

G. Dinu, M. Lapata (2010)

# Measuring Distributional Semantics in Context

Melanie Tosik

6. Dezember 2013

# Inhaltsübersicht

## Motivation

## Modell

- Bedeutungsrepräsentation im Kontext

- Parametrisierte Algorithmen

  - Nicht-negative Matrixfaktorisierung

  - Latent Dirichlet Allocation

## Experimentelles Design

- Aufgabenstellungen

- Training des Modells

- Baselines

## Ergebnisse

- Wortähnlichkeit

- Lexikalische Ersetzung

## Zusammenfassung

# Outline

## Motivation

## Modell

Bedeutungsrepräsentation im Kontext

Parametrisierte Algorithmen

Nicht-negative Matrixfaktorisierung

Latent Dirichlet Allocation

## Experimentelles Design

Aufgabenstellungen

Training des Modells

Baselines

## Ergebnisse

Wortähnlichkeit

Lexikalische Ersetzung

## Zusammenfassung

# Distributionelle Semantik

Was machen wir hier nochmal?

Wir versuchen, **Wortbedeutungen** anhand der **Wortverteilung in Texten** zu modellieren.

Achja, und warum?

Typischerweise um **semantische Ähnlichkeit** von Wörtern zu modellieren.

Und was nützt uns das?

Wir können zum Beispiel automatisch:

- ▶ Synonyme und Paraphrasen generieren  
(Natural Language Generation, data-to-text Systeme)
- ▶ Wortsinne disambiguieren  
(Suchmaschinen, Maschinelle Übersetzung, Diskursstrukturen)
- ▶ Komplexe Inhalte erschließen (Information Retrieval)

# Distributionelle Semantik

Also?

- ▶ Berechnen wir Bedeutungsähnlichkeiten mithilfe von **Vektorraum-Modellen**.
- ▶ Solche Modelle sind am besten geeignet, um **Wörter in Isolation** zu repräsentieren.
- ▶ Sie können außerdem **unüberwacht** gelernt werden und sind mit **wenig Aufwand** zu berechnen.

Wie genau funktioniert das?

- ▶ Ein Wort wird als Punkt in hochdimensionalem Vektorraum repräsentiert, der auf **Kookurrenzen** im Text basiert.
- ▶ Ähnlichkeit wird dann z.B. als Kosinusähnlichkeit zwischen den Vektoren dargestellt.

# Distributionelle Semantik

Was machen wir dann noch hier?

Einfache Vektorraummodelle erfassen Wortähnlichkeiten in der Regel **unabhängig vom Kontext**.

(**Recap**: Polysemie-Problem, *Bank* als ein Punkt im Vektorraum)

→ Sie erfassen keine verschiedenen Wortsinne.

Beispiel: *heavy*

Generelle Bedeutung: *dense, massive*

Im Kontext aber auch andere Bedeutungen möglich:

1. *She is short and heavy but she has a heart of gold.*  
→ schwer an Gewicht
2. *Some heavy users develop a psychological dependence on cannabis.*  
→ schwerwiegend, besonders stark

# Distributionelle Semantik

**Indirekte** Lösungsversuche:

Mitchell & Lapata, 2008; Erk & Padó, 2008; Thater et al., 2010

Entwickeln **spezialisierte Modelle**, die Wortbedeutung im Kontext repräsentieren. Erst werden Kookurrenz-Vektoren extrahiert, die noch **alle** Wortsinne enthalten. Nach Anwendung verschiedener Vektor-Operationen erhalten wir anschließend:

- ▶ Kontextualisierte Repräsentationen eines Zielwortes (Erk & Padó)
- ▶ Kontextualisierte Repräsentationen für eine Menge von Wörtern (Mitchell & Lapata)

**Grundproblem:** Kompositionalität, Wortbedeutung im Kontext

# Kompositionalität

**Kompositionalitätsprinzip:** Die Bedeutung eines komplexen Ausdrucks ist eine Funktion der Bedeutung seiner Teile und ihrer morpho-syntaktischen Kombination.

$$[[\text{Wort}_1 \circ \text{Wort}_2]] = f \circ ([[ \text{Wort}_1 ]], [[ \text{Wort}_2 ]])$$

Beispiel:  $[[\text{jeder Student arbeitet}]] = f([[ \text{jeder Student} ]], [[ \text{arbeitet} ]])$

Mitchell & Lapata, 2008:

- ▶  $v(\text{Wort}) =$  Vektor eines Zielwortes
- ▶  $v(\text{Wort}_1 \circ \text{Wort}_2) = v(\text{Wort}_1) + v(\text{Wort}_2)$
- ▶  $v(\text{Wort}_1 \circ \text{Wort}_2) = v(\text{Wort}_1) \times v(\text{Wort}_2)$
- ▶ ...

→ Bag-of-words Vektoren erfassen aber nicht nur Kompositionalität.



## Wortbedeutung im Kontext

Typvektoren eines Wortes repräsentieren die Kontexte jedes Wortvorkommens im Korpus.

→ Die Kontexte der verschiedenen Wortsinne stehen alle in ein und demselben Vektor.

Wir wollen aber Vektoren, die die verschiedenen **Wortsinne im jeweiligen Kontext** repräsentieren.

Erk & Padó, 2008:

Strukturierte Vektorrepräsentationen; Wörter werden von einem **“lexikalischen” Vektor** und einem **Vektor für selektionale Präferenzen der syntaktischen Rollen** repräsentiert

$$w = \langle v_w, R_{subj}, R_{obj}, \dots, R_{subj-1}, R_{obj-1}, \dots \rangle$$

Beispiel:  $v(Maus \mid frisst) = v(Maus) \times R_{subj-1}(frisst)$

## Wortbedeutung im Kontext

Weiterentwicklung:

Thater et al., 2010:

Grundidee: Bedeutungsrepräsentation des Zielwortes in Vektor zweiter Ordnung, der die Vektoren der Kontextwörter aufsummiert

Bedeutungsrepräsentation:

- ▶  $v_1(\textit{frisst})$  = Vektor für *frisst*
- ▶  $v_2(\textit{Maus}) = v_1(\textit{Käse}) + v_1(\textit{Eingabegerät}) + v_1(\textit{Daumen}) + \dots$

Wortbedeutung im Kontext:

- ▶  $v_2(\textit{Maus} \mid \textit{frisst}) = v_2(\textit{Maus}) \times v_1(\textit{frisst})$

Selektionale Präferenzen werden dabei wie eben mit eingerechnet.

# Outline

## Motivation

## Modell

Bedeutungsrepräsentation im Kontext

Parametrisierte Algorithmen

Nicht-negative Matrixfaktorisierung

Latent Dirichlet Allocation

## Experimentelles Design

Aufgabenstellungen

Training des Modells

Baselines

## Ergebnisse

Wortähnlichkeit

Lexikalische Ersetzung

## Zusammenfassung

# Jetzt aufgepasst!

Modell von Dinu & Lapata, 2010:

- = **Probabilistisches Modell** zur Bedeutungsrepräsentation von Wörtern und Berechnung von Wortähnlichkeiten im Kontext
- ▶ Bedeutungsrepräsentation von einzelnen Wörtern dabei als **Wahrscheinlichkeitsverteilung über latente Wortsinne** (noch kein Kontext)
- ▶ Verschiedene Wortbedeutungen dann anhand der **Veränderung der Wahrscheinlichkeitsverteilung** der Wortsinne  
Beispiel: *Bank* – Geldinstitut 0.5, Sitzgelegenheit 0.5  
→ *Kredit* Kontext: Geldinstitut 0.9, Sitzgelegenheit 0.1
- ▶ Modell außerdem nicht gebunden an eine konkrete Methode zur Induktion von Wortbedeutung

# Framework

Was bisher geschah:

- ▶ Vektorraum-Modelle basierend auf Kookurrenzmatrizen
- ▶ Zeilen: Zielwörter, Spalten: Dokumente oder Kontextwörter, Zellen: Kookurrenzhäufigkeiten
- ▶ Wortbedeutungen anhand der Distanzen im Vektorraum, z.B. durch Berechnung der Kosinusähnlichkeit

Ähnlichkeit geht jetzt anders!

Was bleibt:

- ▶ Input: Kookkurrenzmatrix
- ▶ Zeile: Zielwort  $t_i$ , Spalte: Kontextwort  $c_j$ , Zelle  $(t_i, c_j)$ : Kookurrenzhäufigkeit von  $t_i$  mit  $c_j$

Was NEU ist: Bedeutungsrepräsentation über latente Wortsinne

# Bedeutungsrepräsentation über latente Wortsinne

## Annahme:

Für jedes Zielwort  $t_i$  gibt es eine globale Menge an Wortsinnen:

$$Z = \{z_k | k : 1 \dots K\}$$

$K$  ... Anzahl der Wortsinne (festgelegter Wert)

$z_k$  ... Konkreter Wortsinn

Die Bedeutung von  $t_i$  kann als Wahrscheinlichkeitsverteilung über diese Menge der Wortsinne betrachtet werden.

Vektor  $\mathbf{v}$  repräsentiert dann das Zielwort  $t_i$ :

$$\mathbf{v}(t_i) = (\mathbf{P}(z_1|t_i), \dots, \mathbf{P}(z_K|t_i))$$

$\mathbf{P}(z_1|t_i)$  ... Wahrscheinlichkeit von Wortsinn  $z_1$  gegeben Zielwort  $t_i$

$\mathbf{P}(z_2|t_i)$  ... Wahrscheinlichkeit von Wortsinn  $z_2$  gegeben Zielwort  $t_i$  usw.

# Bedeutungsrepräsentation über latente Wortsinne

$$\mathbf{v}(t_i) = (\mathbf{P}(z_1|t_i), \dots, \mathbf{P}(z_K|t_i))$$

## Note:

- ▶ Repräsentation immer durch das Korpus bedingt, das wir verwenden (Parametrisierung)
- ▶ Wortsinne  $z_1 \dots z_K$  sind latent (nicht beobachtbar) und Hilfsmittel zur Dimensionalitätsreduktion

Analog zur Ausgangsgleichung können wir nun auch eine Gleichung aufstellen, die um **Kontextfeature** erweitert ist:

$$\mathbf{v}(t_i, \mathbf{c}_j) = (\mathbf{P}(z_1|t_i, \mathbf{c}_j), \dots, \mathbf{P}(z_K|t_i, \mathbf{c}_j))$$

→ Wahrscheinlichkeitsverteilung nun kontextabhängig und präziser; Kontextfeature ermöglichen Disambiguierung des Zielwortes, sodass weniger Wortsinne höhere Wahrscheinlichkeiten zugewiesen werden

# Bedeutungsrepräsentation über latente Wortsinne

$$\mathbf{v}(\mathbf{t}_i, \mathbf{c}_j) = (\mathbf{P}(z_1 | \mathbf{t}_i, \mathbf{c}_j), \dots, \mathbf{P}(z_K | \mathbf{t}_i, \mathbf{c}_j))$$

Wie rechnen wir das jetzt aus?



# Bedeutungsrepräsentation über latente Wortsinne

$$v(t_i, c_j) = (P(z_1|t_i, c_j), \dots, P(z_K|t_i, c_j))$$

Wie rechnen wir das jetzt aus?

Um die Bedeutungsrepräsentation zu erhalten, die wir oben definiert haben, müssen wir jeweils  $\mathbf{P}(z_k|t_i, c_j)$  berechnen:

$$\mathbf{P}(z_k|t_i, c_j) = \frac{P(t_i, z_k)P(c_j|z_k, t_i)}{\sum_k P(t_i, z_k)P(c_j|z_k, t_i)}$$

# Bedeutungsrepräsentation über latente Wortsinne

$$v(t_i, c_j) = (P(z_1|t_i, c_j), \dots, P(z_K|t_i, c_j))$$

Wie rechnen wir das jetzt aus?

Um die Bedeutungsrepräsentation zu erhalten, die wir oben definiert haben, müssen wir jeweils  $\mathbf{P}(z_k|t_i, c_j)$  berechnen:

$$\mathbf{P}(z_k|t_i, c_j) = \frac{P(t_i, z_k)P(c_j|z_k, t_i)}{\sum_k P(t_i, z_k)P(c_j|z_k, t_i)}$$

$P(c_j|z_k, t_i)$  ... schwer zu berechnen, da  $K \times I$  J-dimensionale Verteilung, die wir lernen müssten

→ **Unabhängigkeitsannahme**

# Bedeutungsrepräsentation über latente Wortsinne

$$v(t_i, c_j) = (P(z_1|t_i, c_j), \dots, P(z_K|t_i, c_j))$$

Wie rechnen wir das jetzt aus?

Um die Bedeutungsrepräsentation zu erhalten, die wir oben definiert haben, müssen wir jeweils  $\mathbf{P}(z_k|t_i, c_j)$  berechnen:

$$\mathbf{P}(z_k|t_i, c_j) = \frac{P(t_i, z_k)P(c_j|z_k, t_i)}{\sum_k P(t_i, z_k)P(c_j|z_k, t_i)}$$

Unabhängigkeitsannahme:

$$\mathbf{P}(z_k|t_i, c_j) \approx \frac{P(z_k|t_i)P(c_j|z_k)}{\sum_k P(z_k|t_i)P(c_j|z_k)}$$

- ▶ Zielwörter  $t_i$  und Kontextwörter  $c_j$  sind bedingt unabhängig gegeben Wortsinne  $z_k$
- ▶ Zielwörter und Kontextwörter eines Wortsinnes werden demnach unabhängig voneinander generiert

# Bedeutungsrepräsentation über latente Wortsinne

$$v(t_i, c_j) = (P(z_1|t_i, c_j), \dots, P(z_K|t_i, c_j))$$

Wie rechnen wir das jetzt aus?

$$P(\mathbf{z}_k|t_i, c_j) = \frac{P(t_i, z_k)P(c_j|z_k, t_i)}{\sum_k P(t_i, z_k)P(c_j|z_k, t_i)}$$

Unabhängigkeitsannahme:

$$P(\mathbf{z}_k|t_i, c_j) \approx \frac{P(z_k|t_i)P(c_j|z_k)}{\sum_k P(z_k|t_i)P(c_j|z_k)}$$

Und wie rechnen wir DAS aus?

Wir schätzen  $P(z_k|t_i)$  und  $P(c_j|z_k)$  mithilfe verschiedener **latenter Variablenmodelle**.

# Bedeutungsrepräsentation über latente Wortsinne

$$\mathbf{P}(z_k | t_i, c_j) \approx \frac{P(z_k | t_i) P(c_j | z_k)}{\sum_k P(z_k | t_i) P(c_j | z_k)}$$

Wir schätzen  $P(z_k | t_i)$  und  $P(c_j | z_k)$  anhand latenter Variablenmodelle.

## Note:

- ▶ Wortsinne entsprechen hier keinen lexikographischen Wortbedeutungen.
- ▶ Wortsinne können eher als generelle Topics verstanden werden, die in dem Korpus vorkommen, auf dem wir unsere Modelle trainieren.
- ▶ Wortsinne sind nicht Wort-spezifisch, sondern global. (über alle Wörter im Korpus verteilt)
- ▶ Wortsinne können sowohl kontextUNabhängig als  $P(z_k | t_i)$  oder kontextABhängig als  $\mathbf{P}(z_k | t_i, c_j)$  dargestellt werden.

# Parametrisierte Algorithmen

$$P(\mathbf{z}_k | \mathbf{t}_i, \mathbf{c}_j) \approx \frac{P(\mathbf{z}_k | \mathbf{t}_i) P(\mathbf{c}_j | \mathbf{z}_k)}{\sum_k P(\mathbf{z}_k | \mathbf{t}_i) P(\mathbf{c}_j | \mathbf{z}_k)}$$

Wir schätzen  $P(\mathbf{z}_k | \mathbf{t}_i)$  und  $P(\mathbf{c}_j | \mathbf{z}_k)$  anhand **latenter Variablenmodelle**.

Parameter der Algorithmen:

1. Input Kookurrenz-Matrix (Dimension)
  - ▶ Spielraum bzgl. der Definition der Spalten der Kookurrenzmatrix
  - ▶ Anzahl von Kontextwörtern (Kontextfenster), ganze Paragraphen oder Dokumente, syntaktische Abhängigkeiten
2. Algorithmus zur Induktion latenter Sinne (Kontextualisierung)
  - ▶ Spielraum bzgl. verschiedener probabilistischer Modelle
  - ▶ NMF (Nicht-negative Matrixfaktorisierung), LDA (Latent Dirichlet allocation)

# Nicht-negative Matrixfaktorisierung

Es handelt sich im Allgemeinen um Algorithmen, die eine **Input Matrix  $\mathbf{V}$**  unter einer bestimmten Kostenfunktion in zwei Faktoren  $\mathbf{W}$  und  $\mathbf{H}$  approximieren.

$\mathbf{W}$  und  $\mathbf{H}$  sind hier dimensionalitätsreduzierte Matrizen, deren Produkt sich an die Input Matrix  $\mathbf{V}$  annähert:

$$\mathbf{V}_{I,J} \approx \mathbf{W}_{I,K} \mathbf{H}_{K,J}$$

# Nicht-negative Matrixfaktorisierung

**W** und **H** sind hier dimensionalitätsreduzierte Matrizen, deren Produkt sich an die Input Matrix **V** annähert:

$$\mathbf{V}_{I,J} \approx \mathbf{W}_{I,K} \mathbf{H}_{K,J}$$

Recap: Wir wollen  $P(z_k|t_i)$  und  $P(c_j|z_k)$  schätzen.

$$P(z_k|t_i, c_j) \approx \frac{P(z_k|t_i)P(c_j|z_k)}{\sum_k P(z_k|t_i)P(c_j|z_k)}$$

**V**<sub>I,J</sub> ... **P**(t<sub>i</sub>|c<sub>j</sub>)

**W**<sub>I,K</sub> ... **P**(t<sub>i</sub>,z<sub>k</sub>)

**H**<sub>K,J</sub> ... **P**(c<sub>j</sub>|z<sub>k</sub>)



# Nicht-negative Matrixfaktorisierung

**W** und **H** sind hier dimensionalitätsreduzierte Matrizen, deren Produkt sich an die Input Matrix **V** annähert:

$$\mathbf{V}_{I,J} \approx \mathbf{W}_{I,K} \mathbf{H}_{K,J}$$

Recap: Wir wollen  $P(z_k|t_i)$  und  $P(c_j|z_k)$  schätzen.

$$P(z_k|t_i, c_j) \approx \frac{P(z_k|t_i)P(c_j|z_k)}{\sum_k P(z_k|t_i)P(c_j|z_k)}$$

**V**<sub>I,J</sub> ... **P**(t<sub>i</sub>|c<sub>j</sub>)

**W**<sub>I,K</sub> ... **P**(t<sub>i</sub>,z<sub>k</sub>) Noch nicht die benötigte Wahrscheinlichkeit!

**H**<sub>K,J</sub> ... **P**(c<sub>j</sub>|z<sub>k</sub>)

→ Faktorisierung:

$$\mathbf{P}(t_i|c_j) = \sum_k P(t_i)P(z_k|t_i)P(c_j|z_k)$$

## NMF – Beispiel

$$P(\mathbf{t}_i | \mathbf{c}_j) \approx P(\mathbf{t}_i, \mathbf{z}_k) \times P(\mathbf{c}_j | \mathbf{z}_k)$$

$$\mathbf{V}_{I,J} \approx \mathbf{W}_{I,K} \times \mathbf{H}_{K,J}$$

	grau	Treiber	OP	Aktie	Holz
...	...	...	...	...	...
Maus	0.4	0.44	0.26	0.2	0.28
Bank	0.3	0.27	0.49	0.53	0.62
...	...	...	...	...	...

$$\mathbf{V}_{I,J}$$

## NMF – Beispiel

$$P(t_i|c_j) \approx P(t_i, z_k) \times P(c_j|z_k)$$

$$V_{I,J} \approx W_{I,K} \times H_{K,J}$$

	Nagetier	Eingabegerät	Hand	Geldinstitut	Sitzplatz		grau	Treiber	OP	Aktie	Holz
...	...	...	...	...	...	...	...	...	...	...	...
Maus	0.3	0.5	0.1	0	0	Nagetier	0.6	0.1	0.3	0.1	0.5
Bank	0	0.2	0	0.5	0.4	Eingabegerät	0.4	0.8	0.2	0.3	0.2
...	...	...	...	...	...	Hand	0.2	0.05	0.7	0.2	0.3
						Geldinstitut	0.2	0.2	0.5	0.9	0.6
						Sitzplatz	0.3	0.02	0.5	0.05	0.7
						...	...	...	...	...	...

W<sub>I,K</sub>
H<sub>K,J</sub>

## NMF – Beispiel

$$P(\mathbf{t}_i | \mathbf{c}_j) \approx P(\mathbf{t}_i, \mathbf{z}_k) \times P(\mathbf{c}_j | \mathbf{z}_k)$$

$$\mathbf{V}_{I,J} \approx \mathbf{W}_{I,K} \times \mathbf{H}_{K,J}$$

$$\mathbf{V}_{Bank, Aktie} \approx \mathbf{W}_{Bank, Geldinstitut} \times \mathbf{H}_{Geldinstitut, Aktie}$$

	grau	Treiber	OP	<b>Aktie</b>	Holz
...	...	...	...	...	...
Maus	0.4	0.44	0.26	0.2	0.28
<b>Bank</b>	0.3	0.27	0.49	<b>0.53</b>	0.62
...	...	...	...	...	...

$$\mathbf{V}_{I,J}$$

## NMF – Beispiel

$$P(\mathbf{t}_i | \mathbf{c}_j) \approx P(\mathbf{t}_i, z_k) \times P(\mathbf{c}_j | z_k)$$

$$\mathbf{V}_{Bank, Aktie} \approx \mathbf{W}_{Bank, Geldinstitut} \times \mathbf{H}_{Geldinstitut, Aktie}$$

$$0.53 \approx 0.2 \times 0.3 + 0.5 \times 0.9 + 0.4 \times 0.05$$

	Nagetier	Eingabegerät	Hand	Geldinstitut	Sitzplatz
...	...	...	...	...	...
Maus	0.3	0.5	0.1	0	0
Bank	0	0.2	0	0.5	0.4
...	...	...	...	...	...

$$\mathbf{W}_{I,K}$$

	grau	Treiber	OP	Aktie	Holz
...	...	...	...	...	...
Nagetier	0.6	0.1	0.3	0.1	0.5
Eingabegerät	0.4	0.8	0.2	0.3	0.2
Hand	0.2	0.05	0.7	0.2	0.3
Geldinstitut	0.2	0.2	0.5	0.9	0.6
Sitzplatz	0.3	0.02	0.5	0.05	0.7
...	...	...	...	...	...

$$\mathbf{H}_{K,J}$$

# Latent Dirichlet Allocation

<LDA Exkurs>

# Topic Modelle

- = **Generative Wahrscheinlichkeitsmodelle** für Dokumente wie Text- oder Bildkorpora
  - ▶ Jedes Korpuselement (jedes Dokument) setzt sich aus verschiedenen zugrundeliegenden (**latenten**) **Themen** (Topics) zusammen. (z.B. *wissenschaftliche Papiere*)
  - ▶ Jedes (beobachtbare) Wort im Dokument ist einem oder mehreren dieser Topics entnommen. (z.B. *Physik, Informatik, Linguistik*)
  - ▶ Anhand der Topics (festgelegte Anzahl) lassen sich **Ähnlichkeiten zwischen Dokumenten** feststellen.

Note: Es handelt sich um sogenannte **Bag-of-words Modelle**, d.h.

- ▶ Syntax wird vernachlässigt
- ▶ Topics und Wörter sind austauschbar (z.B. Google News)

# Topic Modelle

- = **Generative Wahrscheinlichkeitsmodelle** für Dokumente wie Text- oder Bildkorpora
  - ▶ Jedes Korpuselement (jedes Dokument) setzt sich aus verschiedenen zugrundeliegenden (**latenten**) **Themen** (Topics) zusammen (*z.B. wissenschaftliche Papiere*)
  - ▶ Jedes (beobachtbare) Wort im Dokument ist einem oder mehreren dieser Topics entnommen (*z.B. Physik, Informatik, Linguistik*)
  - ▶ Anhand der Topics (festgelegte Anzahl) lassen sich **Ähnlichkeiten zwischen Dokumenten** feststellen

Ziel: Latente Themen unüberwacht lernen



# Topic Modelle

Grundidee: Jedes Dokument setzt sich aus latenten Topics zusammen, die durch Wortverteilungen charakterisiert sind.

Ziel: Latente Themen unüberwacht lernen

Wie funktioniert das?

- ▶ LSA (Deerwester et al., 1990)
- ▶ pLSI (Hofmann, 1999)  
(keine Annahmen über Topic-Verteilungen)
- ▶ LDA (Blei et al., 2003)  
(Wahrscheinlichkeitsverteilung über Topic-Verteilungen)

Grundbegriffe:

**Wort** ... Einzelnes Wort aus dem Vokabular  $V$ ; Index  $\{1, \dots, V\}$

**Dokument** ... Sequenz von  $N$  Wörtern;  $\mathbf{w} = (w_1, w_2, \dots, w_N)$

**Korpus** ... Sammlung von  $M$  Dokumenten;  $D = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_M\}$

# Formalisierung LDA

## Grundbegriffe:

**Wort** ... Einzelnes Wort aus dem Vokabular  $V$ ; Index  $\{1, \dots, V\}$

**Dokument** ... Sequenz von  $N$  Wörtern;  $\mathbf{w} = (w_1, w_2, \dots, w_N)$

**Korpus** ... Sammlung von  $M$  Dokumenten;  $D = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_M\}$

Es wird angenommen, dass es für jedes Dokument  $\mathbf{w}$  im Korpus  $D$  ein **generativer Prozess** stattgefunden hat:

1. Wähle  $N \sim \text{Poisson}(\xi)$
2. Wähle  $\theta \sim \text{Dir}(\alpha)$
3. Für jedes der  $N$  Wörter  $w_n$ :
  - a. Wähle ein Topic  $z_n \sim \text{Multinomial}(\theta)$
  - b. Wähle ein Wort  $w_n$  aus  $p(w_n | z_n, \beta)$   
(Multinomialverteilung gegeben Topic  $z_n$ )

# Formalisierung

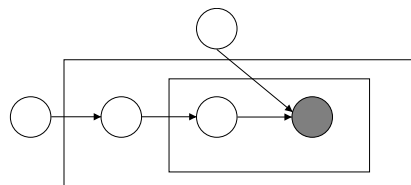
Es wird angenommen, dass es für jedes Dokument  $\mathbf{w}$  im Korpus  $D$  ein **generativer Prozess** stattgefunden hat:

1. Wähle  $N \sim \text{Poisson}(\xi)$  (Anzahl der Wörter im Dokument)
2. Wähle  $\theta \sim \text{Dir}(\alpha)$  ( $k$ -dimensionale Dirichlet Zufallsvariable)
3. Für jedes der  $N$  Wörter  $w_n$ :
  - a. Wähle ein Topic  $z_n \sim \text{Multinomial}(\theta)$
  - b. Wähle ein Wort  $w_n$  aus  $p(w_n|z_n, \beta)$   
(Multinomialverteilung gegeben Topic  $z_n$ )

## Note:

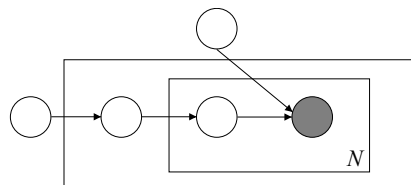
- ▶ Dimensionalität  $k$  der Dirichlet Verteilung wird festgelegt
- ▶ Wortwahrscheinlichkeiten werden durch  $\beta$  parametrisiert, eine  $k \times V$  Matrix mit  $\beta_{ij} = p(w^j=1|z^i=1)$

# Graphische Modelle



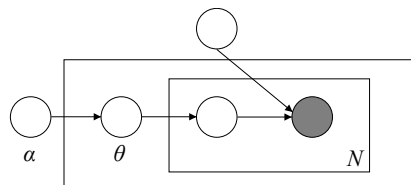
- ▶ Knoten sind Zufallsvariablen
- ▶ Kanten beschreiben mögliche Abhängigkeiten
- ▶ Beobachtete Variablen sind schattiert
- ▶ Tafeln (plates) beschreiben replizierte Struktur

# Graphisches Modell LDA



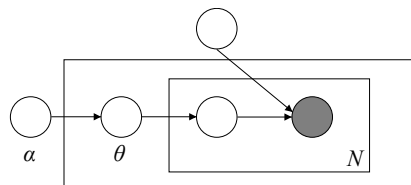
1. Wähle  $N \sim \text{Poisson}(\xi)$

# Graphisches Modell LDA



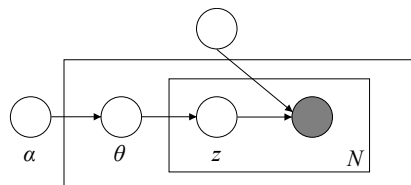
1. Wähle  $N \sim \text{Poisson}(\xi)$
2. Wähle  $\theta \sim \text{Dir}(\alpha)$

# Graphisches Modell LDA



1. Wähle  $N \sim \text{Poisson}(\xi)$
2. Wähle  $\theta \sim \text{Dir}(\alpha)$
3. Für jedes der  $N$  Wörter  $w_n$ :

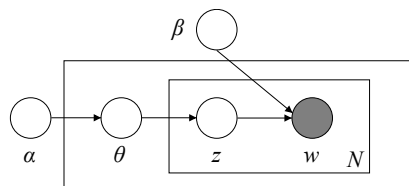
# Graphisches Modell LDA



1. Wähle  $N \sim \text{Poisson}(\xi)$
2. Wähle  $\theta \sim \text{Dir}(\alpha)$
3. Für jedes der  $N$  Wörter  $w_n$ :
  - a. Wähle Topic  $z_n \sim \text{Multinomial}(\theta)$

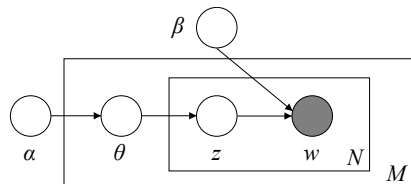


# Graphisches Modell LDA



1. Wähle  $N \sim \text{Poisson}(\xi)$
2. Wähle  $\theta \sim \text{Dir}(\alpha)$
3. Für jedes der  $N$  Wörter  $w_n$ :
  - a. Wähle Topic  $z_n \sim \text{Multinomial}(\theta)$
  - b. Wähle Wort  $w_n$  aus  $p(w_n|z_n, \beta)$

# Graphisches Modell LDA

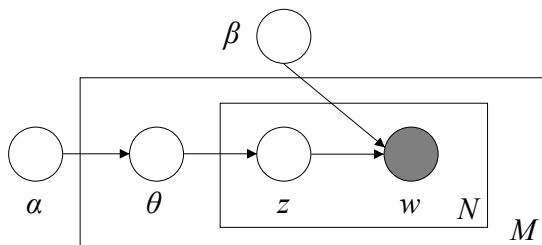


1. Wähle  $N \sim \text{Poisson}(\xi)$
2. Wähle  $\theta \sim \text{Dir}(\alpha)$
3. Für jedes der  $N$  Wörter  $w_n$ :
  - a. Wähle Topic  $z_n \sim \text{Multinomial}(\theta)$
  - b. Wähle Wort  $w_n$  aus  $p(w_n|z_n, \beta)$
 (Repeat)

## Wahrscheinlichkeiten LDA

Gegeben Parameter  $\alpha$  und  $\beta$ , lässt sich Topic-Verteilung  $\theta$ , Menge der Topics  $z$  und die Menge der Wörter  $w$  wie folgt berechnen:

$$p(\theta, z, w | \alpha, \beta) = p(\theta | \alpha) \prod_{n=1}^N p(z_n | \theta) p(w_n | z_n, \beta)$$



# Wahrscheinlichkeiten LDA

$$p(\theta, \mathbf{z}, \mathbf{w} | \alpha, \beta) = p(\theta | \alpha) \prod_{n=1}^N p(z_n | \theta) p(w_n | z_n, \beta)$$

Integrieren über  $\theta$  und summieren über  $\mathbf{z}$ :

$$p(\mathbf{w} | \alpha, \beta) = \int p(\theta | \alpha) \left( \prod_{n=1}^N \sum_{z_n} p(z_n | \theta) p(w_n | z_n, \beta) \right) d\theta$$

Produkt der Randwahrscheinlichkeiten der einzelnen Dokumente ergibt dann die Wahrscheinlichkeit eines Korpus  $D$ :

$$p(D | \alpha, \beta) = \prod_{d=1}^N \int p(\theta_d | \alpha) \left( \prod_{n=1}^{N_D} \sum_{z_{dn}} p(z_{dn} | \theta_d) p(w_{dn} | z_{dn}, \beta) \right) d\theta_d$$

$\alpha, \beta$  ... Korpus-Level, sampeln einmal pro Korpus

$\theta_d$  ... Dokument-Level, sampeln einmal pro Dokument

$z_{dn}, w_{dn}$  ... Wort-Level, sampeln für jedes Wort

## LDA – Beispiel: Paper MIT

1. Wähle  $N \sim \text{Poisson}(\xi)$   
**2000**
2. Wähle  $\theta \sim \text{Dir}(\alpha)$   
 $\theta_1 = \mathbf{(0.5, 0.3, 0.2)} \sim \alpha = \mathbf{0.1}$
3. Für jedes der  $N$  Wörter  $w_n$ :
  - a. Wähle ein Topic  $z_n \sim \text{Multinomial}(\theta)$   
 $\mathbf{z_{1,1} = 1}$
  - b. Wähle ein Wort  $w_n$  aus  $p(w_n|z_n, \beta)$   
 $\mathbf{w_{1,1} = \text{quantum}}$

•	$k \times V$	particle	<b>quantum</b>	algorithm	tree	semantics	...
<b>1</b>	<b>(Physik)</b>	0.8	<b>0.9</b>	0.5	0.4	0.1	...
2	(Informatik)	0.2	0.1	0.9	0.7	0.3	...
3	(Linguistik)	0.1	0.1	0.6	0.9	0.8	...

# LDA – Beispiel: Paper MIT

1. Wähle  $N \sim \text{Poisson}(\xi)$   
**2000**
2. Wähle  $\theta \sim \text{Dir}(\alpha)$   
 $\theta_1 = \mathbf{(0.5, 0.3, 0.2)} \sim \alpha = \mathbf{0.1}$
3. Für jedes der  $N$  Wörter  $w_n$ :
  - a. Wähle ein Topic  $z_n \sim \text{Multinomial}(\theta)$   
 $\mathbf{z_{1,2} = 3}$
  - b. Wähle ein Wort  $w_n$  aus  $p(w_n|z_n, \beta)$   
 $\mathbf{w_{1,2} = tree}$

•	$k \times V$	particle	quantum	algorithm	<b>tree</b>	semantics	...
1	(Physik)	0.8	0.9	0.5	0.4	0.1	...
2	(Informatik)	0.2	0.1	0.8	0.7	0.3	...
3	<b>(Linguistik)</b>	0.1	0.1	0.6	<b>0.9</b>	0.8	...

# LDA – Beispiel: Paper MIT

1. Wähle  $N \sim \text{Poisson}(\xi)$   
**2000**
2. Wähle  $\theta \sim \text{Dir}(\alpha)$   
 $\theta_1 = \mathbf{(0.5, 0.3, 0.2)} \sim \alpha = \mathbf{0.1}$
3. Für jedes der  $N$  Wörter  $w_n$ :
  - a. Wähle ein Topic  $z_n \sim \text{Multinomial}(\theta)$   
 $\mathbf{z_{1,3} = 2}$
  - b. Wähle ein Wort  $w_n$  aus  $p(w_n|z_n, \beta)$   
 $\mathbf{w_{1,3} = \text{algorithm}}$

•	$k \times V$	particle	quantum	<b>algorithm</b>	tree	semantics	...
1	(Physik)	0.8	0.9	0.5	0.4	0.1	...
<b>2</b>	<b>(Informatik)</b>	0.2	0.1	<b>0.8</b>	0.7	0.3	...
3	(Linguistik)	0.1	0.1	0.6	0.9	0.8	...

## LDA – Beispiel: Paper MIT

Ergebnis nach 3 von 2000 Durchläufen für das 1. Dokument (MIT Paper) aus unserem Korpus (z.B. Conference Proceedings):

“quantum tree algorithm ...”

Recap: Eigentlich wollen wir den **generativen Prozess umkehren**, also die **latenten Wahrscheinlichkeitsverteilungen** ( $\theta$  und  $\beta$ ) ausgehend von unseren Beobachtungen ( $\mathbf{w}$ ) **inferieren**.

Wie geht das nun wieder?

Mithilfe verschiedener Inferenzalgorithmen, z.B. **Gibbs sampling**



# Gibbs sampling – Intuition

Was wir haben: ein Korpus und eine bestimmte Anzahl an Topics  
Was wir wollen: latente Topics und deren Wortverteilungen

Also?

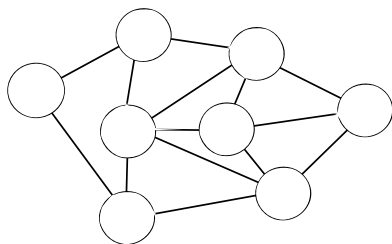
- ▶ Echte Posteriorverteilung lässt sich nicht berechnen, **Approximative Posterior-Inferenz allerdings schon.**
- ▶ Gibbs-Sampling produziert Samples aus dieser echten Verteilung.

Grundidee:

**Posterior für einzelne Zufallsvariablen** ist leicht zu berechnen.

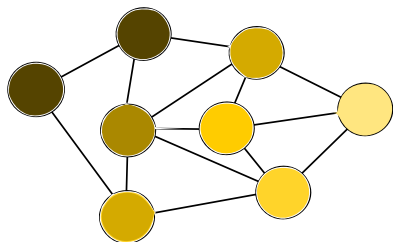
→ Man kann nacheinander jede einzelne Zufallsvariable neu aus ihrem Posterior (gegeben alle anderen Variablen) ziehen.

## Gibbs sampling – Intuition



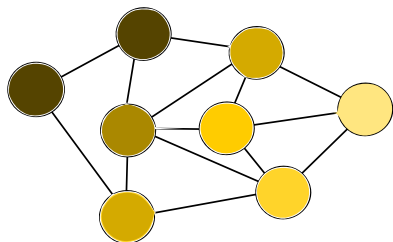
- ▶ Neu Ziehen von einzelnen Variablen entspricht Zustandsübergang.
- ▶ Zustand = komplette Belegung von allen Variablen
- ▶ Iteratives neu Ziehen ist "Random Walk" auf dem Zustandsgraphen.

# Gibbs sampling – Intuition



- ▶ Häufigkeiten des Besuchs von Zuständen entspricht Verteilung über Belegungen der Zufallsvariablen.
- ▶ Theorem: Wenn man sich lange genug auf dem Zustandsgraphen bewegt und Zustandsübergänge aus den Einzel-Posteriors zieht, konvergieren die Besuchshäufigkeiten zur Gesamt-Posterior-Verteilung, egal welchen Ausgangspunkt wir wählen.

# Gibbs sampling – Intuition



Also?

- ▶ Um Samples aus dem Gesamtposterior zu erhalten, mit beliebigem Startwert beginnen, und sampeln, bis die Verteilung konvergiert.
- ▶ Zwischen dem Entnehmen einzelner Samples viele Sampleschritte durchführen, damit die Samples voneinander unabhängig sind.

# Gibbs sampling – Intuition

Etwas genauer:

1. Jedem Wort eines jeden Dokuments wird zufällig ein Topic zugewiesen.  
→ Wir erhalten sowohl Topic- als auch Wortverteilungen.
2. Für jedes Dokument, für jedes Wort:  
Für jedes Topic, berechne:
  - ▶  $p(\text{Topic}|\text{Dokument})$
  - ▶  $p(\text{Wort}|\text{Topic})$
  - ▶ Weise jedem Wort ein neues Topic zu, mit  $p(\text{Topic}|\text{Dokument}) \times p(\text{Wort}|\text{Topic})$   
→ Wir nehmen also für jedes Wort an, dass alle anderen Topics korrekt sind.  
→ Topic-Verteilung nähert sich der Wirklichkeit an.

# Latent Dirichlet Allocation

</LDA Exkurs>

“If you’re interested in the details of inference and priors, Kevin Knight’s ‘Bayesian Inference with Tears’ is excellent.”

## Zurück zum Modell

$$P(\mathbf{z}_k | \mathbf{t}_i, \mathbf{c}_j) \approx \frac{P(\mathbf{z}_k | \mathbf{t}_i) P(\mathbf{c}_j | \mathbf{z}_k)}{\sum_k P(\mathbf{z}_k | \mathbf{t}_i) P(\mathbf{c}_j | \mathbf{z}_k)}$$

Wir schätzen  $P(\mathbf{z}_k | \mathbf{t}_i)$  und  $P(\mathbf{c}_j | \mathbf{z}_k)$  anhand **latenter Variablenmodelle**.

- ▶ Dinu und Lapata verwenden LDA jetzt, um (statt Topics) die **Wortsinne der Zielwörter** zu erhalten
- ▶ Kookurrenzmatrix: jede Reihe  $\mathbf{t}_i$  wird als "Dokument" behandelt, Spalten sind weiterhin Kontextwörter  $\mathbf{c}_j$ , in jeder Zelle  $\mathbf{t}_i \mathbf{c}_j$  steht die Häufigkeit von  $\mathbf{c}_j$  im "Dokument" von  $\mathbf{t}_i$

Mithilfe von Gibbs sampling erhalten wir nun wiederum die benötigten Wahrscheinlichkeiten:

$\theta$  als Verteilung der Wortsinne für jedes Zielwort;  $\theta_{ik} = P(\mathbf{z}_k | \mathbf{t}_i)$   
 $\beta$  als Kontextwort-Verteilung für jeden Wortsinn;  $\beta_{kj} = P(\mathbf{c}_j | \mathbf{z}_k)$

# Outline

## Motivation

## Modell

Bedeutungsrepräsentation im Kontext

Parametrisierte Algorithmen

Nicht-negative Matrixfaktorisierung

Latent Dirichlet Allocation

## Experimentelles Design

Aufgabenstellungen

Training des Modells

Baselines

## Ergebnisse

Wortähnlichkeit

Lexikalische Ersetzung

## Zusammenfassung



# Aufgabenstellungen

Unser probabilistisches Modell zur Repräsentation von Wörtern über die induzierten Wortsinne können wir jetzt mit optimierten Parametern für jeweils NMF und LDA verwenden, um

1. Wortähnlichkeit zu berechnen (unabhängig vom Kontext) (Finkelstein et al., 2002)
2. Automatisch lexikalische Substitute von Zielwörtern zu ermitteln (McCarthy and Navigli, 2007)

# Finkelstein et al., 2002

**Task:** Berechnen der Wortähnlichkeit

$$\text{sim}(t_i, t_i') = \text{sim}((v(t_i)), v(t_i'))$$

**Data set:** 353 Wortpaare und entsprechende Ähnlichkeitswerte

Word 1	Word 2	Human (mean)
tiger	cat	7.35
tiger	tiger	10.00
plane	car	5.77
smart	student	4.62
energy	secretary	1.81

## McCarthy and Navigli, 2007

**Task:** Finden von geeigneten lexikalischen Substituten

→ Systeme sollen eine vorgegebene Menge an lexikalischen Substituten entsprechend der semantischen Ähnlichkeit zum Zielwort ranken:  $\text{sim}((v(t_i, c_j)), v(t_i'))$

(Nur das Zielwort zu kontextualisieren ist diskriminativer als Zielwort und Substitut in ähnlichem Kontext auszuwerten.)

**Data set:** SemEval 2007 Lexical Substitution Task

(200 Zielwörter in 10 verschiedenen Satzkontexten, Annotationen)

Sentences	Substitutes
It is important to apply the herbicide on a <b>still</b> day, because spray drift can kill non-target plants.	calm(5) not-windy(1) windless(1)
A movie is a visual document comprised of a series of <b>still</b> images.	motionless(3) unmoving(2) fixed(1) stationary(1) static(1)

# Training des Modells

## Gleiche Input Daten:

- ▶ Bag-of-words Matrix (GigaWord)
- ▶ Reihen sind Zielwörter, Spalten sind kookkurrierende Kontextwörter
- ▶ Kontextwörter: Kontextfenster von 5 Wörtern, Vokabular besteht aus 3000 häufigsten Wörtern im Korpus

## NMF:

- ▶ Algorithmus aus Lee and Seung (2000)
- ▶ Normalisierte Input Matrix
- ▶ Verschiedene Dimensionen  $K$ : [600–1000] in 200er Schritten
- ▶ 150 Iterationen
- ▶ Mittelwert aus je 2 unabhängigen Durchläufen ( $K$  einziger Parameter)

# Training des Modells

## Gleiche Input Daten:

- ▶ Bag-of-words Matrix (GigaWord)
- ▶ Reihen sind Zielwörter, Spalten sind kookkurrierende Kontextwörter
- ▶ Kontextwörter: Kontextfenster von 5 Wörtern, Vokabular besteht aus 3000 häufigsten Wörtern im Korpus

## LDA:

- ▶ Parameter:  $K, \alpha, \beta$
- ▶ Verschiedene Dimensionen  $K$ : [600–1400] in 200er Schritten
- ▶  $\alpha$ :  $\frac{2}{K}$  (Porteous et al., 2008),  $\frac{50}{K}$  (Griffiths and Steyvers, 2004)
- ▶ Gibbs sampling
- ▶ 1000 Iterationen
- ▶ Mittelwert aus 5 Iterationen [600–1000]

# Semantische Ähnlichkeit

Es wurden verschiedene Ähnlichkeitsmaße ausprobiert:

- ▶ Skalarprodukt
- ▶ Kosinusähnlichkeit
- ▶ Inverse Jensen-Shannon-Abweichung

→ **Skalarprodukt** am besten geeignet:

- ▶ Nur vollständig disambiguierten Wörtern wird ein Ähnlichkeitswert von 1 zugewiesen.
- ▶ Außerdem wird einbezogen, wie “fokussiert” die betrachteten Verteilungen sind, da hochgradig ambige Wörter wahrscheinlich keinen sehr hohen Ähnlichkeitswert erzielen.

# Baselines

## Kontextunabhängig:

Latent Semantic Analysis (Landauer and Dumais, 1997):

- ▶  $U\Sigma V$  Singulärwertzerlegung mit Rang  $k = 1000$
- ▶ Evaluierung anhand Zerlegung mit  $K$ : [200–1000] in 100er Schritten
- ▶ Vorherige Gewichtung: tf-idf, Zeilennormalisierung

Einfache semantische Räume ohne Dimensionalitätsreduktion:

- ▶ Gewichtungsmethoden: PMI, tf-idf, Zeilennormalisierung
- ▶ Außerdem Ähnlichkeit nach Lin (1998)

## Im Kontext:

- ▶ Vektoraddition, Vektormultiplikation anhand des semantischen Raumes (Mitchell and Lapata, 2008)
- ▶ Dimensionalitätsreduzierte Repräsentationen anhand NMF und LDA

# Outline

## Motivation

## Modell

Bedeutungsrepräsentation im Kontext

Parametrisierte Algorithmen

Nicht-negative Matrixfaktorisierung

Latent Dirichlet Allocation

## Experimentelles Design

Aufgabenstellungen

Training des Modells

Baselines

## Ergebnisse

Wortähnlichkeit

Lexikalische Ersetzung

## Zusammenfassung



# Wortähnlichkeit

Model	Spearman $\rho$
SVS	38.35
LSA	49.43
NMF	<b>52.99</b>
LDA	<b>53.39</b>
LSA <sub>MIX</sub>	49.76
NMF <sub>MIX</sub>	51.62
LDA <sub>MIX</sub>	51.97

Beste Parameter:

- ▶ SVS: tf-idf, Kosinusähnlichkeit
- ▶ LSA: Zeilennormalisierung,  $K = 600$ , Skalarprodukt
- ▶ NMF, LDA: je mehr Wortsinne, desto besser
- ▶ NMF:  $K = 1000$ , Skalarprodukt
- ▶ LDA:  $K = 1200$ , Skalarprodukt,  $\alpha = \frac{50}{K}$

(Mixture Modelle: Mittelwert aus den Ergebnissen der Algorithmen mit verschiedenen Parametern)

# Lexikalische Ersetzung

Model	Kendall's $\tau$
SVS	11.05
Add-SVS	12.74
Add-NMF	12.85
Add-LDA	12.33
Mult-SVS	14.41
Mult-NMF	13.20
Mult-LDA	12.90
Cont-NMF	14.95
Cont-LDA	13.71
Cont-NMF <sub>MIX</sub>	<b>16.01</b>
Cont-LDA <sub>MIX</sub>	<b>15.53</b>

- ▶ SVS als Baseline (kein Kontext)
- ▶ NMF, LDA:  
Einzelergebnisse/Mixture
- ▶ Baselines mit  $v(t_i)$  und Vektoroperationen  
Add: PMI, Lin  
Mult: tf-idf, Kosinus
- ▶ Kontextualisiert ( $v(t_i|c_j)$ )

# Qualitative Analyse

Wie beeinflussen Kontextwörter denn nun die Verteilung der Wortsinne des Zielwortes?

Lexikalische Ersetzung und Cont-LDA OHNE Kontext:

- (1) With their transcendent, improvisational **jams** and Mayan-inspired sense of a higher, metaphysical purpose, the band's music delivers a spiritual sustenance that has earned them a very devoted core following.

Senses	Word Distributions
TRAFFIC (0.18)	road, traffic, highway, route, bridge
<b>MUSIC (0.04)</b>	music, song, rock, band, dance, play
FAN (0.04)	crowd, fan, people, wave, cheer, street
VEHICLE (0.04)	car, truck, bus, train, driver, vehicle

# Qualitative Analyse

Lexikalische Ersetzung und Cont-LDA IM Kontext (von *band*):

- (2) With their transcendent, improvisational **jams** and Mayan-inspired sense of a higher, metaphysical purpose, the band's music delivers a spiritual sustenance that has earned them a very devoted core following.

Senses	Word Distributions
TRAFFIC (0.18)	road, traffic, highway, route, bridge
<b>MUSIC (0.88)</b>	music, song, rock, <b>band</b> , dance, play
FAN (0.04)	crowd, fan, people, wave, cheer, street
VEHICLE (0.04)	car, truck, bus, train, driver, vehicle

→ *riff* und *gig* als wahrscheinlichste lexikalische Substitute

# Qualitative Analyse

Lexikalische Ersetzung und Cont-LDA im Kontext, Shortcomings:

- ▶ Unvollständige Disambiguierung  
*bug*: SOFTWARE (0.09), DISEASE (0.06),  
Kontext *client*: SECRET-AGENCY (0.34), SOFTWARE (0.29)
- ▶ Domänenspezifische Wortsinne  
*function*: MATH, SERVICE,  
Kontext *distribution*: SERVICE → Zeitungskorpus
- ▶ Lokale Wortsinne  
(*check, shoulder*): INJURY (0.81), BALL-SPORTS (0.10),  
Kontext: CLOTHING
- ▶ Bag-of-words Annahme, daher weniger gute Ergebnisse für Verben  
(*let, know*): {see, know, think. do}(0.57), {go, say, do, can}(0.20)

# Outline

## Motivation

## Modell

- Bedeutungsrepräsentation im Kontext

- Parametrisierte Algorithmen

  - Nicht-negative Matrixfaktorisierung

  - Latent Dirichlet Allocation

## Experimentelles Design

- Aufgabenstellungen

- Training des Modells

- Baselines

## Ergebnisse

- Wortähnlichkeit

- Lexikalische Ersetzung

## Zusammenfassung



# Zusammenfassung

- ▶ Wir haben uns heute ein allgemeines Modell zur Berechnung von semantischer Ähnlichkeit im Kontext angesehen.
- ▶ Grundidee ist, Wortbedeutungen als Wahrscheinlichkeitsverteilung über globale Wortsinne und kontextuelle Wortbedeutung als Veränderung dieser Verteilung zu modellieren.
- ▶ Es wird die gleiche Vektorrepräsentation für einzelne Zielwörter als auch für Wörter im Kontext verwendet, was das Modell konzeptuell einfach macht.
- ▶ Zwei Instanziierungen des Modells sind nicht-negative Matrixfaktorisierung und Latent Dirichlet Allocation.
- ▶ Diese parametrisierten Algorithmen erzielen bessere Ergebnisse für Wortbedeutungen im Kontext als alle Modelle, die wir uns bisher angesehen haben.
- ▶ Da die Mixture Modelle für beide Algorithmen gute Ergebnisse liefern, kann die Feineinstellung der Parameter vernachlässigt werden.

Vielen Dank!  
Fragen, Anregungen?



# Literatur

-  Dinu, Georgiana & Lapata, Mirella (2010). Measuring Distributional Similarity in Context. *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1162–1172, MIT, Massachusetts, USA, 9-11 October 2010. ©2010 Association for Computational Linguistics.
-  Blei, David M. & Ng, Andrew Y. & Jordan, Michael I. & Lafferty, John (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, Volume 3.