Computer-Linguistische Anwendungen

CLA | B.Sc. | LMU

Vorlesung:
Embeddings via Gradient Descent

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

Übersicht

- word2vec skipgram Versionen
- skipgram negative sampling (SGNS): objective
- Embeddings via Gradient Descent
- Skipgram (Word2Vec): Practical Implementation
- Visualisation
- FastText
- Takeaways

Simple Wort (Cooccurrence) Vektoren vs. Word Embeddings

Simple Wort (Cooccurrence) Vektoren (e.g. WordSpace Model)

- Jede Dimension ist ein spezifisches Wort im Vokabular
- Es gibt so viele Dimensionen wie es Worte im Vokabular gibt → sparse (und meist ineffiziente) Vektoren

C	*ship*	*boat*	*ocean*	*wood*
ship	-	0	1	0
boat	0	-	0	0
ocean	1	1	-	0
wood	1	0	0	-

Word Embeddings (e.g. Word2Vec)

Simple Wort (Cooccurrence) Vektoren vs. Word Embeddings

Simple Wort (Cooccurrence) Vektoren (e.g. WordSpace Model)

- Jede Dimension ist ein spezifisches Wort im Vokabular
- Es gibt so viele Dimensionen wie es Worte im Vokabular gibt → sparse (und meist ineffiziente) Vektoren

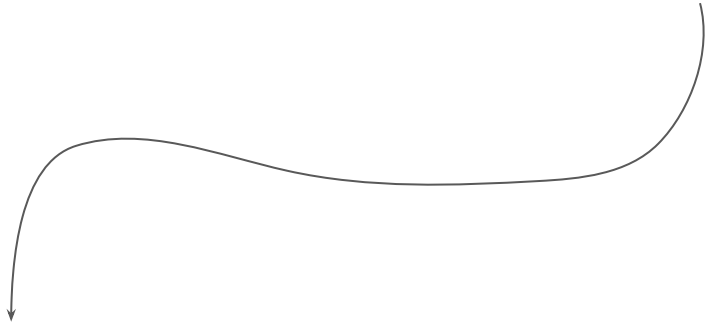
C	*ship*	*boat*	*ocean*	*wood*
ship	-	0	1	0
boat	0	-	0	0
ocean	1	1	-	0
wood	1	0	0	-

Word Embeddings (e.g. Word2Vec)

- Die Dimensionen sind abstrakt, sie repräsentieren semantische Eigenschaften die nicht genau benannt werden können
- Es gibt nur so viele Dimensionen wie wir für unser Model wählen (e.g. 100 oder 300) dense (und meist effiziente) Vektoren

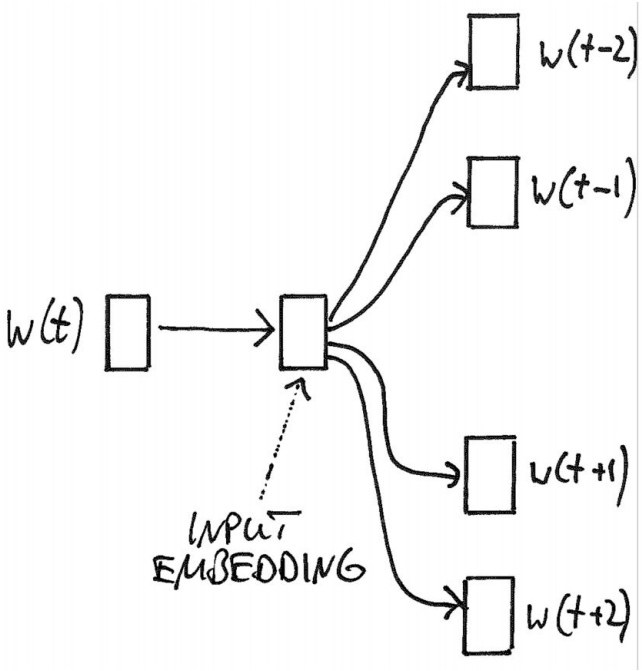
C	*dim*1	*dim*2	*dim*3
ship	3.5	0.5	1.3
boat	1.6	0.33	1.45
ocean	2.3	0.56	0.67
wood	1.5	0.12	1.5

word2vec skipgram Versionen

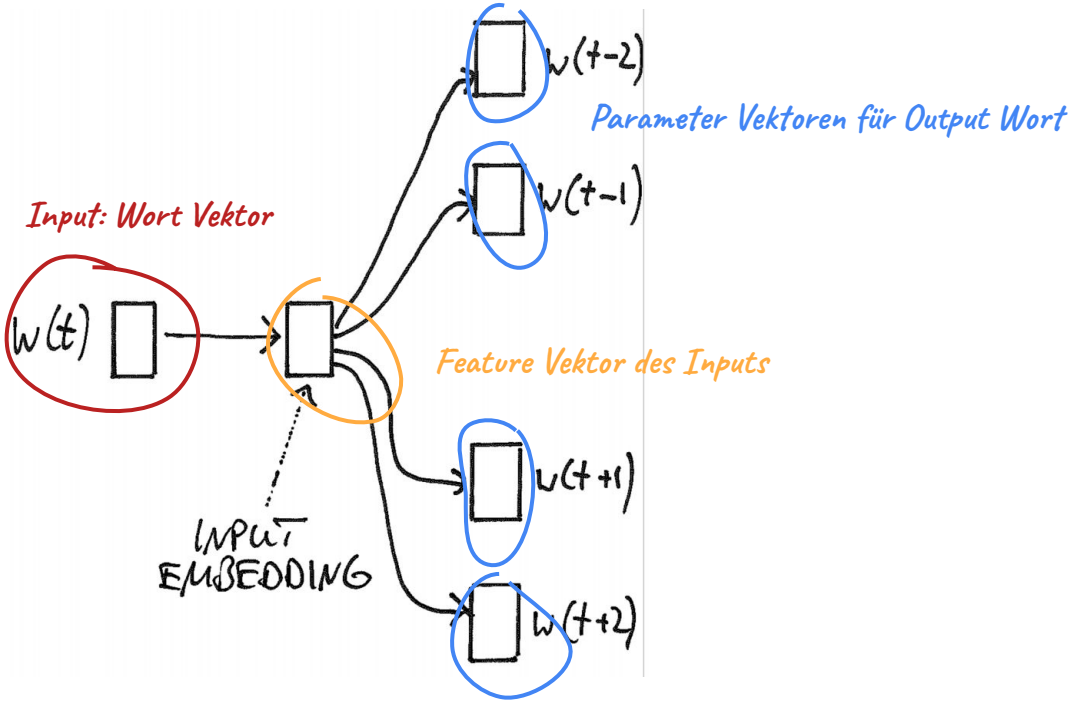


Der Name "skipgram" kommt von der Tatsache, dass das Modell "Skips" oder "Übersprünge" macht mit Wörter im Text, um das Kontextfenster zu definieren, in dem es die Wahrscheinlichkeit von anderen Wörtern vorhersagt.

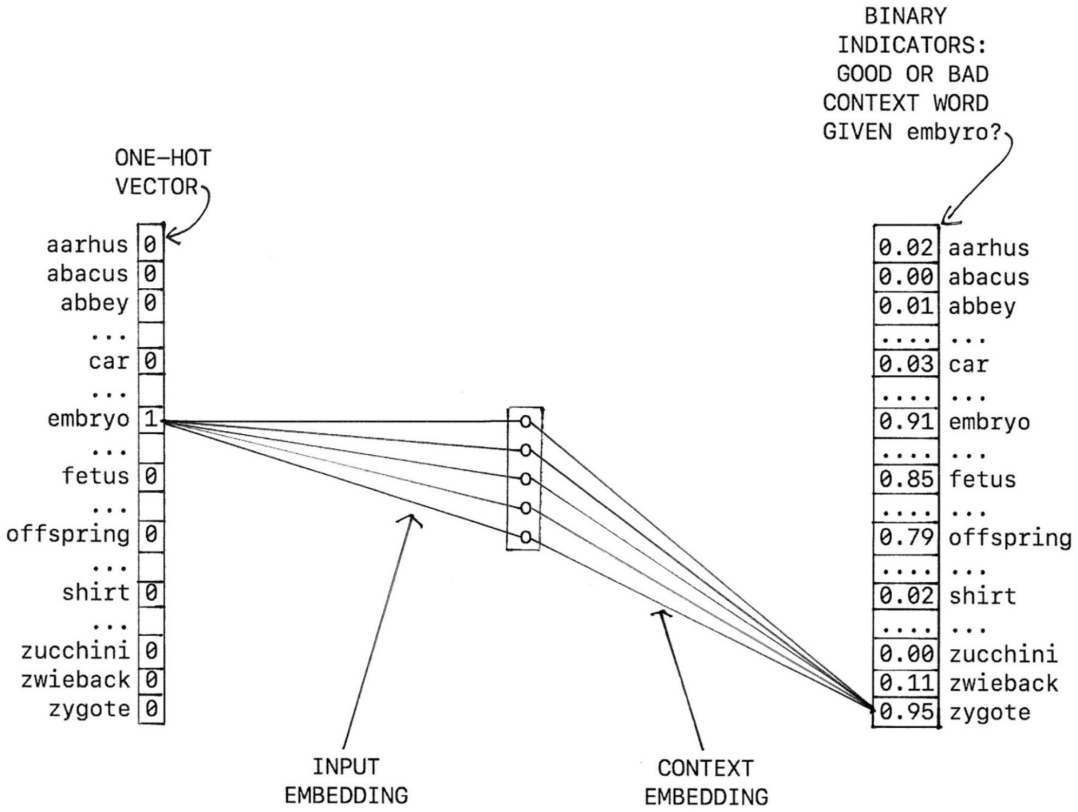
word2vec skipgram → Idee: Sage auf Grund eines Eingabe-Wortes ein Kontext-Wort hervor (und umgekehrt)



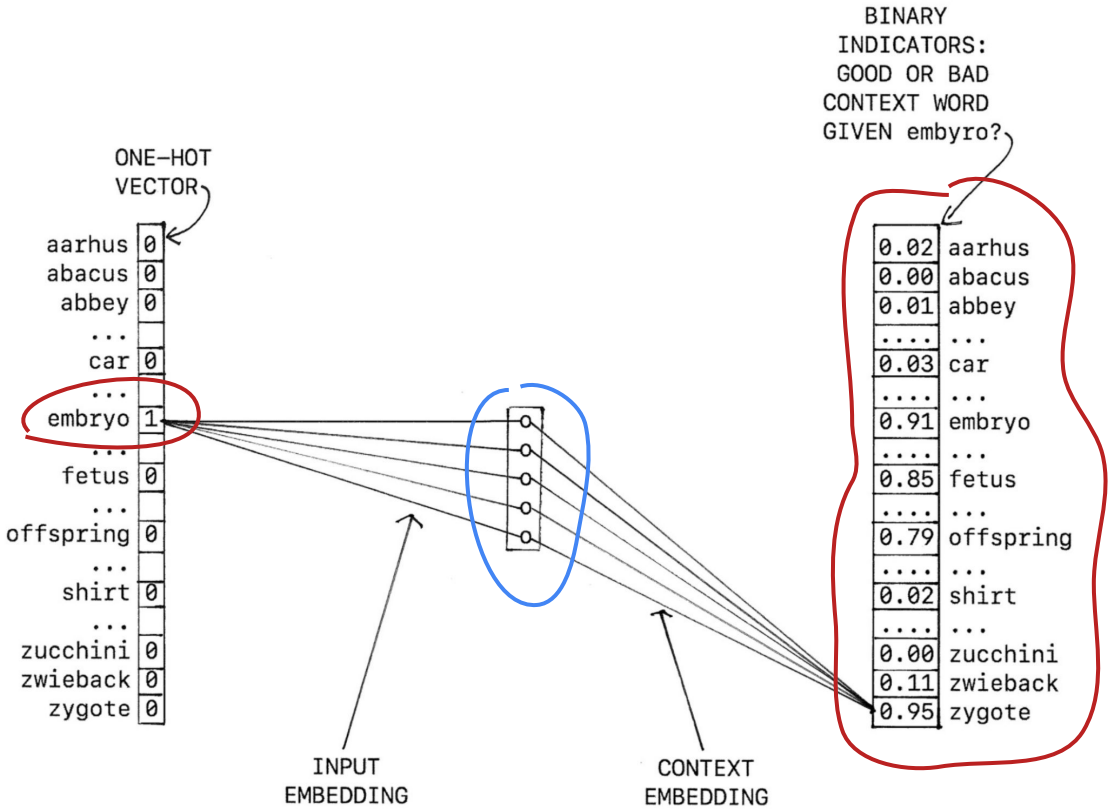
word2vec skipgram → Idee: Sage auf Grund eines Eingabe-Wortes ein Kontext-Wort hervor (und umgekehrt)



word2vec skipgram → Idee: Sage auf Grund eines Eingabe-Wortes ein Kontext-Wort hervor (und umgekehrt)



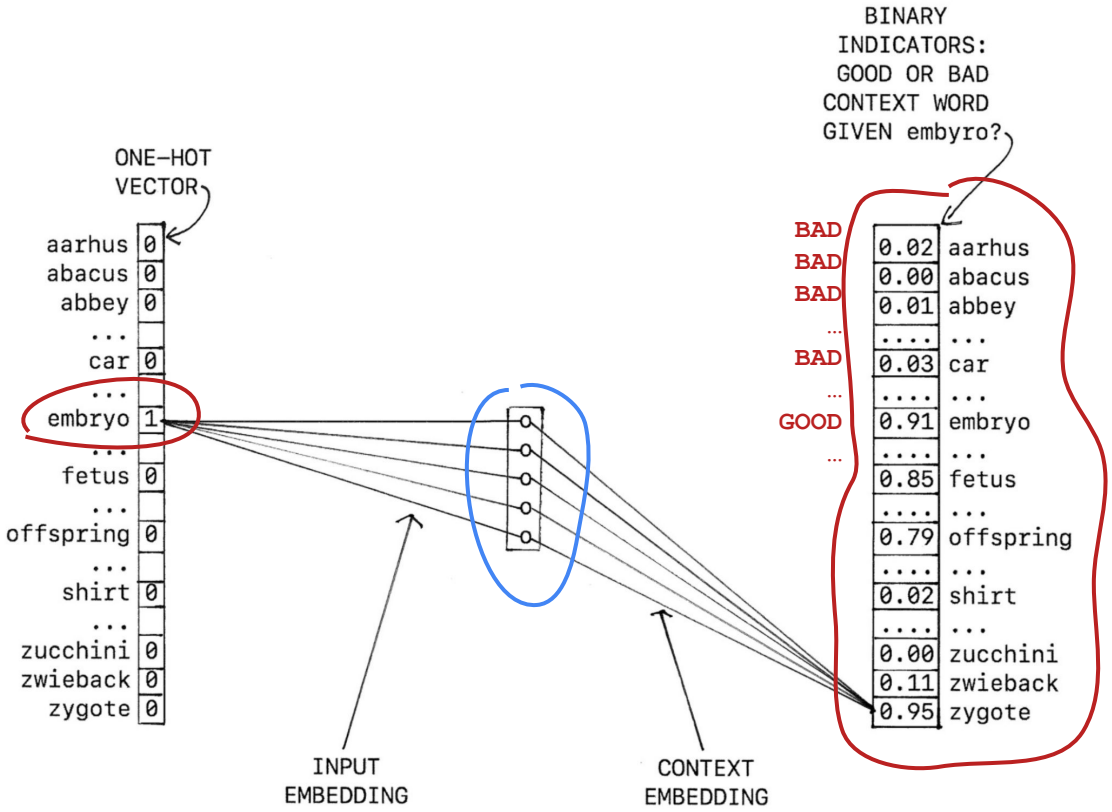
word2vec skipgram → Idee: Sage auf Grund eines Eingabe-Wortes ein Kontext-Wort hervor (und umgekehrt)



Erster Ansatz:

P (Kontext | Eingabe)

word2vec skipgram → Idee: Sage auf Grund eines Eingabe-Wortes ein Kontext-Wort hervor (und umgekehrt)



Input Vektor des Ziel Worters:

Für jedes Kontext Wort wird Kontext-Wort-Vektor erzeugt und Ähnlichkeit zwischen Eingabe und Kontext Embedding verglichen

P (GOOD | Eingabe, Kontext)
P (BAD | Eingabe, Kontext)

e.g., `P(GOOD | embryo, car)`

Kommt das Paar in den Trainingsdaten vor?

Drei Methoden um Wort-Vektoren zu erhalten

- Alle drei haben das skipgram Ziel (previous slide):
  Sage vorher ob ein Eingabe-Wort wahrscheinlich zusammen mit einem Kontext-Wort auftritt

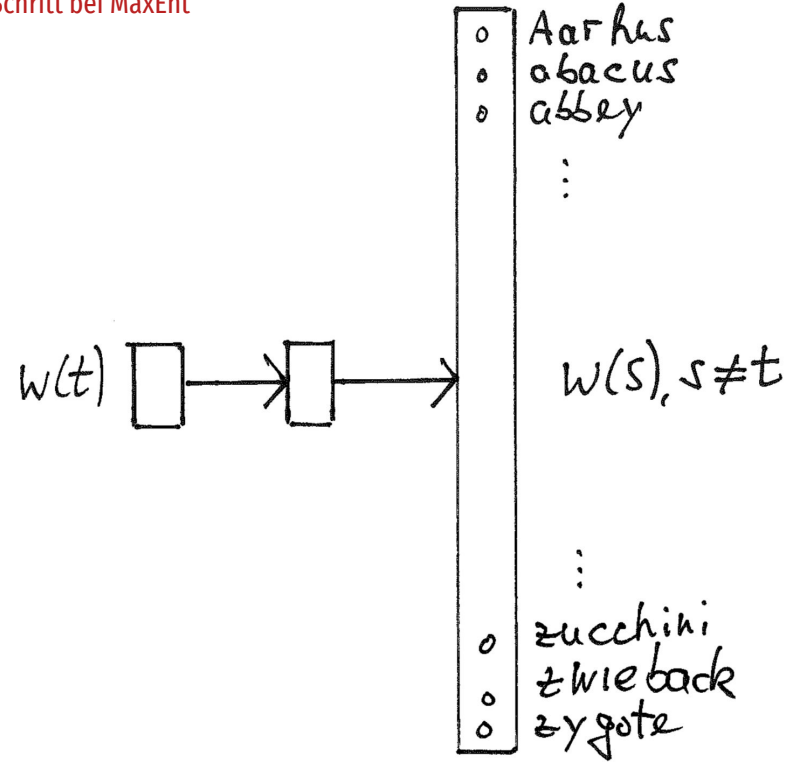
  1. Matrix factorization (SVD) der PPMI matrix
     - Letzte Vorlesung

  2. word2vec skipgram negative sampling (SGNS) using GD
     - Heutiges Thema
     - Levy & Goldberg zeigen annähernde Äquivalenz:
       SGNS ≈ SVD-of-PPMI-matrix

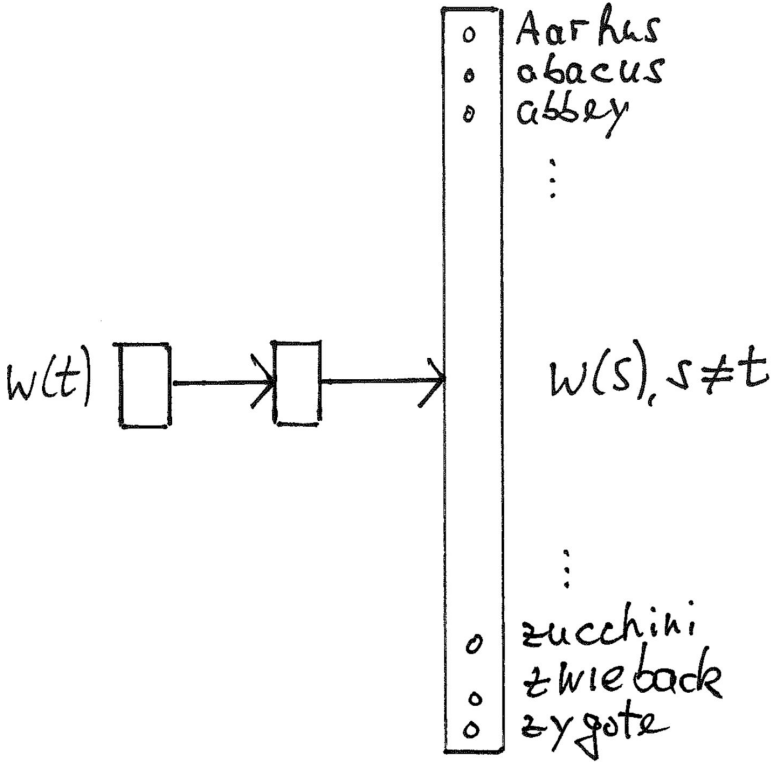
  3. word2vec hierarchical softmax (skipgram HS)
     - skipgram HS vs. SGNS: Unterschiedliche Ziele

Skipgram Softmax

Letzter Schritt bei MaxEnt



Skipgram Softmax



P (Kontext | Eingabe)

Wie wahrscheinlich ist dieses Paar?

P (GOOD | Eingabe, Kontext)
P (BAD | Eingabe, Kontext)

Kommt das Paar in den Trainingsdaten vor?

Skipgram Softmax: Ziel

$$\text{argmax}_\theta \sum_{(w,c)\in D} \log \frac{\exp(\vec{v}_w \cdot \vec{v}_c)}{\sum_{c'\in V} \exp(\vec{v}_w \cdot \vec{v}_{c'})}$$

P (Kontext | Eingabe)

Wie wahrscheinlich ist dieses Paar?

Skipgram Softmax: Ziel

Ineffizient! Für jedes Kontext-Wort muss der Score bei jeder Optimierung berechnet werden

$$\text{argmax}_\theta \sum_{(w,c)\in D} \log \frac{\exp(\vec{v}_w \cdot \vec{v}_c)}{\sum_{c'\in V} \exp(\vec{v}_w \cdot \vec{v}_{c'})}$$

- Die gleiche SoftMax Funktion die wir bei der Linearen Regression gesehen haben (wandelt einen Score für eine Klasse z_i in eine Wahrscheinlichkeit um)
- $\overrightarrow{\boldsymbol{v}_w}$: Vektor des aktuellen Wortes, $\overrightarrow{\boldsymbol{v}_c}$: Vektor des Kontext-Wortes, $\overrightarrow{\boldsymbol{v}_c}$: Vektor aller anderen Kontext-Worte
- $(\vec{v}_w \cdot \vec{v}_c) \approx z_i$ Score der Zielklasse für welches wir die Wahrscheinlichkeit errechnen wollen
- $(\vec{v}_w \cdot \vec{v}'_c) \approx z_j$ Score aller anderen Klassen
- $\frac{\exp(\vec{v}_w \cdot \vec{v}_c)}{\sum_{c'\in V} \exp(\vec{v}_w \cdot \vec{v}_{c'})}$ → P (Kontext | Eingabe-Wort), die Wk des Kontextes gegeben dem Eingabe-Wort
- argmax_θ : Parameter (Wort-Vektoren) die diese log-Wahrscheinlichkeit maximieren

Drei Versionen des Skipgram: Lern-Algorithmen

Methode:	**Lernalgorithmus**
Word2Vec Negative Sampling (Ursprüngliche Version):	Gradient Descent
Word2Vec SVD (Levy & Goldberg):	SVD
Word2Vec Hierarchical Softmax:	Gradient Descent