Computer-Linguistische Anwendungen

CLA | B.Sc. | LMU

Embeddings Learned by Matrix Factorization

Philipp Wicke, PhD
Centrum für Sprach- und Informationsverarbeitung
Ludwig-Maximilians-Universität München
pwicke@cis.lmu.de

Slides recycled from B. Roth / H. Schuetze

Überblick

- WordSpace Limitationen
- Lineare Algebra Wiederholung
- Input matrix
- Matrix Faktorisierung
- Diskussion

WordSpace Limitationen

Embeddings

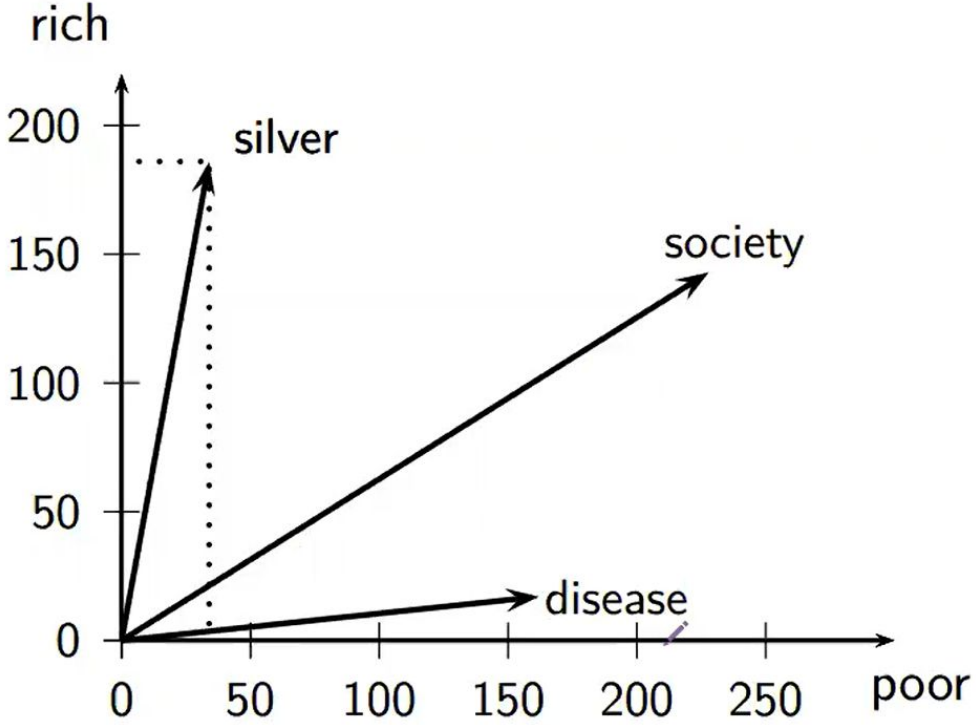
Definition

Das Embedding eines Wortes *w* ist ein dense Vektor (im Gegensatz zu *sparse*). Dieser Vektor $\vec{v}(w) \in R^k$ repräsentiert semantische und weitere Eigenschaften von *w*. Typische Werte sind $50 \leq k \leq 1000$.

- In dieser Hinsicht sind Embeddings nicht anders als WordSpace: Beide, Embedding und WordSpace Vektoren, sind Repräsentationen von Wörtern, hauptsächlich semantisch, können aber auch andere Eigenschaften abbilden.
- Embeddings haben eine viel geringere Dimensionalität als WordSpace Vektoren

- WordSpace Vektoren sind *sparse* (die meisten Einträge sind 0)
- Embedding Vektoren sind *dense* (es gibt fast nie Einträge die 0 sind)

WordSpace


rich
200
150
100
50
0
silver
society
disease
0
50
100
150
200
250
poor

Wort Repräsentationen: Dichte und Dimensionalität

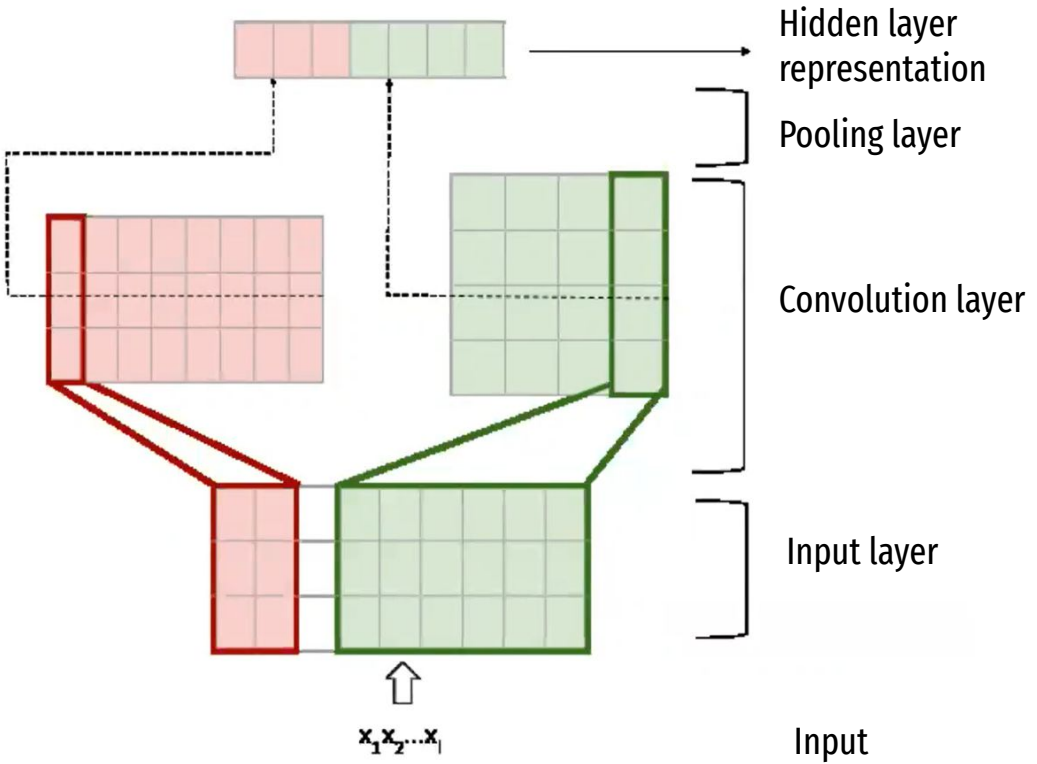
- WordSpace Vektoren sind sparse und hoch multidimensional
- Embedding Vektoren sind dense und weniger multidimensional
- Warum sind Embeddings potentiell besser?
    - Embeddings sind effizienter
    - Embeddings sind oft effektiver

Effizienz von Embeddings

Hohe Dimensionalität

→ langsames Training

Die Zeit, die ein neuronales Netz für Training braucht, ist ungefähr linear zur Dimensionalität des Wort Vektors.



WordSpace Vektoren: Beispiele für Ineffektivität

- Beispiel: Polaritäts-Klassifikation (negativ vs. positives Wort)

- Zusammen Auftreten mit “bad” weist auf negative Polarität hin

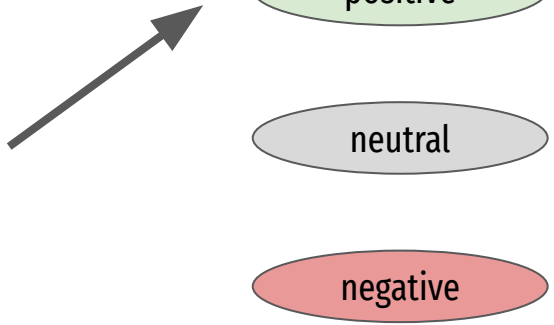
  Mögliches Ergebnis:

  - Inkorrekte Klassifikation basierend auf WordSpace Vektoren

- Embeddings sind robuster und “füllen” fehlende Daten

Effektivität der Embeddings: Polarität

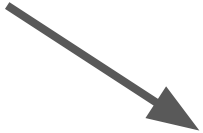
$$\begin{pmatrix} \text{high} & : 0 \\ \text{cold} & : 0 \\ \text{tree} & : 1 \\ \text{good} & : 2 \\ \text{white} & : 0 \\ \text{bad} & : 0 \\ \text{sweet} & : 1 \\ \text{smelly} & : 0 \end{pmatrix}$$



$\overrightarrow{v}(\text{apple})$

Effektivität der Embeddings: Polarität

high : 0
cold : 2
tree : 2
good : 0
white : 1
bad : 0
sweet : 1
smelly : 5



positive

neutral

negative

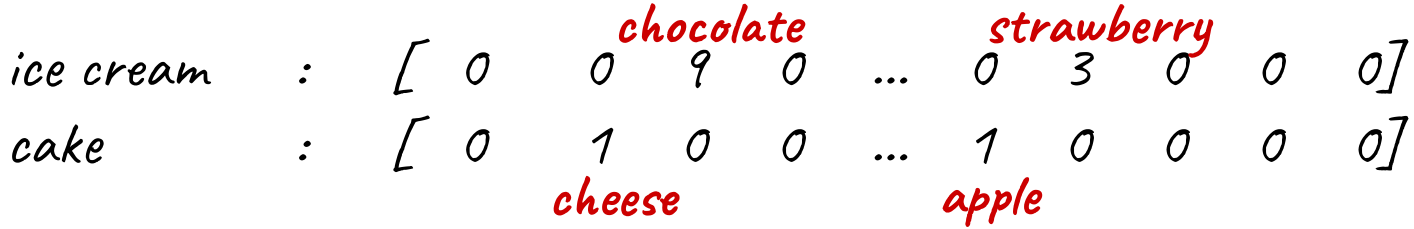
$\vec{v}$(skunk)

Effektivität von WordSpace: Gedankenexperiment

Bauen ein Beispiel für einen Korpus mit zwei Worten w_1 und w_2 mit folgenden Eigenschaften:

- w_1 und w_2 sind **semantisch verwandt**
- Der WordSpace Vektor von w_1 und w_2 sind **nicht ähnlich**

Ziel: Embeddings entfernen die Fehlermomente von WordSpace



Word Embedding Lernen: Parameter Schätzung

- Das Embedding eines Wortes ist ein Vektor $v \in R^K$
- Die Koordinaten sind die Parameter welcher wir vom Korpus lernen/schätzen müssen
- Lernen eines WordSpace models:
    - (i) zähle, dann (ii) PPMI gewichten

- Für Word Embeddings ist das Lernen etwas komplizierter
- Zwei unterschiedliche Methoden:
    - Embeddings werden via Matrix Faktorisierung gelernt
    - Embeddings werden via Gradient Descent (word2vec) gelernt

- Beide dieser Schätzungen sind ungefähr ähnlich

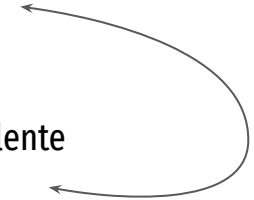
Beispiel eines Embedding Vektors: Die Zahlen sind die Parameter

Embedding für das Wort “skunk” (jede Dimension bildet eine abstrakte Eigenschaft ab):

```
[0.0564, 0.0687, -0.0845, -0.1265, -0.0375, -0.0268, 0.0119, -0.0183,
0.0453, -0.0137, 0.0621, -0.0159, -0.0299, -0.0368, -0.0261, -0.0175,
-0.0198, -0.0329, -0.0523, -0.0323, 0.0281, 0.0418, -0.0549, 0.0345,
-0.0538, -0.0407, -0.0335, -0.0235, -0.0277, -0.0128, 0.0115, -0.0523,
-0.0198, 0.0379, -0.0241, -0.0275, -0.0265, -0.0044, -0.0534, 0.0041,
-0.0628, -0.0084, -0.0456, -0.0431, -0.0406, 0.0431, -0.0026, 0.0542,
0.0191, 0.0229, -0.0476, -0.0251, -0.0286, -0.0366, -0.0455, -0.0431,
0.0376, -0.0437, 0.0019, -0.0423, -0.0312, 0.0367, 0.0242, 0.0386,
-0.0268, -0.0081, -0.0378, 0.0247, -0.0426, -0.0348, -0.0366, 0.0357,
0.0323, 0.0261, -0.0528, 0.0359]
```

Word Embedding Parameter-Schätzung: Historischer Kontext

- Mikolov et al. (2013) zeigen **word2vec**, schätzen die Parameter mit Gradient Descent ab.
  - Dies ist der Lern-Algorithmus welcher bis heute der default für die meisten Fälle ist



- Levy und Goldberg (2014) zeigen, dass ein gewisser Typ der Matrix Faktorisierung nahezu äquivalente Ergebnisse bringt
  - Dies ist besonders wichtig, weil zwei Forschungsfelder davon Nutzen machen können: Neuronale Netze und Distributional Semantics

- Auch wenn die zeitliche Reihenfolge eine andere war, ist es inhaltlich in dieser Vorlesung sinnvoller folgenden Reihenfolge zu betrachten: Distributional Semantics → Embedding Lernen mit Matrix Faktorisierung → Embedding Lernen mit Gradient Descent