Probabilistische online Wissensgraphkonstruktion aus natürlicher Sprache

Clemens Damke

25. September 2017
Version: Entwurf 1

PADERBORN UNIVERSITY
The University for the Information Society
Department of Electrical Engineering,
Computer Science and Mathematics
Warburger Straße 100
33098 Paderborn

INTELLIGENT SYSTEMS
Intelligent Systems Group (ISG)

Bachelorarbeit

Probabilistische online Wissensgraphkonstruktion aus natürlicher Sprache

Clemens Damke

1. Korrektor Prof. Dr. Eyke Hüllermeier
Institut für Informatik
Universität Paderborn

2. Korrektor Prof. Dr. Axel-Cyrille Ngonga Ngomo
Institut für Informatik
Universität Paderborn

Betreuer Dr. Theodor Lettmann und Prof. Dr. Eyke Hüllermeier

25. September 2017

Clemens Damke
Probabilistische online Wissensgraphkonstruktion aus natürlicher Sprache
Bachelorarbeit, 25. September 2017
Korrektoren: Prof. Dr. Eyke Hüllermeier und Prof. Dr. Axel-Cyrille Ngonga Ngomo
Betreuer: Dr. Theodor Lettmann und Prof. Dr. Eyke Hüllermeier

Universität Paderborn
Intelligente Systeme
Institut für Informatik

Pohlweg 51
33098 Paderborn

Abstract

Hallo Welt. Test5.

Inhaltsverzeichnis

1 Einleitung

> *The actual world cannot be distinguished from a world of imagination by any description. Hence the need of pronoun and indices, and the more complicated the subject the greater the need of them.*
>
> — **Charles Sanders Peirce**
> Mathematiker und Philosoph

1.1 Motivation

In den letzten Jahren hat die Repräsentation von Wissensbasen durch Graphen, sog. Wissensgraphen, immer mehr an Bedeutung gewonnen. Google, Bing und IBM Watson benutzen solche Wissensgraphen z. B. zum Beantworten von komplexen Suchanfragen.

Die Grundidee dabei ist es, Entitäten durch Knoten und Relationen durch Kanten abzubilden. Entitäten können konkrete Dinge, wie z. B. Personen, aber auch abstrakte Konzepte, wie z. B. historische Epochen, sein. Relationen beschreiben beliebige Beziehungen zwischen den Entitäten, z. B. $person(\text{Da Vinci}) \xrightarrow{\text{lived in}} epoch(\text{Renaissance})$. Die Entität, von der eine solche Relation ausgeht, wird als Subjekt und die Zielentität als Objekt der Relation bezeichnet.

Die Typen von Entitäten bzw. Relationen (z. B. $person$ bzw. $lived\ in$) und deren Bedeutung sind dabei i. d. R. formal in einer sog. Ontologie spezifiziert. Die Ontologie beschränkt also die Menge gültiger Wissensgraphen, was eine effiziente maschinelle Verarbeitung der im Graph enthaltenen Informationen ermöglicht.

Da Wissensgraphen in zahlreichen Domänen einsetzbar sind, wird deren automatisierte Konstruktion bereits seit Jahren erforscht. Manuelles Konstruieren und vor allem anschließendes Warten und Aktualisieren von Wissensgraphen, ist aufgrund der abzubildenden Datenmengen nicht praktikabel. Bei einer maschinellen automatisierten Konstruktion sind insbesondere zwei Anforderungen problematisch:

1. Das Verarbeiten von unstrukturierten Eingaben, wie z. B. natürlichsprachlichen Texten.

2. Effizientes Eingliedern neuer Informationen in einen bestehenden Wissensgraphen. Dieses Eingliedern von Informationen umfasst im Speziellen:

    - **Entity Resolution:** Hinzukommende Entitäten, die bereits im Graphen enthalten sind, müssen als Duplikate erkannt werden. Dies ist i. d. R. nicht trivial, da die selbe Entität durch viele verschiedene, oftmals vom Kontext abhängige, Token repräsentiert werden kann; z. B. *Bob* vs. *Robert* oder *Der Papst* vs. *Franziskus*.

    - **Link Prediction:** Hinzukommende Entitäten müssen mit bereits bestehenden Entitäten in Relation gesetzt werden. Hinzukommende Relationen können zudem benutzt werden um andere Relationen zu inferieren; z. B.

$$female(A) \wedge B \xrightarrow{\text{son of}} A \implies A \xrightarrow{\text{mother of}} B$$

Die Kombination dieser beiden Anforderungen ist interessant, da das meiste verfügbare Wissen in natürlichsprachlicher Textform vorliegt und zudem permanent neues Wissen entsteht. Ein automatisiertes Wissensgraphkonstruktionsverfahren, welches beide Anforderungen berücksichtigt, ist daher in diversen Domänen von Nutzen. Ein Beispiel hierfür ist die Auswertung von Kommunikationsdaten aus E-Mails oder Chat-Nachrichten mit dem Ziel die sozialen Beziehungen und Intentionen der Kommunikationpartner zu ermitteln.

1.2 Ziele der Arbeit

Das übergeordnete Ziel dieser Arbeit ist es, ein Verfahren zu finden, welches das soeben beschriebene Problem der automatisierten Wissensgraphkonstruktion für E-Mail-Daten löst. Konkret sei ein Stream von E-Mails gegeben, denen jeweils ein Inhalt, ein Absender, eine Menge von Empfängern und ggf. weitere Metadaten, wie z. B. Absendezeit, Absendeort oder IP-Adresse, zugeordnet ist. Die Nachrichteninhalte werden der Einfachheit halber als ausschließlich englischsprachig angenommen. Außerdem wird eine, für E-Mails und andere Kurznachrichten typische, eingeschränkte Sprachkomplexität angenommen. Die Nachrichten sollen nacheinander in das zu konstruierende System eingefügt werden, welches sukzessive einen Wissensgraphen daraus erzeugt.

Für diese Erzeugung muss eine Reihe von Teilproblemen gelöst werden:

1. **Onotologie:** Spezifikation einer Wissensgraphontologie, die mächtig genug ist, um die Diversität natürlichsprachlich beschriebener Informationen abzubilden.

2. **Repräsentation:** Spezifikation der maschinellen Repräsentation des Wissensgraphen.

3. **Sprachverarbeitung:** Finden eines Verfahrens, welches die natürlichsprachlichen Inhalte der Nachrichten in eine für die Wissensgraphkonstruktion geeignete Form bringt.

4. **Grapherweiterung:** Finden eines Verfahrens, um eine eintreffende Nachricht in den bestehenden Wissensgraphen einzufügen.

Das aus den Teillösungen zusammengesetzte Verfahren muss, neben der offensichtlichen Anforderung einen Wissensgraphen zu konstruieren, zudem folgende technische Anforderungen erfüllen:

1. **Erweiterbarkeit:** Es sollen Schnittstellen eingeplant sein, um neben der Sprachverarbeitung auch andere Verarbeitungsverfahren, z. B. für Bilder, hinzufügen zu können. Die Graphontologie, Graphrepräsentation und das Grapherweiterungsverfahren dürfen also nicht zu sehr auf die Struktur natürlicher Sprache zugeschnitten sein.

2. **Parallelisierbarkeit:** Das Verfahren soll in der Lage sein die Rechenleistung mehrerer Prozessorkerne zu nutzen. Diese Anforderung betrifft insbesondere das Grapherweiterungsverfahren.

1.3 Aufbau der Arbeit

Kapitel 2

Kapitel 3

Kapitel 4

Kapitel 5

Kapitel 6

2 Verwandte Arbeiten

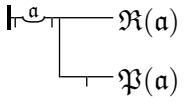
Die in 1.2 beschriebenen Ziele werden bereits seit langem erforscht. Der Begriff *Wissensgraph* wurde 2012 durch Google popularisiert, die Ideen dahinter lassen sich allerdings bis ins Ende des 19. Jahrhunderts zurückverfolgen. Dieses Kapitel zeigt auf, wie sich die Themen dieser Arbeit in die bisherige Forschung einfügen. 2.1 ordnet hierfür das Konzept des Wissensgraphen in die Entwicklungsgeschichte der Wissensrepräsentation ein. In 2.2 wird ein Überblick über die momentan verbreiteten *Natural Language Processing* (NLP) Werkzeuge gegeben. Das Wissensgraphkonzept und NLP wird schließlich in 2.3 kombiniert und es werden die aktuell verwendeten Verfahren zur Wissensgraphkonstruktion beschrieben.

2.1 Ansätze zur Wissensrepräsentation

2.1.1 Logische Grundlagen

Die Entwicklung der Wissensrepräsentation hängt eng mit der Entwicklung der Logik zusammen. Während in der formalen Logik und Mathematik die Prädikatenlogik das allgemein verwendete Kalkül ist und alternative Formalismen kaum verbreitet sind, finden im Bereich der Wissensrepräsentation bis heute diverse andere Kalküle Verwendung. Diese werden im Folgenden kurz vorgestellt.

Begriffsschrift (1879) Gottlob Freges Buch über die *Begriffsschrift* [Fre79] gilt als eines der bedeutsamsten Werke der Logik. Sie ist der erste Formalismus mit der Mächtigkeit der modernen Prädikatenlogik zweiter Ordnung mit Identität. Frege benutzt hierfür eine zweidimensionale Notation, die sich stark von der heute gebräuchlichen, linearen, an die Algebra angelehnte Notation unterscheidet.



$$\Leftrightarrow \quad \exists\, a : P(a) \vee R(a)$$

Im Gegensatz zur Prädikatenlogik gibt es keine eigene Syntax für *UND* und *ODER*; diese Operatoren müssen durch die Kombination von Negation und Implikation abgebildet werden. Zudem gibt es ausschließlich den Allquantor; eine existenzquantisierte

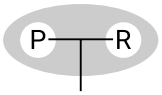
Aussage muss durch Negation der negierten allquantisierten Aussage ausgedrückt werden.

Prädikatenlogik: Peirce Notation (1885) Unabhängig von Frege entwickelte der amerikanische Mathematiker Charles Sanders Peirce ebenfalls ein prädikatenlogisches Kalkül [Pei85]. Peirces Notation hatte starke Ähnlichkeiten mit der heute benutzten linearen Schreibweise. Statt den modernen Symbolen hat Peirce allerdings die algebraischen Operatoren benutzt, um die Analogien zwischen Logik und Algebra auszudrücken.

$$\Sigma_a \, P_a + R_a \quad \Leftrightarrow \quad \exists \, a : P(a) \vee R(a)$$

Existential Graphs (1897) Neben seiner zuvor entwickelten linearen prädikatenlogischen Notation, hat Peirce zudem viele Jahre an einem alternativen graphischen Kalkül gearbeitet, welches er *Existential Graphs* (zu dt. Existenzgraphen) [Sow11] nannte. Ähnlich wie die Begriffsschrift werden Existenzgraphen zweidimensional dargestellt. Von dieser Gemeinsamkeit abgesehen, funktionieren sie allerdings fundamental verschieden. Ein logischer Ausdruck wird hier durch einen ungerichteten Graphen beschrieben. Die konkrete räumliche Anordnung der Knoten und Kanten hat dabei keine semantische Relevanz.

Peirce hat Existenzgraphen als ein dreistufiges aufeinander aufbauendes System konzipiert. Die erste Stufe, die sog. α-Graphen, umfasst alle notwendigen syntaktischen Elemente, um ein Kalkül mit der Mächtigkeit der Aussagenlogik zu erhalten. Die β-Graphen bilden die zweite Stufe und erweitern die Syntax der α-Graphen, sodass die Mächtigkeit der Prädikatenlogik erster Ordnung erreicht wird. Sowohl für α-, als auch für β-Graphen, ist die Vollständigkeit und Korrektheit bewiesen. Die dritte Stufe (γ-Graphen) wurde von Pierce nie vollendet; sie deckt in etwa die Mächtigkeit der heutigen Prädikatenlogik höherer Ordnung sowie der Modallogik ab.



$\Leftrightarrow \quad \exists \, a : P(a) \vee R(a)$

Wie schon die Begriffsschrift, sind Existenzgraphen syntaktisch minimal. Direkt ausdrücken lässt sich lediglich *UND*, der Existenzquantor und die Negation. Ein weiterer Unterschied zur heutigen Prädikatenlogik ist die Beschreibung logischer Inferenzen. Im Gegensatz zu den prädikatenlogischen Ersetzungsaxiomen, die auf der syntaktischen Struktur von logischen Ausdrücken operieren (z. B. für Kommutativität), lassen sich die Ersetzungsaxiome für Existenzgraphen als Graphtransformationsregeln verstehen, die bestimmte Teilmengen der Knoten und Kanten eines Ausdrucks durch andere äquivalente Knoten- und Kantenmengen ersetzen.

Prädikatenlogik: Peano-Russell Notation (1910) Die zweidimensionalen Notationen wurde häufig kritisiert, da sie die lineare, algebraische Notation der symbolischen Logik von Boole und De Morgan verwarfen [Slu87]. Freges Begriffsschrft und Peirces Existenzgraphen konnten sich daher nicht durchsetzen. Peirces algebraische prädikatenlogische Notation hingegen, stieß auf größere Akzeptanz. Giuseppe Peano hat auf deren Basis eine ähnliche Notation entwickelt, welche allerdings nicht die algebraischen Operatoren benutzt, damit sich logische Ausdrücke besser mit mathematischen Ausdrücken kombinieren lassen. Bertrand Russell hat Peanos Notation anschließend in leicht abgewandelter Form in den *Principia Mathematica* [WR10] benutzt. Diese sog. Peano-Russell-Notation ist im Wesentlichen identisch mit der modernen Schreibweise.

Trotz des Verschwindens der zweidimensionalen Notationen, finden sich noch heute Anlehnungen daran [Wik]. So ist z. B. die Negation $\neg A$ auf Freges negierten Inhaltsstrich —┬— A und der Ableitungsoperator $\vdash$ auf Freges Urteilsstrich mit angefügtem Inhaltsstrich ├── zurückzuführen.

2.1.2 Entwicklung maschineller Wissensrepräsentation

Die Idee Computer zur Lösung beliebiger Probleme zu benutzen ist nicht neu. Da ein solches maschinelles Problemlösen die Verfügbarkeit von Hintergrundwissen über die Problemdomäne erfordert, wurden Methoden zur Wissensrepräsentation immer im Zusammenhang mit Problemlösern erforscht [HR+83]. So wie effiziente Datenstrukturen die Implementation effizienter Algorithmen ermöglichen, ermöglichen gute Wissensrepräsentationen die Implementation guter Problemlöser. Was genau nun als ein guter Problemlöser verstanden wird, hat sich im Laufe der Jahre allerdings immer wieder verändert.

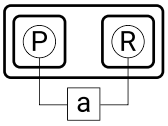
Universelle Problemlöser Einer der ersten maschinellen Problemlöser war der von Newell und Simon 1955 entwickelte *Logic Theorist* (LT) [NS56]. LT war in der Lage logische Aussagen zu beweisen, indem er systematisch Ersetzungsaxiome auf eine gegebene Aussage angewandt hat, bis die gesuchte Lösung abgeleitet wurde.

Die Grundidee des LT haben Newell, Simon und Shaw 1959 im *General Problem Solver* (GPS) [New+59] erweitert. Es wurden Heuristiken hinzugefügt, um den Suchraum geschickter zu durchlaufen. GPS war ein universeller Problemlöser, konnte also jedes Problem lösen, das sich durch eine Menge von Horn-Klauseln ausdrücken lässt. Zwar war es so theoretisch möglich Probleme aus diversen Domänen zu lösen, aufgrund der kombinatorischen Explosion war GPS alledings nicht zur Lösung komplexer praktischer Probleme geeignet.

Expertensysteme Aufgrund der Misserfolge universeller Problemlöser für praktische Probleme, hat die Forschung begonnen sich mehr auf die Entwicklung von Expertensystemen zu fokussieren. Expertensysteme besitzen für gewöhnlich eine Wissensbasis in der domänenspezifisches Wissen in Form von Regeln und Fakten kodiert ist. Eine sog. Inferenzmaschine benutzt diese Regeln und Fakten um Probleme zu lösen.

Semantic Networks (1956) Die Idee, Graphen als Datenstruktur für Wissensbasen zu verwenden, taucht erstmal in den sog. *Semantic Networks* [Leh92] (zu dt. semantische Netzwerke) auf. Dieser Ansatz beschreibt Wissen als Menge von $(subject, predicate, object)$-Tripeln. Es gibt darüber hinaus allerdings keine klaren Regeln, wie ein semantisches Netz strukturiert sein muss. Semantische Netzwerke sind daher allenfalls als ein Oberbegriff für die große Vielfalt konkreter graphbasierter Wissensbasen zu verstehen.

Conceptual Graphs (1976) Wie genau mit Graphen komplexes Wissen beschrieben werden kann, das über eine reine Taxonomie hinaus geht, blieb bei semantischen Netzen unklar. John F. Sowas *Conceptual Graphs* [Sow76][Har+07] (zu dt. Konzeptgraphen) lösen dieses Problem. Statt Wissen lediglich als eine einfache Menge von Beziehungen abzubilden, wird es als prädikatenlogischer Ausdruck verstanden. Hierfür baut Sowa auf Peirces Existenzgraphen auf, die bis dahin weitestgehend unbeachtet waren.



$$\Leftrightarrow \quad \exists\, a : P(a) \vee R(a)$$

Dieser Ansatz erlaubt es komplexe Wissensbasen zu konstruieren, in denen nicht nur gespeichert werden kann, ob ein Konzept existiert, sondern auch, ob es nicht oder nur möglicherweise existiert. Da Konzeptgraphen, so wie schon die Existenzgraphen, ein vollständiges und korrektes Logikkalkül sind, lassen sich zudem Inferenzregeln für sie definieren. Der Vorteil hierfür einen Graphen statt eines prädikatenlogischen Ausdrucks zu verwenden ist, dass eine Graphstruktur einen deutlich effizienteren Zugriff auf gespeichertes Wissen ermöglicht.

Knowledge Graphs (1987) Der Begriff *Knowledge Graph* [RM92] (zu dt. Wissensgraph) bezeichnete ursprünglich eine Klasse semantischer Netze, deren Relationsmenge formal spezifiziert ist. Die Menge erlaubter Graphen wird hierdurch so eingeschränkt, dass das repräsentierte Wissen eindeutig interpretierbar und ohne Redundanz ist. Dies erlaubt die Definition von Inferenzregeln, um Schlussfolgerungen aus einem gegebenen Graphen zu ziehen. Im Laufe der Jahre ist die Grenze zwischen semantischen Netzen und Wissensgraphen allerdings so stark verschwommen, dass heute auch mehrdeutige, redundanzbehaftete semantische Netze, die lediglich we-

nige Relationstypen benutzen, als Wissensgraphen bezeichnet werden [McC+16]. Wissensgraphen und Konzeptgraphen müssen weiterhin streng unterschieden werden, da erstere oftmals Negation und Modalität nicht unterstützen.

2.1.3 Aktuelle Wissensrepräsentationsprojekte

Manuelle Ansätze

Semantic Web Das sog. *Semantic Web* bezeichnet eine Menge von W3C-Standards, die das bestehende Web um eine formale Wissensbeschreibungssyntax erweitern [BH]. Zentral ist dabei das *Resource Description Framework* (RDF), mit dem sich beliebige Konzepte, auch Resourcen genannt, beschreiben und verknüpfen lassen. Ziel ist es über die unstrukturierte Netzstruktur des bestehenden Webs, eine strukturierte, leicht maschniell verarbeitbare, Netzstruktur zu legen. Durch die Anfragesprache *SPARQL* ist es möglich Wissen aus diesem Netz auszulesen. Das Web würde somit zu einem großen denzentralen Wissensgraphen. Tim Berners-Lee beschreibt diese Idee als das "Web 3.0" [Sha06]. Obwohl die Technologien hierfür bereits seit Jahren existieren, sind bislang nur wenige Webseiten mit RDF-Tags annotiert. Häufige Kritik ist, dass das Semantic Web zu viel theoretisches Hintergrundwissen über Wissensrepräsentationsverfahren erfordert, um für die meisten Webseitenbetreiber zugänglich zu sein [MS03].

WordNet Das *WordNet* der Universität Princeton [Wor] ist ein frei verfügbares lexikalisch-semantisches Netz für die englische Sprache, d. h. ein semantisches Netz, welches die Bedeutung von Worten in Relation zueinander setzt. Relationen werden dabei z. B. für Synonyme, Hypernyme (Oberbegriffe) und Meronyme (Bestandteile) eingefügt. Der Datenbestand des WordNets wird manuell gepflegt und resultiert aus der Kombination der Einträge verschiedener Wörterbücher.

Automatisierte Ansätze

Neben den manuellen Grapherzeugungsansätzen des Semantic Webs und des WordNets, gibt es diverse voll- und semiautomatische Ansätze. Diese bauen die Graphstruktur selbstständig aus gegebenen Datenquellen auf.

NELL Das *Never-Ending Language Learning* (NELL) System [Car+10] traviersiert selbstständig das Internet und fügt die gefundenen textuellen Informationen in einen Wissensgraphen ein. Hierfür wird eine Kombination verschiedener Modelle

verwendet, die regelmäßig angepasst wird. Menschen können optional Feedback für die extrahierten Fakten geben, um die Inferenzqualität weiter zu verbessern.

Google Knowledge Graph Basierend auf den Ideen der in 2.1.2 vorgestellten Wissensgraphen, stellte Google 2012 eine eigene, ebenfalls *Knowledge Graph* genannte, Wissensgraphtechnologie vor [Sin12]. Sie wird benutzt, um Suchanfragen semantisch, statt per String-Matching, zu beantworten. So können z. B. zum Suchbegriff verwandte Ergebnisse angezeigt werden, selbst wenn es keine textuelle Ähnlichkeit zu jenem gibt. Laut Googles Aussagen stammen die Quelldaten u. a. aus Wikipedia Infoboxen, Wikidata und dem CIA World Factbook. Da es sich hierbei, im Gegensatz zu NELL, primär um strukturierte Daten handelt, ist das automatisierte Einpflegen mit hoher Genauigkeit möglich. Wie genau die Daten im Graph repräsentiert werden, ist nicht öffentlich bekannt.

Abb. 2.1. Popularität des Begriffs “knowledge graph” (Quelle: Google Trends [Goo])

2.2 NLP-Werkzeuge

Neben der Repräsentation von Wissen, ist auch die Verarbeitung natürlicher Sprache ein Kernbestandteil dieser Arbeit. Hierfür existiert bereits eine Vielzahl von *Natural Language Processing* (NLP) Werkzeugen. Trotz dieser Vielfalt lassen sich Kernverarbeitungsschritte festmachen, die in den meisten Werkeugen verwendet werden.

1. **Tokenization:** Oftmals einer der ersten Verarbeitungsschritte einer NLP Bibliothek. Eine Eingabezeichenkette wird dabei in eine Liste von Tokens zerlegt. Token sind u. a. Wörter, Satzzeichen und numerische Literale.

$$\text{“Today I’m testing myself.”} \rightarrow (\text{Today}, \text{I}, \text{’m}, \text{testing}, \text{myself}, .)$$

2. **Lemmatization:** Abbildung von Tokens auf ihre Lemmata (Grundformen).

$$(\text{Today}, \text{I}, \text{'m}, \text{testing}, \text{myself}, .) \rightarrow (\text{today}, \text{I}, \text{be}, \text{test}, \text{myself}, .)$$

3. **Part-of-speech Tagging:** Abbildung von Tokens auf ihre Wortarten und Flexionen (POS-Tags).

Today	I	'm	testing	myself	.
adverb	personal pronoun	present tense first-person singular verb	present participle verb	personal pronoun	sentence terminator

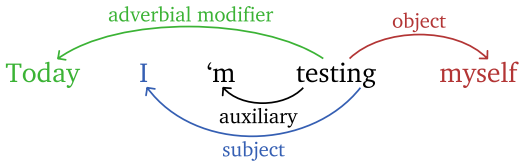
4. **Named Entity Recognition:** Klassifizierung von Tokens in Kategorien wie z. B. Person, Ort oder Zeitpunkt.

$$(\underbrace{\text{Today}}_{\text{date}}, \text{I}, \text{'m}, \text{testing}, \text{myself}, .)$$

5. **Coreference Resolution:** Bestimmung von Token-Äquivalenzklassen, die jeweils auf das selbe Konzept verweisen (insbesondere Pronomina und ihr Antezedens).

$$(\text{Today}, \text{I}, \text{'m}, \text{testing}, \text{myself}, .)$$

6. **Dependency Parsing:** Eine auf den POS-Tags aufbauende syntaktische Analyse, welche die grammatikalischen Abhängigkeiten der Token untereinander ausgibt. Die Menge dieser Abhängigkeiten bildet einen Baum oder baumähnlichen Graphen, der *Treebank* bzw. *Dependency Graph* genannt wird.



Wie sich erkennen lässt, bauen die Verarbeitungsschritte sukzessive aufeinander auf und bilden eine Art Pipeline [Usz]. Dieses Pipeline-Modell findet sich auch in vielen NLP-Werkzeugen wieder. Ein solches ist z. B. das quelloffene Stanford CoreNLP Projekt [Man+14][Cor], welches u. a. Module für alle der soeben vorgestellten Verarbeitungsschritte beinhaltet. Ein alternatives NLP-Toolkit ist Apache OpenNLP [Ope]; es bietet ähnliche Module wie CoreNLP an.

2.3 Wissensgraphkonstruktionsverfahren

Wie in 2.1.3 gezeigt, gibt es diverse Ansätze um Graphen aus Daten zu konstruieren. Da für das Thema dieser Arbeit insbesondere automatisierte Verfahren relevant sind, die mit unstrukurierten Daten, wie z. B. natürlicher Sprache, umgehen können, werden diese im Folgenden näher beschrieben.

Üblicherweise arbeiten Wissensgraphkonstruktionsverfahren nicht direkt mit den unstrukturierten Eingabedaten, wie z. B. den Inhalten von E-Mails, sondern mit einer Knoten- bzw. Konzeptmenge und ggf. auch einer Kanten- bzw. Relationsmenge, die zuvor, z. B. mittels eines in 2.2 vorgestellten NLP-Verfahrens, aus den Rohdaten extrahiert wurden. Die Wissensgraphkonstruktion ist somit äquivalent zum Problem der *Link Prediction*, also dem Finden von Relationen zwischen den gegebenen Konzepten. Die Link Prediction wiederum lässt sich als ein Problem des *Statistical Relational Learnings* (SRL) auffassen. In der Literatur finden sich im Wesentlichen drei Klassen von SRL-Verfahren [Nic+16], die auf verschiedenen Annahmen über die Korrelation der zu verknüpfenden Informationen basieren:

1. **Latent Feature Models:** Alle Relationen werden als bedingt unabhängig angenommen, sofern bestimmte Eigenschaften über Subjekte und Objekte der Relationen gegeben sind.

2. **Graph Feature Models:** Alle Relationen werden als bedingt unabhängig angenommen, sofern bestimmte Eigenschaften der Struktur des Graphen gegeben sind.

3. **Markov Random Fields:** Es wird angenommen und erlaubt, dass alle Relationen lokale Abhängigkeiten voneinander haben können.

Latent Feature Models Ein Beispiel für ein Latent Feature Modell ist das RESCAL-Verfahren [Nic13], welches auf Tensorfaktorisierung basiert. Die Grundidee dabei ist es, allen Entitäten i einen Feature-Vektor $e_i \in \mathbb{R}^H$ zuzuordnen und für alle Relationen k eine Gewichtsmatrix $W_k \in \mathbb{R}^{H \times H}$ zu finden. Die Konfidenz in die Existenz einer Relation $i \xrightarrow{k} j$ wird durch $e_i^\top \, W_k \, e_j$ beschrieben. Diese Definition ermöglicht eine sehr schnelle Link Prediction, da lediglich ein Vektor-Matrix-Vektor-Produkt berechnet werden muss. RESCAL liefert gute Ergebnisse, wenn die vorherzusagenden Relationen globale Abhängigkeiten aufweisen. Lokal stark zusammenhängende Teilgraphen werden allerdings schlecht erkannt, da nur der Feature-Vektor und nicht die Nachbarschaft einer Entität berücksichtigt wird; ein Beispiel hierfür sind symmetrische Relationen.

$$A \xrightarrow{\text{married to}} B \implies B \xrightarrow{\text{married to}} A$$

Graph Feature Models Komplementär zu den Latent Feature Modellen sind die Graph Feature Modelle. Statt Entitäten in einen Feature-Raum einzubetten, wird hier die Nachbarschaft der Entitäten betrachtet. Ein Beispiel hierfür ist der *Path Ranking Algorithmus* (PRA) [Lao+11]. PRA ermittelt Relationen durch zufälliges Durchwandern des Graphen. Um die Stärken der Latent Feature und Graph Feature Modelle zu kombinieren, wurden Hybrid-Modelle, wie z. B. das *Additive Relational Effects* (ARE) Verfahren [Nic+14], entwickelt, welches die Konfidenzen von RESCAL und PRA addiert.

Markov Random Fields Fundamental verschieden von diesen beiden Verfahren sind *Markov Random Fields* (MRFs). Hier sind prinzipiell Abhängigkeiten zwischen allen Relationen möglich, was MRFs sehr flexibel macht. Da dies hinsichtlich der Laufzeit schnell impraktikabel wird, wird das Modell um einen Abhängigkeitsgraphen erweitert, der die Anzahl von betrachteten Abhängigkeiten reduziert. Der Abhängigkeitsgraph darf dabei nicht mit dem Wissensgraphen verwechselt werden: Ersterer beschreibt statistische Abhängigkeiten zwischen Relationen, während letzterer Relationen zwischen Konzepten beschreibt. Zur Modellierung von Abhängigkeitsgraphen werden i. d. R. Kalküle verwendet, die an eine Prädikatenlogik erster Ordnung angelehnt sind. Das Finden eines Wissensgraphen ist in diesem Modell analog zum Lösen des MAX-SAT-Problems. Wählt man ein Kalkül in dem die Atome (i. e. Zufallsvariablen des MRFs) aus $[0, 1]$ sind, und Formeln zudem ausschließlich Disjunktionen und Negationen gemäß Łukasiewicz S-Norm enthalten, erhält man ein sog. *Hinge-Loss-MRF* [Bac+13][Bac+15], da die Loss-Funktion der Disjunktion in dieser S-Norm ein Hinge-Loss ist.

Ein konkretes Kalkül, welches sich zur Spezifikation von HL-MRFs eignet, ist die *Probabilistic Soft Logic* (PSL) [Br10][Bac+15]. MAX-SAT lässt sich für solche HL-MRFs effizient und parallelisierbar mit dem konvexen *Alternating Direction Method of Multipliers* (ADMM) Optimierungsverfahren [Boy+11] lösen. In dessen ursprünglichen Form ist ADMM allerdings ausschließlich für offline Inferenz geeignet; der Wissensgraph müsste also bei jeder Eingabe neu konstruiert werden. Um dieses Problem zu lösen, wurde das *Budgeted Online Collective Inference* (BOCI) Verfahren [Puj+15] entwickelt. BOCI nutzt Metadaten, die während der Ausführung von ADMM anfallen, um eine Bewertung für jedes Atom zu berechnen. Die Bewertung eines Atoms beschreibt, wie groß die erwartete Wertveränderung beim Eintreffen neuer Informationen ist. Kommen nun neue Informationen an, müssen ausschließlich die m höchstbewerteten Atome betrachtet werden, die Werte aller anderen Atome werden fixiert. Je höher das Budget m, desto höher ist die Qualität im Vergleich zu einer Neukonstruktion des Graphen. Es wurde empirisch gezeigt, dass die Inferenzqualität mit BOCI oft nur unwesentlich schlechter ist als bei einer kompletten offline Inferenz.

Die Kombination von PSL, ADMM und BOCI ist daher ein guter Ausgangspunkt für den Entwurf eines online Wissensgraphkonstruktionsverfahrens. Der Vorteil dieses Ansatzes gegenüber eines Latent Feature oder Graph Feature Modells ist, dass sich andere domänenspezifische Expertensysteme leicht in eine PSL Inferenz integrieren lassen. PSL erlaubt nämlich die Inklusion von benutzerdefinierten Funktionen und Prädikaten. Diese können benutzt werden, um z. B. die Levenshtein-Distanz zweier Zeichenketten oder domänenspezifisches Hintergrundwissen, wie die Distanz zwischen zwei namentlich genannten Orten, mit in die Entity Resolution einfließen zu lassen.

3 Theoretische Grundlagen

In Kapitel 2 wurde ein Überblick über das Problemumfeld der Wissensgraphkonstruktion gegeben. Diese Arbeit baut insbesondere auf den bereits kurz vorgestellten Konzeptgraphen, Stanfords CoreNLP Bibliothek und der PSL auf. Für die folgenden Kapitel ist ein Grundverständnis dieser drei Themen notwendig. Sie werden daher in den folgenden Abschnitten näher beschrieben.

3.1 Wissensmodellierung mit Konzeptgraphen

John F. Sowas Konzeptgraphen bilden die Basis der Graphontologie dieser Arbeit. Wie in 2.1.2 beschrieben, sind sie ein auf Existenzgraphen basierendes logisches Kalkül. Die vollständige Konzeptgraphsyntax geht allerdings weit über die Prädikatenlogik hinaus, da auch Modallogik und natürlichsprachliche Konzepte, wie z. B. Fragen und Betonungen, unterstützt werden. Da Sowas eigene Beschreibungen diesbezüglich teils etwas unklar sind, werden im folgenden lediglich die sog. *Conceptual Graphs with Cuts* [Dau03] vorgestellt. Sie sind eine zur Prädikatenlogik erster Ordnung äquivalente, formal spezifizierte Teilmenge der Konzeptgraphen, deren Vollständigkeit und Korrektheit bewiesen ist.

3.1.1 Syntax

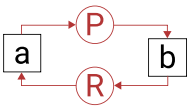
In ihrer einfachsten Form lassen sich Konzeptgraphen als Graphen mit drei Arten von Knoten und zwei Arten von Kanten beschreiben.

Konzeptknoten (*concepts*) Entsprechen in etwa existenzquantisierten gebundenen Variablen. Wie auch in der Prädikatenlogik, haben die Bezeichner von Konzeptknoten keine semantische Relevanz und können frei gewählt werden.

$$[\mathrm{a}]\ [\mathrm{b}] \quad \Leftrightarrow \quad \exists\, a, b \tag{3.1}$$

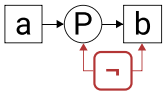
Relationsknoten (*conceptual relations*) und Argumentkanten (*arguments*) Relationsknoten entsprechen prädikatenlogischen Atomen. Das Symbol innerhalb eines Relationsknoten gibt die Relation des Atoms an. Für die Repräsentation der Argumente werden

sog. Argumentkanten zwischen Relationsknoten und Konzeptknoten verwendet. Die Position der Argumente bei mehrstelligen Relationen werden durch Nummerierung der Argumentkanten oder bei zweistelligen Relationen durch gerichtete Argumentkanten abgebildet. Wenn in einem Graphen mehrere Relationsknoten bzw. Atome auftauchen, werden diese als *UND*-verknüpft interpretiert; für die Abbildung von *ODER* wird die Negation verwendet.



$$\Leftrightarrow \quad \exists\, a, b:\ P(a,b) \wedge R(b,a) \tag{3.2}$$

Negationskontexte (*negation contexts* oder *cuts*) Für die Negation von Aussagen werden in Konzeptgraphen sog. Kontexte verwendet. Sie lassen sich neben der Negation auch zur Modellierung anderer Zusammenhänge nutzen, diese werden hier allerdings ausgelassen, um den Vergleich mit der Prädikatenlogik zu ermöglichen.

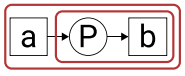


$$\Leftrightarrow \quad \exists\, a\ \neg\exists\, b : P(a,b) \tag{3.3}$$

$$\Leftrightarrow \quad \exists\, a\ \forall\, b : \neg P(a,b)$$

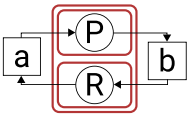
Die Darstellung von Kontexten mit Knoten und Kanten wird schnell unübersichtlich, daher werden stattdessen Boxen verwendet, die die Kindknoten umschließen.

Kontexte können nicht nur Konzeptknoten und Relationsknoten enthalten, sondern auch andere Kontexte. Hierbei ist zu beachten, dass alle Knoten und Kontexte höchstens einen Elternkontext haben können; die Linien zweier Kontextboxen dürfen sich also nicht schneiden.



$$\Leftrightarrow \quad \neg\exists\, a\ \neg\exists\, b : P(a,b) \tag{3.4}$$

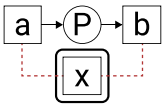
$$\Leftrightarrow \quad \forall\, a\ \exists\, b : P(a,b)$$



$$\Leftrightarrow \quad \exists\, a, b : \neg(\neg P(a,b) \wedge \neg R(b,a)) \tag{3.5}$$

$$\Leftrightarrow \quad \exists\, a, b : P(a,b) \vee R(b,a)$$

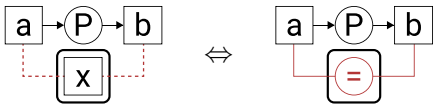
Koreferenzkanten (*coreference links*) Entspricht der Äquivalenzrelation.



$$\Leftrightarrow \quad \exists\, a, b : P(a, b) \wedge \neg\exists\, x : a = x \wedge x = b \tag{3.6}$$

$$\Leftrightarrow \quad \exists\, a, b : P(a, b) \wedge a \neq b$$

Prinzipiell ließe sich die Äquivalenz auch durch Relationsknoten ausdrücken. Um syntaktisch zu kennzeichnen, dass es sich nicht um eine beliebige Relation, sondern um eine Äquivalenzrelation handelt, wird dies jedoch i. d. R. nicht getan. Koreferenzkanten können also als eine Kurzschreibweise verstanden werden, die den Zweck hat die für die Inferenz relevanten Symmetrie-, Transitivitäts- und Reflexivitätseigenschaften zu kennzeichnen.



3.1.2 Dominierende Knoten

So wie die Syntaxelemente prädikatenlogischer Ausdrücke nicht beliebig kombiniert werden können, unterliegen auch Konzeptgraphen gewissen Einschränkungen. Die Einschränkung, dass alle Knoten und Kontexte höchstens einen Elternkontext haben dürfen, wurde bereits erwähnt. Die zweite wichtige Einschränkung ist das Verbot nicht dominierender Knoten (*dominating nodes*). Was genau dies bedeutet, wird im Folgenden erläutert. Zuerst müssen Konzeptgraphen jedoch formal spezifiziert werden.

$$\begin{aligned}
G &:= (V, E), \text{ mit globalem Kontext } \top \in V \\
concept(v) &:\Leftrightarrow v \in V \text{ ist ein Konzeptknoten} \\
relation(v) &:\Leftrightarrow v \in V \text{ ist ein Relationsknoten} \\
context(v) &:\Leftrightarrow v \in V \text{ ist ein Kontext, es gilt } context(\top) \\
neg(v) &:\Leftrightarrow v \in V \text{ ist ein Negationskontext} \\
nest(c, v) &:\Leftrightarrow context(c) \wedge (c, v) \in E \\
coref(v_1, v_2) &:\Leftrightarrow concept(v_1) \wedge concept(v_2) \wedge (v_1, v_2) \in E \\
arg(r, v) &:\Leftrightarrow relation(r) \wedge concept(v) \wedge ((r, v) \in E \vee (v, r) \in E) \\
E &\supseteq \{(\top, v) : v \in V \wedge \neg\exists\, c \in V : nest(c, v)\} \\
a \leq b &:\Leftrightarrow (\exists\, x \in V : a \leq x \wedge x \leq b) \\
&\quad \vee nest(b, a) \vee (\exists\, c \in V : nest(c, a) \wedge nest(c, b))
\end{aligned} \tag{3.7}$$

Um die nachfolgenden Definitionen einfacher zu machen, wird der globale Kontext $\top$ eingeführt, der, in Anlehnung an Peirce, *sheet of assertion* genannt wird. $\top$ enthält alle Knoten, die keinen explizit dargestellten Elternkontext haben. Die Kontextbox von $\top$ umschließt also den gesamten Konzeptgraphen. $\leq$ ist eine Quasiordnung und bildet die *enthalten-in*-Relation zwischen Knoten ab, d. h. $a \leq b$ gdw. a innerhalb der Box des Elternkontextes von b liegt. Das größte Element gemäß $\leq$ ist also immer $\top$.

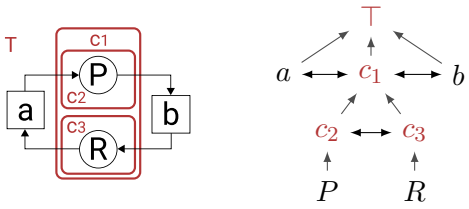
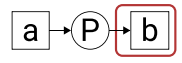


Abb. 3.1. Zusammenhang zwischen Kontexten und der $\leq$-Ordnung. Die Existenz eines Pfades von x nach y im obigen baumartigen Graphen, entspricht $x \leq y$.

Auf Basis von $\leq$ lässt sich nun das Konzept dominierender Knoten definieren.

$$dom(G) :\Leftrightarrow \forall\, r, v \in V : arg(r, v) \rightarrow r \leq v \qquad (3.8)$$
$$\wedge \forall\, v_1, v_2 \in V : coref(v_1, v_2) \rightarrow (v_1 \leq v_2 \vee v_2 \leq v_1)$$

Für jeden Konzeptgraphen G muss $dom(G)$ gelten. Eine Intuition für diese Einschränkung ist, dass es nicht sinnvoll ist die Existenz eines Atoms auszudrücken, welches durch nicht existente Variablen parametrisiert ist. Eine detaillierte Untersuchung des Zwecks dominierender Knoten und eine Beschreibung der entstehenden Probleme, wenn auf die Notwendigkeit dominierender Knoten verzichtet wird, findet sich in [Dau03, Abschnitt 14.3].



$\lightning\ \neg dom(G)$, da $arg(P, b) \wedge \neg(P \leq b)$.

$\lightning\ \neg dom(G)$, da $coref(a, b) \wedge \neg(a \leq b \vee b \leq a)$.

Abb. 3.2. Beispiele für fehlerhafte Konzeptgraphen ohne dominierende Knoten.

3.2 Stanford CoreNLP

Um natürlichsprachliche Daten in einen Konzeptgraphen zu transformieren, ist im ersten Schritt eine Sprachanalyse notwendig. Hierfür wurde die Stanford CoreNLP [Cor] und die Apache OpenNLP [Ope] Bibliothek in Erwägung gezogen, da beide häufig genutzt, aktiv weiterentwickelt, frei verfügbar und JVM-basiert sind. Die JVM-Integration ist wichtig, um mit anderen verwendeten Bibliotheken kompatibel zu sein; mehr hierzu in Abschnitt 4.1. Für die Implementation wurde schließlich CoreNLP gewählt, da es mit den mitgelieferten Modellen häufig bessere Ergebnisse als OpenNLP liefert. Da beide Bibliotheken bzgl. ihrer Funktionalität allerdings recht ähnlich sind, kann die NLP Komponente als substituierbar angesehen werden. Ein Wechsel von CoreNLP auf OpenNLP wäre mit relativ geringem Aufwand möglich.

Im Folgenden wird nun die grundlegende Architektur von CorenNLP beschrieben. CoreNLP verwendet das in 2.2 vorgestellte Pipeline-Modell. Die verschiedenen Verarbeitungsstufen der Pipeline werden Annotatoren genannt. Da die genaue Funktionsweise der Annotatoren ist für diese Arbeit weniger relevant, wichtiger ist ein Überblick über die Art der Ergebnisse, die die Annotatoren liefern.

Tokenization und Lemmatization Liefern, wie erwartet, eine Liste von Token bzw. eine Liste der Lemmata der Token. Es werden neben Englisch zahlreiche andere Sprachen und ein Großteil des Unicode Zeichensatzes unterstützt. Der Tokenizer verwendet zum Finden der Tokens intern einen deterministischen endlichen Automaten.

POS-Tagging Ordnet jedem Token eine Wortart und Flexion (POS-Tag) zu. Die Menge der möglichen POS-Tags wurde aus dem *Penn Treebank Tag Set* [Atw] übernommen. Für das Finden der Tags benutzt CoreNLP sog. *Cyclic Dependency Networks* [Tou+03], eine Erweiterung bayesscher Netze, in denen zyklische Abhängigkeiten erlaubt sind.

Named Entity Recognition (NER) Findet sog. Entitäten. CoreNLP benutzt hierfür eine Menge von Entitätsklassen, die sich in drei Kategorien von Klassen unterteilen lässt:

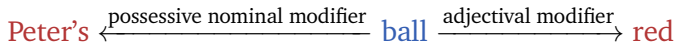
1. **Benannte Entitäten:** Person, Ort, Organisation und Sonstige. Diese Entitätsklassen werden mittels *Conditional Random Fields* [Fin+05], einer Variante von *Markov Random Fields* (siehe 3.3.1), erkannt.

2. **Numerische Entitäten:** Geldbetrag, Zahl, Ordinalzahl und Prozentzahl. Hierfür wird ein regelbasiertes System verwendet. Die so erkannten Token werden zudem normalisiert, um eine leichtere Weiterverarbeitung zu ermöglichen.

3. **Temporale Entitäten:** Datum, Uhrzeit, Dauer und Menge von Zeitpunkten. Diese Klassen werden ebenfalls mit einem regelbasierten System erkannt. Mittels SUTime [CM12] werden die erkannten Token anschließend normalisiert und relative Zeitangaben in absolute Zeitpunkte aufgelöst, sofern ein Referenzzeitpunkt gegeben ist. Für die Normalisierung wird das TimeML TIMEX3-Format [Tim] benutzt, mit dem sich auch komplexe Zeitangaben, wie *"twice a week"* (`type="set" value="P1W" freq="2X"`), formal ausdrücken lassen.

Coreference Resolution Ermittelt Äquivalenzklassen von Token, die auf das selbe Konzept bzw. die selbe Entität verweisen. CoreNLP stellt hierfür drei verschiedene Systeme bereit: Ein schnelles, regelbasiertes, deterministisches System, ein etwas langsameres statistisches System und zuletzt ein langsames System, das auf neuronalen Netzen basiert. Die langsameren Systeme liefern im Schnitt bessere Ergebnisse.

Dependency Parsing Dieser Annotator ermittelt die grammatikalischen Beziehungen zwischen Worten. Das Ergebnis ist ein sog. Abhängigkeitsgraph (*Dependency Graph*), in dem die Knoten Token und die Kanten Beziehungen repräsentieren. CoreNLP verwendet für die Kantentypen *Universal Dependencies Version 2* (UD v2) [Udv], eine Menge von 37 Arten grammatikalischer Beziehungen, die für eine vielzahl natürlicher Sprachen nutzbar ist. Die Struktur der zurückgegebenen Abhängigkeitsgraphen, basiert auf einem "*head*-*modifier*"-Pattern; d. h. dass, ausgehend von einem *head*-Token, Kanten zu *modifier*-Token gehen, die die Bedeutug des *heads* verändern.



Der CoreNLP Dependency Parser nutzt ein sog. *Transition-based Parsing* [Niv04], bei dem alle Token der Reihe nach aus einem Buffer auf einen Stack von aktuell betrachteten Token gelegt werden. Ein Klassifikator (im Falle von CoreNLP ist dies ein neuronales Netz) wählt dabei in jedem Schritt einen von drei Zustandsübergängen:

1. **LEFT-ARC:** Fügt eine Abhängigkeitskante (i, j) vom ersten Token i des Stacks zum zweiten Token j des Stacks ein und entfernt dann j vom Stack.

2. **RIGHT-ARC:** Fügt eine Abhängigkeitskante (j, i) vom zweiten Token j des Stacks zum ersten Token i des Stacks ein und entfernt dann i vom Stack.

3. **SHIFT:** Verschiebt das erste Token des Buffers auf den Stack.

Diese drei Zustandsübergänge werden so lange angewandt, bis der Buffer leer ist. Durch die richtige Kombination von Übergängen lässt sich jeder beliebige Abhängigkeitsgraph beschreiben.

3.3 Modellierung von HL-MRFs mit PSL

In 3.1 wurde beschrieben, wie komplexes Wissen durch Konzeptgraphen repräsentiert werden kann; in 3.2 wurde beschrieben, wie der Inhalt natürlichsprachlicher Texte extrahiert und durch eine Menge von Abhängigkeiten repräsentiert werden kann. Dieser Abschnitt beschreibt nun, wie aus einer Menge gegebenener Abhängigkeiten und Fakten neue Abhängigkeiten und Fakten inferiert werden können. Konkrekt werden hierfür *Hinge-Loss Markov Random Fields* (HL-MRFs) und die *Probabilistic Soft Logic* (PSL) vorgestellt.

3.3.1 Markov Random Fields

MRFs sind, so wie auch bayessche Netze, eine Klasse von *Probabilistischen Graphischen Modellen* (PGM); d. h. sie sind Graphen, deren Knoten als Zufallsvariablen und deren Kanten als stochastische Abhängigkeiten interpretiert werden. Im Gegensatz zu bayesschen Netzen, sind die Kanten in MRFs allerdings ungerichtet, es sind also zyklische Abhängigkeiten erlaubt. Formal beschreibt ein MRF G die multivariate Verteilung P eines Zufallsvektors X gemäß einer Potentialfunktion Φ:

$$
\begin{aligned}
X &:= (X_1, \ldots, X_n) = \text{Zufallsvektor} \\
\mathcal{X} &:= \text{Menge aller möglichen Werte } (x_1, \ldots, x_n) \text{ von } X \\
G &:= (X, E) \\
\mathcal{C} &:= \{c_1, \ldots, c_m\} = \text{Menge der maximalen Cliquen in } G \\
X_c &:= \text{Vektor der Zufallsvariablen in der Clique } c \in \mathcal{C} \\
\Phi_c(x_c) &:= \text{Cliquenpotential} \in \mathbb{R}_0^+ \text{ der Werte } x_c \text{ von } X_c \\
P(X = x) &= \frac{1}{Z} \prod_{i=1}^{m} \Phi_{c_i}(x_{c_i}), \text{ mit Normalisierkonst. } Z := \sum_{x \in \mathcal{X}} \prod_{i=1}^{m} \Phi_{c_i}(x_{c_i}) \qquad (3.9)
\end{aligned}
$$

Für MRFs gelten drei wichtige Eigenschaften bzgl. der Unabhängigkeit der Zufallsvariablen in X:

1. **Globale Markov-Eigenschaft:**

   $$\forall X_A, X_B, X_S \subseteq X : sep_{X_A, X_B}(X_S) \rightarrow (X_A \perp X_B \mid X_S)$$

   Alle Paare (X_A, X_B) von Teilmengen von X sind bedingt unabhängig, sofern die Werte einer separierenden Teilmenge X_S gegeben sind. X_S ist separierend ($\Leftrightarrow sep_{X_A, X_B}(X_S)$), wenn alle Pfade von $a \in X_A$ nach $b \in X_B$ einen Knoten $s \in X_S$ enthalten.

2. **Lokale Markov-Eigenschaft:**

$$\forall X_i \in X : X_i \perp (X \setminus \Gamma(X_i) \setminus \{X_i\}) \mid \Gamma(X_i)$$

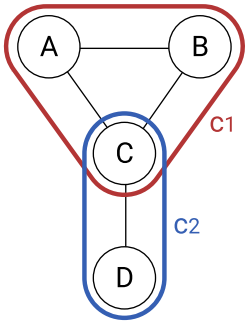
Eine direkte Konsequenz der globalen Markov-Eigenschaft ist die lokale Markov-Eigenschaft. Jede Variable X_i ist bedingt unabhängig von ihren nicht benachbarten Variablen, sofern ihre Nachbarschaft $\Gamma(X_i)$ gegeben ist.

3. **Paarweise Markov-Eigenschaft:**

$$\forall X_i, X_j \in X : \{X_i, X_j\} \notin E \rightarrow (X_i \perp X_j \mid X \setminus \{X_i, X_j\})$$

Aus der lokalen Markov-Eigenschaft folgt, dass jedes nicht adjazente Variablenpaar (X_i, X_j) bedingt unabhängig voneinander ist, sofern alle anderen Variablen gegeben sind.

Beispiel Die obigen Definitionen sind bislang noch recht abstrakt. Ein exemplarisches praktisches Einsatzgebiet von MRFs ist das Lösen von SAT-Problemen. Gegeben sei die SAT-Instanz $(\neg A \vee B \vee C) \wedge (\neg C \vee \neg D)$. MRFs können benutzt werden, um dieses Problem zu modellieren und eine erfüllende Belegung zu finden.



$$\begin{aligned} X &:= (A, B, C, D), \text{ mit } Bild(X) = \{0,1\}^4 \\ \mathcal{C} &:= \{\underbrace{\{A, B, C\}}_{c_1}, \underbrace{\{C, D\}}_{c_2}\} \\ \Phi_{c_1}(a, b, c) &:= \min\{1 - a + b + c, 1\} \\ \Phi_{c_2}(c, d) &:= \min\{2 - c - d, 1\} \\ P(X = (a, b, c, d)) &:= \frac{1}{Z} \Phi_{c_1}(a, b, c)\ \Phi_{c_2}(c, d) \end{aligned}$$

Eine Clique repräsentiert in diesem MRF eine Disjunktionsklausel und das Cliquenpotential gibt an, ob eine gegebene Belegung die Klausel erfüllt. Gemäß dieser Definition, lässt sich die Erfüllbarkeit durch $\max_x P(X = x) > 0$ ausdrücken, d. h. die Formel ist erfüllbar, gdw. es eine Variablenbelegung mit Eintrittswahrscheinlichkeit > 0 gibt. Die Normalisierungskonstante Z hat auf das Ergebnis keinen Einfluss und kann daher ignoriert werden.

Das MRF-Inferenzproblem Da sich SAT, wie soeben exemplarisch gezeigt, auf MRFs reduzieren lässt, ist das Inferenzproblem, d. h. das Finden einer maximal wahrscheinlichen Belegung der Zufallsvariablen, ein NP-schweres Problem. Allgemeine, exakte und effiziente Lösungsalgorithmen existieren daher nicht. Durch Einschränken der Struktur von G und Φ, oder durch das erlauben von Approximationen, lassen sich

MRF-Inferenzen jedoch deutlich effizienter finden. Wenn z. B. G ein Baum ist, kann mit dem *Belief Propagation* Algorithmus eine exakte Lösung in polynomieller Zeit gefunden werden.

3.3.2 Hinge-Loss MRFs

Eine Unterart von MRFs, sind die sog. Hinge-Loss MRFs. Sie sind so strukturiert, dass sich das Inferenzproblem effizient und exakt durch konvexe Optimierungsverfahren lösen lässt. Es gibt drei wesentliche Unterschiede zu allgemeinen MRFs:

1. Die Bedeutung von Zufallsvariablen und Kanten zwischen Zufallsvariablen ist klar definiert, da Φ nicht mehr frei wählbar ist. Ähnlich zum Beispiel aus 3.3.1, repräsentieren Zufallsvariablen aussagenlogische Variablen und Cliquen Disjunktionsklauseln.

2. Für den Zufallsvektor X muss $Bild(X) = [0,1]^n$ gelten. Diese Einschränkung besteht, da jede HL-MRF-Zufallsvariable als die Wahrscheinlichkeit, dass eine aussagenlogische Variable wahr ist, interpretiert wird.

3. Die Definition der Verteilung P ist etwas anders:

$$P(X = x) := \frac{1}{Z} \prod_{i=1}^{m} e^{w_i \, \Phi_i(x)} \propto \exp\left(\sum_{i=1}^{m} w_i \, \Phi_i(x)\right) = \exp\left(w \, \Phi(x)^\top\right) \quad (3.10)$$
$$w := (w_1, \ldots, w_m) \in [0, \infty]^m, \; \Phi := (\Phi_1, \ldots, \Phi_m)$$

   Auf die Cliquenpotentiale Φ_i wird nun die Exponentialfunktion angewandt, zudem erhalten alle Cliquen bzw. Klauseln c_i ein Gewicht w_i. Das Inferenzproblem $\arg\max_x P(X = x)$ ist somit äquivalent zu $\arg\max_x w \, \Phi(x)^\top$.

KNF-Formel Interpretation Aufgrund der Einschränkung von HL-MRFs auf aussagenlogische Ausdrücke, wird im Folgenden die Graphterminologie fallen gelassen und stattdessen die entsprechende aussagenlogische Terminologie verwendet. Ein HL-MRF wird nun als Repräsentation einer KNF-Formel $C_1 \wedge \cdots \wedge C_m$ interpretiert. Jede Disjunktionsklausel $C_j \in C$ wird durch eine Menge von Variablenindizes positiver Atome $I_j^+ \subseteq \{1, \ldots, n\}$ und eine Menge von Variablenindizes negativer Atome $I_j^- \subseteq \{1, \ldots, n\}$ beschrieben.

$$C_j \cong \left(\bigvee_{i \in I_j^+} X_i\right) \vee \left(\bigvee_{i \in I_j^-} \neg X_i\right)$$

Łukasiewicz Logik Da die Variablen der KNF-Formel, gemäß obiger Definition, Werte $\in [0,1]$ annehmen können, ist nun noch unklar, wie die Operatoren $\wedge$, $\vee$ und $\neg$ funktionieren sollen. HL-MRFs benutzen hierfür die sog. Łukasiewicz Logik aus der Klasse der T-Norm Fuzzy Logiken; sie ist eine Erweiterung der booleschen Logik, d. h. die Łukasiewicz Operatoren verhalten sich für die Extrema 0 und 1 so, wie die booleschen Operatoren, sind aber ebenfalls für alle dazwischen liegenden Eingabewerte definiert.

$$x_1 \wedge x_2 := \max\{x_1 + x_2 - 1, 0\} \tag{3.11}$$

$$x_1 \vee x_2 := \min\{x_1 + x_2, 1\} \tag{3.12}$$

$$\neg x := 1 - x \tag{3.13}$$

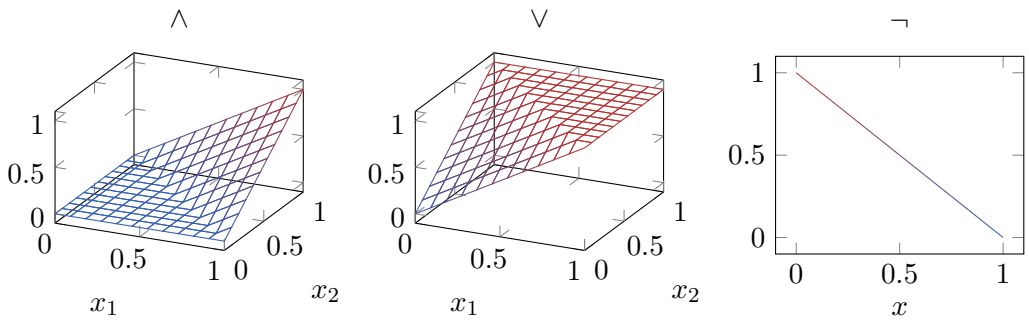


Abb. 3.3. Visualisierung der Łukasiewicz Operatoren $\wedge$, $\vee$ und $\neg$.

Mittels der Łukasiewicz Logik können Disjuktionsklauseln $C_j \in C$ für eine gegebene Variablenbelegung x nun Wahrheitswerte $\in [0,1]$ zugeordnet werden. Dieser Wahrheitswert wird als Klauselpotential $\Phi_j(x)$ verwendet. Das HL-MRF-Inferenzproblem für KNF-Formeln in Łukasiewicz Logik ist demnach

$$\begin{aligned}
&\arg\max_{x\in[0,1]^n} \sum_{C_j \in C} w_j \quad \Phi_j(x) \\
&= \arg\max_{x\in[0,1]^n} \sum_{C_j \in C} w_j \quad \left(\left(\bigvee_{i \in I_j^+} x_i \right) \vee \left(\bigvee_{i \in I_j^-} \neg x_i \right) \right) \\
&= \arg\max_{x\in[0,1]^n} \sum_{C_j \in C} w_j \min \left\{ \left(\sum_{i \in I_j^+} x_i \right) + \left(\sum_{i \in I_j^-} (1 - x_i) \right), 1 \right\}
\end{aligned} \tag{3.14}$$

Statt die Summe der Wahrheitswerte $\Phi_j(x)$ zu maximieren, kann alternativ auch die Summe der Distanzen zur Erfüllung $\ell_j(x)$, genannt *Distance to Satisfaction*, minimiert werden; es gilt $\ell_j(x) = 1 - \Phi_j(x)$. Gemäß dieser Interpretation ist ein HL-MRF somit

eine Menge von gewichteten Constraints $\ell_j(x) \leq 0$, für die eine Lösung mit möglichst wenigen Verletzungen gesucht wird. Das Inferenzproblem ist also

$$\begin{aligned} &\arg\min_{x\in[0,1]^n} \sum_{C_j\in C} w_j \max\{\ell_j(x), 0\} \\ &= \arg\min_{x\in[0,1]^n} \sum_{C_j\in C} w_j \max\left\{1 - \left(\sum_{i\in I_j^+} x_i\right) - \left(\sum_{i\in I_j^-} (1-x_i)\right), 0\right\} \end{aligned} \tag{3.15}$$

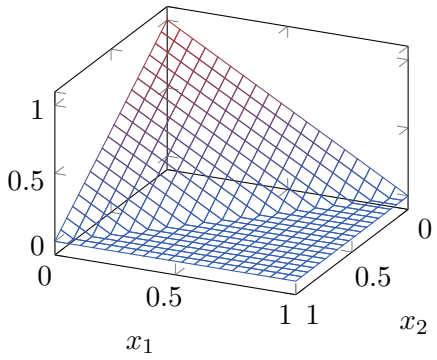


Abb. 3.4. Visualisierung der Loss-Funktion $\ell_j(x_1, x_2)$ für $C_j \cong X_1 \vee X_2$

Wie Abbildung 3.4 für den zwei-elementigen Klauselfall veranschaulicht, handelt es sich bei ℓ um eine Hinge-Loss Funktion. Hierher rührt die Bezeichnung Hinge-Loss MRF. Da Hinge-Loss Funktionen konvex sind und die Summe konvexer Funktionen ebenfalls konvex ist, handelt es sich bei der HL-MRF-Inferenz um ein konvexes Optimierungsproblem. Es existieren also effiziente und exakte Lösungsalgorithmen. Einer dieser Algorithmen ist das *Alternating Direction Method of Multipliers* Verfahren (ADMM), es wird in 3.3.4 näher vorgestellt.

MAX-SAT Äquivalenz In 3.3.1 wurden MRFs anhand des Beispiels der SAT-Instanz $(\neg A \vee B \vee C) \wedge (\neg C \vee \neg D)$ veranschaulicht. Diese KNF-Formel hat folgendes Inferenzproblem, wenn sie als HL-MRF repräsentiert wird:

$$\arg\min_{(a,b,c,d)\in[0,1]^4} w_1 \max\{a - b - c, 0\} + w_2 \max\{c + d - 1, 0\}$$

Da in der Verteilung P von HL-MRFs die Exponentialfunktion auf die Potentiale angewandt wird, bewirkt eine unerfüllte Klausel mit $\Phi_j(x) = 0$ nicht, dass $P(X = x) = 0$. Stattdessen ist $P(X = x)$ proportional zu der Summe der Wahrheitswerte der Klauseln. Die HL-MRF-Inferenz beschreibt also nicht SAT, sondern eine Fuzzy-Logik-Entsprechung von MAX-SAT. Ein wichtiger Unterschied zum booleschen MAX-SAT ist, dass Klauseln gewichtet sind; das Erfüllen einer Klausel mit hohem Gewicht kann das Nicht-Erfüllen mehrerer anderer Klauseln mit niedrigem Gewicht ausgleichen.

3.3.3 Probabilistic Soft Logic

Wie soeben beschrieben, sind HL-MRFs ein flexibles Werkzeug, um Probleme, die sich durch MAX-SAT ausdrücken lassen, zu lösen. Der Schritt von einem konkreten domänenspezifischen Problem in eine Menge von Klauseln C und einen Gewichtsvektor w ist bislang allerdings noch unklar. An dieser Stelle setzt die *Probabilistic Soft Logic* (PSL) an. PSL ist eine formale Sprache, um mit einer intuitiven Syntax Klassen von HL-MRFs zu beschreiben.

PSL Syntax

Die Syntax von PSL ist an die Prädikatenlogik angelehnt und besteht aus sieben Arten von Elementen:

1. **Konstanten:** Repräsentieren konstante Werte, wie z. B. Strings oder Zahlen. Werden für domänenspezifische Daten, wie z. B. Namen benutzt.

2. **Variablen:** Werden während einer Inferenz mit Konstanten belegt. PSL Variablen sind nicht zu verwechseln mit den Zufallsvariablen in MRFs.

3. **Terme:** Ein Term ist entweder eine Konstante oder eine Variable.

4. **Prädikate:** Entsprechen in etwa den prädikatenlogischen Prädikaten. Jedes PSL Prädikat hat einen eindeutigen Bezeichner und eine Signatur aus Konstantentypen.
$$Person \subseteq \text{UUID}, \quad Name \subseteq \text{UUID} \times \text{String}$$

5. **Atome:** Ein Atom ist ein Prädikat der Arität n, kombiniert mit einem n-Tupel von Termen. Das Term-Tupel enthält die Argumente des Prädikates. Wenn alle Argumente Konstanten sind, spricht man von einem Grundatom (*ground atom*).
$$Person(x), \quad Name(x, \text{“Alice”})$$

6. **Literale:** Ein Literal ist entweder ein Atom oder ein negiertes Atom.
$$Name(x, \text{“Alice”}), \quad \neg Name(x, \text{“Bob”})$$

7. **Regeln:** Eine Regel ist eine gewichtete Disjunktionsklausel von Literalen. Die negativen Atome der Klausel bilden dabei den sog. Körper B (*body*), die positiven Atome den sog. Kopf H (*head*) der Regel. Die so zerlegte Disjunktionsklausel lässt sich als Implikationsregel interpretieren:

$$\left(\bigvee_{b \in B} \neg b\right) \vee \left(\bigvee_{h \in H} h\right) \Leftrightarrow \left(\bigwedge_{b \in B} b\right) \rightarrow \left(\bigvee_{h \in H} h\right)$$

   Mit Implikationen lassen sich nun intuitiv Zusammenhänge zwischen Prädikaten modellieren.

$$0.65 : Person(x) \wedge Name(x, \text{“Alice”}) \rightarrow Female(x)$$

Eine Menge von PSL-Regeln wird PSL-Programm genannt. Als Input erwartet ein PSL-Programm R eine sog. Basis $\mathcal{A}$. Die Basis ist dabei eine Menge von Grundatomen, die während der Inferenz in Betracht gezogen werden sollen, und setzt sich aus zwei disjunkten Teilmengen $\mathbb{C} \dot{\cup} \mathbb{O} = \mathcal{A}$ zusammen. $\mathbb{C}$ ist die Menge der geschlossenen (*closed*) Grundatome, d. h. der Wahrheitswert $\in [0, 1]$ dieser Atome ist bekannt. $\mathbb{O}$ umfasst die offenen (*open*) Grundatome, deren Wahrheitswert noch unbekannt ist und daher inferiert werden soll.

Beispiel Ein Anwendungsgebiet von PSL ist z. B. die Modellierung sozialer Beziehungen. Angenommen, es sollen Freundschaftsbeziehungen zwischen Personen inferiert werden. Ein stark vereinfachtes PSL-Programm hierfür könnte so aussehen:

$$\begin{aligned}
0.4 &: \neg Friends(A, B) && (r_1)\\
0.5 &: Friends(A, B) \wedge Friends(B, C) \wedge A \neq C \rightarrow Friends(A, C) && (r_2)\\
1 &: Interest(A, X) \wedge Interest(B, X) \rightarrow Friends(A, B) && (r_3)\\
&\quad Friends(A, B) \rightarrow Friends(B, A) && (r_4)\\
&\quad \neg Friends(A, A) && (r_5)
\end{aligned}$$

In dieser Regelmenge sind fünf Annahmen über das Verhalten der `Friends`-Relation kodiert:

1. r_1 bildet ab, dass zwei Personen a priori nicht miteinander befreundet sind. Eine solche PSL-Regel, die, in Ermanglung weiteren Wissens, das Nichtvorhandensein einer Relation postuliert, wird *Prior* genannt.

2. r_2 bildet die Transitivität der $Friends$-Relation ab. Zwei Personen mit einem gemeinsamen Freund, sind evtl. befreundet. Das Infix-Prädikat $\neq$ ist üblicherweise vordefiniert.

3. r_3 bildet ab, dass Personen mit einem gemeinsamen Interesse eine höhere Wahrscheinlichkeit haben befreundet zu sein.

4. r_4 bildet die Symmetrie der $Friends$-Relation ab. Im Gegensatz zu den anderen Regeln, ist r_4 ein sog. *Constraint*, d. h. sie hat ein implizites Gewicht $w_4 = \infty$. Das Nichterfüllen von r_4 hat somit zur Folge, dass die gewichtete Distance to Satisfaction ebenfalls den Wert ∞ annimmt und durch die Erfüllung anderer Regeln nicht ausgeglichen werden kann. Es ist also garantiert, dass jede MAX-SAT Lösung r_4 einhält; d. h. $Friends(A, B) = Friends(B, A)$.

5. r_5 ist ein weiterer Constraint, der die Irreflexivität von $Friends$ erzwingt.

Für das PSL-Programm $R = \{r_1, \ldots, r_5\}$ sei nun folgender Input $\mathcal{A}$ gegeben:

$$\mathbb{C} = \left\{ \begin{array}{ll} Interest(\text{“Alice”}, \text{“Reading”}) = 0.5, & Interest(\text{“Dave”}, \text{“Reading”}) = 1, \\ Interest(\text{“Alice”}, \text{“Skiing”}) = 0.5, & Interest(\text{“Charlie”}, \text{“Skiing”}) = 1, \\ Interest(\text{“Bob”}, \text{“Tennis”}) = 1, & Interest(\text{“Charlie”}, \text{“Tennis”}) = 1, \\ Friends(\text{“Alice”}, \text{“Bob”}) = 1 & \end{array} \right\}$$

$$\mathbb{O} = \{Friends(x, y) = \ ? : x, y \in \{\text{“Alice”}, \text{“Bob”}, \text{“Charlie”}, \text{“Dave”}\}\} \setminus \mathbb{C}$$

Die Interessen der Personen und die Freundschaft zwischen Bob und Alice aus $\mathbb{C}$ werden als bekannt angenommen. Die gesuchten Freundschaftsrelationen aus $\mathbb{O}$ sollen inferiert werden. Abbildung 3.5 zeigt ein mögliches MAX-SAT Ergebnis

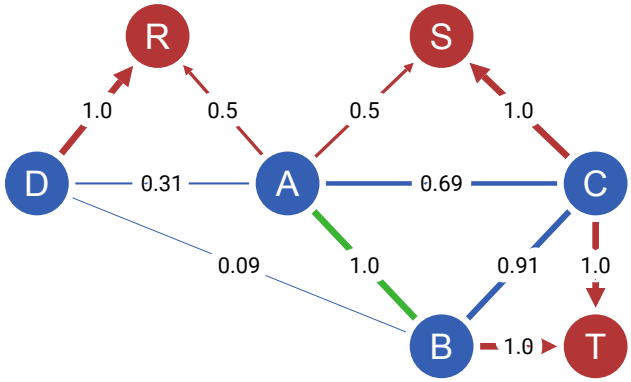


Abb. 3.5. Graph der inferierten $Friends$-Relation und der gegebenen $Interest$-Relation.

der Inferenz. Wegen des Priors r_1 werden Personen generell als nicht befreundet eingeordnet. Personen mit gemeinsamen Interessen werden hingegen, aufgrund von r_3, als Freunde erkannt. Die durch r_2 modellierte Transitivitäteigenschaft ist ein weiterer verstärkender Faktor. Dies lässt sich in der Clique *Alice-Bob-Charlie* beobachten, die $Friends$-Beziehungen haben sich dort gegenseitig verstärkt.

PSL $\rightarrow$ HL-MRF Übersetzung

Wie soeben gezeigt, ist PSL ein intuitiver Formalismus zur Definition relationaler Modelle. Um Inferenzen durchzuführen, ist allerdings zuerst eine Übersetzung in ein HL-MRF notwendig. Gegeben ist hierbei immer ein PSL-Programm $R = \{r_1, \ldots, r_k\}$ und eine Eingabe $\mathcal{A} = \mathbb{C} \dot{\cup} \mathbb{O}$. Die Übersetzung erfolgt in zwei Schritten:

1. **Zufallsvariablen:** Die Grundatome aus $\mathbb{C}$ werden zu Zufallsvariablen X, deren Belegung x gegeben ist. Die Grundatome aus $\mathbb{O}$ werden zu Zufallsvariablen Y, für die die optimale Belegung $y \in \mathcal{Y}$ aus der Menge aller möglichen Belegungen $\mathcal{Y}$ gesucht wird. Der Zufallsvektor des HL-MRFs ist also $X \cup Y$.

2. **Disjunktionsklauseln:** In der bisherigen HL-MRF Definition wurde stets davon ausgegangen, dass die optimale Belegung *aller* Zufallsvariablen gesucht ist. Im Falle von PSL ist allerdings bereits bekannt, dass $X = x$ gilt. Das Inferenzproblem lautet daher wie folgt:

$$\begin{aligned} \arg\max_{y \in \mathcal{Y}} P(Y = y \mid X = x) &= \arg\max_{y \in \mathcal{Y}} P(X \cup Y = x \cup y) \\ &= \arg\max_{y \in \mathcal{Y}} w\, \Phi(y, x)^\top \end{aligned} \tag{3.16}$$

   Um ein HL-MRF zu erhalten, muss nun Φ und w, d. h. die Menge der Klauseln C und ihre Gewichte, definiert werden. Die PSL-Regeln in R sind zwar Klauseln, können aber nicht direkt als C verwendet werden, da sie potentiell freie PSL-Variablen enthalten, die PSL-Atome also potentiell nicht in $\mathcal{A}$ sind. Es erfolgt daher ein sog. *Grounding* der PSL-Regeln. In jede Regel aus R wird dabei jede mögliche Belegung der PSL-Variablen eingesetzt, sodass die resultierende Regelinstanz nur Atome aus $\mathcal{A}$ enthält. Die Menge dieser Regelinstanzen wird Grundregeln (*ground rules*) genannt und als Klauselmenge C benutzt. Aus C wiederum lässt sich Φ nun mittels der Łukasiewicz Logik eindeutig ableiten. Das Gewicht jedes Potentials Φ_i ist dabei gleich dem Gewicht der PSL-Regel aus der es resultierte.

3.3.4 Inferenzverfahren

In Abschnitt 3.3.2 wurde die Konvexität des HL-MRF-Inferenzproblems bereits diskutiert. Prinzipiell kann daher jedes konvexe Optimierungsverfahren im Kontext von HL-MRFs verwendet werden. In der Praxis hat sich jedoch ein Verfahren, als besonders effizient erwiesen.

ADMM Das sog. *Alternating Direction Method of Multipliers* Verfahren (ADMM) ist eine Variante des *Method of Multipliers* Verfahrens, in der die dualen Variablen partiell aktualisiert werden. Durch diese Variation lässt sich ADMM gut parallelisieren und ist somit geeignet für große Datenmengen und den Einsatz in Cluster-Umgebungen. Das ursprüngliche Paper [Boy+11, Kapitel 10] beschreibt explizit, wie sich ADMM mit verteilten Datensets und verteilten Programmiermodellen wie z. B. *Map-Reduce* oder *Pregel* implementieren lässt.

Die ADMM-Laufzeit ist proportional zur Klauselanzahl $|C|$. Die Klauselanzahl wiederum wächst gemäß $\mathcal{O}(|\mathcal{A}|^r)$, wobei r die maximale Anzahl von Nicht-Grund-Atomen in einer PSL-Regel ist. Insgesamt wächst die Inferenzdauer also polynomiell in Abhängigkeit zur Eingabe $\mathcal{A}$.

BOCI Um effiziente Updates eines gegebenen Inferenzergebnises für eine veränderte Eingabe $\mathcal{A}'$ zu ermöglichen, wurde das *Budgeted Online Collective Inference* (BOCI) Verfahren entwickelt. Wenn ein Inferenzergebnis für $\mathcal{A} = \mathbb{C} \dot{\cup} \mathbb{O}$ existiert und anschließend die Wahrheitswerte für einen Teil der bislang offenen Atome aus $\mathbb{O}$ bekannt werden, muss mittels BOCI keine weitere vollständige Inferenz durchgeführt werden.

Die Grundidee dabei ist es Metadaten, die während der letzten ADMM-Inferenz angefallen sind, für eine Bewertung jedes Atoms zu benutzen. Die Bewertungen spiegeln die Volatilität der Wahrheitswerte der Atome, in Abhängigkeit von $\mathbb{C}$, wider. Welche Metadaten für die Bewertung verwendet werden sollten, hängt stark von der Semantik der Wahrheitswerte der PSL-Prädikate ab. In [Puj+15, Kapitel 4] werden verschiedene mögliche Bewertungsverfahren beschrieben, für diese Arbeit sind jene allerdings nicht näher relevant.

Gegeben sei also eine Bewertung der Atome aus $\mathcal{A}$. Werden nun die Wahrheitswerte bislang offener Atome aus $\mathbb{O}$ bekannt, müssen ausschließlich die m höchstbewerteten Atome aus $\mathbb{C}$ neu inferiert werden, die Wahrheitswerte aller anderen Atome werden fixiert. Je höher das sog. Budget m, desto höher ist die Qualität im Vergleich zu einer vollständigen Inferenz. Es wurde empirisch gezeigt, dass die Inferenzqualität mit BOCI gegenüber einer vollständigen Inferenz meist nur unwesentlich schlechter ist, während sich die Inferenzdauer teils um über 60% verringert [Puj+15, Kapitel 5].

4 Vorgeschlagenes Wissensgraph-konstruktionsverfahren

Auf Basis der vorgestellten Konzeptgraphen, CoreNLP und PSL wird im folgenden Kapitel ein Verfahren für die online Konstruktion eines Wissensgraphen aus natürlichsprachlichen Textnachrichten aufgebaut. Der Fokus liegt dabei primär auf der generellen Architektur des Verfahrens. Das Resultat ist also als Proof-of-Concept zu verstehen, auf dessen Basis praxistaugliche Systeme konzipiert werden können.

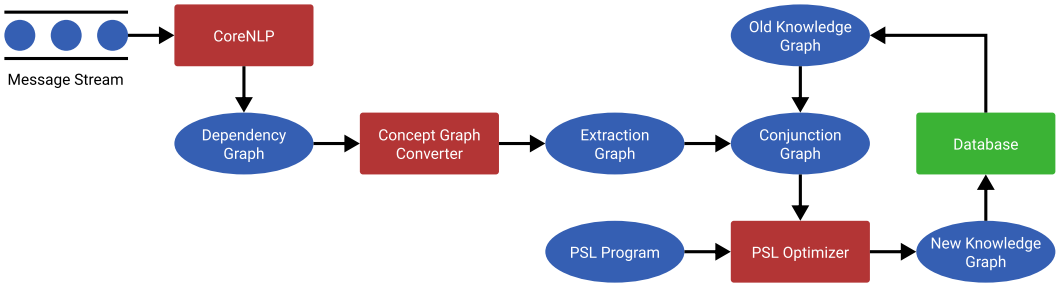


Abb. 4.1. Grobes Architekturdiagramm des Konstruktionsverfahrens

Das im Folgenden vorgestellte Verfahren folgt einem dreistufigen Pipeline-Modell:

1. Mittels CoreNLP wird eine eintreffende Textnachricht in einen Abhängigkeitsgraphen transformiert.

2. Der resultierende Abhängigkeitsgraph wird in einen sog. Extraktionsgraphen umgewandelt. Hierbei handelt es sich um einen Konzeptgraphen, der den Inhalt der Nachricht formal repräsentiert.

3. Der Extraktionsgraph wird mit dem bestehenden Wissensgraphen verschmolzen. Dies entspricht der Konjuktion der durch die beiden Graphen repräsentierten logischen Ausdrücke. Der resultierende Konjunktionsgraph wird als Eingabe für ein PSL-Programm verwendet, welches auf Basis des hinzugekommenen Wissens neue Beziehungen im Wissensgraphen inferiert.

Die Beschreibung dieser Pipeline erfolgt in vier Abschnitten. In Abschnitt 4.1 wird kurz die technische Umsetzung erläutert. Abschnitt 4.2 beschreibt anschließend die Ontologie der konstruierten Wissensgraphen. Nach diesen strukturellen Betrachtun-

gen wird schließlich in Abschnitt 4.3 und Abschnitt 4.4 die Transformation von Text zu Extraktionsgraph, bzw. von Extraktionsgraph zu Wissensgraph beschrieben.

4.1 Implementation

Das in den folgenden Abschnitten beschriebene Verfahren wurde im Rahmen dieser Arbeit prototypisch implementiert. Bei der Wahl der hierfür verwandten Technologien wurde darauf Wert gelegt, dass eine möglichst nahtlose Integration der Komponenten möglich ist. Sowohl CoreNLP [Cor] als auch die PSL-Referenzimplementation [Psl] sind JVM-Bibliotheken. Diese Arbeit wurde daher ebenfalls in einer JVM-Sprache implementiert.

Hierfür wurde Clojure gewählt, ein moderner Lisp-1-Dialekt, mit einem Fokus auf funktionale Programmierung, unveränderliche Datenstrukturen und gleichzeitiger Interoperabilität mit objektorientierten Bibliotheken. Andere JVM-Sprachen, wie z. B. Java, Scala oder Groovy, wurden ausgeschlossen, da sie sich während des Entwicklungsprozesses als hinderlich erwiesen haben. Der Hauptgrund hierfür ist, dass CoreNLP bei der Initialisierung diverse Modelle laden muss. Bei Verwendung eines modernen Desktop-Rechners benötigt dies ca. 20 Sekunden, auf langsamerer Hardware teils mehrere Minuten; diese Wartezeiten waren ein stark verlangsamender Faktor beim entwickeln. Da Clojure ein Lisp ist, unterstützt es traditionsgemäß *REPL Driven Development*. Statt nach jeder Änderung die Anwendung neu zu starten und die Modelle erneut zu laden, kann so lediglich der geänderte Bytecode in den laufenden Prozess injiziert werden; die geladenen Modelle bleiben dabei im Speicher und die Änderung kann ohne weitere Wartezeit getestet werden. Durch die Wahl von Clojure konnte die Entwicklung deutlich beschleunigt werden.

4.2 Wissensgraphontologie

4.3 NLP-Phase

4.4 Graphkonstruktionsphase

5 Auswertung

5.1 Testmethode

5.2 Ergebnisse

6 Zusammenfassung

Anhang

A

Literatur

[Bac+13] Stephen H. Bach, Bert Huang, Ben London und Lise Getoor. „Hinge-loss Markov Random Fields: Convex Inference for Structured Prediction“. In: Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence. UAI'13. Bellevue, WA: AUAI Press, 2013, S. 32–41 (zitiert auf Seite 15).

[Bac+15] Stephen H. Bach, Matthias Broecheler, Bert Huang und Lise Getoor. „Hinge-Loss Markov Random Fields and Probabilistic Soft Logic“. In: (2015). arXiv: 1505.04406v2 [cs.LG] (zitiert auf Seite 15).

[Boy+11] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato und Jonathan Eckstein. „Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers“. In: Foundations and Trends in Machine Learning 3.1 (Jan. 2011), S. 1–122 (zitiert auf den Seiten 15, 32).

[Br10] Matthias Bröcheler, Lilyana Mihalkova und Lise Getoor. „Probabilistic Similarity Logic“. In: Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence. UAI'10. Catalina Island, CA: AUAI Press, 2010, S. 73–82 (zitiert auf Seite 15).

[Car+10] Andrew Carlson, Justin Betteridge, Bryan Kisiel et al. „Toward an Architecture for Never-ending Language Learning“. In: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence. AAAI'10. Atlanta, Georgia: AAAI Press, 2010, S. 1306–1313 (zitiert auf Seite 11).

[CM12] Angel X. Chang und Christopher D. Manning. „Sutime: A library for recognizing and normalizing time expressions.“ In: LREC. Bd. 2012. 2012, S. 3735–3740 (zitiert auf Seite 22).

[Dau03] Frithjof Dau. The Logic System of Concept Graphs with Negation. Springer Berlin Heidelberg, 2003 (zitiert auf den Seiten 17, 20).

[Fin+05] Jenny Rose Finkel, Trond Grenager und Christopher Manning. „Incorporating non-local information into information extraction systems by gibbs sampling“. In: Proceedings of the 43rd annual meeting on association for computational linguistics. Association for Computational Linguistics. 2005, S. 363–370 (zitiert auf Seite 21).

[Fre79] Gottlob Frege. Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens. L. Nebert, 1879 (zitiert auf Seite 7).

[Har+07] Frank van Harmelen, Vladimir Lifschitz und Bruce Porter. Handbook of Knowledge Representation. San Diego, USA: Elsevier Science, 2007, S. 213–237 (zitiert auf Seite 10).

[HR+83] Frederick Hayes-Roth, Donald A. Waterman und Douglas B. Lenat. Building Expert Systems. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1983, S. 6–7 (zitiert auf Seite 9).

[Lao+11] Ni Lao, Tom Mitchell und William W. Cohen. „Random Walk Inference and Learning in a Large Scale Knowledge Base“. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. EMNLP '11. Edinburgh, United Kingdom: Association for Computational Linguistics, 2011, S. 529–539 (zitiert auf Seite 15).

[Leh92] F. Lehmann. Semantic Networks in Artificial Intelligence. New York, NY, USA: Elsevier Science Inc., 1992, S. 6 (zitiert auf Seite 10).

[Man+14] Christopher D. Manning, Mihai Surdeanu, John Bauer et al. „The Stanford CoreNLP Natural Language Processing Toolkit“. In: Association for Computational Linguistics (ACL) System Demonstrations. 2014, S. 55–60 (zitiert auf Seite 13).

[MS03] Catherine C. Marshall und Frank M. Shipman. „Which Semantic Web?“ In: Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia. HYPERTEXT '03. Nottingham, UK: ACM, 2003, S. 57–66 (zitiert auf Seite 11).

[New+59] Allen Newell, John C. Shaw und Herbert A. Simon. „Report on a general problem solving program“. In: IFIP congress. Bd. 256. Pittsburgh, PA. 1959, S. 64 (zitiert auf Seite 9).

[Nic+14] Maximilian Nickel, Xueyan Jiang und Volker Tresp. „Reducing the Rank in Relational Factorization Models by Including Observable Patterns“. In: Advances in Neural Information Processing Systems 27. Hrsg. von Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence und K. Q. Weinberger. Curran Associates, Inc., 2014, S. 1179–1187 (zitiert auf Seite 15).

[Nic+16] Maximilian Nickel, Kevin Murphy, Volker Tresp und Evgeniy Gabrilovich. „A Review of Relational Machine Learning for Knowledge Graphs“. In: Bd. 104. 1. Institute of Electrical und Electronics Engineers (IEEE), Jan. 2016, S. 11–33 (zitiert auf Seite 14).

[Nic13] Maximilian Nickel. „Tensor Factorization for Relational Learning“. Diss. Ludwig-Maximilians-Universität München, 2013 (zitiert auf Seite 14).

[Niv04] Joakim Nivre. „Incrementality in deterministic dependency parsing“. In: Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together. Association for Computational Linguistics. 2004, S. 50–57 (zitiert auf Seite 22).

[NS56] Allen Newell und Herbert Simon. „The logic theory machine–A complex information processing system“. In: IRE Transactions on information theory 2.3 (1956), S. 61–79 (zitiert auf Seite 9).

[Pei85] C. S. Peirce. „On the Algebra of Logic: A Contribution to the Philosophy of Notation“. In: American Journal of Mathematics 7.2 (Jan. 1885), S. 180 (zitiert auf Seite 8).

[Puj+15] Jay Pujara, Ben London und Lise Getoor. „Budgeted Online Collective Inference“. In: Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence. UAI'15. Amsterdam, Netherlands: AUAI Press, 2015, S. 712–721 (zitiert auf den Seiten 15, 32).

[RM92] R. P. Van de Riet und R. A. Meersman. Linguistic Instruments in Knowledge Engineering. New York, NY, USA: Elsevier Science Inc., 1992 (zitiert auf Seite 10).

[Slu87] Hans Sluga. „Frege against the Booleans.“ In: Notre Dame Journal of Formal Logic 28.1 (Jan. 1987), S. 80–98 (zitiert auf Seite 9).

[Sow11] John F. Sowa. „Peirce's tutorial on existential graphs“. In: Semiotica 2011.186 (Jan. 2011) (zitiert auf Seite 8).

[Sow76] John F. Sowa. „Conceptual Graphs for a Data Base Interface“. In: IBM Journal of Research and Development 20.4 (Juli 1976), S. 336–357 (zitiert auf Seite 10).

[Tou+03] Kristina Toutanova, Dan Klein, Christopher D Manning und Yoram Singer. „Feature-rich part-of-speech tagging with a cyclic dependency network“. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1. Association for Computational Linguistics. 2003, S. 173–180 (zitiert auf Seite 21).

[WR10] Alfred North Whitehead und Bertrand Russell. Principia mathematica. 1910 (zitiert auf Seite 9).

Webseiten

[Atw] Eric Atwell. The University of Pennsylvania (Penn) Treebank Tag-set. Leeds University. URL: http://www.comp.leeds.ac.uk/amalgam/tagsets/upenn.html (besucht am 14. Sep. 2017) (zitiert auf Seite 21).

[BH] Dan Brickle und Ivan Herman. Semantic Web Interest Group. W3C. URL: https://www.w3.org/2001/sw/interest/ (besucht am 10. Sep. 2017) (zitiert auf Seite 11).

[Cor] Stanford CoreNLP. Stanford University. URL: https://stanfordnlp.github.io/CoreNLP (besucht am 7. Sep. 2017) (zitiert auf den Seiten 13, 21, 34).

[Goo] Google Trends "knowledge graph". Google. 2017. URL: https://trends.google.com/trends/explore?date=2004-01-01%202017-08-31&q=knowledge%20graph (besucht am 7. Sep. 2017) (zitiert auf Seite 12).

[McC+16] James P. McCusker, Deborah L. McGuinness, John S. Erickson und Katherine Chastain. What is a Knowledge Graph? 2016. URL: https://www.authorea.com/users/6341/articles/107281-what-is-a-knowledge-graph (besucht am 9. Sep. 2017) (zitiert auf Seite 11).

[Ope] Apache OpenNLP. URL: http://opennlp.apache.org/ (besucht am 7. Sep. 2017) (zitiert auf den Seiten 13, 21).

[Psl] PSL Referenzimplementation. LINQS. URL: https://github.com/linqs/psl (besucht am 7. Sep. 2017) (zitiert auf Seite 34).

[Sha06] Victoria Shannon. A 'more revolutionary' Web. 23. Mai 2006. URL: http://www.nytimes.com/2006/05/23/technology/23iht-web.html (besucht am 10. Sep. 2017) (zitiert auf Seite 11).

[Sin12] Amit Singhal. Introducing the Knowledge Graph: things, not strings. Google. 16. Mai 2012. URL: https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html (besucht am 7. Sep. 2017) (zitiert auf Seite 12).

[Tim] Guidelines for Temporal Expression Annotation for English. TimeML Working Group. 14. Aug. 2009. URL: http://www.timeml.org/tempeval2/tempeval2-trial/guidelines/timex3guidelines-072009.pdf (besucht am 18. Sep. 2017) (zitiert auf Seite 22).

[Udv] Universal Dependency Relations. Stanford University. URL: http://universaldependencies.org/u/dep/ (besucht am 15. Sep. 2017) (zitiert auf Seite 22).

[Usz] Hans Uszkoreit. Repräsentationen und Prozesse in der Sprachverarbeitung. Einführung in die Computerlinguistik. URL: http://www.coli.uni-saarland.de/~hansu/Verarbeitung.html (besucht am 13. Sep. 2017) (zitiert auf Seite 13).

[Wik] Wikipedia. Begriffsschrift. URL: https://de.wikipedia.org/wiki/Begriffsschrift (besucht am 7. Sep. 2017) (zitiert auf Seite 9).

[Wor] WordNet. A lexical database for English. Princeton University. URL: http://wordnet.princeton.edu (besucht am 10. Sep. 2017) (zitiert auf Seite 11).

Abbildungsverzeichnis

Tabellenverzeichnis

Erklärung zur Bachelorarbeit

Ich, Clemens Damke (Matrikel-Nr. 7011488), versichere, dass ich die Bachelorarbeit mit dem Thema *Probabilistische online Wissensgraphkonstruktion aus natürlicher Sprache* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen der Arbeit, die ich anderen Werken dem Wortlaut oder dem Sinn nach entnommen habe, wurden in jedem Fall unter Angabe der Quellen der Entlehnung kenntlich gemacht. Das Gleiche gilt auch für Tabellen, Skizzen, Zeichnungen, bildliche Darstellungen usw. Die Bachelorarbeit habe ich nicht, auch nicht auszugsweise, für eine andere abgeschlossene Prüfung angefertigt. Auf § 63 Abs. 5 HZG wird hingewiesen.

Paderborn, 25. September 2017

Clemens Damke