_{ij}}} \times \log\left(\frac{N-n_j+0.5}{f_{ij}+0.5}\right) )</td>
</tr>
<tr>
<td>ATC</td>
<td>( w_{ij} = \sqrt{\sum_{n=1}^{N} \left(0.5+0.5 \times \frac{f_{ij}}{\max_j - f_{ij}}\right) \log\left(\frac{N}{n_j}\right)^2} )</td>
</tr>
<tr>
<td>LTU</td>
<td>( w_{ij} = \frac{0.8+0.2 \times f_{ij}}{f_{ij}} )</td>
</tr>
<tr>
<td>MI</td>
<td>( w_{ij} = \log \frac{P(t_{ij}</td>
</tr>
<tr>
<td>PosMI</td>
<td>( \max(0, \text{MI}) )</td>
</tr>
<tr>
<td>T-Test</td>
<td>( w_{ij} = \frac{P(t_{ij}</td>
</tr>
<tr>
<td>( \chi^2 )</td>
<td>see (Curran, 2004, p. 83)</td>
</tr>
<tr>
<td>Lin98a</td>
<td>( w_{ij} = \frac{f_{ij} \times f_j}{f_{ij} \times f_j} )</td>
</tr>
<tr>
<td>Lin98b</td>
<td>( w_{ij} = -1 \times \log \frac{n_j}{N} )</td>
</tr>
<tr>
<td>Gref94</td>
<td>( w_{ij} = \frac{\log f_{ij} + 1}{\log n_j + 1} )</td>
</tr>
</tbody>
</table>
Similarity Measures

So far we used **cosine similarity**

**Several other options** for computing similarity are available

In most cases **Correlation** is the best choice (cosine similarity of vectors normalized by their mean)

Kiela and Clark, *A systematic study of Semantic Vector Space Parameters*, 2014, is a good review

<table>
<thead>
<tr>
<th>Measure</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>( \frac{1}{1+\sqrt{\sum_{i=1}^{n}(u_i-v_i)^2}} )</td>
</tr>
<tr>
<td>Cityblock</td>
<td>( \frac{1}{1+\sum_{i=1}^{n}</td>
</tr>
<tr>
<td>Chebyshev</td>
<td>( \frac{1}{1+\max_{i}</td>
</tr>
<tr>
<td>Cosine</td>
<td>( \frac{u \cdot v}{</td>
</tr>
<tr>
<td>Correlation</td>
<td>( \frac{(u-\mu_u)(v-\mu_v)}{</td>
</tr>
<tr>
<td>Dice</td>
<td>( \frac{2 \sum_{i=0}^{n} \min(u_i,v_i)}{\sum_{i=0}^{n} u_i+v_i} )</td>
</tr>
<tr>
<td>Jaccard</td>
<td>( \frac{\sum_{i=0}^{n} u_i+v_i}{\sum_{i=0}^{n} \max(u_i,v_i)} )</td>
</tr>
<tr>
<td>Jaccard2</td>
<td>( \frac{\sum_{i=0}^{n} \min(u_i,v_i)}{\sum_{i=0}^{n} \max(u_i,v_i)} )</td>
</tr>
<tr>
<td>Lin</td>
<td>( \frac{\sum_{i=0}^{n} u_i+v_i}{</td>
</tr>
<tr>
<td>Tanimoto</td>
<td>( \frac{u \cdot v}{</td>
</tr>
<tr>
<td>Jensen-Shannon Div</td>
<td>( 1 - \frac{1}{2} \left( D(u</td>
</tr>
<tr>
<td>( \alpha )-skew</td>
<td>( 1 - \frac{D(u</td>
</tr>
</tbody>
</table>
Evaluation

Intrinsic

- evaluate **word pairs** similarities \(\rightarrow\) compare with similarity judgments given by humans (**WordSim**, **MEN**, **Mechanical Turk**, **SImLex**)
- evaluate on **analogy tasks**
  
  'Paris is to France as Tokyo is to \(x\)' (**MSR analogy**, **Google analogy**)

Extrinsic

- use the vectors in a **downstream task**
  (classification, translation, ...) and evaluate the **final performance** on the task
Best parameters configuration?

(context, similarity measure, weighting, ...)
Depends on the task!
Methods overview
Methods

Semantic Differential (Osgood et al. 1957)
Semantic features (Smith et al. 1974)
Mechanisms of sentence processing assigning roles to constituents (McLelland and Kawamoto 1986)
Learning Distributed Representations of Concepts (Hinton et al. 1986)
Forming Global Representations with Extended Back-Propagation [FGREP] (Mikkulainen and Dyer 1987)
Sparse Distributed Memory [SDM] (Kanerva 1988)
Hyperspace Analogue to Language [HAL] (Lund and Burgess 1995)
Random Indexing (Kanerva et al. 2000)
Latent Dirichlet Allocation [LDA] (Blei et al. 2003)
A neural probabilistic language model (Bengio et al. 2003)
Infomap (Widdows et al. 2004)
Correlated Occurrence Analogue to Lexical Semantic [COALS] (Rohde et al. 2006)
Dependency Vecotrs (Padó and Lapata 2007)
Explicit Semantic Analysis (Gabrilovich and Markovich 2007)
Distributional Memory (Baroni and Lenci 2009)
Non-Negative Matrix Factorization [NNMF] (Van de Cruys et al. 2010) originally: (Paatero and Tapper 1994)
JoBimText (Biemann and Riedl 2013)
word2vec [SGNS and CBOW] (Mikolov et al. 2013)
vLBL and ivLBL (Mnih and Kavukcuoglu 2013)
Hellinger PCA (HPCA) (Lebret and Collobert 2014)
Global Vectors [GloVe] (Pennington et al. 2014)
Infinite Dimensional Word Embeddings (Nalisnick and Ravi 2015)
Gaussian Embeddings [GloVe] (Vilnis and McCallum 2015)
Diachronic Word Embeddings (Hamilton et al. 2016)
WordRank (Ji et al. 2016)
Exponential Family Embeddings (Rudolph et al. 2016)
Multimodal Word Distributions (Athiwaratkun and Wilson 2017)
Explicit vs Implicit

Explicit vectors: big sparse vectors with interpretable dimensions

Implicit vectors: small dense vectors with latent dimensions

Count vs Prediction

Alessandro Lenci, *Distributional models of word meaning*, 2017
Hyperspace Analogue to Language [HAL]

**Target:** a specific word

**Context:** window of ten words

**Weighting:** (10 - distance from target) for each occurrence

**Similarity:** euclidean distance

**Dimensionality reduction:** sort contexts (columns of the matrix) by variance and keep top 200

---

**the dog barked at the cat**

weight \( \text{dog barked} = 10 \) (no gap)

weight \( \text{dog cat} = 7 \) (3 words gap)

<table>
<thead>
<tr>
<th>( c_2 )</th>
<th>( c_7 )</th>
<th>...</th>
<th>( c_3 )</th>
<th>( c_5 )</th>
<th>...</th>
<th>...</th>
<th>( c_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_1 )</td>
<td>54</td>
<td>23</td>
<td>...</td>
<td>8</td>
<td>4</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>( w_1 )</td>
<td>21</td>
<td>82</td>
<td>...</td>
<td>10</td>
<td>6</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>( w_n )</td>
<td>32</td>
<td>47</td>
<td>...</td>
<td>9</td>
<td>3</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>variance</td>
<td>30</td>
<td>25</td>
<td>...</td>
<td>5</td>
<td>3</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**top 200 keep**

**201+ discard**
Hyperspace Analogue to Language

**Advantages**
- Simple
- Fast $O(n)$

**Disadvantages**
- No higher order interactions (only direct co-occurrence)
Latent Semantic Analysis [LSA]

**Target:** a specific word

**Context:** document id

**Weighting:** tf-idf (term frequency - inverse document frequency), *but can use others*

**Similarity:** cosine

**Dimensionality reduction:** Singular Value Decomposition (SVD)

\[
weight_{ij} = \log(f_{ij}) \cdot \log\left(\frac{N}{n_j}\right)
\]

- \(f_{ij}\): frequency of word \(j\) in document \(i\)
- \(N\): total documents
- \(n_j\): total documents containing word \(j\)

**Intuition:** the more frequency in the document, the better. The less frequent in the corpus, the better
SVD in a nutshell

\[ W = U_k \Sigma_k V_k^T \]

**Intuition**
- keep top \( k \) singular values as they contain most of the variance
- \( k \) can be interpreted as the number of topics

**Target matrix**
\[ T_{SV D} = U_k \Sigma_k \]
- Trick from (Levy at al. 2015): throw \( \Sigma \) away for better performance
\[ T_{SV D} = U_k \]

**Context matrix**
\[ C_{SV D} = V_k^T \]
Latent Semantic Analysis

**Advantages**

- Reduced dimension $k$ can be interpreted as topics
- Reducing the number of columns unveils higher order interactions

**Disadvantages**

- Static $\rightarrow$ can't easily add new documents, words and topics
- SVD is one time operation, without intermediate results
- Expensive in terms of memory and computation $O(k^2m)$
Random Indexing [RI]

Locality-sensitive hashing method that approximates the distance between points

Generates **random matrix** $R$ and projects the **original matrix** $A$ to it to obtain a **reduced matrix** $B$

**Reduced space** $B$ preserves the **euclidean distance** between points in **original space** $A$ (*Johnson-Lindenstrauss lemma*)

$B^{n,k} \approx A^{n,m} R^{m,k}$ \quad $k \ll m$

$(1 - \epsilon)d_r(v, u) \leq d(v, u) \leq (1 + \epsilon)d_r(v, u)$
Random Indexing [RI]

Algorithm

- For every word in the corpus create a **sparse random context** vector with values in \{-1, 0, 1\}

- **Target** vectors are the **sum** of the **context** vectors of the words they co-occur with multiplied by the **frequency** of the co-occurrence

**Dataset**

I drink beer
You drink a glass of beer

**Context Vectors**

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>drink</th>
<th>beer</th>
<th>you</th>
<th>glass</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>drink</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>beer</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>you</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>glass</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Target Vectors**

\[
t_{\text{beer}} = 1c_{i} + 2c_{\text{drink}} + 1c_{\text{you}} + 1c_{\text{glass}}
\]

| beer | 0 | -1 | 2 | 0 | 1 | -1 | 1 |
Random Indexing

**Advantages**

- Fast $O(n)$
- Incremental → can add new words any time, just create a new context vector

**Disadvantages**

- In many intrinsic tasks doesn't perform as well as other methods
- Stochasticity in the process → random distortion
- Negative similarity scores
Latent Dirichlet Allocation [LDA]

**Target**: a specific word

**Context**: document id

**Assumptions**:

- **Latent topics** (same idea as \( k \) in LSA)
- Each **topic** is a **Dirichlet distribution** over words
- Each **document** is a **mixture** of corpus-wide **topics**
- Each **word** is drawn from one of the **topics**

![Diagram of LDA model]

The values of \( T \) and \( C \) are probabilities
Latent Dirichlet Allocation

Seeking Life’s Bare (Genetic) Necessities

“are not all that far apart,” especially in comparison to the 75,000 genes in the human genome, notes Me Adams, an evolutionary biologist at Cold Spring Harbor Laboratory in New York. “But coming up with a coherent answer may be more than just a matter of adding up the numbers,” he says. "More and more genes are rapidly being sequenced. It may be a way of organizing the newly sequenced genome," explains Mark Eisen, a computational genomic biologist at the National Center for Biotechnology Information (NCBI), in Bethesda, Maryland.

Although the numbers don’t match precisely, those predictions are important for understanding the evolution of complex organisms. The more genes an organism has, the more complex its biology becomes. However, the exact number of genes can vary widely depending on the organism’s environment and lifestyle.

The diagram illustrates the relationship between different genes and their functions. By analyzing the data, researchers can identify patterns and connections that might not be apparent at first glance. This information is crucial for understanding the complex workings of living organisms.
Latent Dirichlet Allocation

**Advantages**

- Dirichlet prior → each document is about few topics
- Easy to interpret

**Disadvantages**

- Expensive to compute $O(nk^2)$
- Static → can't easily add new documents, words and topics (although some extensions do it)
Explicit Semantic Analysis [ESA]

**Target**: a specific word

**Context**: Wikipedia article

**Assumption**: Wikipedia articles are explicit topics

**Weighting**: tf-idf

**Similarity**: cosine

**Dimensionality Reduction**: discard *too short* articles and articles with few other articles linking them

---

<table>
<thead>
<tr>
<th></th>
<th>Mouse (Rodent)</th>
<th>Mouse (computing)</th>
<th>Mickey Mouse</th>
<th>Button</th>
<th>Janson Button</th>
<th>Drag and Drop</th>
</tr>
</thead>
<tbody>
<tr>
<td>mouse</td>
<td>0.95</td>
<td>0.89</td>
<td>0.81</td>
<td>0.50</td>
<td>0.01</td>
<td>0.60</td>
</tr>
<tr>
<td>button</td>
<td>0.10</td>
<td>0.81</td>
<td>0.20</td>
<td>0.95</td>
<td>0.89</td>
<td>0.70</td>
</tr>
<tr>
<td>mouse button</td>
<td>0.50</td>
<td>0.85</td>
<td>0.50</td>
<td>0.72</td>
<td>0.45</td>
<td>0.65</td>
</tr>
</tbody>
</table>

Average of 2 vectors $→$ emerges disambiguated meaning

<table>
<thead>
<tr>
<th></th>
<th>cat</th>
<th>leopard</th>
<th>jaguar</th>
<th>car</th>
<th>animal</th>
<th>button</th>
</tr>
</thead>
<tbody>
<tr>
<td>Panther</td>
<td>0.83</td>
<td>0.72</td>
<td>0.65</td>
<td>0.3</td>
<td>0.92</td>
<td>0.01</td>
</tr>
</tbody>
</table>
Explicit Semantic Analysis

**Advantages**
- Simple
- Fast $O(n)$
- Interpretable

**Disadvantages**
- The assumption doesn't always hold
- Doesn't perform as good as other methods
- Vectors are really high dimensional, although quite sparse
Generic holing @ operation

Apply it to any tuple to obtain targets (jo) and contexts (bim)

Weighting: custom measure similar to Lin

Similarity: Lexicographer Mutual Information (PMI x Frequency) (Kilgarriff et al. 2004)
Advantages

- Generic preprocessing operation deals with many context representations and types of data
- Deals with complex contexts (example: several steps in a tree)

Disadvantages

- No dimensionality reduction → vectors are high dimensional
- No uncovering of higher order relations
- MapReduce implementation only effective on clusters
Skip Gram with Negative Sampling (SGNS)

**Target:** a specific word

**Context:** window of $n$ words

Vectors are obtained by training the model to predict the **context** given a **target**

The **error** of the prediction is **back-propagated** and the **vectors updated**

Probability that if you randomly pick a word nearby "ant" you will get "car"
The quick brown fox jumps over the lazy dog

They should be the same
Example

The quick brown fox jumps over the lazy dog

Target vectors

Context vectors

softmax → prediction

ground truth

They are different
Back-propagate the error and update the vectors to improve prediction
Model and Loss

\[ p(w_j | w_i) = \text{softmax}(T_i \cdot C_j) = \frac{e^{T_i \cdot C_j}}{\sum_k e^{T_i \cdot C_k}} \]

\[ H(y, \hat{y}) = - \sum_k y_k \log \hat{y}_k \]
Calculating the full softmax is **expensive** because of **large vocabulary**

*The quick brown fox jumps over the lazy dog*

1. Create pairs of **target** and **context** words and predict the probability of them co-occurring to be **1**

   - (fox, quick) → **1**
   - (fox, brown) → **1**
   - (fox, jumps) → **1**
   - (fox, over) → **1**

2. Sample **false context** words from their unigram distribution and predict the probability of them co-occurring with **true target** word to be **0**

   - (fox, quick) → **1** (fox, the) → **0**
   - (fox, brown) → **1** (fox, lazy) → **0**
   - (fox, jumps) → **1** (fox, dog) → **0**
   - (fox, over) → **1** (fox, the) → **0**
Negative Sampling Loss

\[ \log \sigma(T_i \cdot C_j) + \sum_{k \sim P(w)} \mathbb{E} \log \sigma(-T_k \cdot C_j) \]

- **target**: Sample from the distribution of words
- **context**: Number of negative samples
- **Vector of the negative sample**: Sample from the distribution of words
SGNS as matrix factorization

\[
\begin{align*}
\text{Target vectors} & \times \quad \text{Features} \\
\text{Words} & \quad \quad \quad \quad \text{Context vectors} \\
\text{Features} & \quad \quad \quad \quad \text{Contexts} \\
\end{align*}
\]

= ?
SGNS as matrix factorization

\[ \text{log}(k) = \text{PMI} \]

\[ \text{Target vectors} \times \text{Context vectors} = \text{PMI} \]

- Words \times Features = Context vectors
- Words \times Contexts = PMI
- \(-\log(k)\)
word2vec

Advantages

- Iterative way for factorizing a matrix
- Fast $O(nm)$, great implementations
- Several parameters to improve performance (negative samples, subsampling of frequent words, ...)
- Default parameters can go a long way

Disadvantages

- Inflexible definition of context
- Doesn't use dataset statistics in a smart way
- Columns are hard to interpret as topics
Are neural word embeddings better than classic DSMs?

**Yes**
With vanilla parameters
Baroni et al., *Don’t count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors*, 2014

**No**
With optimal parameters
Levy et al., *Improving Distributional Similarity with Lessons Learned from Word Embeddings*, 2015

**Maybe**
Trained on 1 billion+ words
GloVe

Explicit factorization of **target** x **contexts** matrix

Precomputes the matrix (unlike SGNS)

Uses **directly** the **statistics** of the dataset (frequencies of co-occurrences)

\[ J = \sum_{i,j} f(W_{ij})(w_i^T \tilde{w}_j - \log W_{ij})^2 \]

- frequency of word \(i\) in context \(j\)
- **target**
- **context**
- like SGNS
GloVe

**Advantages**
- Better use of dataset statistics
- Converges to good solutions with less data
- Simple to apply on different contexts

**Disadvantages**
- Recent comparisons show that on many tasks it doesn't perform as well as LSA or SGNS
Instead of representing words as points, represent them as **distributions**

**Mean** and **variance** in every dimension

Multimodal **mixes** a **fixed** number of gaussian distributions
Gaussian Embeddings and Multimodal Word Distributions

**Advantages**

- Words as distributions instead of point in a space is a promising direction
- Better treatment of polysemy

**Disadvantages**

- More expensive than previous models
- Still brittle → fixed number of mixtures
Takeaways from literature*

No single algorithm consistently outperforms the others: all models in the same ballpark

SGNS is only slightly better when there is more than 1 billion words in the corpus

iSVD is slightly better in most other cases

SVD better on similarity, SGNS better on analogy

Hyperparameter settings are more important than algorithm choice

Training on a larger corpus helps

*Levy, Goldberg and Dagan, Improving Distributional Similarity with Lesson Learn from Word Embeddings, 2015
Recommendations from literature*

**DON’T** use shifted PPMI with SVD

**DON’T** use SVD “correctly”, i.e. without eigenvector weighting, throwing away Sigma

**DO** use PPMI and SVD with short contexts (window size of 2)

**DO** use many negative samples with SGNS

**DO** always use context distribution smoothing (raise unigram distribution to the power of $\alpha=0.75$)

**DO** use SGNS as a baseline (robust, fast and cheap to train)

**DO** try adding context vectors in SGNS and GloVe

*Levy, Goldberg and Dagan, *Improving Distributional Similarity with Lesson Learn from Word Embeddings*, 2015*
Open questions and current trends
So far we represented words as vectors, how to represent sentences?

Can't use the co-occurrences of sentences in their context as sentences are sparse, most of them occur once.

Should represent their meaning combining word representations.

*The meaning of an utterance is a function of the meaning of its parts and their composition rules* - Gottlob Frege, *Über Sinn und Bedeutung*, 1892.
Composition operators

**Simple solution**, just **sum** the vectors of the words in a sentence!

Other **operators**: product, weighted sum, convolution, ... (Mitchell and Lapata, 2008)

It's hard to perform better than the simple **sum**

**Sum** can't be the real answer as it's **commutative** → doesn't consider **word order**
Learn to compose

Recursive Matrix Vector Network (Socher at al. 2012)

Recursive Neural Tensor Network (Socher et al. 2013)

Recurrent Neural Network (Elman 1990) and others

Legend

vector | matrix | tensor
sum | product | concat

sentence

[w, w_1, \ldots, w_i, \ldots, w_n]

\[C \times X + \text{vector}

sentence

[w_1, w_2, \ldots, w_n]

sentence

[w_1, w_2, \ldots, w_n]

\[C \times X + \text{vector}

\[o_i, o_n\]

\[w_i, w_j\]

\[X\]
Subword structure

**Assumption:** similar words are similarly spelled *(player / played)*

Exploit **characters** and **character sequences**

Useful to deal with **misspells** and **rare / new words** *(player ~ pleyer)*

Beware of **pitfalls** *(pray / prey)*

- **CharCNN** (Zhang, Zhao and LeCun 2015)
- **LSTM with word CharCNN** (Kim 2016)
- **FastText** (Bojanowski 2016)
Embeddings for Graphs
Embeddings for Graphs
Knowledge Graph

A knowledge graph is a kind of information model that represents information as a graph, consisting of nodes and edges. Nodes represent entities, while edges represent relationships between these entities. These relationships are denoted by predicates, and the combinations of subjects, predicates, and objects are called SPO triples. For example, in the graph shown:

- **Leonard Nimoy** played **Spock**.
- **Star Wars** is a **Science Fiction** movie.
- **Alec Guinness** starred in **Star Wars**.

Knowledge graphs can be represented using different standards, such as the Resource Description Framework (RDF). RDF triples are of the form (subject, predicate, object), where:

- **subject** is the starting point of the relationship.
- **predicate** indicates the type of relationship.
- **object** is the end point of the relationship.

In a knowledge graph, nodes represent entities, and edges represent relationships between the entities. Directed edges indicate the direction of the relationship, and undirected edges represent symmetric relationships.

---

**TABLE I**

<table>
<thead>
<tr>
<th>Construction Type</th>
<th>Creation Method</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automated Semi-Structured Yes</td>
<td>Knowledge Vault, DBpedia</td>
<td>YAGO, DBpedia</td>
</tr>
<tr>
<td>Automated Semi-Structured No</td>
<td>NELL, Probase</td>
<td>Semi-Structured Wiki, Freebase</td>
</tr>
<tr>
<td>Collaborative Yes</td>
<td>Wikidata, Freebase</td>
<td>YAGO, DBpedia</td>
</tr>
<tr>
<td>Collaborative No</td>
<td>Prisma, Google Knowledge Graph</td>
<td>YAGO, DBpedia</td>
</tr>
<tr>
<td>Closed World Assumption</td>
<td>Knowledge Vault, DBpedia</td>
<td>Knowledge Vault, DBpedia</td>
</tr>
<tr>
<td>Open World Assumption</td>
<td>Auto. Unstructured No</td>
<td>Knowledge Vault, DBpedia</td>
</tr>
</tbody>
</table>

**VII-B**

Knowledge graphs are used in various applications, such as search engines, recommendation systems, and answering complex queries. They can be constructed in different ways, ranging from fully automated to manually curated approaches. The quality of knowledge graphs depends on factors such as completeness, accuracy, and data quality.

**B. Open vs. closed world assumption**

In knowledge graphs, the choice of world assumption (OWA) can affect the interpretation of relationships. The closed world assumption (CWA) assumes that all non-existing facts are unknown, while the open world assumption (OWA) assumes that all non-existing facts are false. The choice of world assumption can influence the performance of reasoning systems.

**C. Knowledge base construction**

The construction of knowledge bases is a critical step in the development of knowledge graphs. It involves the collection and integration of data from various sources. The resulting knowledge base is a set of triples that represent facts and relationships.

<table>
<thead>
<tr>
<th>Construction Type</th>
<th>Creation Method</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automated Semi-Structured Yes</td>
<td>Knowledge Vault, DBpedia</td>
<td>YAGO, DBpedia</td>
</tr>
<tr>
<td>Automated Semi-Structured No</td>
<td>NELL, Probase</td>
<td>Semi-Structured Wiki, Freebase</td>
</tr>
<tr>
<td>Collaborative Yes</td>
<td>Wikidata, Freebase</td>
<td>YAGO, DBpedia</td>
</tr>
<tr>
<td>Collaborative No</td>
<td>Prisma, Google Knowledge Graph</td>
<td>YAGO, DBpedia</td>
</tr>
<tr>
<td>Closed World Assumption</td>
<td>Knowledge Vault, DBpedia</td>
<td>Knowledge Vault, DBpedia</td>
</tr>
<tr>
<td>Open World Assumption</td>
<td>Auto. Unstructured No</td>
<td>Knowledge Vault, DBpedia</td>
</tr>
</tbody>
</table>

**VII-B**

In conclusion, knowledge graphs are powerful tools for representing and reasoning about complex information. They are used in various applications, and their quality depends on the construction methods and world assumptions used in their development.
Knowledge Graphs

Tensor Factorizations
(Nickel et al. 2015)

Universal Schema
(Riedel et al. 2013)
Exotic applications

item2vec - recommender systems (Barkan and Koenigstein 2016)

node2vec - graph embeddings (Grover and Leskovec 2016)

dna2vec (Ng 2017)

Predicting drug-drug interactions (Fokoue 2016)

Movies, music, playlists, recipes, ...
Conclusions

**Know the theory** *structuralism* and everything **makes sense**

Distributional Semantics and Embeddings have a **long rich history**

**Context is king**

**No algorithm** to *rule them all*, but a **great toolset** to chose from

Many aspects of reality can be seen in terms of **targets** and **contexts**

Go out and apply them to your business!
Thanks

Influenced this talk:
Magnus Sahlgren
Alfio Gliozzo
Marco Baroni
Alessandro Lenci

Contacts
piero.molino@gmail.com
http://w4nderlu.st

Yoav Goldberg
Andre Freitas
Pierpaolo Basile
Aurélie Herbelot
Arianna Betti