Distributionelle Ähnlichkeit im Kontext

Melanie Tosik

Universität Potsdam, Department Linguistik

Seminar: Distributionelle Semantik (Prof. Dr. Alexander Koller)

WiSe 2013/14

Abstract

Ziel dieser Arbeit ist ein klarer Überblick über die probabilistischen Verfahren, die Dinu und Lapata (2010) eingeführt haben, um semantische Wortähnlichkeit im Kontext zu berechnen.

1 Einführung

Es gibt inzwischen viele verschiedene Ansätze, um Wortbedeutungen mittels der Wortverteilungen in Texten zu modellieren. Anhand der semantischen Ähnlichkeit von Wörtern können Suchmaschinen nicht nur einzelne Wortsinne disambiguieren, es ist außerdem bereits möglich, automatisch Synonyme und Paraphrasen zu generieren oder auf Textebene komplexe Inhalte zu erschließen.

Typischerweise werden Bedeutungsähnlichkeiten mithilfe von Vektorraum-Modellen berechnet, die am besten geeignet sind, um Wortähnlichkeiten ungeachtet des Kontexts darzustellen. Einzelne Wörter werden als Punkte in einem hochdimensionalen Vektorraum repräsentiert, der sich aus Kookkurrenzen (dem gemeinsamen Auftreten zweier Wörter) im Text ergibt. Semantische Ähnlichkeit kann dann unter Verwendung verschiedener Ähnlichkeitsmaße berechnet werden, die sich an der Distanz der entsprechenden Vektoren zueinander orientieren. Solche Modelle können unüberwacht gelernt werden und sind mit wenig Rechenaufwand verbunden.

Ähnlichkeit ist allerdings nicht nur Synonymie: Antonyme sind sich distributionell ebenfalls sehr ähnlich, und auch Kompositionalität wird so noch nicht erfasst.

Ohne Kontext ist es allerdings kaum möglich, zwischen verschiedenen Wortsinnen zu unterscheiden. Dinu und Lapata (2010) führen als Beispiel das Adjektiv *heavy* an, das sich als *schwer* ins Deutsche übersetzen lässt. Die Verwendungen desselben Wortes in (1) werden alle mit verschiedenen Denotationen assoziiert:

(1) a. Die Aufgabe ist schwer. (*schwierig zu lösen*)
b. Der Lastwagen ist schwer. (*schwer an Gewicht*)
c. Die Enttäuschung ist schwer. (*schwerwiegend, ernstzunehmend*)
d. Das Essen ist schwer. (*schwer verdaulich*)

Vektor-basierte Modelle können diese Wortsinne nicht erfassen. Es gibt nur einen einzigen Vektor für *schwer*, der die diversen Wortbedeutungen gemeinsam enthält und die Gesamtbedeutung invariabel repräsentiert.

Es gab in der Vergangenheit mehrere Versuche, dieses Problem indirekt zu lösen. Da vorgehende Ansätze aber nur unwesentlich zum Verständnis der hier präsentierten Arbeit beitragen, sei der interessierte Leser an dieser Stelle an die Literaturübersicht im Originalpapier von Dinu und Lapata (2010) verwiesen.

In Kapitel 2 wird stattdessen direkt das probabilistische Rahmenkonzept eingeführt, das verwendet wird, um sowohl einzelne Wortbedeutungen als auch Wortähnlichkeit im Kontext zu repräsentieren. Grundidee ist dabei, Wortbedeutungen als Wahrscheinlichkeitsverteilung über latente (nicht beobachtbare) Wortsinne zu modellieren. Wortähnlichkeiten im Kontext werden dann als Veränderung dieser Ausgangswahrscheinlichkeiten der Wortsinne reflektiert.

Ein Beispiel für eine solche Wahrscheinlichkeitsverteilung für das Wort *Bank* ist in (2) illustriert.

(2) Geldinstitut (0.5)
Sitzgelegenheit (0.5)

Kommt *Bank* im Korpus nun im Kontext von *Kredit* vor, verändern sich diese Werte dementsprechend. Wie in (3) dargestellt, ist es nun sehr viel wahrscheinlicher, dass von einem Geldinstitut die Rede ist.

(3) Geldinstitut | *Kredit* (0.9)
Sitzgelegenheit | *Kredit* (0.1)

Instanziiert wird das Modell durch zwei parametrisierte Algorithmen, die in Kapitel 3

diskutiert werden. Details zum experimentellen Design folgen in Kapitel 4, eine Darstellung der Evaluationsergebnisse in Kapitel 5.

Abschließend finden sich Zusammenfassung und Ausblick in Kapitel 6.

2 Bedeutungsrepräsentation im Kontext

Bekannte Vektorraum-Modelle basieren auf Kookkurrenzmatrizen, deren Zeilen Zielwörter, Spalten Kontextwörter (oder ganze Dokumente) und Zellen Kookkurrenzhäufigkeiten enthalten. Häufigkeiten werden in Vektoren übersetzt und Wortähnlichkeiten an deren Distanzen im Vektorraum festgemacht.

Das hier dargestellte Modell unterscheidet sich davon nicht; als Input dienen weiterhin Zielwörter t_i mit $i : 1 \dots I$ in den Zeilen, Kontextwörter c_j mit $j : 1 \dots J$ in den Spalten und die Kookkurrenzhäufigkeit von t_i und c_j in jeder Zelle (i, j) der aus dem Korpus extrahierten Kookkurrenzmatrix.

Im Gegensatz zu bisherigen Versuchen werden Wortbedeutungen nun allerdings über latente Wortsinne dargestellt (wie im *Bank* Beispiel).

Es wird dabei angenommen, dass es für jedes Zielwort t_i eine globale Menge an Wortsinnen $Z = \{z_k | k : 1 \dots K\}$ gibt.

Wortsinne meint hier keine lexikographischen Wortbedeutungen, sondern generelle Topics (Themenbereiche), die im Korpus vorkommen, auf dem das Modell trainiert wird. Wortsinne sind also nicht wortspezifisch, sondern zu jedem Zeitpunkt global über alle Wörter im Korpus verteilt. K entspricht dabei der vorher festgelegten Anzahl der Wortsinne und z_k dem konkreten Wortsinn.

Um Bedeutungsrepräsentationen im Kontext herzuleiten, muss zunächst geklärt werden, wie ein einzelnes Zielwort t_i in Isolation repräsentiert wird.

Die Bedeutung von t_i wird als Wahrscheinlichkeitsverteilung über die Menge der Wortsinne betrachtet.

> "The intuition behind such a representation is that a target word can be described by a set of core meanings and by the frequency with which these are attested." – Dinu und Lapata (2010)

Der Vektor v repräsentiert dieses Zielwort t_i dann wie folgt:

$$v(t_i) = (P(z_1|t_i), \dots, P(z_K|t_i)) \tag{1}$$

$P(z_1|t_i)$ ist die Wahrscheinlichkeit von Wortsinn z_1 gegeben Zielwort t_i, $P(z_2|t_i)$ die Wahrscheinlichkeit von Wortsinn z_2 gegeben Zielwort t_i usw.

Diese Repräsentation ist immer strikt durch das Korpus bedingt, das verwendet wird. Die Wortsinne z_1 bis z_k sind nicht beobachtbar und können als Hilfsmittel zur Dimensionalitätsreduktion verstanden werden.

Die Ausgangsgleichung wird in (2) um ein Kontextfeature c_j erweitert, das die Wahrscheinlichskeitsverteilung entscheidend präzisiert.

$$v(t_i, c_j) = (P(z_1|t_i, c_j), \ldots, P(z_k|t_i, c_j)) \tag{2}$$

Zielwörter können nun disambiguiert werden, da durch die Abhängigkeit vom Kontext weniger Wortsinnen höhere Wahrscheinlichkeiten zugewiesen werden.

Um die kontextsensitive Bedeutungsrepräsentation in (2) zu berechnen, muss jeweils die Wahrscheinlichkeit $P(z_k|t_i, c_j)$ geschätzt werden.

Durch Faktorisierung ergibt sich die Gleichung in (3).

$$P(z_k|t_i, c_j) = \frac{P(t_i, z_k)P(c_j|z_k, t_i)}{\sum_k P(t_i, z_k)P(c_j|z_k, t_i)} \tag{3}$$

Der zweite Faktor $P(c_j|z_k, t_i)$ ist schwer zu berechnen, da eine K $\times$ I J–dimensionale Verteilung gelernt werden müsste. Demzufolge wird eine Unabhängigkeitsannahme gemacht, die in der Gleichung in (4) resultiert.

$$P(z_k|t_i, c_j) = \frac{P(z_k|t_i)P(c_j|z_k)}{\sum_k P(z_k|t_i)P(c_j|z_k)} \tag{4}$$

Es wird nun angenommen, dass Zielwörter und Kontextwörter gegeben eines bestimmten Wortsinnes unabhängig voneinander generiert werden.

Mithilfe dieses Rahmenkonzepts ist es möglich, Wortbedeutung sowohl ohne als auch im Kontext zu erhalten, indem entweder nur $P(z_k|t_i)$ oder $P(z_k|t_i, c_j)$ geschätzt wird. Diese Schätzung ist an keine konkrete Methode zur Induktion von Wortsinnen gebunden. Um Wortbedeutung letztendlich **im Kontext** zu berechnen, werden verschiedene latente Variablenmodelle verwendet, mittels derer die Gleichung in (4) effizient gelöst werden kann.

3 Parametrisierte Algorithmen

Es gibt zwei Parameter, die bei der Initialisierung der Algorithmen variiert werden können und intuitiv Flexibilität in Dimension und Kontextualisierung der Modelle entsprechen.

Die Input-Kookkurrenz-Matrix ermöglicht Spielraum bezüglich der Definition ihrer Spalten. Die Anzahl der Kontextwörter ist abhängig von dem Kontextfenster, das gewählt wird (möglich sind beispielweise ganze Paragraphen oder Dokumente, oder syntaktische Dependenzen).

Der Algorithmus zur Induktion der latenten Sinne ist ebenfalls nicht von vornherein festgelegt. Während aus einer Vielzahl möglicher Verfahren gewählt werden kann, werden nachfolgend nur die beiden probabilistischen Modelle eingeführt, die von Dinu und Lapata (2010) angewandt werden.

3.1 Nicht-negative Matrixfaktorisierung

Bei nicht-negativer Matrixfaktorisierung (NMF) handelt es sich im Allgemeinen um Algorithmen, die eine Input-Matrix V unter einer bestimmten Kostenfunktion in zwei Faktoren W und H approximieren.

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

$$V_{I,J} \approx W_{I,K} H_{K,J} \tag{5}$$

In der gegebenen statistischen Interpretation nicht-negativer Matrixfaktorisierung wird versucht, die Kullback-Leibler-Divergenz zwischen WH und V zu minimieren. Der Vollständigkeit halber findet sich die Formel zur Berechnung in (6).

$$\min \sum_{i,j} (V_{i,j} \log \frac{V_{i,j}}{WH_{i,j}} - V_{i,j} + WH_{i,j}) \tag{6}$$

Die Faktoren in (5) können nun als Faktoren der zu lösenden Gleichung in (4) interpretiert werden. I, J und K denotieren Zielwörter, Kontextfeature und Wortsinne.

$V_{I,J}$ ist die Input-Kookkurrenzmatrix, die sich aus dem Korpus ergibt, und fließt nicht direkt in die Berechnung ein.

$W_{I,K}$ entspricht nach Faktorisierung[1] dem ersten Faktor der zu lösenden Gleichung in (4) $P(z_k|t_i)$. $H_{K,J}$ gleicht bereits dem zweiten Faktor der Gleichung $P(c_j|z_k)$ und muss nicht weiter manipuliert werden.

[1]Die Herleitung der Faktorisierung kann anhand des Originalpapiers von Mitchell und Lapata (2010) eindeutig nachvollzogen werden.

Als Beispiel soll ein kurzer (hypothetischer) Text dienen, aus dem die folgende Kookkurrenzmatrix extrahiert wurde.

Tabelle 1: $V_{I,J}$

	grau	Treiber	OP	Aktie	Holz
...	...	...	...	...	...
Maus	0.23	0.32	0.13	0.06	0.07
Bank	0.11	0.2	0.07	**0.26**	0.18
...	...	...	...	...	...

Tabelle 1 kann entnommen werden, wie groß die Wahrscheinlichkeit ist, dass ein Zielwort *Bank* im Text gemeinsam mit einem Kontextfeature *Aktie* aufgetreten ist:

$$P(Bank|Aktie) = P(t_i|c_j) \approx V_{I,J} \approx V_{Bank,Aktie} \approx \mathbf{0.26} \tag{7}$$

$V_{I,J}$ wird jetzt in die zwei Matrizen $W_{I,K}$ und $H_{K,J}$ zerlegt, die als $P(t_i, z_k)$ und $P(c_j|z_k)$ interpretiert werden.

Tabelle 2: $W_{I,K}$

	Nagetier	Eingabegerät	Hand	Geldinstitut	Sitzplatz
...	...	...	...	...	...
Maus	0.3	0.5	0.1	0	0
Bank	0	0.2	0	0.4	0.35
...	...	...	...	...	...

Tabelle 3: $H_{K,J}$

	grau	Treiber	OP	**Aktie**	Holz
...	...	...	...	...	...
Nagetier	0.4	0.05	0.1	0	0.2
Eingabegerät	0.2	0.6	0.05	0.1	0
Hand	0.1	0	0.7	0.05	0.1
Geldinstitut	0	0.2	0.1	0.6	0.1
Sitzplatz	0.2	0	0.05	0	0.4
...	...	...	...	...	...

Im Beispiel würden wir also Rechnung (8) aufstellen und für das Zielwort *Bank* den latenten Wortsinn GELDINSTITUT[2] erhalten.

$$V_{Bank,Aktie} \approx W_{Bank,Geldinstitut} \times H_{Geldinstitut,Aktie}$$
$$\mathbf{0.26} \approx 0.2 \times 0.1 + 0.4 \times 0.6 \tag{8}$$

3.2 Latent Dirichlet Allocation

Zur Berechnung von Wortähnlichkeiten im Kontext verwenden Dinu und Lapata (2010) eine Variation von Latent Dirichlet Allocation (LDA). Um diese Herangehensweise gänzlich nachvollziehen zu können, wird der Algorithmus zunächst in seiner Standardfassung ausgeführt. Anschließend werden die Modifikationen dargestellt, die in der Versuchsanordnung des Originalpapiers vorgenommen wurden.

3.2.1 Latent Dirichlet Allocation – Exkurs

Bei Latent Dirichlet Allocation handelt es sich um ein statistisches Topic Model, ein generatives Wahrscheinlichkeitsmodell für Dokumente wie Text– oder Bildkorpora.

Jedes Korpuselement (jedes Dokument) setzt sich aus verschiedenen zugrundeliegenden, latenten Themen (Topics) zusammen. Jedes Wort im Dokument ist einem oder mehreren dieser Topics entnommen. Anhand der Topics lassen sich dann Ähnlichkeiten zwischen Dokumenten feststellen.

Wählt man als Dokument zum Beispiel ein wissenschaftliches Papier vom Massachusetts Institute of Technology (MIT), gehören die einzelnen Wörter wahrscheinlich zu Topics wie Physik, Informatik oder Linguistik.

Topic Models sind grundsätzlich Bag-of-words-Modelle, die keinerlei Syntax erfassen können. Topics sind inhaltlich stets austauschbar. Google News[3] zeigt dem Leser beispielsweise stets gleich viele Themenbereiche an (Wirtschaft, Unterhaltung, Sport usw.), aktuelle Beiträge werden allerdings laufend durch neuere Beiträge ersetzt.

Mithilfe von Latent Dirichlet Allocation kann eine vorher festgelegte Anzahl an latenten Themen unüberwacht gelernt werden.

Die Grundidee ist, dass sich jedes Dokument aus latenten Topics zusammensetzt, die durch bestimmmte Wortverteilungen charakterisiert sind. Das Ziel ist, diese latente Topics unüberwacht zu lernen.

[2] Die Topics wurden von Hand mit Namen versehen, der Algorithmus ist dazu nicht in der Lage.
[3] `https://news.google.de/`

Es gab verschiedene Anläufe, dieses Ziel zu erreichen. Deerwester et al. (1990) entwickelten Latent Semantic Analysis (LSA)[4], ein Verfahren, das latente Konzepte nach Dimensionsreduktion[5] einer Wort-Dokument-Matrix explizit macht. Hofmann (1999) erweiterte diese Verfahrensweise um eine Wahrscheinlichkeitsverteilung über Dokumente, resultierend in pLSA (Probabilistic Latent Semantic Analysis). Blei et al. (2003) führten dann Latent Dirichlet Allocation ein, ein Modell, das außerdem noch eine Wahrscheinlichkeitsverteilung über Topic-Verteilungen annimmt.

Zur Formalisierung von LDA werden folgende Grundbegriffe definiert:

Ein Wort w_i denotiert ein einzelnes Wort aus dem Vokabular V mit $i : 1 \dots V$.
Ein Dokument $\mathbf{w}$ besteht aus einer Sequenz von N Wörtern, $\mathbf{w} = (w_1, w_2, \dots, w_N)$.
Ein Korpus D repräsentiert eine Sammlung von M Dokumenten;
$D = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_M)$.

Es wird nun angenommen, dass ein generativer Prozess stattgefunden hat. Generativ bedeutet in diesem Falle, dass Parameter gesucht werden, um die Daten im Korpus zu rekonstruieren.

Für jedes Dokument $\mathbf{w}$ im Korpus D sieht dieser Prozess folgendermaßen aus:

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

[4] In der Literatur ist im Kontext von Information Retrieval auch von Latent Semantic Indexing (LSI) die Rede, ein Unterschied besteht ausschließlich im Anwendungsgebiet.

[5] Dimensionsreduktion durch Singulärwertzerlegung (SVD)

Am einfachsten lässt sich diese Idee wieder anhand eines Beispiels nachvollziehen. Im ersten Schritt wird N zufällig aus einer Poisson-Verteilung gezogen. Eine Poisson-Verteilung ist eine diskrete[6] Wahrscheinlichkeitsverteilung, die sich aus der Binomialverteilung herleiten lässt.

Storrer (1995) führt folgende Definition an:

> "Die diskrete Zufallsgröße X folgt der Poisson-Verteilung mit dem Parameter μ ($\mu > 0$), wenn sie die (abzählbar unendlich vielen) Werte 0, 1, 2, ... annimmt und wenn für die zugehörigen Wahrscheinlichkeiten gilt:
>
> $$P(X = k) = \frac{\mu^k}{k!} e^{-\mu}, \qquad k = 0, 1, 2, \ \ldots \tag{9}$$
>
> Der Parameter μ kann eine beliebige reelle Zahl > 0 sein."

Modelliert wird eine Wahrscheinlichkeit von Ereignissen, die für einzelne Elemente sehr gering, für eine große Anzahl an Elementen aber beobachtbar ist, wenn ein Wert für das durchschnittliche Auftreten in einem festen Zeit– oder Raumintervall angegeben wird.

Betrachten wir als Dokument $\mathbf{w}$ ein wissenschaftliches Papier vom MIT, das dem Berichtsband einer Konferenz (unserem Korpus D) entnommen wurde. Als Wert für N erhalten wir dann beispielsweise 2000; das Papier bestand also aus 2000 Wörtern.

Im zweiten Schritt muss θ bestimmt werden, eine k-dimensionale[7] Dirichlet-verteilte Zufallsvariable. Eine Dirichlet-Verteilung ist eine multivariate (mehrdimensionale) Verteilung, die grob gesagt eine Wahrscheinlichkeit über eine Wahrscheinlichkeitsverteilung angibt.

Bei letzterer handelt es sich um eine Multinomialverteilung, die Wahrscheinlichkeiten für das Auftreten von k verschiedenen Ereignissen (für das Werfen einer Münze beispielsweise Kopf oder Zahl) bestimmt.

Eine Dirichlet-Verteilung gibt nun an, wie wahrscheinlich eine solche Verteilung auftritt (a priori sollte die Multinomialverteilung am wahrscheinlichsten sein, die beiden Seiten der Münze die gleiche Wahrscheinlichkeit zuweist).

Gesteuert wird dieser Prozess über einen Hyperparameter α. Für einen fairen Münzwurf würde α große und symmetrische Werte annehmen. Ist die Münze gezinkt, erhielten wir Werte $\alpha < 1$.

[6] Diskrete Wahrscheinlichkeitsverteilungen operieren auf endlichen oder abzählbar unendlichen Mengen.

[7] Die Dimensionalität k der Dirichlet-Verteilung wird festgelegt.

Wir wissen bereits, dass unser Beispielpapier aus 2000 Wörtern besteht. Analog zur Einführung haben wir uns für $k = 3$ entschieden, die latenten Topics sind Physik, Informatik und Linguistik. Dem Vektor α weisen wir einen Wert von 0.1 zu. Er enthält kleine Werte, was die Wahrscheinlichkeit auf wenige Topics, die ungleich über das Dokument verteilt sind, erhöht. Zufällig ziehen wir daraus dann die tatsächliche Verteilung der Topics: $\theta_1 = (0.5, 0.3, 0.2)$. Während α im Laufe des generativen Prozesses unverändert bleibt, variiert θ dabei für jedes Dokument.

Für jedes Wort w_n wird anhand unserer Multinomialverteilung θ_1 nun zunächst zufällig ein Topic z_n ausgewählt. Da der Algorithmus nicht weiß, worum es sich bei den Topics inhaltlich handelt, werden diese einfach durchnummeriert. Das Topic für das erste Wort im ersten Dokument entspricht also zum Beispiel $z_{1,1} = 1$ (Physik).

Zuletzt müssen wir uns noch auf ein konkretes Wort w_n aus $p(w_n|z_n, \beta)$ festlegen. Bei β handelt es sich um eine $k \times V$-dimensionale Matrix, die die beobachtbaren Wortwahrscheinlicheiten wie in Formel (10) parametrisiert.

Eine Beispielmatrix ist in Tabelle 4 illustriert.

$$\beta_{i,j} = p(w^j = 1|z^i = 1) \tag{10}$$

Tabelle 4: $\beta_{i,j}$

	k × V	rule	**quantum**	algorithm	tree	semantics	...
1	**(Physik)**	0.3	**0.4**	0.15	0.1	0.05	...
2	(Informatik)	0.05	0.05	0.5	0.25	0.15	...
3	(Linguistik)	0.02	0.02	0.2	0.46	0.3	...

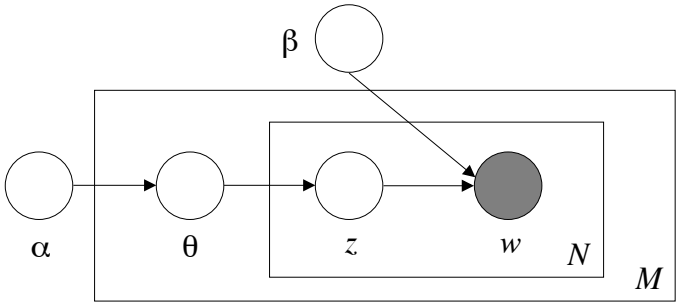
Die latenten Konzepte, die dem Dokument zugrunde liegen, werden zeilenweise von k repräsentiert. V denotiert das gesamte Vokabular der Sprache, in der das Dokument verfasst ist, mit je einem Wort in jeder Spalte.

Das erste Wort unseres Beispieldokuments könnte also *quantum* sein. Der Algorithmus geht dann über zum nächsten Wort. Der Zufallsprozess generiert $z_{1,2} = 3$ (Linguistik) und wir erhalten $w_{1,2} = tree$. Im nächsten Durchlauf wird $z_{1,3} = 2$ (Informatik) und $w_{1,3} = algorithm$ festgelegt usw.

Das Ergebnis nach 3 von 2000 Wiederholungen für das erste Dokument im Korpus wäre also "*quantum tree algorithm ...* "

Der generative Prozess, auf dem LDA basiert, lässt sich in einem graphischen Modell zusammenfassen. Knoten repräsentieren hier Zufallsvariablen, Kanten beschreiben mögliche Abhängigkeiten. Beobachtbare Variablen sind schattiert dargestellt. Tafeln (*plates*) zeigen replizierte Strukturen an.

Abbildung 1: Graphisches Modell LDA, Blei et.al (2003)



Der generative Prozess ist der Abbildung damit vollständig zu entnehmen. Es gibt zwei Hypervariablen α und β, die einmal pro Korpus gesampelt (aus den entsprechenden Verteilungen gezogen) werden und sich deshalb außerhalb der Tafeln befinden. Innerhalb der ersten Tafel befinden wir uns auf Dokument-Ebene, wir samplen θ einmal für jedes der M Dokumente. Die innerste Tafel repräsentiert die Vorgänge auf Wort-Ebene; für jedes der N Wörter ziehen wir zufällig ein Topic und ein beobachtbares Wort.

Gegeben die Parameter α und β definieren Blei et al. (2003) die bedingte Wahrscheinlichkeit der Topic-Verteilung θ, die Menge der Topics z und die Menge der Wörter $\mathbf{w}$ wie in (11) angegeben. Integrieren über θ und summieren über $\mathbf{z}$ ergibt (12).

$$p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta) \tag{11}$$

$$p(\mathbf{w}|\alpha, \beta) = \int p(\theta|\alpha) \left(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) p(w_n|z_n, \beta) \right) d\theta \tag{12}$$

$$p(D|\alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha) \left(\prod_{n=1}^{N_d} \sum_{z_d n} p(z_{dn}|\theta_d) p(w_{dn}|z_{dn}, \beta) \right) d\theta_d \tag{13}$$

Das Produkt der Randwahrscheinlichkeiten der einzelnen Dokumente resultiert abschließend in (13) in der Wahrscheinlichkeit eines Korpus D.

Das eigentliche Ziel ist allerdings, diesen generativen Prozess umzukehren, um die latenten Wahrscheinlichkeitsverteilungen (θ, β) ausgehend von unseren Beobachtungen $(\mathbf{w})$ zu inferieren. Möglich ist das mithilfe verschiedener Inferenzalgorithmen. Während Blei et al. (2003) einen Algorithmus zur variationalen Inferenz einführen, verwenden Dinu und Lapata (2010) einen Gibbs Sampler[8].

3.2.2 LDA im Kontext

Zur Erinnerung ist in (14) noch einmal die Gleichung abgebildet, die durch die parametrisierten Algorithmen gelöst werden soll.

$$p(z_k|t_i, c_j) \approx \frac{P(z_k|t_i)P(c_j|z_k)}{\sum_k P(z_k|t_i)P(c_j|z_k)} \tag{14}$$

$P(z_k|t_i)$ und $P(c_j|z_k)$ werden anhand der beiden latenten Variablenmodelle geschätzt.

Dinu und Lapata (2010) verwenden LDA jetzt, um (statt Topics) die Wortsinne der Zielwörter zu inferieren. Dies geschieht, indem die Input-Kookkurrenzmatrix anders interpretiert wird als zu Beginn.

Jede Reihe t_i (korrespondierend den Zielwörtern) wird jetzt als "Dokument" behandelt. Spalten enthalten weiterhin Kontextwörter c_j. In jeder Zelle (t_i, c_j) steht die Häufigkeit von c_j im "Dokument" von t_i.

Mittels Gibbs Sampling werden die benötigten Wahrscheinlichkeiten in (14) inferiert:

θ in (15) als Verteilung der Wortsinne eines jeden Zielwortes.

$$\theta_{i,k} = P(z_k|t_i) \tag{15}$$

β in (16) als Kontextwort-Verteilung für jeden Wortsinn.

$$\beta_{k,j} = P(c_j|z_k) \tag{16}$$

[8]Eine ausführliche Herleitung übersteigt leider den Rahmen dieser Arbeit. Einen guten Einstieg bietet das Tutorial "Gibbs Sampling for the Uninitiated" von Reisnik und Hardisty (2010).

4 Experimentelles Design

Im folgenden Kapitel werden die Experimente vorgestellt, die Dinu und Lapata (2010) damit durchgeführt haben. Dazu werden Aufgabenstellungen, Training der Modelle und Evaluationsmethoden beschrieben.

4.1 Aufgabenstellungen

Das probabilistische Modell zur Repräsentation von Wörtern über die induzierten Wortsinne wird mit optimierten Parametern für jeweils nicht-negative Matrixfaktorisierung und Latent Dirichlet Allocation verwendet, um

1. Wortähnlichkeit zu berechnen (Finkelstein et al., 2002)

2. Automatisch lexikalische Ersetzungen von Zielwörtern zu ermitteln (McCarthy and Navigli, 2007).

4.1.1 Wortähnlichkeiten

Das Modell, das in Kapitel 2 eingeführt wurde, ist in der Lage, Wortähnlichkeiten unabhängig vom Kontext zu erfassen.

Die Ähnlichkeit zweier Wörter t_i und t_i' wird wie folgt berechnet:

$$sim(t_i, t_i') = sim(v(t_i), v(t_i')) \tag{17}$$

Angewandt wurden die Modelle auf einen Datensatz von Finkelstein et al. (2002). Dieser enthält 353 Wortpaare mit dazugehörigen Ähnlichkeitswerten, die durch manuelle Annotation ermittelt wurden. Ein Auszug ist in Tabelle 5 illustriert.

Tabelle 5: Datensatz nach Finkelstein et al. (2002)

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

4.1.2 Lexikalische Ersetzung

Wie gut die Modelle mit Wortbedeutungen im Kontext umgehen, wird mittels lexikalischer Ersetzung eines Wortes in einem bestimmten Kontext evaluiert.

Die Modelle erhalten eine vorgegebene Menge an lexikalischen Ersetzungen und gewichten diese entsprechend der semantischen Ähnlichkeit zum Zielwort:

$$sim(v(t_i, c_j), v(t'_i)) \tag{18}$$

Wie bisher repräsentiert t_i das Zielwort, c_j das Kontextwort und t'_i das (mehr oder weniger) geeignete lexikalische Substitut[9].

Das zweite Experiment basiert auf dem SemEval-2007 Lexical Substitution Task. Der Datensatz besteht aus 200 Zielwörtern in 10 verschiedenen Satzkontexten sowie geeigneten lexikalischen Ersetzungen, die von Hand annotiert wurden.

Tabelle (6) enthält beispielhaft die Daten für das englische Adjektiv *still*.

Tabelle 6: Datensatz SemEval-2007

Sentences	Substitutes
It is important to apply the herbicide on a **still** 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 **still** images.	motionless(3) unmoving (2) fixed (1) stationary (1) static (1)

4.2 Training der Modelle

NMF und LDA arbeiten mit denselben Input-Daten, konkret einer Bag-of-words-Matrix, die aus dem GigaWord Korpus extrahiert wurde. Wie zu Beginn stellen Reihen dieser Matrix Zielwörter dar. Spalten entsprechen kookkurrierenden Kontextwörtern, die in einem Kontextfenster von 5 Wörtern auftreten. Es wurden dafür nur die 3000 häufigsten Wörter im Korpus verwendet.

[9] Nur das Zielwort zu kontextualisieren ist diskriminativer als Zielwort und Substitut in ähnlichem Kontext auszuwerten, vgl. Thater et al. (2010).

Nicht-negative Matrixfaktorisierung wurde nach Lee and Seung (2000) implementiert. Experimentiert wurde mit verschiedenen Dimensionen $K : [600 - 1000]$, je in 200er Schritten. In zwei unabhängigen Durchläufen gab es 150 Iterationen des Algorithmus, der Mittelwert der beiden Ergebnisse steht als Endergebnis.

Latent Dirichlet Allocation setzt neben der Anzahl der Topics K auch α und β als Hyperparameter voraus. K variierte zwischen [600 – 1400], β wurde auf 0.01 festgesetzt, Werte für α entsprachen ausgehend von Porteous et al. (2008) $\alpha : \frac{2}{K}$ und basierend auf Griffiths und Steyvers (2004) $\alpha : \frac{50}{K}$. Gibbs Sampling ergab angewendet auf die "Dokumente" in der Input-Matrix nach 1000 Iterationen und Berechnung des Mittelwertes aus 5 Iterationen [600–1000] die gewünschten Ergebnisse.

Dinu und Lapata (2010) haben mit drei verschiedenen Ähnlichkeitsmaßen experimentiert, um Wortähnlichkeit zu berechnen: dem Skalarprodukt, Kosinusähnlichkeit und inverser Jensen-Shannon-Abweichung. Das Skalarprodukt ist darunter am ehesten geeignet, Wortähnlichkeiten zu repräsentieren:

$$sp(v, w) =< v, w >= \sum_i v_i w_i \tag{19}$$

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.

Als Baselines für die Berechnung von Wortähnlichkeiten in Isolation dienten Latent Semantic Analysis (LSA) nach Landauer und Dumais (1997) und ein Vektorraum ohne jegliche Dimensionalitätsreduktion.

Für LSA wurde die ursprüngliche Matrix mittels $U\Sigma V$ Singulärwertzerlegung mit Rang $k = 1000$ komprimiert und anschließend durch tf-idf[10] bzw. Zeilennormalisierung gewichtet. Auf den einfachen semantischen Raum ohne Dimensionalitätsreduktion wurden drei verschiedenen Gewichtungsmethoden (PMI[11], tf-idf und Zeilennormalisierung) und Ähnlichkeit nach Lin (1998) angewandt.

Für die Berechnung von Wortähnlichkeiten im Kontext wurde von verschiedenen Vektoroperationen (Addition und Multiplikation) auf dem semantischen Vektorraum und auf den dimensionalitätsreduzierten Matrizen von NMF und LDA[12] ausgegangen.

[10] Term frequency–inverse document frequency, vgl. Jones (1972)

[11] Pointwise Mutual Information, vgl. Church and Hanks (1990)

[12] Die Algorithmen wurden an dieser Stelle noch nicht vollständig durchlaufen.

4.3 Evaluationsmethoden

Die Berechnung semantischer Wortähnlichkeiten wird mithilfe des Spearman Rangkorrelationskoeffizienten evaluiert. Ein Rangkorrelationskoeffizient ist ein parameterfreies Maß für statistische Abhängigkeiten zwischen zwei Variablen.

Für das erste Experiment wird gemessen, wie gut eine beliebige monotone Funktion den Zusammenhang zwischen den berechneten Ähnlichkeitswerten und den Ähnlichkeitswerten, die in Finkelstein et al. (2002) durch die Annotatoren zur Verfügung gestellt wurden, beschreiben kann. Einen Spearman Korrelationskoeffizienten von +1 oder -1 erhalten wir also, wenn zwischen diesen beiden Variaben eine monotone Abhängigkeit besteht[13].

Lexikalische Ersetzung wird anhand Kendalls Tau evaluiert, ebenfalls ein Rangkorrelationskoeffizient. Er erfasst den Zusammenhang zwischen dem Goldstandard des SemEval Datensatzes und den Rankings, die die Algorithmen ausgeben.

5 Ergebnisse

Die Performanz der Modelle auf Wortähnlichkeiten ist in Tabelle 7 dargestellt.

Tabelle 7: Ergebnisse Wortähnlichkeit

Model	Spearmans ρ
SVS	38.35
LSA	49.43
NMF	**52.99**
LDA	**53.39**
LSA_{MIX}	49.76
NMF_{MIX}	51.62
LDA_{MIX}	51.97

SVS ist ein einfaches Vektorraummodell, das die besten Resultate mit tf-idf Gewichtung und Kosinusähnlichkeit liefert. Latent Semantic Analysis (LSA) wurde mit normalisierten Input-Zeilen, $K = 600$ Dimensionen und dem Skalarprodukt zur Berechnung der Wortähnlichkeiten in Matrix U initiiert.

[13]Diese Abhängigkeit muss nicht zwingend linear sein.

Auf nicht-negative Matrixfaktorisierung wurde dasselbe Ähnlichkeitsmaß angewandt, die Anzahl der Dimensionen ist optimalerweise $K = 1000$. Die besten Parameter für Latent Dirichlet Allocation sind ebenfalls das Skalarprodukt, $K = 1200$ und $\alpha = \frac{50}{K}$. Die Ergebnisse von sowohl NMF als auch LDA verbessern sich mit steigender Anzahl der Wortsinne.

Neben der Verwendung der Modelle mit festgelegtem Parameter-Setting wurde außerdem getestet, inwiefern sich das Gesamtergebnis verändert, wenn die Ergebnisse verschiedener Durchläufe mit verschiedenen Parametern gemittelt werden (LSA_{MIX}, NMF_{MIX} und LDA_{MIX}). Insgesamt werden die besten Ergebnisse mit nicht-negativer Matrixfaktorisierung und Latent Dirichlet Allocation erzielt, sowohl mit festgesetzten als auch mit jeweils veränderten Parametern.

Die Ergebnisse zur lexikalischen Ersetzung sind in Tabelle 8 illustriert.

Tabelle 8: Ergebnisse lexikalische Ersetzung

Model	Kendalls τ
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_{MIX}	**16.01**
Cont-LDA_{MIX}	**15.53**

Als Baselines werden Modelle verwendet, die Wortähnlichkeiten noch nicht im Kontext repräsentieren. SVS entspricht erneut einem semantischen Vektorraum. Add-SVS, Add-NMF und Add-LDA sind die Ergebnisse der Anwendung von Vektoraddition auf die entsprechenden Kookkurrenzmatrizen, die von den drei Modellen generiert werden. Mult-SVS, Mult-NMF und Mult-LDA denotieren die kompositionellen Modelle in Kombination mit Vektormultiplikation.

Für Cont-NMF und Cont-LDA sind die Ergebnisse der kontextualisierten Modelle mit individuellen Parameter Settings dargestellt. Cont-NMF$_{MIX}$ und Cont-LDA$_{MIX}$ stehen für die kontextualisierten Modelle mit variierenden Parametern. Die Ergebnisse der letzten beiden Modelle übertreffen die Ergebnisse aller anderen Konstellationen[14].

Dinu und Lapata (2010) haben diese Evaluationsergebnisse nun auch qualitativ analysiert. Wie sehr Kontextwörter die Verteilung über die Wortsinne eines Zielwortes beeinflussen, kann am Beispiel in (4) deutlich gemacht werden.

(4) 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.

Mithilfe von Latent Dirichlet Allocation werden für *jam* ohne Kontext zunächst die Wortsinne in Tabelle 9 induziert.

Tabelle 9: Wortsinne *jam* ohne Kontext

Wortsinne	Wortverteilungen
Traffic (0.18)	*road, traffic, highway, route, bridge*
Music (0.04)	*music, song, rock, band, dance, play*
Fan (0.04)	*crowd, fan, people, wave, cheer, street*
Vehicle (0.04)	*car, truck, bus, train, driver, vehicle*

Kommt *jam* nun im Kontext von *band* vor, erhöht sich die Wahrscheinlichkeit für den latenten Wortsinn MUSIC von 0.04 auf 0.88; als geeignetste lexikalische Substitute werden *riff* und *gig* ausgegeben.

Dem vielversprechenden Anschein zum Trotz gibt es allerdings auch einige Phänomene, die von den kontextualisierten Modellen noch nicht gänzlich erfasst werden.

Es kommt vor, dass Wortsinne nur unvollständig disambiguiert werden. Die Ausgangswahrscheinlichkeiten der verschiedenen zugrundeliegenden Wortsinne von *bug* sind Software (0.09) und Disease (0.06). Im Kontext von *client* wird als Ergebnis Secret–Agency (0.34) und Software (0.29) angegeben. In Bezug auf Secret–Agency ist immer noch nicht klar, ob es sich um einen Programmfehler oder den nächsten Spionageskandal handelt.

[14]Die Differenzen sind mit $p < 0,01$ statistisch signifikant.

Auch domänenspezifische Wortsinne sind problematisch. Während *function* zunächst MATH und SERVICE zugeordnet wird, ergibt der mathematische Kontext *distribution* den Wortsinn SERVICE, geschuldet der Tatsache, dass es sich bei dem GigaWord Korpus um ein Zeitungskorpus handelt, das *function* überwiegend in seiner Bedeutung als Aufgabe oder Tätigkeit einer Person enthält.

Dem Wortpaar *(check, shoulder)* in Satz (5) werden von unserem Modell lokale Wortsinne zugewiesen, die im globalen Kontext nicht zutreffen.

(5) Check the shoulders so it hangs well, stops at hips or below, and make sure the pants are long enough.

INJURY (0.81) und BALL–SPORTS (0.10) sind im CLOTHING Kontext unangebracht.

Außerdem resultieren aus der Bag-of-words-Annahme grundsätzlich weniger gute Ergebnisse für Verben. Für das Wortpaar *(let, know)* werden beispielsweise die sehr uninformativen Wortklassen {*see, know, think, do*} (0.57) und {*go, say, do, can*} (0.20) vorgeschlagen.

6 Zusammenfassung und Ausblick

Ausgeführt wurde ein allgemeines Rahmenkonzept zur Berechnung von semantischer Ähnlichkeit im Kontext. Die Grundidee ist, Wortbedeutungen ohne Kontext als Wahrscheinlichkeitsverteilung über globale Wortsinne und kontextuelle Wortbedeutung als Veränderung dieser Verteilung zu modellieren. Es wird die gleiche Vektorrepräsentation für Zielwörter in Isolation sowie für Wörter im Kontext verwendet, was das Modell konzeptuell einfach macht. Dinu und Lapata (2010) instanziieren das Modell mithilfe nicht-negativer Matrixfaktorisierung und Latent Dirichlet Allocation. Beide parametrisierte Algorithmen erzielen bessere Ergebnisse für Wortbedeutungen im Kontext als vorausgehende Modelle.

Verbesserungsspielraum besteht in großem Maße im Hinblick auf die Kontextwörter. Es werden weitergehende Untersuchen angeregt, die den kollektiven Einfluss mehrerer Kontextwörter sowie die Repräsentation der Kontextwörter im Allgemeinen einschließen. Idealerweise würden zukünftige Modelle darüber hinaus syntaktische Dependenzen einbeziehen, die die Unzulänglichkeiten der Bag-of-words-Annahme revidieren.

Ein interessanter Beitrag zum Thema Disambiguierung verschiedener Wortsinne wurde kürzlich auf der *Conference of the German Society for Computational Linguistics*

and Language Technology (GSCL)[15] präsentiert.

Houlsby and Ciaramita (2013) haben unter dem Schlagwort *Distributed Wikipedia LDA* ein Verfahren entwickelt, das Wörter anhand der Wissensdatenbank disambiguiert, die mit Wikipedia[16] im großen Stil frei verfügbar ist. Die strukturierten Informationen werden anhand eines neu erprobten Algorithmus verarbeitet, der sich basierend auf einem Rahmenkonzept zu verteilter Inferenz und Bedeutungsrepräsentation aus Latent Dirichlet Allocation und MapReduce[17] zusammensetzt. Die Idee dahinter ist, Wikipedia-Artikel als Dokumenten zugrundeliegenden Topics zu behandeln. Während Houlsby and Ciaramita (2013) den Schwerpunkt auf die Verarbeitung großer Datenmengen legen, soll an dieser Stelle herausgestellt werden, welchen qualitativen Vorteil ihre Herangehensweise gegenüber vorhergehenden Ansätzen bietet.[18]

Zum Einen wird auf diese Weise das Problem der Benennung der latenten Topics gelöst. Vorausgehend wurde angemerkt, dass die parametrisierten Algorithmen, die Dinu und Lapata (2010) verwenden, nicht im Stande sind, die festgelegte Anzahl an Topics inhaltlich bzw. semantisch zu erfassen. Um die Ergebnisse interpretieren zu können, ist das aber unerlässlich. Dieses Defizit kann ausgeglichen werden, indem die Topics nach den Titeln der Wikipedia-Artikel benannt werden, denen sie zugeordnet sind. Es wurde außerdem diskutiert, wie domänenspezifische und lokale Wortsinne die korrekte Disambiguierung eines Zielwortes erschweren. Aufgrund der Vielzahl an Kontextwörtern, die in Wikipedia-Artikeln enthalten sind, wird die Wahrscheinlichkeit domänenspezifischer Wortsinne reduziert. Lokale Wortsinne sind ebenfalls relativ unwahrscheinlich, da sie durch Wikipedias Begriffsklärung (engl. *Disambiguation*) von vornherein ausgeschlossen werden können.

Das Modell von Houlsby and Ciaramita (2013) ist andererseits alleinig auf bereits benannte Entitäten (engl. *Named Entities*) ausgelegt. Dies hat zur Folge, das beispielsweise keinerlei Wortähnlichkeiten zwischen Verben berechnet werden können.

Auf dem Aida-CoNLL Datensatz von Hoffart et al. (2011) werden allerdings sowohl in Bezug auf Skalierbarkeit von LDA als auch in Bezug auf die Genauigkeit der Disambiguierungen die bislang besten Ergebnisse erzielt.

[15] `http://gscl2013.ukp.informatik.tu-darmstadt.de/de/gscl-2013/`

[16] `http://www.wikipedia.org/`

[17] MapReduce ist ein Programmiermodell, das von Google entwickelt wurde, um große Datenmengen auf Computerclustern zu berechnen.

[18] Nichtsdestotrotz lohnt sich ein Blick auf die technischen Details des Originalpapiers.

Literatur

[1] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

[2] Kenneth W. Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Comput. Linguist.* 16, 1 (March 1990), 22–29.

[3] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

[4] Georgiana Dinu and Mirella Lapata. 2010. Measuring Distributional Similarity in Context. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1162–1172, MIT, Massachusetts, USA.

[5] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: the concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

[6] Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235.

[7] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792.

[8] Thomas Hofmann. 2001. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 41(2):177–196.

[9] Neil Houlsby and Massimiliano Ciaramita. 2013. Scalable Probabilistic Entity-Topic Modeling. In *ArXiv e-prints*, arXiv:1309.0337v1 [stat.ML].

[10] Karen Spärck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, Vol. 28, pages 11–21.

[11] Thomas K. Landauer and Susan T. Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240.

[12] Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562.

[13] Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the joint Annual Meeting of the Association for Computational Linguistics and International Conference on Computational Linguistics*, pages 768–774, Montréal, Canada.

[14] Diana McCarthy and Roberto Navigli. 2007. SemEval-2007 Task 10: English Lexical Substitution Task. In *Proceedings of SemEval*, pages 48–53, Prague, Czech Republic.

[15] Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2008. Fast collapsed gibbs sampling for latent Dirichlet allocation. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 569–577, New York, NY.

[16] Philip Resnik and Eric Hardisty. 2010. Gibbs Sampling for the Uninitiated. *Technical Report: LAMP-TR-153*, University of Maryland, College Park.

[17] Hans H. Storrer. 1995. Einführung in die mathematische Behandlung der Naturwissenschaften II. Die Poisson-Verteilung. *Birkhäuser-Skripten, Bd. 2 u. Bd. 8*, S. 151–162, Birkhäuser, Basel.

[18] Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2010. Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 948–957, Uppsala, Sweden.