Generating and Applying Textual Entailment Graphs for Relation Extraction and Email Categorization

Dissertation

zur Erlangung des akademischen Grades eines

Doktors der Philosophie
der Philosophischen Fakultät
der Universität des Saarlandes

vorgelegt von

Kathrin Eichler
aus Kirchheimbolanden

Saarbrücken, 2018

Dekan: Prof. Dr. Roland Marti
Berichterstatter: Prof. Dr. Josef van Genabith, Priv.-Doz. Dr. Günter Neumann
Tag der letzten Prüfungsleistung: 03. Mai 2018

Abstract

Recognizing that the meaning of one text expression is semantically related to the meaning of another can be of help in many natural language processing applications. One semantic relationship between two text expressions is captured by the *textual entailment* paradigm, which is defined as a relation between exactly two text expressions. Entailment relations holding among a set of more than two text expressions can be captured in the form of a hierarchical knowledge structure referred to as *entailment graphs*. Despite the fact that several people have worked on building entailment graphs for different types of textual expressions, little research has been carried out regarding the applicability of such entailment graphs in NLP applications. This thesis fills this research gap by investigating how entailment graphs can be generated and used for addressing two specific NLP tasks: First, the task of validating automatically derived relation extraction patterns and, second, the task of automatically categorizing German customer emails. After laying a theoretical foundation, the research problem is approached in an empirical way, i.e., by drawing conclusions from analyzing, processing, and experimenting with specific task-related datasets. The experimental results show that both tasks can benefit from the integration of semantic knowledge, as expressed by entailment graphs.

Acknowledgements

Firstly, I would like to thank my supervisor, Günter Neumann, for the continuous support of my Ph.D study, especially for all the comments and questions that helped me improve my research and thesis.

I would also like to thank:

- Hans Uszkoreit for encouraging me to pursue a Ph.D and continuously supporting me throughout my years at DFKI,
- Feiyu Xu, Aleksandra Gabryszak, Sebastian Krause, Lili Kotlerman, Vivi Nastase, and Tae-Gil Noh for the fruitful and enjoyable collaboration and for providing code, data, and ideas,
- Stefania Racioppa, Evelyn Nowak, Gergana Popova, Vivien Macketanz, Oliver Marten, and Florian Petersen for their help with data annotation and evaluation,
- Hubert Truckenbrodt, Heike Paschotta, Philippe Thomas, and Jakob Wurster for providing valuable feedback on my writing,
- Josef van Genabith for taking his time to review my work, and
- Leonhard Hennig for his invaluable help with preparing my defense.

Finally, a very special thanks to my family and friends for their love, support, and encouragement.

To: Lunas, Levy, Elias, and Ruben

"Sass: know, be aware of, meet, have sex with"

- Douglas Adams, The Hitchhiker's Guide to the Galaxy

Zusammenfassung

Für viele Anwendungen, die natürliche Sprache verarbeiten, spielt die Erkennung von semantischen Beziehungen zwischen Texten oder Textteilen eine große Rolle. Eine solche semantische Beziehung ist die *textuelle Implikation* (Englisch: textual entailment), die wie folgt definiert ist: Eine textuelle Einheit T impliziert eine Hypothese H (anders formuliert: H folgt aus T oder H kann aus T geschlossen werden), wenn ein Leser von T mit hoher Wahrscheinlichkeit annehmen würde, dass H wahr ist [Dagan and Glickman, 2004].

Die textuelle Implikation unterscheidet sich dadurch von der strikter definierten *logischen Implikation*, dass sie auch Fälle abdeckt, in denen H zwar aus logischer Sicht nicht zwangsläufig aus T folgt, in denen die Folgerungsbeziehung aber dem üblichen Sprachgebrauch entspricht. Folgendes Beispiel veranschaulicht die Beziehung der textuellen Implikation:

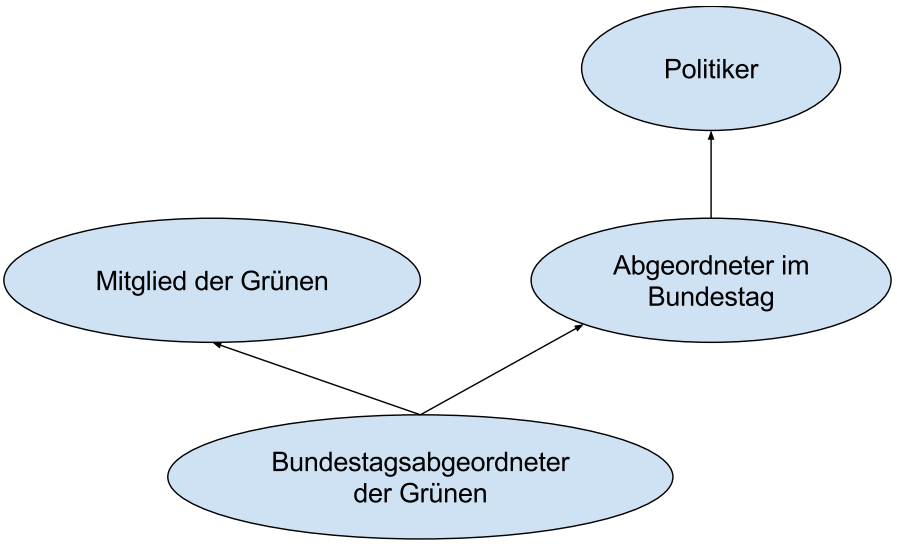
- T: "Rebellisch, intelligent, knurrig – Christian Ströbele war nie bequem. Das bekam auch seine Partei, die Grünen, zu spüren. Jetzt verlässt er den Bundestag."[1]
- H: "Christian Ströbele war Bundestagsabgeordneter der Grünen."

Text T impliziert im obigen Beispiel die Hypothese H, weil in T ausgedrückt wird, dass Ströbeles Partei die Grünen sind und er den Bundestag verlässt und ein Leser aus diesen beiden Informationen typischerweise ableiten würde, dass Ströbele für die Grünen Abgeordneter im Bundestag war.

Die Beziehung der textuellen Implikation (im Folgenden: Entailment) ist eine binäre Relation zwischen genau zwei Texteinheiten (z.B. Sätzen). Um Entailment-Beziehungen für mehr als zwei Texteinheiten zu modellieren, wurden sogenannte *Entailment-Graphen* eingeführt [Berant et al., 2010], die Entailment-Beziehungen in hierarchischer Form abbilden.

Das folgende Schaubild stellt einen Entailment-Graphen dar, der verschiedene Entailment-Beziehungen für den Ausdruck *Bundestagsabgeordneter der Grünen* abbildet (die Pfeilrichtung gibt dabei die Richtung der Entailment-Beziehung an):

[1] Aus der Berliner Zeitung vom 27. Juni 2017



Entailment-Graphen wurden in der Vergangenheit von mehreren Forschungsgruppen eingesetzt, um Entailment-Beziehungen zwischen unterschiedlichen Arten von textuellen Einheiten zu modellieren, beispielsweise Prädikat-Argument-Strukturen [Berant et al., 2010, Levy et al., 2014], getypte Prädikate [Berant et al., 2012, 2011] oder Textfragmente [Kotlerman et al., 2015]. Der Einsatz von Entailment-Graphen in realen Anwendungen wurde bisher jedoch kaum erforscht. Die vorliegende Arbeit hilft dabei diese Lücke zu schließen, indem sie untersucht, wie semantisches Wissen für zwei konkrete Anwendungen in Form von Entailment-Graphen modelliert und verwendet werden kann: Zum einen für die Validierung von automatisch generierten Mustern für die Relationsextraktion, zum anderen für die automatische Kategorisierung von deutschsprachigen Emails aus dem Kundensupport.

Ziel der Relationsextraktion ist es, aus Texten Informationen darüber zu gewinnen, welche im Text genannten Entitäten (z.B. Personen) in einer bestimmten Beziehung (z.B. Ehe) zueinander stehen. So lässt sich beispielweise aus dem Satz *Ich war auf der Hochzeit von Maria und Peter eingeladen* die Information ableiten, dass die beiden mit *Maria* und *Peter* referierten Personen in einer Ehe-Beziehung zueinander stehen (oder standen).

Anwendungen, die solcherlei Relationen automatisch erkennen, nutzen häufig existierendes Wissen, sogenannte Seed-Entitäten, die über die Zielrelation verbunden sind (z.B. Ehepaare), um damit aus Texten, in denen die Entitäten vorkommen, Extraktionsmuster abzuleiten. Diese Muster werden dann verwendet, um neue Entitäten zu erkennen, für die die Relation gilt. Aus dem Beispielsatz *Ich war auf der*

Hochzeit von Maria und Peter eingeladen könnte mit Hilfe der Seed-Entitäten *Maria* und *Peter* beispielsweise das Extraktionsmuster [Hochzeit von PERSON$_1$ und PERSON$_2$] abgeleitet werden, mit dem dann weitere Ehepaare gefunden werden könnten.

Dieses Verfahren ist jedoch fehleranfällig, da Seed-Entitäten, die zum Lernen von Extraktionsmustern verwendet werden, auch in anderen Satzzusammenhängen vorkommen können. So könnten die beiden mit *Maria* und *Peter* referenzierten Entitäten auch in einem Satz wie *Maria spricht mit Peter* gemeinsam auftauchen. Würde man aus diesem Satz das Muster [PERSON$_1$ spricht mit PERSON$_2$] extrahieren, so wäre dies vermutlich kein geeignetes Muster, um verheiratete Menschen zu erkennen. Die Filterung der automatisch generierten Muster ist daher ein wichtiger Schritt, um eine Teilmenge von Mustern zu erhalten, die korrekte Entitäten der Zielrelation mit hoher Genauigkeit extrahieren können. Filterstrategien können unterschiedlicher Natur sein. So können sie beispielsweise Informationen darüber nutzen, wie häufig ein Muster mit verschiedenen Seed-Entitäten vorkommt, oder externe Wissensdatenbanken, z.B. lexikalisch-semantische Netze, zu Hilfe nehmen, um zu beurteilen, ob ein Muster semantisch mit der Zielrelation verwandt ist.

Die Erkennung von syntaktischen oder semantischen Überschneidungen in verschiedenen Mustern kann helfen, mit Hilfe von Cluster-Verfahren ähnliche Muster zu identifizieren. Oft wird hierbei jedoch außer Acht gelassen, dass Beziehungen zwischen Mustern nicht symmetrisch sein müssen. So könnte über ein Clustering zwar beispielsweise herausgefunden werden, dass die Muster [Hochzeit von PERSON$_1$ und PERSON$_2$] (Muster 1) und [Scheidung von PERSON$_1$ und PERSON$_2$] (Muster 2) semantisch miteinander verwandt sind. Ein Clustering von Mustern kann aber nicht ausdrücken, dass sich zwar aus der mit Hilfe von Muster 2 ausgedrückten Beziehung (Scheidung) die mit Muster 1 ausgedrückte Beziehung (Hochzeit) ableiten lässt, sich aber umgekehrt aus einer Ehe-Beziehung keine Scheidungsbeziehung ableiten lässt.

Für die Filterung von Relationsmustern kann das Wissen über textuelle Inferenzen zwischen den erkannten Mustern sehr hilfreich sein, weil es ermöglicht zu unterscheiden zwischen Mustern, aus denen sich die Zielrelation ableiten lässt, und solchen, für die das nicht möglich ist. So lassen sich beispielsweise sowohl mit Muster 1 als auch mit Muster 2 Entitäten extrahieren, die geheiratet haben, aber nur mit Muster 2 Entitäten, die sich haben scheiden lassen.

Diese Art von Inferenz-Beziehungen zwischen Extraktionsmustern lassen sich auf natürliche Weise mit den eingangs beschriebenen Entailment-Graphen abbilden. Um das Potential von Entailment-Graphen im Kontext der Relationsextraktion zu untersuchen, werden in der vorliegenden Arbeit zum einen Verfahren entwickelt, um Entailment-Graphen auf der Basis von Extraktionsmustern zu generieren; zum anderen wird experimentell gezeigt, dass die generierten Graphen sich eignen, um aus einer Menge von automatisch generierten Extraktionsmustern qualitativ hochwertige Muster zu selektieren.

Die zweite im Rahmen der Arbeit betrachtete Anwendung ist die der Emailkategorisierung. Diese spielt beispielsweise im Kundensupport eine Rolle, wo eingehende Kundenanfragen automatisch existierenden Kategorien (z.B. Fehlerklassen) zugeordnet werden sollen, um so den Beantwortungsvorgang zu beschleunigen.

Für Anwendungen dieser Art werden die zugrundeliegenden Texte üblicherweise in eine vereinfachte Darstellung überführt, die dann für das Trainieren von Klassifikationsmodellen verwendet wird. Die am häufigsten verwendete Darstellung ist die sogenannte *bag-of-words*-Repräsentation, in der jeder Text als die Menge der in ihm enthaltenen Wörter dargestellt wird. Die zugrundeliegende Annahme ist hier, dass Texte, die ähnlich sind, dieselben Wörter enthalten.

Da die menschliche Sprache eine Vielzahl von Möglichkeiten bereithält, sich auf unterschiedliche Weise auszudrücken, gibt es jedoch Fälle, in denen die oben genannte Darstellung zum Scheitern verurteilt ist. Das folgende Beispiel zweier Kundenanfragen veranschaulicht dies:

- A: *Wenn ich Daten im .mpg-Format öffne, stimmt die Tonspur nicht mit dem Film überein.*
- B: *Bild und Ton einer Videodatei sind asynchron.*

Obwohl beide Anfragen dasselbe Problem beschreiben, teilen sie kein einziges gemeinsames Wort. Um die Ähnlichkeit der beiden Anfragen erkennen zu können, müssen daher Methoden angewendet werden, die über den Vergleich einzelner Wörter hinausgehen und die beiden Anfragen stattdessen auf der Bedeutungsebene miteinander vergleichen.

Existierende Ansätze verwenden hierfür häufig Wissen aus externen Ressourcen, das verwendet wird, um die *bag-of-words*-Darstellung semantisch anzureichern,

z.B. mit aus lexikalisch-semantischen Netzen extrahierten Synonymen oder Hypernymen. Auch auf großen Datenmengen erzeugte statistische Sprachmodelle, mit denen Bedeutungsähnlichkeiten von Wörtern aus der Ähnlichkeit ihrer Verteilung in sprachlichen Kontexten abgeleitet werden können, kommen hier zum Einsatz.

Die meisten dieser Ansätze operieren jedoch auf einer wortbasierten Textdarstellung, die größere semantische Einheiten wie Phrasen oder Sätze nicht berücksichtigt. Das obige Beispiel zeigt jedoch, dass der Vergleich von größeren Einheiten durchaus relevant sein kann. Zwar kann die semantische Beziehung zwischen den Ausdrücken *Ton* und *Tonspur* auf lexikalischer Ebene erkannt werden (z.B. über die Zerlegung des Kompositums). Um zu erkennen, dass der Ausdruck *Daten im .mpg-Format* eine *Videodatei* beschreibt oder dass der Ausdruck *Tonspur stimmt nicht mit dem Film überein* synonym ist zu *Bild und Ton sind asynchron* ist jedoch der Vergleich größerer Texteinheiten erforderlich.

Entailment-Graphen bieten eine natürliche Möglichkeit, um semantische Beziehungen zwischen Texteinheiten unterschiedlicher Art und Größe abzubilden. In dieser Arbeit wird daher ein Verfahren entwickelt, um aus in deutscher Sprache verfassten Kundenanfragen Entailment-Graphen zu erstellen, die Beziehungen zwischen in den Kundenanfragen enthaltenen Texteinheiten darstellen. Es wird außerdem gezeigt, dass das so generierte semantische Wissen verwendet werden kann, um die automatische Kategorisierung von Kunden-Emails zu verbessern.

Im Rahmen der vorliegenden Arbeit wird der Einsatz von Entailment-Graphen in realen Anwendungen, die natürliche Sprache verarbeiten, am Beispiel der beiden oben genannten Anwendungen untersucht. Hierbei werden die folgenden Forschungsfragen adressiert:

1. Wie können Entailment-Graphen als Datenstruktur und die Methodik ihrer automatischen Generierung formalisiert werden?

2. Können mit Hilfe der entwickelten Methodik Entailment-Graphen für die Anwendung der Relationsextraktion generiert werden und dazu beitragen, Instanzen einer Zielrelation mit höherer Genauigkeit zu extrahieren?

3. Können mit Hilfe der entwickelten Methodik Entailment-Graphen für die Anwendung der Emailkategorisierung generiert werden und zur Verbesserung der automatischen Kategorisierung beitragen?

Forschungsfrage 1 beschäftigt sich mit den theoretischen Aspekten der Erstellung von Entailment-Graphen. Zur Beantwortung der Frage werden relevante Konzepte und Methodiken aus vorhergehenden Forschungsarbeiten analysiert, zentrale Konzepte definiert und der allgemeine Prozess der Erstellung von Entailment-Graphen formalisiert. Darauf aufbauend beschäftigen sich die beiden folgenden Forschungsfragen mit der Anwendbarkeit von Entailment-Graphen für die beiden konkreten Aufgaben. Diese Forschungsfragen werden auf empirische Weise behandelt, indem existierende Daten analysiert und verarbeitet werden und als Grundlage für Experimente dienen.

Um die Anwendbarkeit von Entailment-Graphen im Kontext der Relationsextraktion zu untersuchen, wird zunächst ein neuer Typ von Entailment-Graphen definiert, der automatisch gelernte komplexe Extraktionsmuster als Grundelemente (Graphknoten) verwendet und diese basierend auf erkannten Relationsbeziehungen (gerichtete Graphkanten) hierarchisch anordnet. Die zugrundeliegende Annahme ist, dass eine solche Strukturierung der Muster dazu beitragen kann, die Muster zu filtern, die besonders zuverlässig Instanzen einer bestimmten Relation extrahieren. Um diese Annahme zu untersuchen, wird eine Methode entwickelt, um Extraktionsmustern auf der Grundlage von Entailment-Graphen zu filtern. Für die Evaluierung dieser Methode, wird zunächst ein Gold-Standard-Datensatz aufgebaut, der Entailment-Graphen für drei semantische Relationen enthält.

Außerdem werden existierende Technologien aus dem Forschungsfeld *Recognizing Textual Entailment* [Dagan and Glickman, 2004] (RTE) verwendet und angepasst, um Muster-basierte Entailment-Graphen automatisch zu generieren. Hierfür wird unter anderem ein auf aus externen Wissensquellen extrahierten Alignments basierendes RTE-System [Noh et al., 2015] eingesetzt und für die Verarbeitung von Extraktionsmustern angepasst. Das RTE-System wird auf einer Untermenge der handannotierten Muster aus dem Gold-Standard trainiert. Außerdem werden verschiedene Stragien für die Graph-Optimierung eingesetzt und evaluiert.

Um die vorgeschlagene Methode zur Entailment-Graph-basierten Filterung mit anderen Filterungsmethoden zu vergleichen, wird ein Evaluationsdatensatz aufgebaut, mit dem sich messen lässt, wie gut die mit der jeweiligen Methode selektierten Muster sich für die Relationsextraktion eignen. Die Experimente belegen den Nutzen einer Entailment-Graph-basierten Filterung, inbesondere im Vergleich zu Methoden, die zwar die semantische Ähnlichkeit von Mustern ermitteln, hierbei aber nicht die asymmetrische Natur von semantischen Beziehungen berücksichtigen.

Ein weiterer Datensatz [Toutanova et al., 2015] wird verwendet, um zu ermitteln, wie generisch das trainierte RTE-Modell ist. Hier zeigen die Experimente, dass eine überschaubare Menge an handannotierten Mustern ausreicht, um ein Modell zu trainieren, das auch eingesetzt werden kann, um sinnvolle Entailment-Graphen für andere semantische Relationen zu erzeugen.

Um die Anwendbarkeit von Entailment-Graphen im Kontext der Klassifizierung von Emails zu untersuchen, wird zunächst ein Datensatz aus Emails aus dem Kundensupport analysiert, um zu erkennen, in welcher Form textuelle Inferenz bei der Zuordnung von Emails zu Kategorien eine Rolle spielt. Hierfür werden Emailtexte mit den Beschreibungen der zugeordneten Kategorien verglichen, um zu ermitteln, welche Inferenzphänomene der Zuordnung zugrundeliegen. Die Verteilung der erkannten Inferenzphänomene in den untersuchten Daten lässt darauf schließen, dass vor allem (domänenspezifische) lexikalische Semantik sowie die Erkennung von aus mehreren Wörtern bestehenden lexikalischen Einheiten eine Rolle spielen.

Aufbauend auf dieser Analyse werden Entailment-Graphen erzeugt, die semantische Beziehungen zwischen in Emailtexten und Kategoriebeschreibungen enthaltenen Textausdrücken abbilden. Gold-Standard-Graphen werden direkt aus den Ergebnissen der durchgeführten Analyse abgeleitet, für die automatische Erzeugung der Graphen werden RTE-Technologien eingesetzt. Außerdem wird eine Methode vorgestellt und evaluiert, die das in Form der Entailment-Graphen dargestellte semantische Wissen beim Trainieren eines Email-Klassifizierers verwendet. Insbesondere wird das abgebildete Wissen eingesetzt, um unterschiedliche Text-Repräsentationen für den Klassifizierer zu generieren, und experimentell ausgewertet, welche Auswirkung die Berücksichtigung einzelner Inferenztypen auf das Klassifikationsergebnis hat.

Die Ergebnisse dieser Evaluierung zeigen, dass das generierte Wissen zu einer erhöhten Performanz des Email-Klassifizierers beitragen kann, insbesondere in Szenarien, in denen wenig Trainingsmaterial, z.B. in Form von manuell kategorisierten Emails, vorhanden ist. Die Verbesserungen in der Performanz lassen sich hier vor allem auf die Erkennung von Beziehungen zwischen lexikalischen Einheiten (einschließlich Mehrwort-Ausdrücken) zurückführen. Die Erkennung von Beziehungen zwischen komplexeren Texteinheiten trägt in den durchgeführten Experimenten nicht zu einer Performanz-Verbesserung bei.

Der erfolgreiche Einsatz von Entailment-Graphen im Rahmen der beiden in der Arbeit adressierten Anwendungen zeigt, dass Entailment-Graphen eine probate Möglichkeit bieten, das für eine bestimmte Aufgabe relevante, gegebenenfalls domänenspezifische, semantische Wissen abzubilden und zur automatisierten Bewältigung der Aufgabe zu verwenden. Die Ergebnisse zeigen außerdem, dass vorhandene RTE-Technologien in der Lage sind, aussagekräftige Entailment-Graphen automatisch zu erzeugen. Vorhandene RTE-Systeme können allerdings bisher ausschließlich Entailment-Beziehungen erkennen, die sich aufgrund von lexikalisch-semantischen Beziehungen oder syntaktischen Transformationen ableiten lassen. Um Entailment-Beziehungen für komplexere Fälle automatisch erkennen zu können, z.B. für Fälle, die die Berücksichtigung von verschiedenen Zeitformen und Modi oder domänenspezifischem Wissen erfordern, ist die Entwicklung fortgeschrittener Technologien beziehungsweise die Einbindung von Ressourcen, die das erforderliche Domänenwissen beinhalten, erforderlich.

Mit der Beantwortung der oben genannten Forschungsfragen leistet die vorliegende Arbeit folgenden Beitrag:

1. Sie schafft ein allgemeines Verständnis für die Fragestellungen, die sich im Zusammenhang mit der Erstellung von Entailment-Graphen ergeben.
2. Sie stellt zwei neue Arten von Entailment-Graphen vor sowie Methoden, das dargestellte Wissen in reale Anwendungen einzubinden.
3. Sie zeigt mit Hilfe von Experimenten, dass Entailment-Graphen in zwei realen Anwendungen erfolgreich eingesetzt werden können.
4. Sie untersucht das Potential existierender Technologien zur Erkennung von Entailment-Beziehungen für die Generierung von Entailment-Graphen.

Folgende Publikationen sind aus der der vorliegenden Arbeit zugrundeliegenden Forschung entstanden:

- **Eichler, K.**, Gabryszak, A., and Neumann, G. (2014). An analysis of textual inference in German customer emails. In *Proceedings of the Third Joint Conference on Lexical and Computational Semantics (*SEM)*, Dublin, Ireland (Abschnitt 5.2)

- Magnini, B., Zanoli, R., Dagan, I., **Eichler, K.**, Neumann, G., Noh, T.-G., Padó, S., Stern, A., and Levy, O. (2014). The Excitement Open Platform for Textual Inferences. In *Proceedings of the 52nd Annual Meeting of the Association of Computational Linguistics*, Baltimore, USA (Abschnitt 3.4.1)

- Noh, T.-G., Padó, S., Shwartz, V., Dagan, I., Nastase, V., **Eichler, K.**, Kotlerman, L., and Adler, M. (2015). Multi-Level Alignments As An Extensible Representation Basis for Textual Entailment Algorithms. In *Proceedings of the 4th Joint Conference on Lexical and Computational Semantics*, Denver, USA (Abschnitte 4.4.2 und 5.4.2.2)

- **Eichler, K.**, Xu, F., Uszkoreit, H., Hennig, L., and Krause, S. (2016). TEGREP: A Corpus of Textual Entailment Graphs based on Relation Extraction Patterns. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, Portorož, Slovenia (Abschnitt 4.3)

- **Eichler, K.**, Xu, F., Uszkoreit, H., and Krause, S. (2017). Generating pattern-based entailment graphs for relation extraction. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics*, Vancouver, Canada (Abschnitte 4.4 und 4.5)

- **Eichler, K.** and Gabryszak, A. (2017). Evaluating text representations for the categorization of German customer emails. In *Theorie, Semantik und Organisation von Wissen*, Passau, Germany (Abschnitt 5.3)

Desweiteren werden folgende weitere Publikationen der Autorin der vorliegenden Arbeit zitiert:

- **Eichler, K.** (2005). *Automatic classification of Swedish email messages.* Bachelor thesis, Eberhard-Karls-Universität, Tübingen, Germany

- **Eichler, K.**, Hemsen, H., and Neumann, G. (2008). Unsupervised relation extraction from web documents. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, Paris, France

- **Eichler, K.** and Neumann, G. (2010). DFKI KeyWE: Ranking Keyphrases Extracted from Scientific Articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, Los Angeles, USA

- **Eichler, K.**, Meisdrock, M., and Schmeier, S. (2012). Search and Topic Detection in Customer Requests - Optimizing a Customer Support System. *KI - Künstliche Intelligenz*, 26(4):419–422

Contents

List of Figures

List of Tables

Abbreviations

BOW	**B**ag **O**f **W**ords
CAS	**C**ommon **A**nalysis **S**ystem
EDA	**E**ntailment **D**ecision **A**lgorithm
EG	**E**ntailment **G**raph
EOP	**E**XCITEMENT **O**pen **P**latform
IE	**I**nformation **E**xtraction
LAP	**L**inguistic **A**nalysis **P**ipeline
NLP	**N**atural **L**anguage **P**rocessing
RE	**R**elation **E**xtraction
POS	**P**art **O**f **S**peech
RTE	**R**ecognizing **T**extual **E**ntailment
STS	**S**emantic **T**extual **S**imilarity
T/H	**T**ext/**H**ypothesis

Chapter 1

Introduction

1.1 Motivation

Recognizing that the meaning of one text expression is semantically related to the meaning of another can be of help in many natural language processing applications, such as question answering, information retrieval, information extraction, and document summarization. One semantic relationship between two text expressions is captured by the *textual entailment* paradigm. According to the first PASCAL Recognising Textual Entailment (RTE) challenge, a textual expression T is defined to *entail* a hypothesis H if "typically, a human reading T would infer that H is most likely true" [Dagan et al., 2006]. An example, taken from the dataset underlying this challenge, is the following:

- T: *The two suspects belong to the 30th Street gang, which became embroiled in one of the most notorious recent crimes in Mexico: a shootout at the Guadalajara airport in May, 1993, that killed Cardinal Juan Jesús Posadas Ocampo and six others.*
- H: *Cardinal Juan Jesús Posadas Ocampo died in 1993.*

In the example above, T entails H because T states, among other things, that a person called Cardinal Juan Jesús Posadas Ocampo was killed in a shootout in May, 1993, and knowing this, a reader would typically infer that this person died in 1993, as stated by H.

Entailment relations can be unidirectional, as in the example above, or bidirectional, as in the following example:

- A: *Cardinal Juan Jesús Posadas Ocampo and six others were killed at the Guadalajara airport in May, 1993.*
- B: *In May 1993, Posadas Ocampo, along with six other people, was assassinated at Guadalajara International Airport.*

In this example, expression A entails expression B and vice versa. Bidirectional entailment holds if the two expressions are paraphrases of each other.

Textual entailment is defined as a relation between exactly two text expressions. For capturing the entailment relations holding among a set of more than two text expressions, Berant et al. [2010] introduced the notion of *entailment graphs*, which represent entailment relations in a hierarchical way. A sample entailment graph from Berant et al. [2010] is depicted in Figure 1.1, which represents entailment relations holding between propositional templates. In this visualization, $\rightarrow$ denotes a unidirectional and $\leftrightarrow$ a bidirectional entailment relation holding between the connected propositional templates.

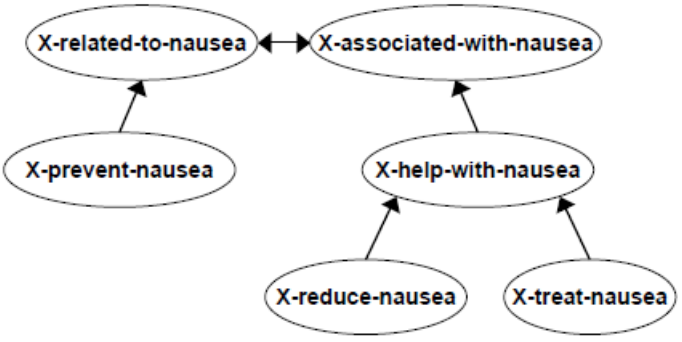


FIGURE 1.1: Entailment graph for propositional templates [Berant et al., 2010]

The given graph shows how the underlying templates are semantically related and allows for drawing inferences. For example, the graph states that the template *X-prevent-nausea* entails the template *X-related-to-nausea*, from which we can derive that any X that *prevents nausea* is also *related to nausea*. The graph also states that the two templates *X-related-to-nausea* and *X-associated-with-nausea* bidirectionally entail each other, meaning that the two templates carry the same meaning.

Apart from propositional templates, entailment graphs have also been built for other types of text expressions, including typed predicates [Berant et al., 2012, 2011], open IE propositions [Levy et al., 2014], and text fragments extracted from customer interactions [Kotlerman et al., 2015].

Despite the fact that several people have worked on building entailment graphs for different types of textual expressions, little research has been carried out regarding the applicability of such entailment graphs in natural language processing (NLP) applications. In this work, my goal is to investigate how entailment graphs, expressing knowledge about the semantic relationship holding between text expressions, can be used for two specific NLP tasks: First, the task of validating automatically derived relation extraction patterns and, second, the task of automatically categorizing German customer emails. In the following, I introduce these two tasks and motivate the idea of addressing them using entailment graphs.

1.1.1 Relation extraction

The task of relation extraction is to recognize and extract relations between entities or concepts in texts. For example, from the sentence *I went to Mary and Peter's wedding party*, we may want to recognize the fact that the two entities referred to by *Mary* and *Peter* got married. A common way to approach this task is to derive extraction patterns using seed entities and use these extraction patterns for recognizing additional entities [Brin, 1998]. For example, using Mary and Peter as seed entities, from the sentence above, we could derive the extraction pattern [PERSON$_1$ and PERSON$_2$'s wedding]. However, automating this process can easily lead to wrong extraction patterns. For example, using the same seed entities, we may also derive an extraction pattern like [PERSON$_1$ talked to PERSON$_2$] from a sentence like *Mary talked to Peter*. Thus, pattern filtering is a crucial step in order to identify high-quality extraction patterns. Filtering strategies can for example be based on frequency heuristics [Krause et al., 2012] or make use of external knowledge resources, such as semantic knowledge bases [Moro et al., 2013]. Assessing the quality of extraction patterns can also be based on identifying that extraction patterns are syntactically or semantically related. Work into this direction usually aims at clustering patterns or relation mentions based on similarity measures [Banko et al., 2007, Yates and Etzioni, 2009]. However, the fact that relationships holding among extraction patterns may not be bidirectional

has hardly been exploited so far. For example, identifying the patterns [PERSON$_1$ married PERSON$_2$] (pattern 1) and [PERSON$_1$ divorced PERSON$_2$] (pattern 2) as being semantically related can be helpful, but falls short of expressing that the relation expressed by pattern 2 *entails* the relation expressed by pattern 1, but not vice versa.

Being aware of these assymetric relationships between patterns can be of help in pattern selection, as it allows us to distinguish between patterns, from which the target relation can be inferred, and those for which this inference is not possible. For example, both patterns 1 and 2 can be used for extracting people that are or were married. However, only pattern 2 is a valid pattern for extracting couples whose marriage ended in a divorce.

With entailment graphs being a natural way of structuring textual expressions for drawing inferences, I suggest the generation and exploitation of entailment graphs for expressing inference relations holding among extraction patterns and evaluate the usability of these graphs in the context of relation extraction.

1.1.2 Email categorization

Assigning customer emails to matching categories can be approached as a text categorization task. The most commonly used text representation for this kind of tasks is the bag-of-words representation, which has been shown to work well in many applications. The underlying assumption made is that a text can be represented by the set of words contained in it, and that texts expressing similar meaning share the same words. However, as human language allows us to express the same idea in many different ways, for example by using synonyms or paraphrases, this representation can be too simple to capture the semantic relatedness of texts. This is illustrated by the following example, which is taken from a dataset of German customer emails:

- A: *Wenn ich Daten im .mpg-Format öffne, stimmt die Tonspur nicht mit dem Film überein.* [When opening data in .mpg format, the audio track does not match the film.]
- B: *Bild und Ton einer Videodatei sind asynchron.* [Image and sound of the video file are asynchronous.]

Sentences A and B represent two ways of expressing exactly the same customer problem, even though none of the words in B match words in A. In order to identify the semantic relationship between the two sentences, we need to go beyond the surface level of words, and, instead, compare the two texts at the meaning level.

Numerous methods have been proposed and applied to address this issue. Approaches range from enriching the bag-of-words representation semantically, for example by incorporating knowledge from external knowledge resources, such as lexical-semantic nets or ontologies, to statistical language models exploiting distributional similarities among words (e.g., latent semantic indexing) or topic models (e.g., latent Dirichlet allocation). Still, most of these approaches rely on a word level representation of the text. However, our example shows that matching the two sentences requires more than comparing the semantics of individual words. For example, even if we can derive the relatedness of lexical items such as *Ton* [sound] and *Tonspur* [sound track], this information will not be enough to determine that the expression *Daten im .mpg-Format* [data in .mpg format] refers to a *Videodatei* [video file], and that the expression *Tonspur stimmt nicht mit dem Film überein* [audio track does not match the film] is equivalent to *Bild und Ton sind asynchron* [image and sound are asynchronous]. This requires the semantic comparison of larger text units, such as phrases or sentences. Entailment graphs are a natural way of expressing semantic relations among text expressions of any kind and size. For addressing the task of automatically assigning categories to emails, I therefore suggest to generate entailment graphs expressing relationships holding among relevant text expressions, and make use of the generated knowledge when building the classifier.

1.2 Contribution

The general research problem addressed in this work is to investigate how entailment graphs can be generated and applied for solving NLP tasks. I approach this problem by evaluating the usefulness of entailment graphs in the context of two specific NLP applications, namely email categorization and relation extraction. The two tasks differ with respect to various dimensions (e.g., language and text type) and are therefore well-suited to demonstrate different issues relating to the construction and usage of entailment graphs, thus allowing for some generalization to other NLP tasks.

Addressing the research problem in an application-oriented way leads to the following research questions, to which this thesis will provide answers:

1. How can we formalize the notion of entailment graphs and the general methodology for constructing them?
2. How can we design entailment graphs for the task of relation extraction and can we make use of the generated knowledge for improving extraction accuracy?
3. How can we design entailment graphs for the task of email categorization and can we make use of the generated knowledge for improving categorization performance?

While research question 1 deals with theoretical aspects of entailment graph creation, questions 2 and 3 focus on the applicability of entailment graphs in the context of the two specific NLP tasks. By providing answers to these three questions, this work:

- Provides a general understanding of the issues related to entailment graph generation,
- Proposes two new types of entailment graphs and methods of integrating entailment graph knowledge into NLP applications,
- Shows how entailment graphs can be successfully applied in the context of two NLP applications,
- Assesses the potential of existing RTE technology and resources to generate entailment graphs that can be integrated into real-world applications,

and thus makes a substantial contribution to the rather new field of entailment graph research.

The work underlying this thesis resulted in several publications. These are listed in the following, along with the sections of the thesis, in which the respective work is described.

- **Eichler, K.**, Gabryszak, A., and Neumann, G. (2014). An analysis of textual inference in German customer emails. In *Proceedings of the Third Joint Conference on Lexical and Computational Semantics (*SEM)*, Dublin, Ireland (Section 5.2)

- Magnini, B., Zanoli, R., Dagan, I., **Eichler, K.**, Neumann, G., Noh, T.-G., Padó, S., Stern, A., and Levy, O. (2014). The Excitement Open Platform for Textual Inferences. In *Proceedings of the 52nd Annual Meeting of the Association of Computational Linguistics*, Baltimore, USA (Section 3.4.1)

- Noh, T.-G., Padó, S., Shwartz, V., Dagan, I., Nastase, V., **Eichler, K.**, Kotlerman, L., and Adler, M. (2015). Multi-Level Alignments As An Extensible Representation Basis for Textual Entailment Algorithms. In *Proceedings of the 4th Joint Conference on Lexical and Computational Semantics*, Denver, USA (Sections 4.4.2 and 5.4.2.2)

- **Eichler, K.**, Xu, F., Uszkoreit, H., Hennig, L., and Krause, S. (2016). TEGREP: A Corpus of Textual Entailment Graphs based on Relation Extraction Patterns. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, Portorož, Slovenia (Section 4.3)

- **Eichler, K.**, Xu, F., Uszkoreit, H., and Krause, S. (2017). Generating pattern-based entailment graphs for relation extraction. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics*, Vancouver, Canada (Sections 4.4 and 4.5)

- **Eichler, K.** and Gabryszak, A. (2017). Evaluating text representations for the categorization of German customer emails. In *Theorie, Semantik und Organisation von Wissen*, Passau, Germany (Section 5.3)

Other related publications that I (co-)authored and that are referred to in this thesis are:

- **Eichler, K.** (2005). *Automatic classification of Swedish email messages.* Bachelor thesis, Eberhard-Karls-Universität, Tübingen, Germany

- **Eichler, K.**, Hemsen, H., and Neumann, G. (2008). Unsupervised relation extraction from web documents. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, Paris, France

- **Eichler, K.** and Neumann, G. (2010). DFKI KeyWE: Ranking Keyphrases Extracted from Scientific Articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, Los Angeles, USA

- **Eichler, K.**, Meisdrock, M., and Schmeier, S. (2012). Search and Topic Detection in Customer Requests - Optimizing a Customer Support System. *KI - Künstliche Intelligenz*, 26(4):419–422

1.3 Justification

In recent years, the creation of technologies and resources for a deeper understanding of text has become a major topic of research, one prominent example being work in the field of textual inference. Textual inference technologies have been successfully used in various applications, including question answering, information extraction, summarization, and machine translation. Recently, entailment graphs were proposed as a data structure to capture semantic relationships holding among textual expressions. As such, entailment graphs have the potential to be a valuable resource for textual inference. However, so far little research has been carried out on the usefulness of integrating entailment graphs into NLP tasks. Also, as the focus of most researchers is on English, little research has been carried out on German language data.

With the research described in this thesis, I aim at filling this gap by (1) developing a generic, application- and language-independent procedure for building entailment graphs and (2) investigating how knowledge about the semantic relationship between textual expressions, expressed in the form of entailment graphs, can be used for addressing two specific NLP tasks: First, the task of validating and selecting relation extraction patterns, and, second, the task of automatically categorizing German customer emails. I am not aware of any previous work related to approaching either of the two tasks using entailment graphs.

For investigating the usefulness of entailment graphs for solving NLP tasks, I chose these two tasks because they differ with respect to several aspects, including text type (relation patterns versus email texts), language (English versus German), domain-specificness (general versus highly domain-specific), as well as the specific usage of entailment graph information for the task. This allows me to identify and address various aspects relating to the generation and usage of entailment graphs

for different NLP tasks. While the focus in this work is clearly on these specific tasks, the findings described in this thesis are thus expected to be relevant to a range of other applications involving textual inferencing.

1.4 Methodology

After reviewing related literature, I address the research questions listed in Section 1.2 in the following way. Research question 1 is addressed by looking into theoretical aspects related to entailment graph generation. Generalizing and extending notions and methodologies described in previous work in the field of entailment graphs, I provide relevant definitions and a formalized procedure for building entailment graphs, thus laying the grounds for the remainder of the work.

Having provided the theoretical foundation, I approach research questions 2 and 3 in an empirical way, i.e., by drawing conclusions from analyzing, processing, and experimenting with specific task-related datasets. Empirical techniques have been prominent in NLP since the 1990s [Cohen, 1995] and keep gaining importance, especially due to the ever-growing availability of language data as well as new technologies for processing them.

In order to investigate the usefulness of entailment graphs for the two tasks, I proceed in the following way: For each task, I first analyze an existing real-world dataset and design and create entailment graphs according to the outcome of the analysis, showing how the general procedure can be applied for building entailment graphs for the respective tasks. For either task, I then propose a methodology for using the knowledge represented by the entailment graphs for solving the respective task and evaluate the usefulness of this knowledge in an experimental way. I also evaluate the potential of state-of-the-art RTE systems, knowledge resources, and graph optimization algorithms to create entailment graphs for the two tasks automatically. This allows me to assess the usefulness of entailment graphs as such, but also the potential of existing RTE technology and resources to automatically generate entailment graphs that can be of help for solving NLP tasks.

1.5 Outline

The remainder of this thesis is structured as follows: Chapter 2 provides an overview of the field with related work on all relevant areas, including research related to semantic knowledge representations, textual entailment and entailment graphs, as well as previous semantic-based approaches to the tasks of email categorization and relation extraction. Chapter 3 formalizes the notion of entailment graphs and the general procedure for building them. It also describes a framework developed for building entailment graphs automatically, using existing NLP and RTE technology, which forms an essential part of the experiments performed in the context of the two applications. In chapter 4, entailment graphs are designed and used in the context of relation extraction. After constructing a gold-standard entailment graph dataset for three semantic relations, I evaluate the potential of the constructed entailment knowledge as well as automatically generated graphs for improving the accuracy of extracting relation instances. Similarly, in chapter 5, entailment graphs are designed and used for categorizing German customer emails. Following an analysis of a dataset of German customer emails, I develop entailment graphs for the task of email categorization, propose a method of using the generated knowledge for the categorization task, and present experimental results. Applying state-of-the-art RTE technology, I then build entailment graphs automatically and evaluate the resulting graphs on the task, showing the effects on categorization accuracy. Chapter 6 summarizes my contributions and presents directions for future work.

1.6 Delimitations

For addressing the above research questions, I made use of existing RTE technology, including several algorithms for deciding on entailment, two graph optimization strategies, and various linguistic knowledge resources. Within the scope of this thesis, no research was carried out regarding the development of new RTE technology. Rather, existing technology was applied and adapted to evaluate the applicability of entailment graphs for solving NLP tasks.

This was achieved by analyzing data and carrying out experiments in the context of two specific NLP tasks: relation extraction and email categorization. Due to

the vastness and diversity of NLP tasks, there may be aspects relevant in other NLP tasks that are not covered in this work. However, the two tasks were chosen to differ with respect to various dimensions, including language, type of input, and usage of entailment graph knowledge, thus ensuring that the results obtained allow for some generalization.

Chapter 2

Literature Review

2.1 Introduction

This chapter presents an overview of the literature related to the research problem addressed in this thesis: the generation of entailment graphs and their applicability for solving natural language processing (NLP) tasks. In the first three sections, I will give an overview of work related to semantic knowledge representations, textual entailment and entailment graphs. In the remaining two sections, I will summarize previous work related to the two NLP tasks addressed in this work: text categorization and relation extraction, focussing on approaches that make use of semantic knowledge.

2.2 Graph-structured semantic knowledge representations

The generation and usage of machine-readable representations of semantic knowledge has been a major theme of research in NLP for several decades. Research directions range from the manual and (semi-)automatic construction of semantic knowledge resources to the integration of these resources into a wide variety of NLP applications. A powerful means of representing knowledge is in the form of graph structures. A well-known example of a graph-structured resource is the semantic lexicon WordNet [Fellbaum, 1998, Miller, 1995], which represents relations between word meanings and has been heavily applied for a wide variety of tasks.

A common structuring principle for semantic lexicons is to define semantic classes, each expressing a distinct concept, and model the semantic relations holding between semantic classes as a hierarchy of super and sub-concepts [Frank and Padó, 2012]. In WordNet, words are grouped into sets of cognitive synonyms callsed synsets, and synsets are interlinked by means of semantic relations. Some of these relations, such as the hyperonymy/hyponymy relation for nouns and specificity of manner for verbs, arrange the linked synsets in the form of a hierarchy. WordNet-like resources have been created for dozens of other languages, including many monolingual resources, such as GermaNet [Hamp and Feldweg, 1997] for German, but also resources linking WordNet resources across different languages [Pianta et al., 2002].

Unlike general language resources like WordNet, which aim at modelling a specific type of linguistic knowledge exhaustively for one or several languages, domain ontologies model relations among concepts relevant for a particular domain or application. They are particularly useful in domains with large amounts of domain-specific terminology, e.g., in biology [Bard and Rhee, 2004]. An important relation type in ontologies is the *is-a* relation, which creates a hierarchically structured taxonomy of concepts along the hyperonymy relation. Unlike linguistic knowledge resources, which are usually hand-crafted, domain ontologies are often created (semi-)automatically, for example using lexico-syntactic patterns [Hearst, 1992, Snow et al., 2004] or clustering [Cimiano and Völker, 2005]. In addition to pattern-based methods for extracting hypernym-hyponym relations, researchers have investigated the usage of directional distributional similarity measures [Kotlerman et al., 2010] and word embeddings [Fu et al., 2014].

Even though lexical semantic resources and ontologies have shown useful in many applications, their expressivity is limited by the fact that their basic units are usually single, isolated words [Slodzian, 2001]. These resources usually do not cover semantic relationships holding between concepts expressed as linguistic units more complex than a single word.

Resources such as FrameNet [Baker et al., 1998] and VerbNet [Kipper-Schuler, 2005] go beyond lexical meaning and model predicate-argument structures, thus capturing the constitutive meaning relations holding between predicates and their arguments. FrameNet groups verbs, nouns, and adjectives into semantic classes (so-called frames), corresponding to abstract situations or events. The frames are linked according to a system of frame relations, which define relation hierarchies

(relating more general frames to more specific ones), but also semantic relations such as the succession of events and states. VerbNet is a verb lexicon of hierarchically arranged verb classes where each verb class is described by thematic roles, selectional restrictions on the arguments, and frames. Though more expressive in terms of grammatical constraints, these hand-crafted resources are still centered around lexical items.

While hierarchical structures representing knowledge at the lexical level have been created and used massively, similar knowledge structures encoding relations between larger text units have appeared only recently [Nastase et al., 2015]. One such structure are entailment graphs [Berant et al., 2010], which are based on the paradigm of textual entailment. They go beyond the lexical level, modelling relations holding between larger text units, such as predicate-argument structures or text fragments. The type of text expression, for which entailment relations are modelled, depends on the application, for which entailment graphs are built. Entailment graphs focus on a single relation type, textual entailment, which, unlike a simple *is-a*-hierarchy is able to express semantic relations holding between predicates and other complex text units. While hand-crafted entailment graphs have been created for evaluation purposes [Bentivogli and Magnini, 2014, Kotlerman et al., 2015], the ultimate goal is to create entailment graphs in an automatic way. The automatic generation of entailment graphs is commonly approached using entailment decision algorithms, (linguistic) knowledge resources, and graph optimization algorithms [Berant et al., 2010, 2011]. In the following, I will introduce the notion of *textual entailment* and present related work in the fields of recognizing textual entailment and building entailment graphs.

2.3 Recognizing textual entailment

2.3.1 Definition of the task

In semantics, the notion of *entailment* is defined as a relationship between two statements holding "when the second is a logically necessary consequence of the first" [Brinton, 2000], i.e., a sentence A entails another sentence B if A is true in every possible state of affairs in which B is true. Viewed in a strict sense, entailment is a binary relation between declarative expressions, to which a truth

value can be assigned in a given context. However, entailment can also be extended to include relations between other types of textual expressions, such as words and phrases [Sánchez Valencia, 1996].

The NLP task of Recognizing Textual Entailment (RTE) was established in and evolved around the RTE challenges [Dagan et al., 2006] with the goal of determining whether textual entailment holds for a given text/hypothesis pair. The notion of *textual entailment* refers to an application-oriented notion of entailment: It considers a text T to entail a statement H (also called hypothesis) in cases in which H is highly plausible, given the truth of T, i.e., whenever "a human reading T would typically infer that H is most likely true" [Dagan and Glickman, 2004]. This allows for cases, in which the truth of H may be considered slightly uncertain, at least hypothetically [Dagan et al., 2013], as in the following example, taken from the dataset of the RTE-2 challenge:

- T: *About two weeks before the trial started, I was in Shapiro's office in Century City.*
- H: *Shapiro works in Century City.*

Given the T/H pair above, we may think of a situation, in which Shapiro has an office in Century City, but does not actually work there. However, this situation would be highly unlikely. Rather, according to the typical expectations of a reader of T, one would say that the truth of H can be inferred from the truth of T.

RTE relates to two other NLP tasks, namely Paraphrase Recognition and Semantic Textual Similarity. The term *paraphrase* is used when referring to textual expressions that convey the same meaning using different surface forms. Bhagat and Hovy [2013] distinguish between strict paraphrases, which must be exactly logically equivalent, and quasi-paraphrases, i.e., text expressions conveying *approximately* the same meaning. As most complex paraphrases, i.e., the ones with alternations going beyond lexical synonymy and local syntactic changes, exhibit at least some degree of difference in content [Dolan and Brockett, 2005], the broader definition of paraphrase is the one that is commonly applied in NLP. Paraphrases may occur at several levels [Madnani and Dorr, 2010]: At the lexical level, paraphrases are usually referred to as synonyms, for example the verbs *(to) buy* and *(to) purchase*. Phrasal paraphrases may be patterns with linked variables, for example *[Y bought X from Z]* and *[X sold X to Y]*. Sentential paraphrases may require

complex transformations going beyond the substitution of individual words and phrases in the original sentence with their respective semantic equivalents, e.g., *He needed to make a quick decision in that situation* and *The scenario required him to make a split-second judgment.*

Another related notion is text similarity, which can be measured with respect to different dimensions, including structure, style, and content [Bär et al., 2011]. The task of measuring the similarity of texts according to the content dimension is referred to as *semantic textual similarity* (STS), which "measures the degree of semantic equivalence" [Agirre et al., 2012]. STS thus differs from paraphrase recognition in that its output is defined as being graded, rather than binary. Due to this vagueness, it is not directly applicable to inference tasks: As Dagan et al. [2013] illustrates, the sentence *monkeys like bananas* is semantically similar to *monkeys like mangoes*, but neither can be inferred from the other.

The RTE task differs from both Paraphrase Recognition and STS in that it assumes the relation between two texts as being *directional*, whereas both paraphrase recognition and STS measure a symmetric relation: If A is a paraphrase of B, then B is also a paraphrase of A. Similarly, the degree of similarity measured in STS for a pair of texts A and B does not consider any directional relationship between A and B. This symmetry does not hold in RTE: If a text A entails a text B, B can, but does not have to entail A. As both the paraphrase relation and the entailment relation are defined as binary relations, for which the the output is either TRUE or FALSE, the paraphrase relation can be viewed as bi-directional textual entailment.

2.3.2 Approaches and technology

Dagan et al. [2013] distinguishes five types of approaches to solving the RTE task: Edit distance-based approaches, approaches based on logical representation and inference, transformation-based approaches, alignment-based approaches, and paired-similarity approaches. Edit distance-based approaches, as applied by Mehdad et al. [2009] and Heilman and Smith [2010], define edit operations at the level of strings, tokens or syntactic dependency trees, as well as schemes reflecting the cost associated to each edit rule. This framework allows the computation of edit distance scores reflecting the amount of editing required to transform the text (T) into the hypothesis (H). The second type of approaches induce a logical

representation of T and H and combine this with a knowledge base representing background knowledge for determining entailment. Approaches along this line include the ones proposed by Bos and Markert [2005], Hodges et al. [2006] and Clark and Harrison [2009]. Transformation-based approaches [Bar-Haim et al., 2007, de Salvo Braz et al., 2005, Harmeling, 2009, MacCartney and Manning, 2009, Stern and Dagan, 2011] aim at making Text and Hypothesis identical using a set of transformation rules, which are constructed in a way that each transformation yields a valid natural language sentence as their result. Alignment-based approaches align portions of T (tokens, phrases, or more complex text constituents) to relating portions of H and were applied by Chambers et al. [2007], Chang et al. [2010], Hickl and Bensley [2007], Iftene and Dobrescu [2007], MacCartney et al. [2006], Madhumita [2016], Noh et al. [2015], Sammons et al. [2009]. Unlike the former approaches, paired-similarity approaches model similarities *across* different T/H pairs, e.g., using kernel-based methods [Wang and Neumann, 2007, Zanzotto and Moschitti, 2006].

Irrespective of the chosen approach, most RTE systems underlyingly make use of the same two types of sources: First, knowledge derived by analyzing the text representing H and T using linguistic preprocessing tools, and second, knowledge derived from external knowledge resources, such as linguistic or semantic knowledge stored in (hand-crafted) resources, knowledge derived using corpus-based methods, and transformation rules.

As linguistic analysis is an essential part of recognizing entailment, RTE systems build upon preprocessing tools such as stemmers, part-of-speech taggers, and named-entity recognizers for creating additional lexical knowledge, and parsers for RTE systems operating at the syntactic level. External knowledge resources are crucial to recognize entailment in cases where text and hypothesis use different, but entailment-preserving surface forms (words or phrases). Linguistic knowledge resources such as CatVar [Habash and Dorr, 2003], NOMLEX [Meyers et al., 1998] and DErivBase [Zeller et al., 2013] can provide information about words belonging to the same derivational family. The most commonly used hand-crafted knowledge resources are lexical nets, such as WordNet Fellbaum [1998], which can help retrieve information about synonymy or hyperonymy relations. Other semantic knowledge resources capture information at higher grammatical levels, such as VerbNet [Kipper-Schuler, 2005], or FrameNet [Baker et al., 1998], the latter a resource applied by Burchardt [2008].

Apart from the above-mentioned hand-crafted knowledge resources, other resources can be (semi-)automatically derived based on statistics computed over large language corpora, e.g., Wikipedia. Corpus-based methods for knowledge acquisition are either based on distributional similarity or co-occurrence [Dagan et al., 2013]. Co-occurrence-based methods rely on the assumption that words that occur frequently together are topically related. They were introduced by Hearst [1992] who identified linguistic patterns connecting pairs of words that express a particular relation type, e.g., hyponymy. Distributional similarity methods follow the *distributional hypothesis* by Harris [1954], suggesting that words that tend to occur in similar contexts have similar meanings. They assess the degree of similarity between two words by measuring the degree of similarity between their context representations. A widely used similarity measure for distributional similarity is the one by Lin [1998]. A collection of English paraphrase expressions automatically learned based on the distributional hypothesis is DIRT, created by Lin and Pantel [2001a].

Recently, with the rising availability of technological means to process large amounts of data, much attention has been put on data-driven methods. In particular, with the re-advent of neural networks in the context of "deep learning", the RTE task was addressed using neural models [Bowman et al., 2015, Rocktäschel et al., 2015]. Unlike previous approaches, they do not make use of processing pipelines or external resources, but require large high-quality datasets to achieve good performance on the RTE task. In the last two years, different neural network models were evaluated on the SNLI corpus [Bowman et al., 2015]. Earlier approaches were evaluated mostly on the datasets made available through the PASCAL RTE Challenges [Dagan et al., 2006], on which a study by Bos and Markert [2005] found human agreement to be 95.25%. State-of-the-art systems are still far from this performance. The best result reported on the RTE-3 test dataset was an accuracy of 80% [Hickl and Bensley, 2007]. The best reported accuracy on the SNLI test corpus was 88.8% [Zhiguo Wang, 2017].

A number of RTE systems have become publicly available over the last years, including Nutcracker [Bos and Markert, 2005], EDITS [Kouylekov and Negri, 2010] and BIUTEE [Stern and Dagan, 2012]. Linguistic tools and knowledge resources commonly used by RTE systems were collected at the Textual Entailment Resource Pool [1] , a website created by the RTE Challenge organizers. In the context

[1] `http://aclweb.org/aclwiki/index.php?title=Textual_Entailment_Resource_Pool`

of the EU project EXCITEMENT, an open platform providing RTE technology was developed and made available to the community. The platform called Excitement Open Platform [Magnini et al., 2014b] (EOP) provides a generic architecture and implementations for recognizing textual entailment in multiple languages, including English, German, Italian, and Bulgarian. The platform includes state-of-art algorithms following different approaches (tranformation-based, edit-distance-based, and classification-based), a large number of knowledge resources (lexical and syntactic), tools for creating new resources, as well as experimentation and testing facilities.

2.3.3 Applications

As drawing conclusions from natural language texts is highly relevant when processing natural language text automatically, RTE-based approaches have been investigated in the context of various NLP applications, including question answering, intelligent tutoring, and machine translation. Harabagiu and Hickl [2006] use an RTE system to re-rank candidate answers in a Question Anwering system. Applying a rule-based approach to transform the input question into a short statement, they determine whether the transformed question (H) entails any of the candidate answers, and move entailing candidates to the top of the output list. Similarly, Celikyilmaz et al. [2009] compute entailment scores between feature representations of user questions and candidate sentences returned by a search engine to improve the ranking of the provided answers. In the QALL-ME system [Ferrández et al., 2011], an RTE engine is used for mapping natural language questions against a structured database of prototypical question patterns. Entailment-based approaches have also been applied in the area of intelligent tutoring: Nielsen et al. [2009] and Sukkarieh and Stoyanchev [2009] measure the quality of student answers by checking whether they entail the concepts of the recorded reference answer. In machine translation, Mirkin et al. [2009] use entailment to address the task of translating unknown terms. Replacing unknown terms in the source sentence by hypernyms, they increased the coverage of unknown terms by 50% compared to allowing only synonyms as substitutes, losing only a few percentage points in precision. RTE techniques have also been used in the context of evaluation. Padó et al. [2009] propose an entailment-based automated measure of translation quality. Entailment-based parser evaluation was introduced in the

SemEval-2010 Shared Task [Yuret et al., 2013]. The functionality of the EXCITEMENT platform was applied to and evaluated in the context of two industrial use cases processing customer interactions [Noh et al., 2015]. Whereas previous experiments had been performed solely in research environments, this was the first time RTE systems were evaluated in a real-world context.

2.4 Entailment graphs

Entailment graphs are hierarchical representations of entailment decisions and can be viewed as a special type of knowledge graph. Knowledge graphs are "graph-structured knowledge bases which store information in the form of relationships between entities" [Nickel et al., 2015] and consist of nodes (representing entities) and edges (representing relationships between entities). An entailment graph encodes information about entailment relations holding between text expressions. The text expressions function as the nodes of the graph. Unlike in typical knowledge graphs, the edges of an entailment graph represent a single relation type: textual entailment. The edges are directed, pointing from the entailing text expression to the entailed text expression.

The notion of *entailment graph* was introduced by Berant et al. [2010]. However, the concept of directed graph structures representing entailment relations was already brought up earlier, by Heylighen [2001], who referred to these graphs as *entailment nets*. His work provides a valuable discussion on the interpretation of nodes and edges in such a graph, but stays on the theoretical side, whereas Berant et al. [2010] lays his focus on the automatic construction of entailment graphs. Entailment graphs have been built for various types of textual expressions, including propositional templates [Berant et al., 2010], predicates [Berant et al., 2012], predicates instantiated with arguments [Levy et al., 2014], and more complex textual expressions (including modifiers) [Kotlerman et al., 2015, Magnini et al., 2014a].

Following the procedure proposed by Berant et al. [2010], entailment graphs are built in two steps: First, decisions on entailment for individual T/H pairs are made (the traditional RTE task) to determine whether an edge from the node representing T to the node representing H should be added. These decisions can be associated with a confidence score. Second, some optimization procedure is

used to derive a consistent transitive entailment graph. For the latter step, Berant et al. [2010] proposes a global strategy, which searches for the most probable transitive graph given the local decisions. Another approach to constructing entailment graphs is to generate entailed sentences from source sentences, as proposed by [Kolesnyk et al., 2016]. An important step that needs to precede the actual graph building, is the identification or generation of relevant text expressions forming the nodes of the graph. This is a highly application-specific task, which can range from extracting predicates [Berant et al., 2011] to simplifying complex statements by removing arguments and modifiers [Kotlerman et al., 2015].

In previous work, the notion of entailment graphs as well as the procedure for constructing them has only been examined with regard to specific types of graphs. The present work provides a more general understanding of entailment graphs and their generation, which constitutes the prerequisite for using entailment graphs for a wider range of tasks.

As entailment graphs have only been proposed very recently, literature related to the application of entailment graphs is very limited. Adler et al. [2012] present an entailment-based text exploration system applied to the health-care domain. Their system allows a user to explore the result returned for a specified query by exploring related propositions according to a set of entailment relations described by an entailment graph. Kotlerman et al. [2015] generate textual entailment graphs for analyzing customer complaints. Evaluating their system in the form of a user study, they conclude that the entailment graphs allowed the study participants to better identify different dissatisfaction reasons.

Mehdad et al. [2013] construct entailment graphs over phrases (token n-grams) in order to improve topic labeling. Structuring phrases in the form of entailment graphs allows them to aggregate the phrase candidates by eliminating synonymous phrases and generalizing patterns that are too specific (e.g., selecting the phrase *animals laugh* over the more specific phrases *miece chuckle* and *rats giggle*). Evaluating their approach on two conversational datasets, they demonstrate that their approach helps to increase the number of correct labels.

Young et al. [2014] build what they call *denotation graphs* over image descriptions, which represent an entailment-based subsumption hierarchy and thus structurally correspond to entailment graphs. They apply their graphs in the context of two semantic inference tasks: The task of deciding the appropriateness of a brief image

caption given a set of correct captions for the image, and the task of assigning similarity scores to sentence pairs from a video description corpus. For both tasks, their results show their approach to be competitive compared with distributional similarity methods.

The work presented in this thesis extends research in the area of applying entailment graphs in that it investigates the usage of entailment graphs for two additional NLP tasks: relation extraction and text categorization. In particular, it is the first attempt to make use of entailment graphs generated from German language data.

2.5 Relation extraction

The task of relation extraction (RE) is to recognize and extract relations between entities or concepts in texts. This task is commonly approached using extraction patterns, for example based on dependency parse trees encoding the grammatical relations among the phrases that jointly express relation instances. In rule-based RE, the patterns are directly applied to extract relation mentions from parsed sentences of free texts [Alfonseca et al., 2012, Yangarber et al., 2000]. Other methods treat RE as a classification or sequence-labeling problem, but even for those techniques parse tree patterns have proven useful as key classification features [Bunescu and Mooney, 2005, Zelenko et al., 2003].

As the manual definition of large amounts of extraction patterns is tedious and time-consuming, various approaches have been proposed to derive these patterns automatically. In order to circumvent manual annotation work needed for supervised learning, recent work in RE concentrates on approaches that learn patterns in an unsupervised or weakly supervised way. Unsupervised approaches aim at identifying salient patterns fully automatically, i.e., they do not require target relations to be pre-specified and do not require manually labeled training data. Work in this area goes back to Riloff [1996] who identifies patterns using a set of heuristic rules and scores the extracted patterns based on their frequency in the data. Banko et al. [2007] introduce a system for what they call ”Open Information Extraction” that works on a web-scale corpus. Applying a parser on a small corpus to generate positive and negative training samples, the training data is used to

train a classifier, which is then applied to score relation candidates extracted from a large text corpus.

A weakly supervised approach to relation extraction is to start with a small amount of seed knowledge and unlabeled data, and perform data labelling and system training in several iterations using bootstrapping (e.g., Brin [1998], Agichtein and Gravano [2000], Agichtein [2006], Ravichandran and Hovy [2002]). However, these systems often have to cope with low recall and precision, the latter partially due to semantic drift. A more recent approach is to apply so-called distant supervision [Bauer et al., 2014, Krause et al., 2012, Mintz et al., 2009, Snow et al., 2004], i.e., to utilize large volumes of pre-existing knowledge (e.g., from knowledge bases such as Freebase) for heuristically labeling data and to acquire candidate patterns from these suspected mentions of relation instances.

Due to the variability of human language, the number of unique patterns potentially expressing the target relation is usually huge. In order to derive extraction patterns achieving a high level of precision in the extracted relation instances, an essential step is thus to filter the set of automatically extracted candidate patterns. The task of estimating the quality of extraction patterns has been dealt with in various ways. Agichtein [2006] uses integrity constraints and seed entities to estimate the confidence of an extraction pattern. Krause et al. [2012] apply frequency-based heuristics, in particular in order to decide on the correct target relation for patterns extracted as candidates for several relations. Another approach is the exploitation of semantic information. Moro et al. [2013] acquire relation-relevant word senses in the patterns and extract corresponding lexical semantic subgraphs from BabelNet, thus generating sets of words semantically associated to the target relation. These are then utilized as semantic knowledge for filtering out patterns not containing any relation-relevant word.

In order to mitigate the issue of language variability, research has also addressed the conflation of related patterns, for example by merging patterns based on syntactic criteria or by clustering patterns that are semantically related. For merging redundant patterns, Banko et al. [2007] create a normalized form (e.g., omitting non-essential modifiers). Shinyama and Sekine [2006] generalize patterns using a predicate-argument structure representation, for example by regularizing linguistic phenomena such as participial constructions and coordination. Thomas et al. [2011] utilize the fact that not all dependency types are of equal importance for relation extraction and improve extraction performance by collapsing dependency

links or unifying dependencies. Angeli et al. [2015] extract relation triples by producing coherent clauses entailed by the original sentence using a classifier and natural logic.

Approaches for grouping patterns or relation mentions include the DIRT system [Lin and Pantel, 2001a], which uses a similarity measure based on mutual information statistics to identify relations that are semantically related. Eichler et al. [2008] cluster relations based on information derived from WordNet, syntactic dependencies, named-entity recognition, and coreference resolution. Kok and Domingos [2008] generate semantic networks by extracting tuples from text, and then inducing general concepts and relations from them by jointly clustering the objects and relational strings in the tuples. In the context of Open information extraction, Yates and Etzioni [2009] present a system for discovering synonymous relations by clustering predicates and arguments based on distributional similarity. Similarly, Yao et al. [2011] cluster equivalent textual expressions by exploiting entity type constraints within a relation as well as features on the dependency path between entity mentions. Akbik et al. [2013] address the problem of pattern ambiguities, proposing a method to model selectional restrictions using phrase clustering of n-grams.

However, even though clustering relations can help gain generalization or reduce ambiguities, its ability to express semantic relationships holding between different extraction patterns is limited. A major limitation is that it cannot capture non-symmetric inference relations. For example, clustering can help us identify the patterns [PERSON$_1$ manager at ORGANIZATION$_1$] (pattern 1) and [PERSON$_1$ employed at ORGANIZATION$_1$] (pattern 2) as being semantically related. However, it falls short of expressing that the two entities linked by pattern 1, are also in the relation expressed by pattern 2, but not vice versa.

Recent work by Riedel et al. [2013] addresses the issue of asymmetry using a matrix factorization approach. Combining variants of surface patterns with pre-existing structured knowledge from Freebase, their method derives the probability of an entity tuple to be an instance of a particular relation. Their model takes into account the assymetric nature of relationships between patterns and learns that certain patterns, or combinations thereof, entail others in one direction, but not necessarily the other.

Non-symmetric relationships have also been studied intensively in the context of RTE. Wang and Neumann [2008] use textual entailment for relation validation, i.e., for estimating the quality of extracted relation instances. For determining whether a given relation instance is valid based on a given text, they construct simple sentences using the named entities of the relation instance and use an RTE system to check if these sentences are entailed by the input text. Their RTE system is based on tree skeletons extracted from dependency parse trees and is shown to perform better than two baseline approaches (bag-of-words and triple matching). Romano et al. [2006] investigate the usage of textual entailment for relation extraction, exploiting the fact that patterns that entail the target relation will extract mentions for which the target relation holds. Using a syntactic matcher based on transformation rules, they identify instances of entailing templates in sentences, i.e., templates appearing as subgraphs in the sentence dependency graph. Based on experiments on a dataset of protein interactions, they find their system's performance to be weaker compared to supervised methods, due to insufficient syntactic matching. A manual analysis still shows a high potential of the template-based approach. Bar-Haim et al. [2007] extend Romano et al. [2006]'s work by presenting a manually created set of entailment rules that aim at transforming a text into an hypothesis through a sequence of intermediate parse trees. They evaluate their framework in the context of relation extraction and present empirical results showing the feasibility of their approach. Roth et al. [2009] define the problem of Entailed Relation Recognition, which aims at finding text fragments entailing a particular information need, expressed as a statement. Their results show that using predicate-based entailment components improves the results over the lexical-level system.

As the above examples show, relation extraction can clearly benefit from considering semantic relationships holding among relation extraction patterns. However, previous work in relation extraction has either focussed on grouping related patterns without considering asymmetric relations, or, on determining asymmetric relations such as implicature or entailment for individual T/H pairs. The possible benefits of structuring relation extraction patterns in the form of entailment graphs, which are investigated in this thesis, have not been considered yet.

2.6 Text categorization

Text categorization is the task of assigning input documents to predefined categories and has been a topic of research since the early 1990s. Text categorization plays a role in various areas involving the organization of documents, including the assignment of news stories to topic categories [Hayes et al., 1988], spam filtering [Sahami et al., 1998], the categorization of medical reports [Wilcox, 2000], or the automatic foldering of emails [Bekkerman et al., 2004, Segal and Kephart, 1999], in particular in the customer support domain [Busemann et al., 2000, Eichler, 2005, Neumann and Schmeier, 1999].

In text categorization, a set of categories is defined beforehand. Given some input documents, the task is then to determine, for each document-category pair $< d_i, c_j >$, whether d_i should be assigned to c_j or not. Depending on the task, the goal may also be to output a list of categories ranked according to their estimated appropriateness to d_i [Sebastiani, 2002].

Text categorization is commonly approached using machine learning, with previously categorized documents being exploited as training data. Popular categorization methods range from rule-based algorithms [Scott and Matwin, 1999], k-NN [Paolo et al., 2004] and Naïve Bayes [Rennie, 2000] to support vector machines [Bekkerman et al., 2004, Busemann et al., 2000, Joachims, 1998], or boosting algorithms [Cai and Hofmann, 2003, Schapire and Singer, 2000]. Recently, there has been a focus on approaches based on (convolutional) neural networks [Johnson and Zhang, 2015, Zhang et al., 2015]. Early work in this area goes back to Ruiz and Srinivasan [1997], but the usage of neural networks for text categorization gained popularity only recently, when advancements in technology and techniques allowed for efficient training.

In order to apply machine learning, the content of the textual documents needs to be converted to a compact representation. This is typically done by transforming documents and categories into feature vectors. A crucial aspect with regard to feature representation is the choice and weighting of features. In the widely used bag-of-words (BOW) representation, features correspond to words identified in the input data and are weighted based on occurrence frequencies in the data. This representation, though simple, has proved effective for many applications. Nevertheless, much research has been carried out to derive more sophisticated

features, representing morphological, syntactic or semantic information present in the input documents.

In order to augment or replace the BOW representation for text categorization and information retrieval, people have researched into various directions.

One direction is to reduce or cluster features based on morphological or semantic criteria. Approaches range from simple methods such as filtering non-content or low-frequency words via collapsing features using stemming or lemmatization to approaches making use of semantic information, in particular information about relationships between words. A common approach, similar to query expansion in information retrieval, is to incorporate semantic information by extending terms with synonymous or semantically related terms. Numerous studies are based on using WordNet as a source of external knowledge [Gonzalo et al., 1998, Kehagias et al., 2003, Paolo et al., 2004, Papka and Allan, 1998, Scott and Matwin, 1999, Ureña et al., 2001]. However, being a manually created resource, the coverage of WordNet is limited. In recent years, researchers have turned to making use of larger-scale structured knowledge bases, such as the Open Directory Project [Gabrilovich and Markovitch, 2005, Gupta and Ratinov, 2008], or large corpora of unstructured language data such as Wikipedia [Gabrilovich and Markovitch, 2006, Gupta and Ratinov, 2008, Milne et al., 2007]. From the latter, semantic similarities between linguistic expressions can be derived based on their distributional properties in the data. Computational models implementing distributional semantics include latent semantic analysis [Deerwester et al., 1990], which derives the semantic similarity of words by analyzing the occurrence of words in documents in order to identify underlying concepts. Other models, such as probabilistic latent semantic analysis [Cai and Hofmann, 2003] and latent Dirichlet allocation[Blei et al., 2003], create document representations based on automatically discovered topics. Gabrilovich and Markovitch [2006] proposed a methodology called explicit semantic analysis, relying on the identification and encoding of human-defined concepts, which can be used to generate additional features [Egozi et al., 2008] or replace the BOW representation altogether [Ratinov et al., 2008]. As word-based models ignore grammatical structure and fail to capture the semantics of larger linguistic units, compositional distributional semantic models have been proposed, which aim to characterize the semantics of entire phrases or sentences. Different approaches have been explored, in particular in the context of the SemEval workshop [Marelli et al., 2014].

Of particular relevance to the work described in this thesis, is research on the exploitation of taxonomic ontologies for going from specific concepts to more general concept representations. Early work in this direction includes Scott [1998], who experimented on text classification using Wordnet hypernyms. Others, like Bloehdorn and Hotho [2006], Wang et al. [2003] and Bloehdorn et al. [2006], investigated the usage of hierarchical domain ontologies when extracting features, with some of their reported results suggesting that the integration of conceptual features improves categorization performance. Previous work in this direction, however, has focussed on using knowledge about semantic relationships holding between individual words or concepts.

The usage of larger text units has been addressed by incorporating features representing terms that go beyond the token level, e.g., n-grams [Beckers et al., 2009, Cavnar and Trenkle, 1994, Fürnkranz, 1998] or phrases [Apté et al., 1994, Lewis, 1992, Schütze et al., 1995, Scott and Matwin, 1999]. These turned out successful in some cases, for example in Fürnkranz [1998]'s work, whose results indicate that n-grams of length up to three usually improve classification performance. However, this is not generally the case. Scott and Matwin [1999] conduct experiments using several phrase-based representations and conclude that phrases do not add any classification power. Work by Beckers et al. [2009] on using bigrams supports this conclusion.

Making use of knowledge about semantic relationships holding between larger text units, as represented by entailment graphs, has not been investigated so far.

2.7 Conclusions

In this chapter, I presented an overview of work relating to knowledge representations, textual entailment and entailment graphs, as well as the role of textual entailment in the context of the two application areas addressed in this work, namely relation extraction and text categorization.

For the text categorization task, the usage of lexical semantic knowledge has been widely studied. Hierarchical knowledge resources have been exploited in the form of lexical semantic nets and domain ontologies. Entailment graphs provide knowledge going beyond these resources, as they represent semantic relations holding

between more complex text expressions. They are also more expressive then paraphase tables, as they store the direction of the inference relation. It is thus worth investigating their usefulness in the context of text categorization.

Structuring textual expressions is also a task addressed in relation extraction, with the expressions to be structured being relation extraction patterns. Most state-of-the-art approaches aim at clustering patterns. However, a major limitation of pattern clusters is that they fall short of capturing unidirectional relationships holding between patterns, a gap that can be filled by structuring patterns using entailment graphs.

The usage of entailment graphs as a way of structuring textual input along the semantic relation of textual entailment is a novel and promising direction for both the text categorization and the relation extraction task and will be investigated in the remainder of this work.

Chapter 3

Building Entailment Graphs

3.1 Introduction

As we have seen in the previous chapter, various research directions address semantic relatedness between text expressions from different angles. Some consider bidirectional relationships between text expressions, such as work aiming at detecting paraphrases or measuring semantic textual similarity. Others focus on directional relationships, such as work in the area of recognizing textual entailment, which addresses the task of identifying whether one text expression can be inferred from another. Viewing semantic relatedness as a directional task, as in textual entailment, allows us to express the relationships holding among a set of text expressions in the form of hierarchical graph structures, such as entailment graphs. In this chapter, I investigate the task of entailment graph generation. Starting with a definition of the relevant concepts, I describe the general procedure for building entailment graphs for various types of textual expressions from natural language texts. In the last part of the chapter, I present the architecture and implementation of a software library developed for building entailment graphs automatically. This library was used in the experiments described in the following chapters.

3.2 Entailment graphs

Entailment graphs were introduced by Berant et al. [2010] and represent entailment relations holding among textual expressions. Entailment graphs can be viewed as a special type of knowledge graph. Knowledge graphs are graph-structured knowledge bases storing information in the form of relationships between entities [Nickel et al., 2015]. When building an entailment graph from natural language input data, the entities forming the vertices (also called nodes) of the graph represent textual expressions identified in the input data. The edges connecting the vertices of the graph are directed and represent a single relation type: entailment relations holding between the expressions linked via the edges.

Entailment graphs arose in the context of the *textual entailment* paradigm. Following Dagan and Glickman [2004], the textual entailment relation is defined as follows:

Definition 3.1 (*Textual entailment*)**.** Textual entailment captures the semantic relationship holding between two textual expressions T (text) and H (hypothesis), if the meaning of H, as interpreted in the context of T, can be inferred from the meaning of T.

Thus, the definition of textual entailment differs from that of logical entailment in that it is more relaxed, application-oriented (see Section 2.3 for details about this difference).

Textual entailment (in the following: entailment) is defined as a directed relation holding between exactly two textual expressions T and H. When building entailment graphs, we go beyond a single T/H pair and model entailment relations for a larger set of expressions. Referring to $V = \{v_1, ..., v_n\}$ as the set of all these expressions, and to $E = \{(v_1, v_2), (v_1, v_3), ..., (v_2, v_1), (v_2, v_3), ..., (v_n, v_{n-1})\}$ as the set of all possible ordered pairs generated from these expressions, we can model each tuple in E as a boolean variable $ent_{ab} \in \{true, false\}$ indicating whether v_a entails v_b or not[1]:

[1]Note that the underlying assumption here is that entailment decisions are binary (entailment vs. non-entailment). Additional entailment types (e.g., contradiction) are not considered in this work.

$$ent_{ab} = \begin{cases} \text{true if } v_a \text{ entails } v_b \\ \text{false otherwise} \end{cases} \tag{3.1}$$

Entailment is a transitive relation, i.e., for expressions v_a, v_b, and v_c, if $ent_{ab} = true$ and $ent_{bc} = true$, then $ent_{ac} = true$.

V and E naturally transform into a graph structure with V representing the vertices of the graph and E representing the edges connecting the vertices (Definition 3.2).

Definition 3.2 (*Entailment Graph*)**.** An entailment graph is a directed graph $G = (V, E)$ consisting of a set of n vertices $V = \{v_i | i = 1..n\}$, representing textual expressions, and, a set of edges $E = \{(v_i, v_j) | v_i, v_j \in V\}$, which connect pairs of nodes and represent entailment relations holding among the associated expressions.

The edges of an entailment graph can either represent the full set of entailment decisions (positive and negative) or the subset of edges expressing positive entailment decisions. In the latter case, an edge starting at node v_a and ending at node v_b encodes that $ent_{ab} = true$. Considering only those tuples (v_a, v_b) in E as edges, for which $ent_{ab} = true$, the resulting graph is hierarchical in nature. In this case, given the inherent properties of textual entailment, directedness and transitivity, a well formed entailment graph should preserve transitivity among connected nodes [Magnini et al., 2014a].

Entailment graphs can be characterized along two dimensions: First, the type of expressions representing the nodes of the graph. Second, the way the entailment relation is defined.

Entailment graphs have been built for various types of expressions, including typed predicates [Berant et al., 2011], propositions [Adler et al., 2012, Levy et al., 2014], phrases [Mehdad et al., 2013], image descriptions [Young et al., 2014], and text fragments expressing customer complaints [Kotlerman et al., 2015]. In the following, I take a deeper look at the characteristics of these expressions in order to define a general notion of the expressions that can act as nodes in an entailment graph.

Looking at previous work in the field of entailment graphs and textual entailment in general, we can distinguish two types of expressions that can be considered

as input for determining entailment relations: First, expressions that are purely textual, e.g., *laugh* [Mehdad et al., 2013] or *staff was not nice* [Kotlerman et al., 2015], and, second, expression templates containing variable parts, such as simple slots, e.g., *X is the author of Y* [Lin and Pantel, 2001b], or typed variables, such as *invade(country,city)* [Berant et al., 2011]. All of these expressions contain at least one textual component, which determines the semantics of the expression and is thus decisive for the entailment relation holding among different expressions. In the following, let's look at the constraints on these expressions as well as their interpretation.

Dagan and Glickman [2004] define entailment as a relationship between a coherent *text* (T) and a *hypothesis* (H) formed by a syntactically coherent text fragment. A *syntactically coherent* text fragment is defined as having a well-formed fully connected syntactic analysis. This means that the definition of H is stricter than the definition of T. As an entailment graph represents all entailment relations holding among a set of expressions, each of the expressions representing a node in the graph should form a valid hypothesis.

A commonly used definition of textual entailment is that "T entails H if, typically, a human reading T would infer that H is most likely true" [Dagan et al., 2006], i.e., whenever the truth of H follows from T. However, following this definition we can only interpret the relations represented in an entailment graph by assuming that truth values can be assigned to all expressions in the graph nodes. This is naturally the case with sentential expressions to which a truth value that can be assigned in a given setting.

For capturing entailment relations holding between non-sentential expressions, e.g., the hyperonym relation between the words *dog* and *animal*, Dagan and Glickman [2004] propose to assign an *existential meaning*. An existential meaning can be assigned by introducing an existential qualifier and interpreting the expression using existential closure Schwarzschild [1999]. For example, the truth value assigned to the word dog would denote whether there exists a dog in the given setting or not.

A more general view on entailment is taken by MacCartney [2009] whose definition of entailment is based on the semantic types distinguished in type theory [Montague, 1970]: Two basic types representing expressions denoting entities (e) and truth values (t) and complex types formed by combining the two basic types to

functional types. The functional type represents expressions $a \rightarrow b$, which combine with an argument of type a (the input type) to produce an expression of type b (the output type). For example, predicates taking a single argument, such as common nouns, adjectives and intransitive verbs, are represented by type $< e, t >$, meaning that they combine with an entity to form an expression associated with a truth value.

MacCartney [2009] defines entailment relations for expressions of every semantic type in terms of set relations, providing a straightforward way of modelling entailment for expressions that are intuitively interpreted as sets rather than associated to a truth value. For example, interpreting *dog* as the set of dogs and *animal* as the set of animals, we can say that the denotation of *dog* entails the denotation of *animal* because the set of dogs is a subset of the set of animals.

In this work, I refer to the expressions representing the nodes in an entailment graph as *entailment units*. I define the notion of entailment units in such a way that it covers the various types of expressions that can function as graph nodes, and also allows for both of the above-mentioned interpretations of entailment (truth-based and set-based). The definition also takes into account that, as stressed by Heylighen [2001] concerning the interpretation of nodes in an entailment net, nodes are defined by the way they are distinguished, i.e., each node represents a concept that an observer considers distinct from all other concepts in the given context.

Definition 3.3 (*Entailment unit*)**.** An entailment unit representing a node in an entailment graph is a textual expression, to which, in the given context, a distinct semantic meaning can be assigned.

By *textual expression*, I refer to expressions containing at least one linguistic component, which makes the expression semantically interpretable. Textual expressions may, however, contain variable parts, for example simple slots, as in Lin and Pantel [2001b] and Berant et al. [2010], or typed variables, as in Berant et al. [2011].

According to MacCartney [2009], expressions having different semantic types are unrelated. In fact, most entailment graphs presented in the literature do not mix semantic types. However, we also find entailment graphs containing nodes of different semantic types. For example, Mehdad et al. [2013] builds entailment

graphs over phrases, with phrases denoting truth values (e.g., *animals laugh*) entailing phrases of the functional type $E \rightarrow T$ (e.g., *laugh*). Similarly, the denotation graphs by Young et al. [2014] and the textual entailment graphs by Kotlerman et al. [2015] contain declarative sentences as well as single words or phrases. These entailment relations can be considered valid assuming that, even though the surface form of two expressions are of different semantic types, when interpreting them, they are intuitively mapped onto an expression of a corresponding type. For example, in the case of *animals laugh* (t) and *laugh* ($< e, t >$), we could interpret the second expression as an expression of type t ("there exists some X that laughs") using existential type shifting Partee and Rooth [1983].

As in this work the focus is on the application of entailment graphs and mixing semantic types may be useful from an application perspective, I follow this relaxed interpretation of the entailment relation, I thus allow for both single-type and mixed-type graphs, assuming that in the latter case entailment relations between expressions of different semantic types are determined by deriving corresponding semantic types using type shifting.

Having laid the theoretical foundations, I will proceed with the procedure for building entailment graphs from natural language data.

3.3 Procedure

This section describes the general procedure for building entailment graphs from natural language input data. It reviews the methodology described by Kotlerman et al. [2015] for building entailment graphs for textual fragments with regard to creating entailment graphs for various types of expressions and from the perspective of developing a software library for building these entailment graphs automatically. The input to the procedure can be any collection of textual data. This could be a single or several coherent texts (e.g., news articles or customer emails), a set of dependency parse trees, or a list of individual words. The output is an entailment graph created from the input data, which represents entailment knowledge considered relevant for a particular application.

3.3.1 Step 1: Identifying relevant expressions

When building entailment graphs from a corpus of textual input data, the first step is to identify the relevant textual expressions from which the nodes of the entailment graph, i.e., entailment units, will be created. I refer to the text expressions identified in the input as *fragments* [2].

Definition 3.4 (*Fragment*)**.** A fragment represents a textual expression, which was identified as relevant in a given input text and can serve as a valid entailment unit.

The relevancy of an expression as well as its type is determined by the application, for which the entailment graph is created. A fragment can for example be a complete sentence, a single content word token (i.e., a lexical word with an assigned meaning), or a token n-gram, but it can also be a combination of non-contiguous tokens in a sentence, linked via syntactic dependencies. A fragment may even contain variable parts, for example a named-entity type replacing a named-entity in the original input. A fragment should correspond to a valid entailment unit, i.e., it should be semantically interpretable. However, as we will see later, more than one entailment unit may be created from a single fragment.

3.3.2 Step 2: Identifying modifiers

According to Definition 3.4 above, fragments can contain more than a single token. In this case, some parts of the fragment may be more, others less essential to the meaning of the fragment. For example, if a customer complains that *the internet is very slow*, one may say that the word *very* is less essential to the overall meaning of the sentence, because removing it from the sentence preserves the general meaning of the complaint. Identifying these less essential parts in a fragment allows us to reduce the fragment to its core meaning.

For reducing a text expression to its core meaning, Kotlerman et al. [2015] introduces the notion of *modifiers*. They consider a modifier to be a single token and annotate dependencies between all tokens in a text expression specifying the

[2]Note that I am adopting the term by Kotlerman et al. [2015] for reasons of comprehensibility of the following sections. Nevertheless, I use the term in a broader sense, referring to textual expressions of any kind, rather than only "textual units for which textual entailment in terms of truth values can be attributed" [Kotlerman et al., 2015].

conditions for removing modifiers from the expression: Tokens without any dependency refer to tokens that cannot be removed. Tokens for which one or more dependencies exist denote modifiers and can only be removed from the text after all dependent modifiers have been removed. For example, for the expression *light inside train is not relaxing*, the tokens *light*, *is*, *not*, and *relaxing* would refer to tokens without dependencies, forming the essential meaning of the text expression. The tokens *inside* and *train* reciprocally depend on each other, as they can only be removed together.

I define the notion of a modifier in a slightly different way, similar to Bentivogli and Magnini [2014], generalizing it from single tokens to expressions containing several tokens reciprocally depending on each other, i.e., tokens that can only be removed as a whole. This way, the annotation of dependencies can be reduced to the ones that are essential in the graph creation process, i.e., the dependencies holding across different modifiers.

For illustration, let us consider the example used by Kotlerman et al. [2015]: *Bright light inside train is not very relaxing.* Here, the tokens *inside* and *train* can be considered a single modifier, as they can only be removed as a unit. Assuming *inside train* to be a single modifier, and *bright* and *very* to be additional modifiers, no dependencies need to be annotated for this sentence, as all modifiers can be removed without dependency constraints on other modifiers.

The annotation of dependencies is required, however, for dependencies holding *across* different modifiers, i.e., whenever the removal of one modifier depends on the removal of another. For example, in the sentence *seats are uncomfortable as very old*, we can annotate two modifiers: *as ... old* and *very*, where the modifier *very* depends on the modifier *as ... old*. This means that *seats are uncomfortable very* is not a valid sentence because we can only remove *as ... old* if we also remove the dependent modifier *very*, yielding the sentence *Seats are uncomfortable.* The dependent modifier *very* can in turn be removed without any dependency constraints, i.e., *seats are uncomfortable as old* is also a valid sentence.

Modifier information can then easily be represented as two sets: one set M holding the modifiers and their span within the fragment, the second set D holding the dependency information. Modifiers in the first sample sentence, *Bright light inside train is not very relaxing*, can then be represented as $M = \{m_1 : 1; m_2 : 3, 4; m_3 : 7\}$ and $D = \{\}$, meaning that the first modifier spans token 1 (*bright*), the second

modifier spans tokens 3 and 4 (*inside train*), and the third modifier spans token 7 (*very*), with no dependencies across these modifiers. Accordingly, the second sample sentence, *seats are uncomfortable as very old*, would be represented as $M = \{m_1 : 4, 6; m_2 : 5\}$ and $D = \{2 \rightarrow 1\}$, meaning that the first modifier spans tokens 4 and 6 (*as ... old*)[3], the second modifier spans token 5 (*very*) and the second modifier depends on the first.

In order to use modifiers in the process of building entailment graphs, the identification and annotation of modifiers needs to be guided by semantics. I therefore define the notion of modifiers as follows:

Definition 3.5 (*Modifier*)**.** A modifier is a token or set of tokens within an entailment unit E, for which it is true that removing it, satisfying the constraints imposed by dependencies on other modifiers in E, results in an expression whose meaning is subsumed by the meaning of E.

This definition is general enough to be applicable not only to purely textual fragments, but also to fragments containing variables. For example, assuming a fragment such as [PERSON_1 *married* PERSON_2 *in* DATE_1], we can consider [*in* DATE] as a modifier, as the complete expression subsumes the meaning of [PERSON_1 *married* PERSON_2], regardless of the instantiation of the variables. Note that this definition of a *modifier* does not necessarily refer to the grammatical notion of a modifier.

While the identification of fragments is application-specific, the identification of modifiers is driven by semantics. However, it is still an application-specific task, as the decision of whether a token is a modifier or not may depend on the application domain. The identification of modifiers is only needed if the input fragments are sufficiently complex. As modifiers can occur in various forms, ranging from single words of different parts of speech to complex phrases, identifying them automatically is a highly challenging task.

3.3.3 Step 3: Building fragment graphs

Following Definition 3.5, we can automatically derive simple entailment graphs based on the modifiers identified in our fragments. Kotlerman et al. [2015] refer

[3]Note that this refers to a modifier consisting of non-contiguous tokens.

to these graphs as *fragment graphs*. Given that the meaning of an entailment unit E subsumes the meaning of all expressions resulting from removing a modifier m from E according to the dependency constraints, we can automatically derive an entailment relation from E (including m) to $E - m$ (the entailment unit with the modifier removed). For example, assuming that in the sentence *the internet is very slow*, *very* is marked as a modifier, we can derive that *the internet is very slow* entails *the internet is slow*. Deriving all possible such entailment relations allows us to build fragment graphs based on the modifiers identified in each fragment. Each fragment corresponds to exactly one fragment graph.

Definition 3.6 (*Fragment Graph*)**.** A fragment graph is a directed acyclic entailment graph created from a single fragment. Given m modifiers identified in the fragment, it consists of at most 2^m nodes, representing all combinations of the base statement (the original fragment with all modifiers removed) with 0 to m modifiers that satisfy the dependency constraints of the modifiers. The edges in a fragment graph connect each node n in the graph to the subset of nodes containing a subset of n's modifiers.

An example of a fragment graph is presented in Figure 3.1, with the top-most node representing the original fragment and the bottom-most node representing the base statement, i.e., the fragment with all modifiers removed[4] For reasons of simplicity, edges that can be derived via transitivity are not shown. Within the texts of the nodes, modifiers are shown in italics.

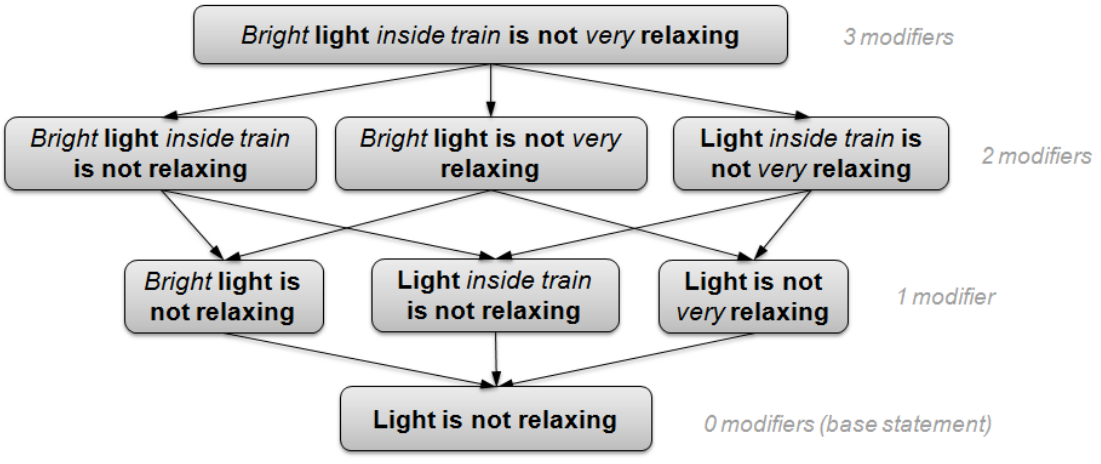


FIGURE 3.1: Example of a fragment graph [Kotlerman et al., 2015]

[4]Note that the entailment graphs we depict in the remainder of this work are interpreted under the closed world assumption, i.e., non-existing edges represent non-entailment decisions.

The generation of fragment graphs is straightforward and directly follows from the modifiers identified in each fragment along with their dependency constraints, i.e. completely application-independent.

3.3.4 Step 4: Computing pair-wise entailment

So far what we have produced is isolated entailment graphs, one for each fragment annotated in the input. In order to produce a complete entailment graph from the input, we need to merge these fragment graphs by adding missing entailment relations holding between the node texts across different fragment graphs [Kotlerman et al., 2015]. For generating entailment graphs in an automatic way, this task can be split into two steps: First, the computation of entailment decisions for individual entailment pairs, and, second, the generation of a transitive graph.

While fragment graph edges can be derived automatically, assuming a correct modifier annotation, external knowledge captured by an entailment decision algorithm (EDA) is required to derive entailment information holding between entailment units belonging to different fragment graphs. EDAs may for example be based on logical reasoning, general language resources, or domain knowledge.

After applying an EDA, the resulting graph structure may look as the graph presented in Figure 3.2. Solid edges correspond to edges copied from fragment graphs, dashed edges represent entailment decisions provided by an EDA. Each EDA decision is associated to a score between 0 and 1 reflecting the system's confidence in its decision. Thus, the output graph is a weighted graph, where edges are associated to weights representing their strengths (the confidence assigned to an edge). I refer to the output graph obtained by merging fragment graphs as *raw entailment graph*.

Definition 3.7 (*Raw Entailment Graph*)**.** A raw entailment graph is a directed entailment graph created by merging fragment graphs. It consists of a node for each distinct expression represented by a fragment graph and entailment decisions computed among expressions from different fragment graphs.

A raw entailment graph refers to an intermediate representation, as it may not necessarily correspond to a well-formed transitive graph. Unlike the fragment graph, it is not necessarily acyclic, with cycles resulting from entailment edges between

synonymous nodes (for example, *sandwiches are overpriced* and *they charge too much for sandwiches* in Figure 3.2) or from incorrect entailment decisions.

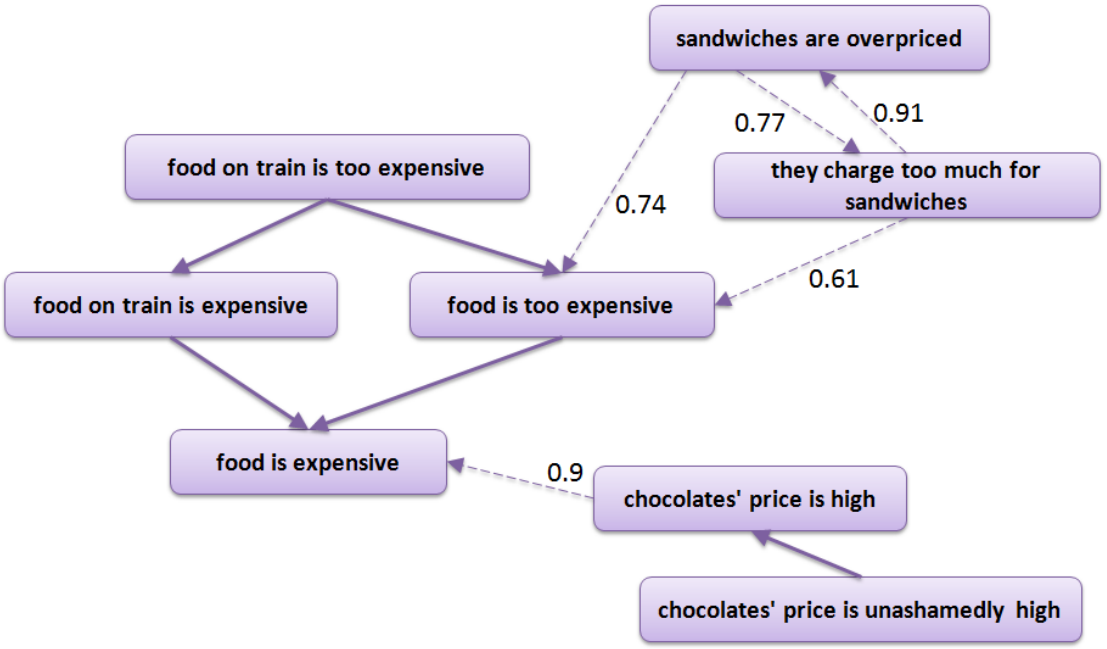


FIGURE 3.2: Sample output of step 4 – A raw entailment graph

The merging step itself does not depend on a particular application. However, the EDA needs to be chosen with respect to the task to be solved.

3.3.5 Step 5: Optimizing the graph

A well-formed entailment graph should preserve transitivity. Assuming that the output of an entailment decision algorithm is not always correct, the output of step 4 is not necessarily a consistent, transitive graph. In order to obtain a well-formed graph, we may have to adapt the raw graph by removing or creating edges to resolve transitivity violations resulting from automatically produced entailment decisions. This task is addressed in the optimization phase, which results in a well-formed transitive graph representing the final output of the procedure. The output may look as the sample graph depicted in Figure 3.3.

In this graph, which I refer to as *collapsed entailment graph*, each strongly connected component (each subset of graph nodes, for which there is a path in each direction between each pair of nodes in the subset) of the optimized graph is depicted as a single node. The number specified at the corner of each node refers to

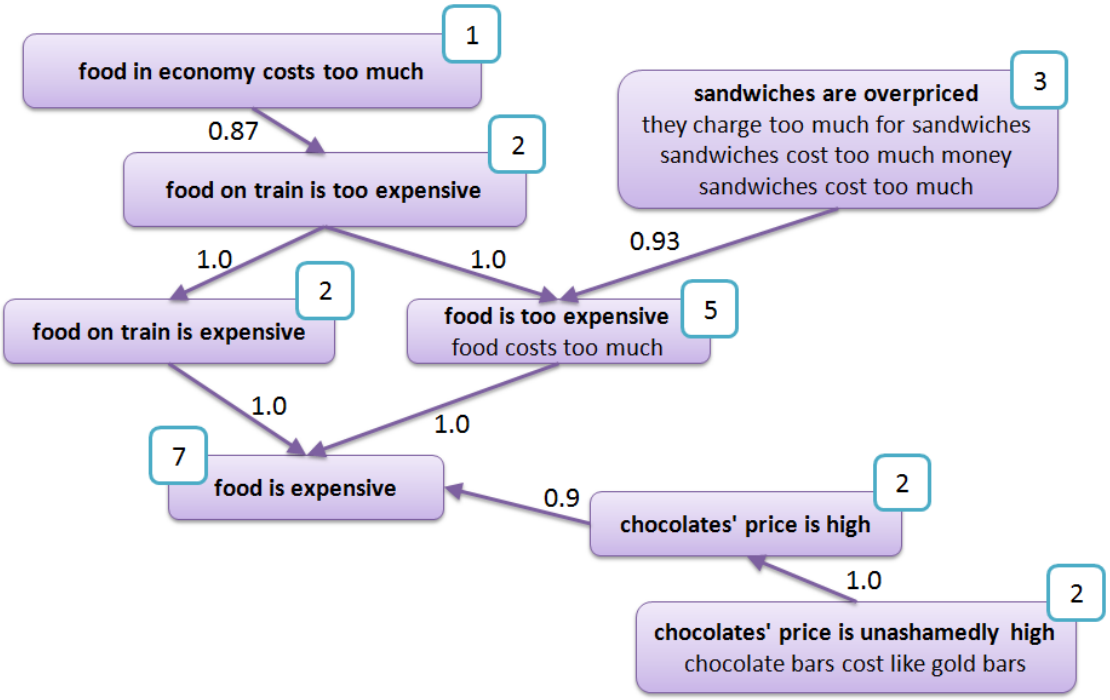


FIGURE 3.3: Sample output of step 5 – A collapsed entailment graph

the number of mentions associated to the node, i.e., the number of occurrences of expressions associated to this node in the original data.

Definition 3.8 (*Collapsed Entailment Graph*)**.** A collapsed entailment graph is a directed acyclic entailment graph consisting of a set of nodes, each representing an equivalence class (i.e., a set of entailment units considered semantically equivalent), and a set of edges representing entailment relations holding among equivalence classes.

Unlike raw entailment graphs, collapsed entailment graphs correspond to well-formed transitive graphs. Also, they do not contain cycles, as mutually entailing statements are combined to a single node. A collapsed entailment graph is therefore a directed acyclic graph. The confidence scores at the edges of the graph may be computed based on the integration of several scores from original EDA decisions in the underlying raw graph.

3.4 Framework for building entailment graphs

The work underlying the contents of this section was carried out as part of the EXCITEMENT project, an EU-funded project carried out between 2012 and 2014,

with two interleaved high-level goals. The first was to set up, for the first time, a generic architecture and a comprehensive implementation for a multilingual textual inference platform and to make it available to the scientific and technological communities. The second goal of the project was to develop a new generation of inference-based industrial text exploration applications for customer interactions, which can enable businesses to better analyze and make sense of their diverse and often unpredicted client content. For achieving the second goal, entailment graphs were used as the central data structure, representing use-case-specific inference knowledge.

This section gives an overview of the framework developed for building entailment graphs in the context of this project, justifies the crucial implementation decisions and presents an overview of the general architecture and implemented modules.[5]

3.4.1 EXCITEMENT open platform

The functionality for building entailment graphs was developed in such a way that algorithms and knowledge resources provided by the Excitement Open Platform (EOP)[6] can easily be integrated and evaluated. The reason behind this decision was that the EOP is not only freely available, but also provides a wide range of different algorithms as well as support for various languages, including the two languages relevant for this work, namely English and German. The architecture of the EOP consists of two major parts: Linguistic analysis pipelines (LAPs) and entailment decision algorithms (EDAs).

An LAP is a collection of NLP annotation components. Component integration is based on the Apache UIMA framework, in oder to enable interoperability among components and to ensure language independence [Magnini et al., 2014b]. Input and output of the components are represented in an extended version of the DKPro type system based on UIMA Common Analysis Structure (CAS) [Gurevych et al., 2007, Noh and Padó, 2013].

[5]The section describes and expands the results of work package 6 of the EXCITEMENT project, which was coordinated by the author of this thesis (as work package leader) and carried out in cooperation with Lili Kotlerman, Vivi Nastase, and Tae-Gil Noh.

[6]http://hltfbk.github.io/Excitement-Open-Platform/

An EDA computes an entailment decision for a given T/H pair. It can make use of components providing standardized algorithms or knowledge resources. Initially, the EOP shipped with three entailment decision algorithms, each following a different approach: BIUTEE [Stern and Dagan, 2011] (transformation-based), TIE [Wang and Neumann, 2007] (classification-based), and EDITS [Negri et al., 2009] (edit-distance based). Lately, additional EDAs have been added to the platform: MultiAlign [Noh et al., 2015] and Nemex [Madhumita, 2016] (both alignment-based).

3.4.2 Graph structures

Following the steps of entailment graph creation, I distinguish three types of graph structures: Fragment graphs, raw entailment graphs, and collapsed entailment graphs. The structures are based on the graph definitions provided in Section 3.3 and summarized in the following. A fragment graph represents a single fragment, all entailment units created based on the modifiers identified in the fragment, and the derived entailment relations. A raw graph contains the entailment units of all processed fragments, all fragment graph edges, and additional edges created by an EDA. EDA edges hold entailment decisions associated to confidence scores. The third graph structure, the collapsed graph, represents a directed acyclic graph (DAG) and is built from the raw graph by resolving transitivity violations and grouping entailment units into equivalence classes, i.e., sets of entailment units considered equivalent (mutually entailing) by the optimization procedure. Note that in this representation, we lose information about entailment confidences within the text expressions contained in the same node. Unlike the raw graph, the collapsed graph is a well-formed transitive graph.

3.4.3 Architecture

The architecture of the platform for building entailment graphs builds on top of the functionality provided by the EOP. It is designed in a modular way, based on the steps for entailment graph building identified in section 3.3. Figure 3.4 visualizes the architecture of the system.

The input to the system is a UIMA CAS object representing the textual input. An entailment graph can be built from a single CAS object or several CAS objects. A

CAS object can be enriched with linguistic annotations (parts-of-speech, lemmata, parse results, etc.), e.g., using LAPs from the EOP. Further annotations are added using fragment and modifier annotators. A *fragment annotator* marks text parts in the CAS text as fragments. A *modifier annotator* marks tokens or sets of tokens within a fragment as modifiers. A modifier annotator receives an input CAS, which holds the corresponding textual input with fragment annotation, and annotates modifiers found in the fragments. Both fragment and modifier annotations can be non-contiguous, i.e., can consist of multiple text parts that do not directly follow each other in the input text. Adding fragment annotation is obligatory, as fragments correspond to the nodes in the graph. Annotating no fragments would thus result in an empty graph. Adding modifier annotation is optional and depends on the application scenario (in particular, the type of fragment).

A *fragment graph generator* creates a fragment graph for each of the annotated fragments, based on the modifier annotation. Each fragment annotated in the CAS results in a single fragment graph. If a fragment does not contain any modifier annotation, the resulting fragment graph consists of a single node representing the complete fragment.

The *graph merger* module builds or extends a raw entailment graph by merging fragment graphs. It receives as input a raw graph (possibly empty), and a set of fragment graphs that are gradually added to the input raw graph. The result of this processing is a richer version of the input raw entailment graph. The raw entailment graph grows with each run of the graph merger module. The fragment graphs created from the input data are merged into the raw graph based on entailment decisions. To merge fragment graphs into the raw graph, the graph merger module can use the entailment decision capability of the EOP. For retrieving entailment decisions, LAP components and EDAs provided by the EOP may be called. Fragment graph edges are copied into the raw graph.

Note that when processing fragments, the same entailment unit may occur in more than one fragment and thus be created several times. When encountering an entailment unit that already exists as node in the raw entailment graph, we do not add a new node, but store the reference to the original text input in this existing node. Thus, each entailment unit text is unique and its corresponding node represents all mentions of this entailment unit in the textual input, from which the graph is built, thus keeping track of frequency information.

The *graph optimizer* module transforms the raw graph into the final output (a collapsed graph). While the raw graph represents all knowledge about the entailment relations between individual node pairs, the purpose of the optimization procedure is to incorporate global constraints to make final decisions on whether an individual entailment relation holds or not, and thus resolve transitivity violations. The module trims an input raw graph by selecting edges based on their associated entailment confidence score. For this, a confidence threshold may be provided to customize the resulting collapsed graph by filtering entailment relations from the input raw graph based on their strength. This step also compresses paraphrasing statements into equivalence class nodes. The graph optimizer module is self-contained, i.e., it transforms the input raw graph into a collapsed graph without relying on external modules or data.

3.4.4 Module implementations

This section describes a collection of implementations of the modules described in Section 3.4.3 that are provided as part of the platform for creating entailment graphs. The implementations are freely available[7] and were partially used in the experiments described in the following two chapters.

3.4.4.1 Fragment annotators

As mentioned before, fragment annotation is a highly application-specific task. In this section, several sample implementations of the fragment annotator module are presented. As all of the presented implementations are based on some kind of linguistic annotation, the decision on a particular implementation depends on the task itself, but also the availability of NLP processing tools. The quality of automatically derived fragments highly depends on the quality of the input (are the texts well-formed or do they contain non-interpretable parts, spelling mistakes, etc.?) and the performance of the used preprocessing tools.

Sentence-based fragment annotation

This simple fragment annotation method annotates each sentence as a separate fragment. The component calls the given LAP or searches the input CAS to find

[7] https://github.com/hltfbk/Excitement-Transduction-Layer

the sentence annotation, and then annotates each sentence as one fragment. It no sentence annotation is found in the input CAS, this fragment annotator requires the availability of a sentence splitter.

Token-based fragment annotation

This is a fragment annotation method that annotates each token as a fragment. The component calls a given LAP or searches the input CAS to find the token annotation, and then annotates each token as one fragment. If no token annotation is provided by the input CAS, this fragment annotator requires the availability of a tokenizer.

Dependency-based fragment annotation

This is a fragment annotation method that annotates a combination of two tokens (except punctuation) linked via a dependency as a fragment. If a filter for dependency types is passed to the constructor, then only dependencies matching the filter are annotated. The component calls the given LAP or searches the input CAS to find the dependency annotation, and then annotates each two-token combination as one fragment. In addition, filters for a) dependency types, b) parts of speech of governor and / or dependent, and 3) particular words can be specified. If filters are passed, only dependencies matching the filters are annotated. If no dependency information is provided by the input CAS, this fragment annotator requires the availability of a dependency parser.

Keyword-based fragment annotation

This method builds fragments starting from keywords provided by the user. Depending on the availability of a dependency parser and the requirements on processing time, it either uses a fixed window around the keyword or makes use of syntactic dependency relations (e.g., as provided by an LAP) to gather the grammatical phrase that encompasses a given keyword. The quality of the output depends on the quality of the dependency relations as well as the provided keyword annotations. As a light-weight alternative to the keyword-based fragment annotator that requires a dependency parser to build a fragment, an additional implementation of this type of annotation builds a fragment centered on the given keyword by adding N tokens to the left and to the right, while respecting sentence boundaries.

3.4.4.2 Modifier annotators

POS-based modifier annotation

For annotating modifiers, the platform provides several implementations relying on part-of-speech (POS) information. When calling the respective method, the annotator iterates over all fragment annotations and adds modifiers according to the specified parts-of-speech (e.g., adverbs or adjectives). The annotators can also verify whether a potential modifier is in the scope of a negation (and thus should not be annotated as a modifier) and whether the given annotation is a "proper" modifier, e.g, in the case of adjectival modifiers, is not in predicative position. If no POS information is available from the input CAS, this modifier annotator requires the availability of a POS tagger.

Phrase-based modifier annotation

Unlike the POS-based modifier annotators, the phrase-based modifier annotators rely on dependency information. A sample implementation of a phrase-based annotator annotates prepositional phrases as modifiers. Invoking the respective method, it builds the prepositional phrase starting from a preposition using dependency relations. If POS and dependency annotation are not provided by the input CAS, this modifier annotator requires the availabililty of a POS tagger and a dependency parser.

3.4.4.3 Fragment graph generators

The fragment graph generator module is the interface between CAS objects holding the textual input data along with different kinds of annotation (fragments, modifiers, linguistic annotations, etc.) and the modules for graph building. The implementation of fragment graph generation is straightforward and directly follows the annotation of fragments and modifiers. For each fragment annotated in the CAS, there will be a single fragment graph, which is further processed within the platform. The implementation is based on the algorithm described by Kotlerman et al. [2015], with a slight adaptation of the modifier definition to cover multi-token modifiers.

3.4.4.4 Graph mergers

All-pairs graph merger

This baseline implementation of the graph merger module merges fragment graphs by obtaining an entailment decision for each possible pair of entailment units. The merging procedure ensures not to call an EDA twice for the same pair of entailment units, as well as for node pairs from the same fragment graph.

Structure-based graph merger

This implementation of the graph merger module automates the procedure described by Kotlerman et al. [2015], which takes into consideration the structure of the input fragment graphs in order to minimize the number of EDA calls needed to perform the merge. When merging two fragment graphs, it starts by comparing the "minimal" entailment unit (i.e., the one containing no modifiers) of the first graph (G_1) to the minimal entailment unit of the second graph (G_2). If no entailment relation is found, no additional comparisons are required and the two graphs are not connected. If entailment was recognized from the minimal entailment unit of G_1 to the minimal entailment unit of G_2, entailment relations between the minimal entailment unit of G_1 and the directly entailing entailment units of the minimal entailment unit of G_2 are checked. Accordingly, for all other upper level nodes, entailment is only checked if an entailment relation at the lower level was determined. In case of bidirectional entailment, the described procedure is performed in both directions.

The described procedure is performed for each new fragment graph with each of the fragment graphs that have already been added to a given raw graph (i.e., were previously merged with the raw graph). The module implementation ensures not to call the EDA twice for the same pair of entailment units.

3.4.4.5 Graph optimizers

Simple graph optimization

This baseline implementation produces a collapsed graph from a raw graph by performing the following steps: First, remove all non-entailing edges, as well as entailing edges with confidence below a threshold specified by the user. Second,

recognize cycles and collapse all the nodes along each cycle's path into a single node. The resulting graph is a valid graph with no transitivity violations.

Global graph optimization

This implementation applies to a given raw graph the global optimization algorithm of Berant et al. [2012], as implemented within the Excitement Open Platform. Based on the entailment decisions and confidence scores stored in the raw graph, the algorithm searches for the best graph under a global transitivity constraint, using Integer Linear Programming to solve the optimization problem. The optimizer ensures that fragment graph edges, both positive (entailing) and negative (non-entailing), are not changed by the optimization algorithm. The resulting graph is a valid transitive graph.

3.5 Summary and discussion

In this chapter, I defined relevant notions and described the general procedure for automatically building entailment graphs from natural language input. The first step, the identification of relevant text fragments, is driven by the specific application, for which the graph is built, and crucially determines the character of the resulting graph: First, it determines the inherent structure of the entailment units and, thus, the complexity of the modifier identification task (step 2) and the resulting fragment graphs (step 3). Second, it determines the definition of entailment underlying the resulting graph (truth-based, containment-based), which becomes relevant when computing entailment relations (step 4). The optimization phase (step 5) is essential in automatic entailment graph creation, as the automization cannot guarantee a transitive graph output resulting from the pair-wise entailment results.

I also presented the design of a software library implementing the proposed procedure and provided a summary of various module implementations, some of which were used for the experiments in the following chapters. In the following two chapters I will investigate the generation and usage of entailment graphs for two NLP tasks: relation extraction and email categorization.

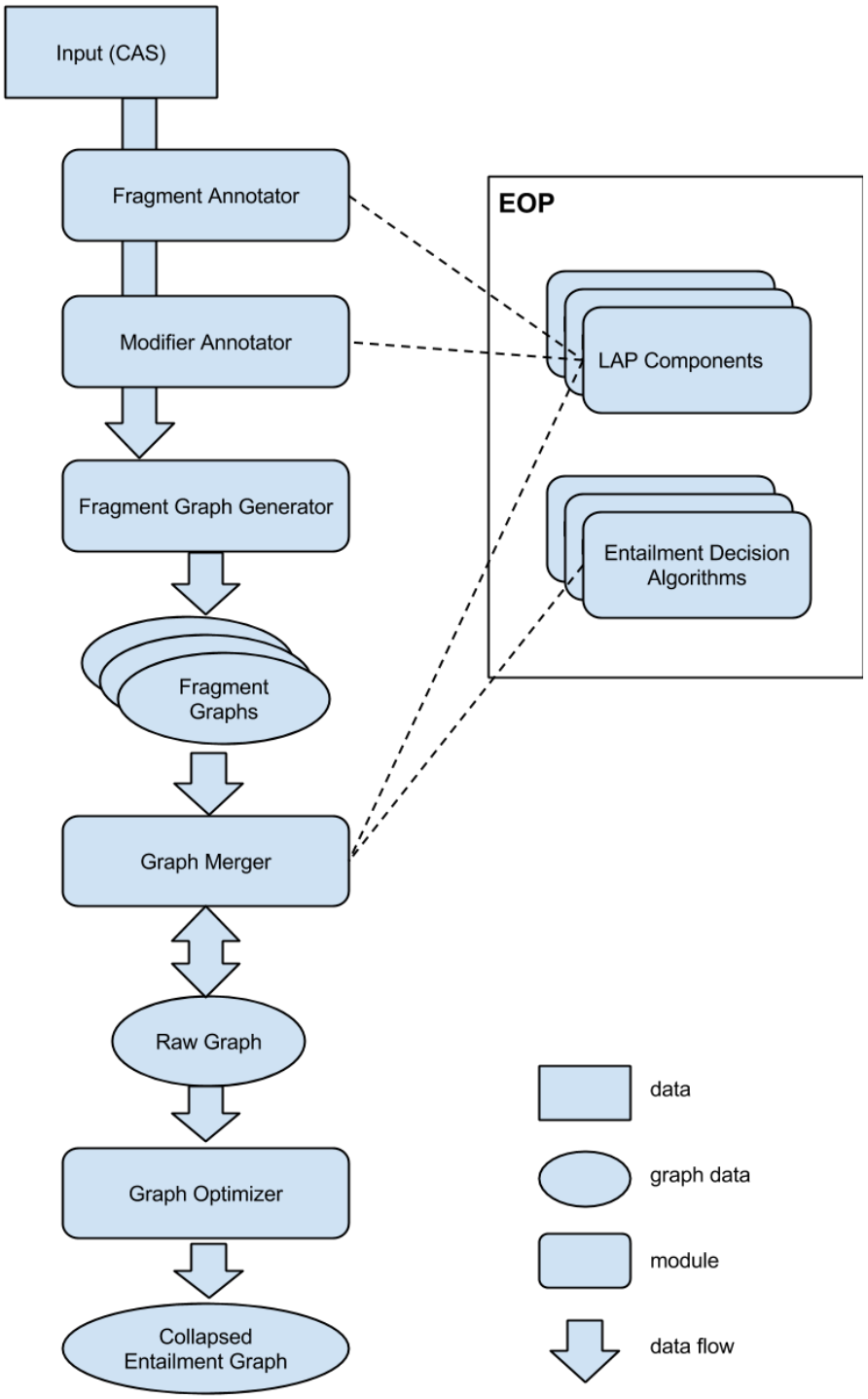


FIGURE 3.4: System architecture

Chapter 4

Entailment Graphs for Relation Extraction

4.1 Motivation

The task of relation extraction (RE) is to recognize and extract relations between entities or concepts in texts. This task is commonly approached by learning linguistic patterns (e.g., in the form of syntactic dependency trees), which potentially express the targeted semantic relations. These patterns are then exploited for extracting new mentions of the target relations. An example of such a pattern, expressing the fact that two people are married, is given in the following:

$$\text{person} \xleftarrow{nsubj} \texttt{marry} \xrightarrow{dobj} \text{person}$$

Nowadays, with technology for processing large amounts of data available, pattern extraction systems typically induce huge amounts of unique pattern candidates, due to the variability of human language. These patterns potentially express the target relation, but only a subset of these pattern candidates reliably extracts correct relation mentions. In order to achieve a high level of extraction performance, pattern selection thus becomes a crucial taks.

As elaborated in Section 2.5, this task can be tackled in various ways, for example based on integrity constraints [Agichtein, 2006], frequency heuristics [Krause et al., 2012] or lexical semantic criteria [Moro et al., 2013]. Other work addresses

the fact the pattern candidates may share common characteristics: Different extraction patterns may be semantically related, but differ in their morphological, lexical-semantic or syntactic form. This issue is addressed by merging patterns based on syntactic criteria [Angeli et al., 2015, Banko et al., 2007, Shinyama and Sekine, 2006, Thomas et al., 2011], by clustering patterns that are semantically related [Eichler et al., 2008, Kok and Domingos, 2008, Yao et al., 2011, 2012, Yates and Etzioni, 2009], or by identifying patterns associated to a given seed relation Bauer et al. [2014].

Such approaches help gain generalization; however, their ability to express semantic relationships is limited, as they cannot capture the asymmetric nature of these relationships. Although this type of semantic filtering can provide clues to evaluating the usefulness of relation extraction patterns, it cannot capture whether the meaning of a given pattern expresses that the target relation R really holds. For example, looking at a sample set of candidate patterns for the *marriage* relation, clustering could help us identify all patterns P1 to P6 below[1] as being semantically related to each other:

P1: PERSON$_1$ <marry> PERSON$_2$[2]

P2: PERSON$_1$ <wed> PERSON$_2$

P3: PERSON$_1$ <be widow of> PERSON$_2$

P4: PERSON$_1$ <divorce from> PERSON$_2$

P5: PERSON$_1$ <love> PERSON$_2$

P6: PERSON$_1$ <be in relationship with> PERSON$_2$

However, it falls short of expressing that the two entities linked by patterns P1 to P4 must have been in a marriage relation, whereas this is not necessarily true of the two entities linked by patterns P5 and P6. Clustering is also not able to capture the fact that patterns P3 and P4 entail pattern P1, but not vice versa.

Being aware of these (directional) semantic relationships among patterns can be of help in many RE applications. For example, in knowledge base population,

[1] I use the infinitive form of verbs here because I ignore tense for the time being, which is also the approach taken in the RTE challenges Dagan et al. [2006].

[2] PERSON$_x$ refers to a slot filler for a person recognized in the input from which the pattern was extracted, <text> refers to a normalized form of the text part of the extracted pattern.

patterns are not only relevant if they express the target relation explicitly, but also if they extract facts from which the target relation can be inferred. For example, all patterns P1 to P4 can be utilized for extracting pairs of people who are or were involved in a marriage relation. However, only pattern P1 expresses the target relation epxlicitly. Pattern P2 is semantically equivalent to P1, patterns P3 and P4 *entail* pattern P1. Patterns 5 and 6 are semantically related to the target relation, but cannot be used for extracting instances of the relation, because they do not entail the target relation.

As illustrated above, RE can clearly benefit from considering semantic relationships holding among extraction patterns. However, previous work in RE has either focussed on grouping related patterns without considering non-symmetric relations, or, on computing entailment decisions for individual T/H pairs. As in Romano et al. [2006], the basic assumption made in this work is that patterns are highly likely to derive correct relation mentions if they express a relation that semantically entails the target relation. However, taking a more global view than previous work, I propose to address the task of deriving reliable extraction patterns by converting all candidate patterns into an entailment graph, which expresses the semantic relationships holding among the patterns, thus creating a knowledge base from which inferences can be made. This structural knowledge can then be used for pattern selection. In particular, the entailment graph structure allows us to straightforwardly identify a set of patterns whose meaning expresses that a target relation R really holds. This pattern set includes patterns that express R explicitly, patterns that express a relation that is semantically equivalent to R, and patterns that express a relation that entails R. The approach of structuring candidate patterns using entailment graphs is related to the work by Nakashole et al. [2012], who create a taxonomy of binary relation patterns. For their syntactic patterns they compute partial orders of generalization and subsumption based on the set of mentions extracted by each pattern. In contrast to their work, I construct pattern-based entailment graphs based on n-ary relation patterns using RTE technology. This is motivated by the fact that entailment is semantic and not mention-based, i.e., one pattern can entail another pattern even if they extract disjoint sets of mentions in a given text corpus.

This chapter is structured as follows: Section 4.2 formalizes the general approach proposed for structuring and selecting patterns using entailment graphs. The construction of a dataset of gold-standard entailment graphs created by structuring

relation extraction patterns is described in Section 4.3. For automatically creating entailment graphs from relation extraction patterns, I use the technology described in Chapter 3 and a state-of-the-art RTE engine, which I adapt to the specific type of input data. This work is described in Section 4.4. In Section 4.5, I evaluate the usage of both gold standard and automatically generated entailment graphs in a relation extraction scenario and present the results of the evaluation. Section 4.6 summarizes the results.

4.2 Approach

As motivated in the previous section, I propose to exploit entailment graphs for relation extraction. In particular, I propose the following approach, which is depicted in Figure 4.1, for identifying valid relation extraction patterns:

1. Create a set of candidate extraction pattern P (applying any method of choice).
2. Generate a pattern-based entailment graph EG expressing entailment relations among the patterns in P.
3. Choose a base pattern[3], expressing the target relation explicitly and select all patterns entailing the base pattern according to EG.
4. Apply the selected patterns to extract relation mentions.

A pattern-based entailment graph refers to a directed graph, in which each node represents a unique RE pattern, and each edge ($\rightarrow$) denotes an entailment relationship. Bidirectional edges ($\leftrightarrow$) denote that the patterns represented by the two linked nodes are considered semantically equivalent. A sample subgraph for the *marriage* relation is given in Figure 4.2[4], which shows all entailment relations with respect to the base pattern [PERSON$_1$ <marry> PERSON$_2$].

[3]Note that the selection of a base pattern can be done manually, but can also be automated. For example, for the relations at hand, the most frequent pattern candidate learned for a particular relation turned out to be an appropriate choice.

[4]For reasons of simplicity, the figure shows the text representation of the patterns, which are in fact represented as dependency structures. Since entailment is transitive, all edges are omitted that can be recovered in the transitive closure.

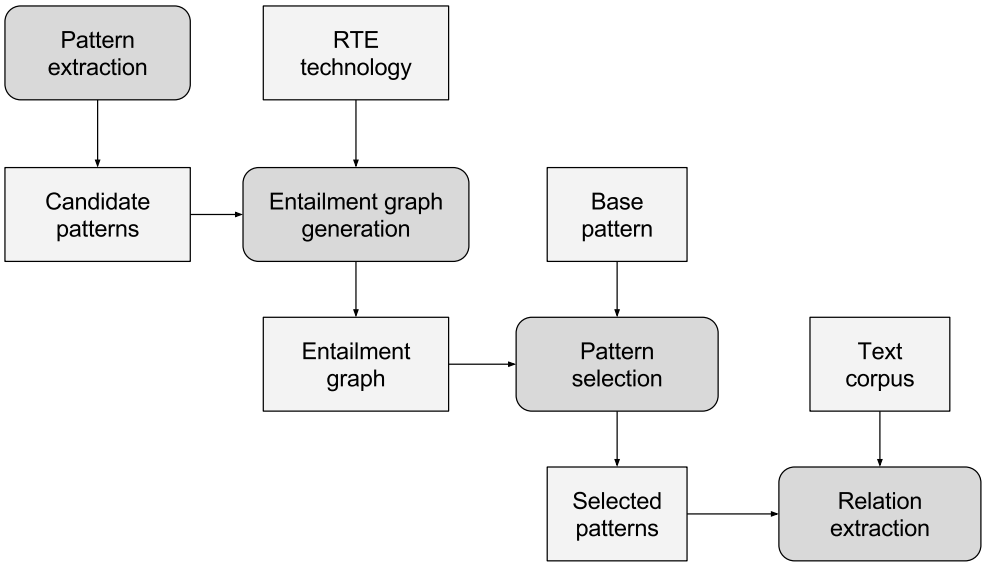


FIGURE 4.1: Procedure for relation extraction using pattern selection based on entailment graphs

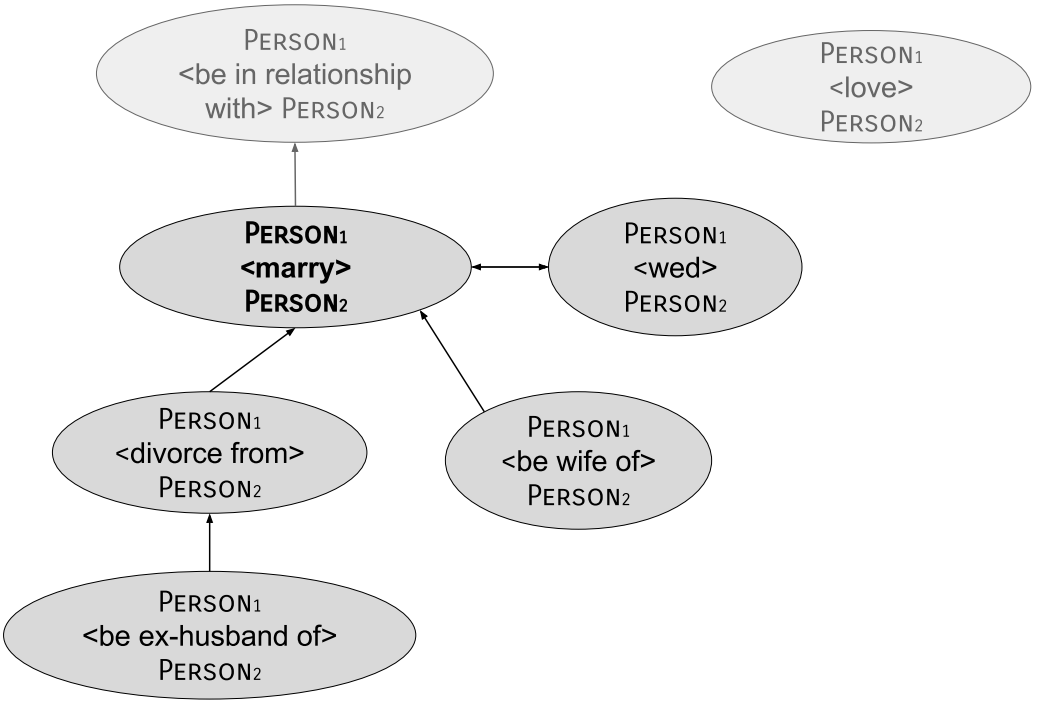


FIGURE 4.2: Subgraph showing entailment relations for the pattern "PERSON$_1$ <marry> PERSON$_2$"

Given this graph, the method would select all patterns entailing H (either directly or via transitivity), including all patterns considered semantically equivalent, but excluding all patterns with no entailment relation to H and those entailed by, but not equivalent to H. For example, selecting [PERSON$_1$ <marry> PERSON$_2$] as the base pattern H, the proposed method would select the pattern [PERSON$_1$ <wed> PERSON$_2$] because it is semantically equivalent to the base pattern. It would also select patterns [PERSON$_1$ <divorce from> PERSON$_2$], [PERSON$_1$ <be wife of> PERSON$_2$], and [PERSON$_1$ <be ex-husband of> PERSON$_2$], because they entail the base pattern, either directly or via transitivity. It would not select the pattern [PERSON$_1$ <love> PERSON$_2$] because it has not entailment relation to H and would also not select the pattern [PERSON$_1$ <be in relationship with> PERSON$_2$] because it is entailed by, but not equivalent to H.

4.3 Creating a gold-standard dataset of pattern-based entailment graphs

This section describes the generation of a resource of gold-standard entailment graphs constructed over dependency-structure based extraction patterns. A summary of the work described in this section and the resulting dataset called TEG-REP was published in Eichler et al. [2016]. The graphs are built based on automatically acquired patterns for three semantic relations typically considered in RE tasks: *marriage*, *acquisition*, and *award honor*. The gold-standard entailment graphs represent the entailment relationships holding among the patterns. After introducing related work in Section 4.3.1, I describe the acquisition of patterns using distant supervision (Section 4.3.2) and the methodology and annotation guidelines used for creating the gold-standard dataset (Section 4.3.3). Section 4.3.4 presents statistics about the dataset, including inter-annotator agreement, dataset size, and distribution of inference types.

4.3.1 Related work

In order to evaluate automatically generated entailment graphs, Bentivogli and Magnini [2014] created a gold-standard entailment graph dataset based on text fragments representing complaints extracted from Italian customer interactions.

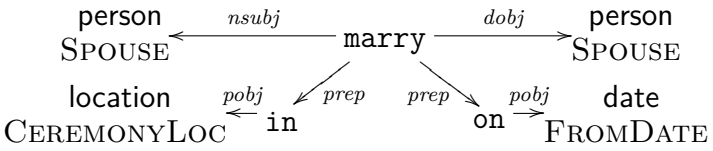
Starting with a manual annotation of so-called modifiers, i.e., tokens that can be removed without affecting the fragment's comprehension, they automatically derive entailment relations holding between a fragment and its subfragments (fragments from which modifiers were removed). All other entailment decisions required for building the entailment graphs are acquired by manually annotating T/H pairs of different fragments or subfragments. The number of T/H pairs to be annotated is minimized by manually clustering fragments into topics and by skipping unnecessary comparisons based on previous annotator decisions. In the final step, transitive closure edges are added to the graph and a consistency check is performed to ensure the transitivity of the resulting graph. The final dataset contains 19 textual entailment graphs, one for each topic cluster, with - altogether - 760 nodes and 2316 entailment edges. Kotlerman et al. [2015] use the same procedure to construct a dataset of entailment graphs based on text fragments extracted from English customer interactions. Their dataset consists of 29 entailment graphs, with 756 nodes and 7862 edges.

4.3.2 Relation extraction patterns

Parse trees have become a popular source for discovering extraction patterns, as they encode the grammatical relations among the phrases that jointly express a relation instance. In rule-based relation extraction methods [Alfonseca et al., 2012, Krause et al., 2012, Yangarber et al., 2000], the patterns are directly applied to extract relation mentions from parsed sentences of free texts, but they have also proven useful as features in classification-based relation extraction [Bunescu and Mooney, 2005, Mintz et al., 2009, Zelenko et al., 2003].

The patterns underlying the gold-standard dataset are a subset of the patterns used by [Moro et al., 2013] and were acquired automatically using the pattern discovery system by Krause et al. [2012]. The system derives candidate patterns from sentence parses generated using MaltParser [Nivre et al., 2007]. Pattern candidates are identified using distant supervision: As other distant supervision systems (e.g., Mintz et al. [2009], Alfonseca et al. [2012]), the system utilizes facts from Freebase [Bollacker et al., 2008] to derive relation instances, based on which relation mentions in the candidate sentences are then annotated. The system regards a sentence as a candidate of a relation mention if it contains the (main) entities of a relation instance of the fact knowledge base. Unlike most

other relation extraction systems, the system can deal with n-ary relations, not only binary relations. Furthermore, just as in the Snowball system [Agichtein, 2006], it assigns semantic role labels to the relation arguments. The following example pattern for the relation *marriage* contains four arguments, two married persons plus the wedding location and the starting date of the marriage:



The notation person|SPOUSE represents a placeholder for an entity mention of type person, which is assigned the role label SPOUSE at extraction time.

Applying the method to 39 relations from the domains *Awards*, *Business* and *People* modeled in Freebase, about 2.8M instances of these relations were retrieved as seed knowledge from Freebase. About 200,000 were turned into search queries, resulting in almost 20M downloaded web pages. 3M sentences matched by seed facts were utilized to learn more than 1.5M pattern candidates for the relation extraction task.

From the dataset described above, I consider three different relations: *acquisition*, *award honor*, and *marriage*. The required (bold) and optional semantic roles for these relations are listed in Table 4.1.

Relation	Role	Description
acquisition	**organization|Buyer**	organization buying another
	organization|Acquired	organization being acquired by another
	date|DATE	date of the acquisition
award honor	**person|Winner**	entity winning the prize
	prize|Award	name of the prize
	date|DATE	date the prize was awarded
marriage	**person|Person$_1$**	first person involved in the marriage
	person|Person$_2$	second person involved in the marriage
	date|FROM	starting date of the marriage
	date|TO	end date of the marriage
	location|CEREMONY	location of the wedding ceremony

TABLE 4.1: Semantic roles per relation

For each relation, I applied Moro et al. [2013]'s semantic filtering with the least restrictive configuration and, from the resulting pattern set, selected 500 of the most frequently occurring patterns for manual annotation.

4.3.3 Annotation procedure

4.3.3.1 General overview

The goal of the annotation procedure is to identify all entailment relations holding among different relation extraction patterns learned for the same target relation. I define entailment between a pattern H and a pattern T to hold if the meaning expressed by H can be inferred from the meaning expressed by T, assuming a typical human interpretation. Note that this definition of entailment follows the definition of textual entailment by Dagan et al. [2006], which is application-oriented and thus more relaxed than the definition of logical entailment. Incorporating a level of uncertainty allows us to deal with patterns that are not fully interpretable from a logical point of view, but are highly likely to carry a particular semantic meaning.

The identification of entailment relations between patterns is done via manual annotation based on the guidelines described in Sections 4.3.3.2 and 4.3.3.3. As with a large number of expressions, the task of comparing each possible T/H pair becomes unfeasible, I looked for ways to reduce the manual annotation workload and the number of inconsistencies in annotation by simplifying the annotation task.One step into this direction was to reduce the complexity of the task by dividing the annotation into two steps:

1. Identification of semantically equivalent patterns
2. Annotation of unidirectional entailment relations

Second, I reduce the manual annotation workload by removing entailment pairs, for which entailment is not possible based on the set of arguments contained in the pattern. The rationale behind this is that a pattern T cannot entail a pattern H if H contains more arguments (i.e., is more specific) than T.

Note that this assumption only holds if we neglect cases of downward entailment von Fintel [1999]. For example, we could imagine cases of negated patterns like person|PERSON$_1$ did not marry person|PERSON$_2$ → person|PERSON$_1$ did not marry person|PERSON$_2$ on date|FROM, where the first pattern entails the second, even though it contains less arguments, i.e., has less semantic strength. This is an interesting issue, but not relevant for this work, because, with our patterns being generated from positive training examples, there were no cases of downward entailment in our data.

As the underlying patterns were generated in a fully automatic way, some of them suffered from incorrect parsing. These patterns were annotated if the part relevant for deciding on entailment was semantically interpretable based on the given dependency tree. The annotation guidelines are summarized in the following.

4.3.3.2 Identification of semantically equivalent patterns

The goal of this first step is to construct sets of semantically equivalent patterns, referred to as *equivalence classes*. Semantically equivalent patterns correspond to patterns, for which textual entailment holds in both directions, i.e., patterns expressing the same meaning. For example, the pattern *organization*|BUYER *bought* *organization*|ACQUIRED[5] is semantically equivalent to *organization*|BUYER *purchased* *organization*|ACQUIRED (in this case, due to the synonymy of the verbs *buy* and *purchase*).

One crucial component determining the semantics of a relation patterns is the set of arguments contained in the pattern. Exploiting this fact, the input to the first annotation step are *argument clusters*, i.e., clusters of patterns grouped automatically based on the number and type of arguments identified in the pattern. For example, for the *marriage* relation, we added all patterns with the argument combination {person|SPOUSE, person|SPOUSE} to one cluster, all patterns with the combination {person|SPOUSE, person|SPOUSE, date|FROMDATE} to another, and so on. The underlying assumption is that patterns can only be semantically equivalent if their arguments are identical.

For each pattern, the annotator first determines whether it entails the base relation, e.g., for the *marriage* relation, if from the semantics of the pattern we can

[5]Note that for the annotation task, named entities were encoded along with the role they fill within the relation pattern.

tell that the pattern links two people that are or were married. Only patterns entailing the base relation are associated to equivalence classes. For grouping patterns into the same equivalence class, we analyse pairs of patterns, distinguishing the following types of equivalence:

Identity The two patterns contain the same set of words, possibly with a differing word order, e.g.,

In date|FROMDATE person|SPOUSE marries person|SPOUSE
↔ person|SPOUSE marries person|SPOUSE in date|FROMDATE

Preposition variations The two patterns contain differing prepositions that assign the same meaning in the given context, e.g.,

person|SPOUSE married to person|SPOUSE in date|FROMDATE
↔ person|SPOUSE married to person|SPOUSE from date|FROMDATE

Morphological variations The two patterns contain words that are morphologically related (e.g., via derivation) and express the same semantics, e.g.,

person|SPOUSE marries person|SPOUSE
↔ person|SPOUSE marriage to person|SPOUSE

This includes cases, where T and H contain different tenses of the same verb[6], e.g.,

person|SPOUSE marries person|SPOUSE"
↔ person|SPOUSE married person|SPOUSE

Note that we do not consider tense variations to be equivalent if they cause a meaning change, e.g.,

[6]Ignoring tense aspects is also the approach taken in the RTE challenges [Dagan et al., 2006]

person|SPOUSE marries person|SPOUSE
$\neq$ person|SPOUSE was going to marry person|SPOUSE

Here, the second pattern, unlike the first, does not express that the marriage actually took place .

Synonymy The two patterns contain words considered synonymous in the given context, e.g.,

person|SPOUSE marries person|SPOUSE
$\leftrightarrow$ person|SPOUSE weds person|SPOUSE

Paraphrase The two patterns contain expressions that are semantically equivalent in the given context, with at least one of the expressions being a multi-word expression, e.g.,

person|SPOUSE marries person|SPOUSE
$\leftrightarrow$ person|SPOUSE established marriage with person|SPOUSE

Passivization One of the two patterns expresses the passive form of the other, e.g.,

Organization|BUYER bought Organization|ACQUIRED
$\leftrightarrow$ Organization|ACQUIRED was bought by Organization|BUYER

Argument labelling variation The arguments of the predicates of the two patterns are aligned by different syntactic functions, e.g.,

Organization|BUYER bought Organization|ACQUIRED
$\leftrightarrow$ sold Organization|ACQUIRED to Organization|BUYER

In this example, Organization|ACQUIRED fills the subject role in the first pattern and the object role in the second pattern, whereas Organization|BUYER fills the role of the indirect object in the first pattern and the subject role in the second pattern.

Patterns are assigned to the same equivalence class if they differ with respect to one or several of the variations described above. Note that, in all cases, we only consider patterns to be semantically equivalent if the dependency structures of the patterns suggest that they in fact express the same meaning.

4.3.3.3 Annotation of unidirectional entailment

For annotating unidirectional entailment relations *within* each argument cluster, I select one representative of each equivalence class and generate all possible pairings of representatives. Based on the logical considerations described in Section 4.3.3.1, for annotating entailment relations holding *across* patterns from different argument clusters, I generate all pairs, for which the set of arguments of H is a subset of the set of arguments of T, as T can only entail H if it is at least as specific as H (assuming upward entailment, as motivated in Section 4.3.3.1). For each generated pair, a human annotator then decides, whether entailment holds or not. During annotation, we encountered cases, in which two patterns contradicted each other, e.g.,

person|SPOUSE person|SPOUSE divorced in date|TODATE
⊥ person|SPOUSE was married to person|SPOUSE until death in date|TODATE

However, these cases were rare and were not annotated separately, as the entailment graphs I construct only capture binary decisions (entailment, non-entailment). Following the manual annotation, I created entailment graphs based on the annotated entailment decisions, checked the transitivity of each graph and identified and removed inconsistencies.

4.3.3.4 Inference types

In order to investigate the types of inference underlying the entailment decisions, I carried out an additional annotation step. For this, I randomly selected a T/H pair of patterns from each pair of equivalence classes linked by an entailment relation, and analysed the nature of entailment, distinguishing among the following inference types:

Additional modifier The inferring pattern contains additional information expressed in the form of a modifier, e.g.,

person|SPOUSE was married to person|SPOUSE until death
→ person|SPOUSE was married to person|SPOUSE

Here, the additional information contained in the second pattern is that the couple was married *until death*.

Ontological The inference is drawn based on ontological knowledge, such as hyperonymy. For example,

person|WINNER's novel won prize|AWARD
→ person|WINNER's book won prize|AWARD

Here, the inference is drawn based on knowing that a novel is a kind of book. When annotating ontological relations, I used the WordNet ontology [Fellbaum, 1998] as reference.

Reasoning The inference is drawn by reasoning, e.g., based on general world knowledge, temporal knowledge, or logical inference. For example, the decision that

person|SPOUSE wife of person|SPOUSE
→ person|SPOUSE married person|SPOUSE

is based on inferencing that *wife* refers to the female role of a married couple. Note that in some cases, inference could be drawn based on a combination of different ontological relations. For example, according to WordNet, *wife* is a hyponym of *spouse*, which is linked to *marriage* via a member holonym relation, which in turn is derivationally related to *marry*. In other cases, reasoning goes beyond lexical inference, as in

was nominated for prize|AWARD lost to person|WINNER
→ person|WINNER won prize|AWARD

In some cases, inferences can even be drawn without any lexical hint, solely on the basis of syntactic reasoning, as in

person|WINNER's prize|AWARD book
→ person|WINNER won prize|AWARD

where the prize winning fact can be derived from the possessive suffix and knowledge about the second argument being a prize.

4.3.4 Resulting corpus

The annotation was conducted by three annotators with a background in linguistics. In step 1 (equivalence class identification), two annotators worked in parallel. Inter-annotator agreement was 0.88 for *marriage*, 0.83 for *acquisition*, and 0.88 for *award honor*, corresponding to almost perfect agreement. The main source of disagreement between annotators were words or expressions with ambiguous semantics. For example, annotators disagreed in cases like

person|SPOUSE eloped with person|SPOUSE
→ person|SPOUSE married person|SPOUSE

where the word *elope* commonly, but not always, refers to marrying secretly, or as to whether

person|WINNER author wins prize|AWARD
↔ person|WINNER writer wins prize|AWARD

where *author* and *writer* belong to the same synset in WordNet, but are slightly different in meaning.

Table 4.2 summarizes the statistics of the corpus. For each relation and for the complete corpus, it lists:

- the number of patterns entailing the target relation ("# of patterns")
- the number of equivalence classes ("# of ECs")

- the number of patterns contained in the largest equivalence class ("max EC size")
- the number of unidirectional entailment relations, i.e., entailment relations holding across equivalence classes ("edges (uni)")
- the number of bidirectional entailment relations, i.e., entailment relations holding across patterns belonging to the same equivalence class) ("edges (bi)")

Table 4.3 shows the distribution of inference types per relation, according to the distinction of types described in section 4.3.3.4 (MOD: additional modifier, ONTO: ontological, REAS: reasoning).

relation	# of patterns	# of ECs	max EC size	edges (uni)	edges (bi)
acquisition	161	77	32	122	1796
marriage	265	117	44	225	3262
award honor	412	224	49	977	4852
overall	838	418	49	571	9910

Table 4.2: Corpus statistics

relation	MOD	ONTO	REAS
acquisition	67%	13%	20%
marriage	68%	5%	27%
award honor	59%	11%	30%

Table 4.3: Distribution of entailment types

Visual representations of the generated graphs for *acquisition* and *marriage* are provided in Figures 4.3 and 4.4. The figures are simplified versions of the complete graphs in that, for the sake of clarity, each equivalence class is represented by a single pattern and semantically equivalent patterns are left out.

4.3.5 Summary and discussion

In this section, I described the creation of a a new linguistic resource, a gold standard dataset of textual entailment graphs based on relation extraction patterns for

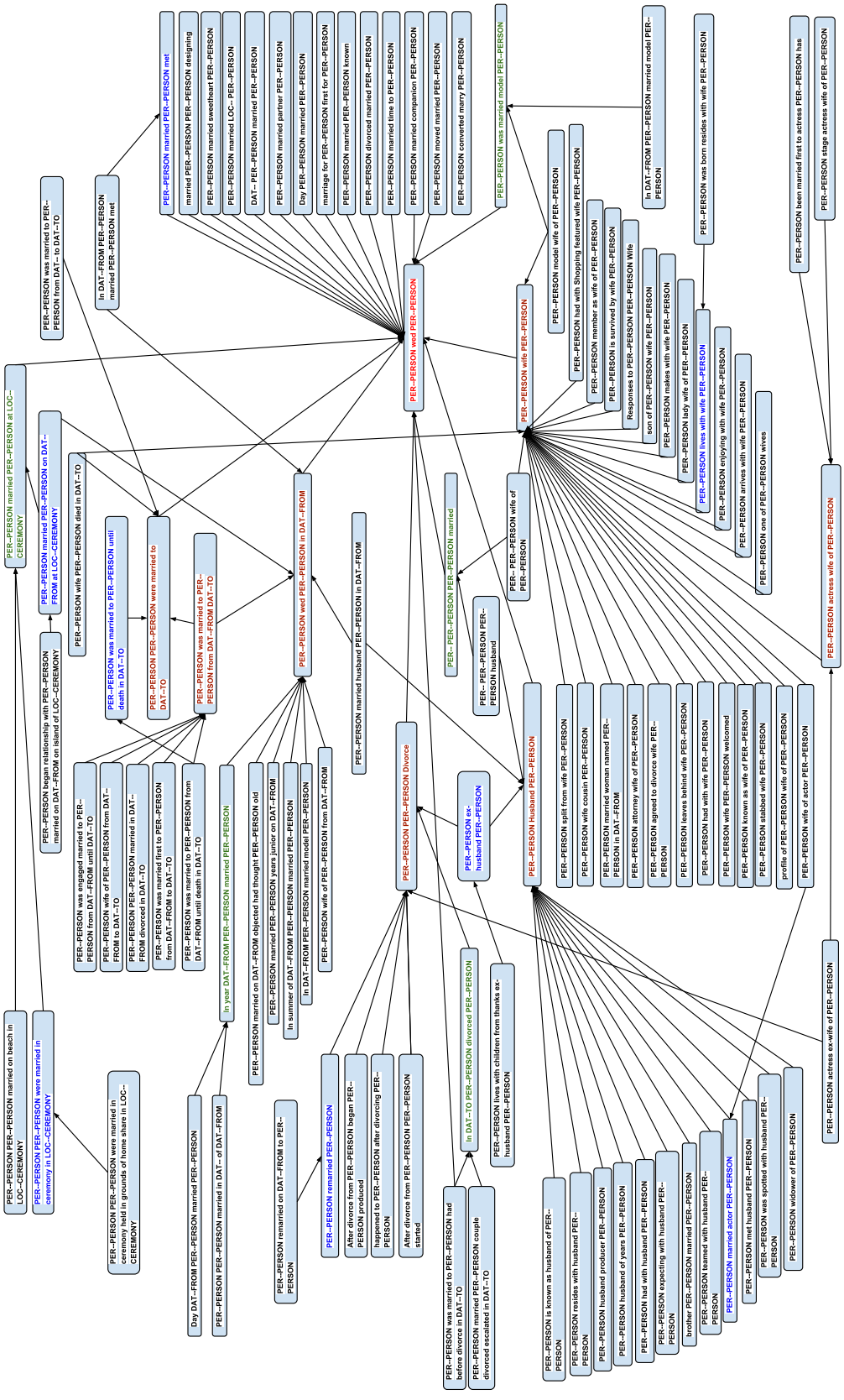


FIGURE 4.3: Entailment graph for *marriage* relation

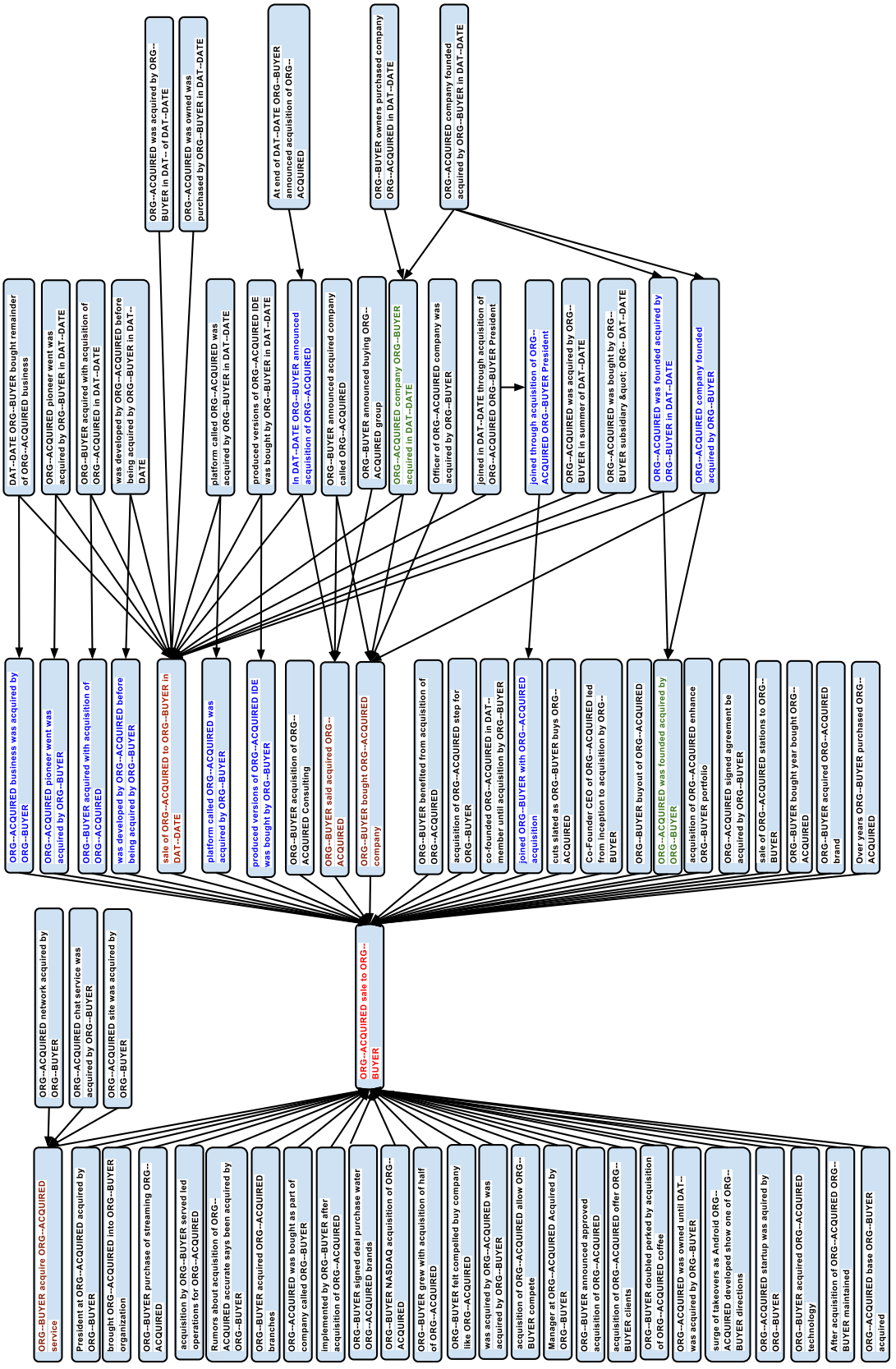


FIGURE 4.4: Entailment graph for *acquisition* relation

three semantic relations. The graphs in the newly created corpus differ from the ones created by Berant et al. [2010] in that they contain n-ary relations, with $n > 2$. In comparison to the entailment graph corpora by Bentivogli and Magnini [2014] and Kotlerman et al. [2015], the new graphs are more generic and have stronger expressiveness since they are not based on textual expressions, but on dependency structures containing semantic arguments of the target relations. The corpus can be utilized in several ways: First, as a gold-standard for evaluating both automatically created entailment graphs and textual entailment systems, in particular systems making use of syntactic information. Second, as a resource for fine-grained modelling of semantic context of the target relation. The graph structure can be utilized to identify relation mentions expressing information that is more specific than the target relation, e.g., instances of prize winners that are authors or those of marriages that ended in a divorce. The resource is publicly available under `http://sargraph.dfki.de/download.html`. Future work includes the extension of the corpus with additional relations and to further automatize the annotation procedure. In particular, by automatically identifying patterns that share syntactic structure the number of pairs to be manually annotated could be further reduced.

4.4 Automatic generation of pattern-based entailment graphs

4.4.1 Procedure

For creating entailment graphs based on extraction patterns in a fully automatic way, I apply the procedure described in Chapter 3. The realization of each of the steps with regard to creating pattern-based entailment graphs is summarized in the following:

Step 1: Identifying relevant expressions For identifying relevant expressions, I apply sentence-based fragment annotation (Section 3.4.4.1), considering the textual representation of the extraction patterns as a sentence. Each named entity is replaced by a variable expressing the respective named entity type and semantic role.

Step 2: Identifying modifiers Modifier identification is an optional step that could be applied for generating additional shorter patterns by removing modifiers. As in my work, I focus on structuring *existing* patterns rather than creating new ones, this step does not apply.

Step 3: Building fragment graphs Fragment graphs are generated according to Section 3.4.4.3. As the optional step of modifier annotation is not performed, all fragment graphs created are single-node graphs, each representing a distinct relation extraction pattern.

Step 4: Computing pair-wise entailment For recognizing entailment relations between individual T/H pairs of patterns, I use the all-pairs graph merger (Section 3.4.4.4), which computes entailment decisions for each possible pair. As entailment decision algorithm, I apply an RTE engine based on multi-level alignments [Noh et al., 2015]. For my experiments, I used the original implementation as well as an adapted version (for details, see Section 4.4.2).

Step 5: Optimizing the graph For creating transitive graphs from the intermediate graphs created in Step 4, I applied two different strategies: The first strategy is the simple graph optimizer described in Section 3.4.4.5. The second strategy is the global graph optimizer described in Section 3.4.4.5.

4.4.2 RTE engine

For recognizing entailment relations between individual T/H pairs, I make use of an RTE engine based on multi-level alignments. This RTE engine, referred to as *MultiAlign*, was implemented on top of the RTE platform EXCITEMENT [Magnini et al., 2014b] and achieved state-of-the-art performance on several RTE corpora [Noh et al., 2015]. I opted for this RTE system because it makes use of external knowledge resources and, unlike more recent systems based on neural networks [Bowman et al., 2015, Rocktäschel et al., 2015], is able to cope with the restricted amount of training data available for the task. *MultiAlign* uses shallow parsing for linguistic preprocessing and the Weka Hall et al. [2009] implementation of logistic regression for entailment classification. Features for the classifier are generated on the basis

of multi-level alignments using four aligners: a lemma aligner (aligning identical lemmas found in T and H), an aligner based on the paraphrase tables provided by the METEOR MT evaluation package Denkowski and Lavie [2014], and two lexical aligners based on Wordnet Fellbaum [1998][7] and VerbOcean Chklovski and Pantel [2004]. As output, it produces a binary decision (entailment, non-entailment) along with a computed confidence score.

For my experiments, I used the original *MultiAlign* implementation as well as an adapted version, in which I made the following changes to the WordNet aligner, addressing alignment recall with the first two, and precision with the third:

1. *T to H alignment* The original implementation retrieved WordNet rules for each lemma in H, with the lemma being the right-hand side of a rule, creating an alignment for each corresponding left-hand side lemma found in T. I adapted this to retrieve rules with a lemma in T being a left-hand side, and aligning matching right-hand side lemmas in H. This allowed for additional alignments not covered by the original algorithm, e.g., *marry* → *divorce*.

2. *All senses* The original implementation only considered the first sense of each WordNet entry. I extend this to cover all senses, enabling the retrieval of additional relevant alignments such as *wed* ↔ *marry*.

3. *No auxiliaries* Rather than retrieving rules for all words in T, I only consider rules for full verbs, nouns, and adjectives. This way, I particularly filter out rules for auxiliary verbs, which tend to produce many irrelevant alignments, especially when considering all senses of an entry.

A sample set of decisions produced by the RTE engine among candidate patterns for the *marriage* relation is depicted in Figure 4.5.

4.4.3 Graph optimization

Automatically derived entailment decisions may contradict each other. For example, as illustrated in Figure 4.5, the RTE engine correctly decides that PERSON$_1$ *<spouse of>* PERSON$_2$ entails PERSON$_1$ *<'s marriage to>* PERSON$_2$ and that

[7]Relations considered by the WordNet aligner: synonym, derivationally related, (instance) hypernym, member / part holonym, entailment, and substance meronym.

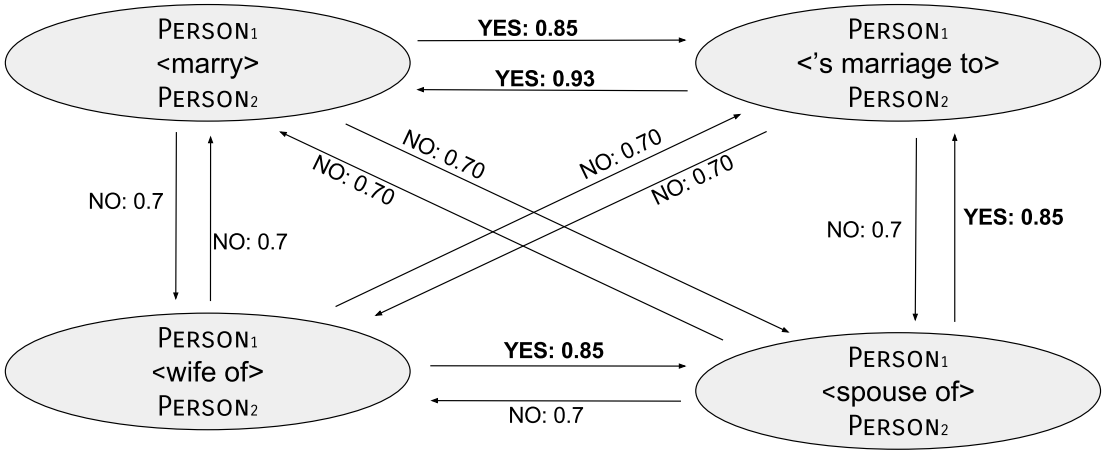


FIGURE 4.5: Sample set of EDA decisions (YES: entailment, NO: no entailment) with associated confidence

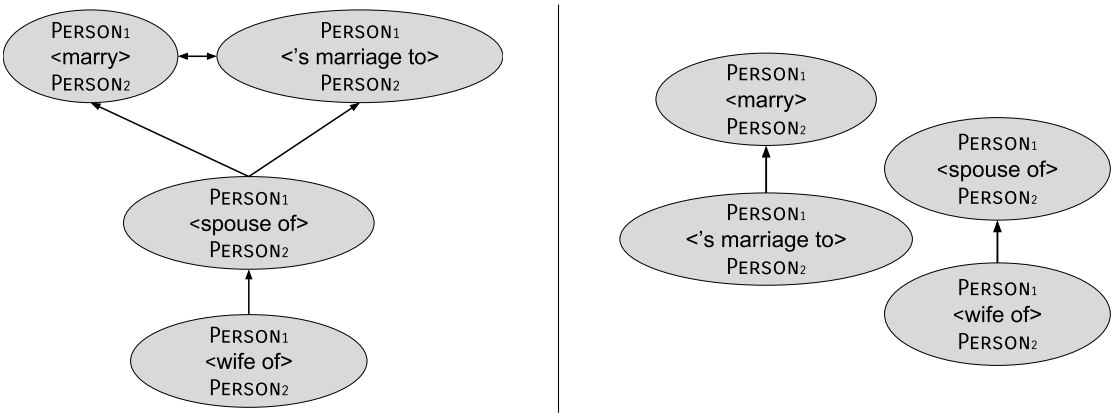


FIGURE 4.6: Sample outputs using greedy (left) and global (right) graph optimizer

PERSON$_1$ *<'s marriage to>* PERSON$_2$ entails PERSON$_1$ *<marry>* PERSON$_2$. However, it misses the entailment relation between PERSON$_1$ *<spouse of>* PERSON$_2$ and PERSON$_1$ *<marry>* PERSON$_2$, because the relationship between *spouse* and *marry* is not covered by the semantic resources underlying the decision. This leads to a set of decisions that is invalid given the transitivity of the entailment relation. For deriving a consistent transitive graph, I applied two different strategies: First, a simple greedy strategy and, second, the global graph optimization algorithm by Berant et al. [2012]. The greedy algorithm assumes a positive entailment relation for the node pair $< A, B >$ whenever any of the expressions in A entails any of the expressions in B, with the confidence of the RTE engine exceeding a pre-defined threshold, and adds missing entailment edges to ensure transitive closure. Berant et al. [2012]'s algorithm searches for the best graph under a global transitivity constraint, approaching the optimization problem by Integer Linear

Programming. The selection of the optimization strategy is crucial, as illustrated in Figure 4.6, which shows sample outputs from each of the two strategies for the decisions in Figure 4.5.

4.5 Evaluation

In this section, I evaluate the usage of knowledge represented by entailment graphs for the selection of relation extraction patterns. Following the procedure described in Section 4.2, I select patterns based on both gold-standard and automatically created entailment graphs and compare the extraction performance achieved with these patterns to the performance achieved with other selection methods, including the semantic filter proposed by Moro et al. [2013]. Extraction performance, i.e., the quality of the selected pattern set, is measured by computing precision and recall over a set of manually annotated relation mentions. In this section, I describe the experimental setup, the evaluation data created for the experiments, and the evaluation results achieved with the gold-standard dataset (Section 4.3) and the automatically generated entailment graphs (Section 4.4).

4.5.1 Experimental setup

For evaluating the proposed method on the relation extraction task, I conducted experiments on two freely available datasets: the

TEG-REP dataset described in Section 4.3 and the FB15k-237 dataset Toutanova et al. [2015]. On the TEG-REG corpus, I carried out a detailed evaluation of several pattern filtering strategies with respect to two semantic relations. On the FB15k-237 corpus, I evaluate the applicability of the proposed method and generated models to structuring extraction patterns for other semantic relations.

4.5.1.1 TEG-REP

Experiments were conducted based on the pattern set underlying the gold-standard corpus created in Section 4.3. The TEG-REP corpus contains gold-standard entailment graphs created from patterns for three relations typically considered in RE tasks: *marriage*, *acquisition*, and *award honor*. The TEG-REP corpus is the

only available corpus of pattern-based entailment graphs and particularly suitable for the evaluation, because it allows for a comparison of patterns selected based on both manually and automatically created entailment graphs.

For my experiments on this corpus, I randomly divided the patterns (around 500 per relation) into two equally-sized portions, one for creating training data for the RTE engine, and one for evaluating pattern selection methods. For creating the evaluation dataset, I applied all extraction patterns in the evaluation split to 14.5 million ClueWeb sentences Lemur Project [2009] linked to Freebase entities Gabrilovich et al. [2013], and manually annotated around 3000 of the extracted mentions. The annotation was done by three annotators. About 10% of the mentions were annotated by two annotators in parallel, who achieved a very high interannotator agreement (Cohens Kappa > 0.9). The remaining mentions were annotated by a single person. A mention was annotated as being correct if we found evidence for the target relation between the entities in the mention to hold. Evidence was drawn either from the source sentence itself, or, in cases were the source sentence did not express the relation explicitly, from external resources such as Freebase or Wikipedia. Based on this dataset, I evaluated several strategies for selecting patterns, measuring precision and recall over the annotated relation instances.

The experiments on the TEG-REP patterns are based on the relations *marriage* and *acquisition*[8]. For the experiments, I split the evaluation set into a development set for optimizing the graph parameters and a test set for the final evaluation. To ensure that development and test split are comparable, with regard to both the quality of patterns and the number of annotated mentions I applied the following splitting procedure: For either relation, I first split the pattern set into two parts based on each pattern's accuracy with regard to the annotated mentions: one set of "good" patterns (accuracy of 0.5 or above) and one set of "bad" patterns (accuracy below 0.5). In each set, I ordered all patterns based on the number of annotated mentions (in descending order). Iterating over the two lists, I then created a development and a test split by checking the number of mentions for each pattern entry and assigning each pattern to the set with the lower number of assigned mentions.

[8]I did not evaluate the *award honor* relation because the vast majority ($> 98\%$) of mentions extracted using these patterns were correct, which would not have allowed for a meaningful evaluation.

In my experiments, I applied the following strategies for selecting patterns and measured performance over the annotated relation mentions in the test dataset.

- *All patterns* All patterns from the test split (baseline).
- *Lexical semantic filter [Moro et al., 2013]* Patterns selected using the lexical semantic filter.
- *Pair-wise entailment (MultiAlignAdapted)* Patterns selected based on pair-wise entailment decisions using the model of *MultiAlignAdapted*.
- *Entailment Graph (MultiAlignOriginal / MultiAlignAdapted)* Patterns selected based on entailment graphs generated with the original / adapted *MultiAlign* implementation.
- *Entailment Graph (TEG-REP gold standard)* Patterns selected based on gold-standard entailment graphs from the TEG-REP corpus.

For selection based on entailment graphs, according to the general procedure described in Section 4.2, the step of generating entailment graphs is followed by the identification of a base pattern per relation, which expresses the target relation explicitly. As base patterns, I selected the patterns [PERSON$_1$ <marry> PERSON$_2$] (for *marriage*) and [ORGANIZATION$_1$ <acquire> ORGANIZATION$_2$] (for *acquisition*). Pattern selection is then performed by reading all patterns entailing the base patterns from the graph. In order to investigate the benefits of the graph structure, I compared the results to those achieved when computing entailment relations at a pair-wise level, i.e., using the base pattern for the relation as H and all other candidate patterns for the relation as T. For lexical semantic filtering, I applied the approach by Moro et al. [2013], who identify relation-relevant word senses based on lexical semantic subgraphs derived from BabelNet and filter out patterns not containing any relevant word sense. Based on a parameter k, they consider a word sense to be relevant if it is at most k-step distant[9] to the core word sense for the target relation.

4.5.1.2 FB15k-237

As training the RTE models requires appropriate training data, which may not be available, I ran additional experiments to investigate if the models trained on one

[9] k=1 includes the most relevant word senses only.

relation are general enough to be used for computing entailment relations among pattern candidates for other semantic relations. To this end, I used the FB15k-237 corpus [Toutanova et al., 2015], which contains knowledge-base relation triples and textual mentions of Freebase entity pairs. For my experiments on this corpus, I generated candidate patterns by extracting the first 1000 tuples matching a particular relation from the pattern files in the corpus, and then extracting all patterns linking any of the tuples in the textual triples used by Toutanova and Chen [2015]. This way, our candidate pattern set contains both patterns expressing the target relation as well as patterns expressing other relations. For creating the entailment graph, I converted all patterns into a textual representation, removed patterns with no lexical item, and, from the remaining patterns, built an entailment graph using the EDA model trained on the *marriage* relation and the parameter setting derived based on the TEG-REP corpus. For evaluating the result, I selected 10 relations, defined a base pattern for each of them, and checked, for each pattern in the graph, whether it entailed the base pattern according to the graph structure and whether the entailment decision was correct based on the semantics expressed by the pattern.

As in this setting, I evaluated the entailment relations expressed by the pattern graph rather than the usage of the patterns for relation extraction, the results are not directly comparable to the figures obtained on the TEG-REP corpus, but still allow for an assessment of the quality of the selected patterns.

4.5.2 Results and discussion

4.5.2.1 TEG-REP

Tables 4.4 and 4.5 summarize the results on the TEG-REP corpus, showing for each of the following pattern selection methods, the number of correct mentions extracted using the selected patterns (TP), and the computed precision (P), recall (R), and F1 scores.

For the lexical semantic filter method, I experimented with different levels of k and noted down the value achieving the best F1 score. The results in the table were produced setting k to 1 for the *marriage* relation and k to 5 for the *acquisition* relation. For the RTE-based methods, I experimented with the two different graph

Configuration	**TP**	**Precision**	**Recall**	**F1**
All patterns	218	0.15	1.00	0.27
Lexical semantic filter Moro et al. [2013]	160	0.61	0.73	0.67
Pair-wise entailment (MultiAlignAdapted)	121	0.97	0.56	0.71
Entailment Graph (MultiAlignOriginal)	128	0.96	0.59	0.73
Entailment Graph (MultiAlignAdapted)	148	0.96	0.68	**0.80**
Entailment Graph (TEG-REP gold standard)	150	0.96	0.69	0.80

TABLE 4.4: Results for *marriage* relation (TEG-REP corpus).

Configuration	**TP**	**Precision**	**Recall**	**F1**
All patterns	259	0.30	1.00	0.46
Lexical semantic filter Moro et al. [2013]	251	0.30	0.97	0.46
Pair-wise entailment (MultiAlignAdapted)	128	0.82	0.49	0.62
Entailment Graph (MultiAlignOriginal)	136	0.81	0.53	0.64
Entailment Graph (MultiAlignAdapted)	242	0.59	0.93	**0.73**
Entailment Graph (TEG-REP gold standard)	127	0.82	0.49	0.62

TABLE 4.5: Results for *acquisition* relation (TEG-REP corpus).

optimization strategies and, for each of them, with different confidence threshold values, and optimized these parameters based on the development split. The figures in the table show the results achieved on the test split using the parameter setting optimized on the development set: the greedy optimization strategy with thresholds of 0.71 (*MultiAlignOriginal*) and 0.77 (*MultiAlignAdapted*) for the *marriage* relation and thresholds of 0.74 (*MultiAlignOriginal*) and 0.75 (*MultiAlignAdapted*) for the *acquisition* relation. On our data, the greedy edge selection strategy produced better results than the global graph optimizer for both relations. This was because the global strategy, even with low confidence thresholds, was more restrictive and removed too many edges from the graph, thus yielding lower recall figures.

For both relations, the best overall results were achieved using the proposed method based on entailment graphs, generated automatically applying the adapted RTE engine. The results show that entailment-based pattern selection is in fact more powerful than the lexical semantic filter. It selects patterns yielding a much higher precision, because it is able to successfully filter out non-entailing patterns, such as [PERSON$_1$ <be in relationship with> PERSON$_2$] for the *marriage* relation, which are wrongly selected using the lexical semantic filter. For the *marriage* relation, the results not only show that our RTE engine adaptations yielded a much

higher recall (with almost no loss in precision) than the original implementation (thanks to an increased number of relevant alignments), but also that pattern selection can in fact benefit from the graph structure: Entailment graphs created using *MultiAlignAdapted* achieved much better performance than a selection based on pair-wise entailment computation using the same RTE model. This was due to a higher recall achieved because the graph structure allowed the algorithm to identify entailment relations that involved the combination of several inference steps and were missed when applying RTE in a pair-wise manner. An example is the relationship between *wife* and *marry*, as shown below:

$$wife \xrightarrow{\substack{hyper-\\nym}} spouse \xrightarrow{\substack{member\\holonym}} marriage \xrightarrow{\substack{deri-\\vation}} marry$$

For the *acquisition* relation, I noticed that the lexical semantic filter performed quite poorly on our corpus. The relation requires a large k-value, i.e., $k >= 5$, since there are many ways in which an acquisition can be described. A company can for instance devour, take-over or purchase another company. Each increase of k allows many additional content words, thus increasing the danger of inappropriate ones. An example is [ORGANIZATION$_1$ trademark of ORGANIZATION$_2$]. Although an acquired company may persist as a brand of its new owner, *trademark* does not generally express a take-over. Although the semantic filter proposed by Moro et al. [2013] is certainly useful as it can provide valuable hints and can be applied without manually annotating training data, it is not powerful enough to discriminate content words as to whether they provide strong evidence for an acquisition or not.

Also patterns selected based on the entailment graph gold-standard performed surprisingly low on the *acquisition* relation. Here, recall was affected negatively because some of the non-entailing patterns that were filtered out were in fact able to extract correct instances with decent precision. In particular, patterns expressing a planned acquisition, such as [ORGANIZATION$_1$ plans to purchase ORGANIZATION$_2$], [ORGANIZATION$_1$ is to acquire ORGANIZATION$_2$], or [ORGANIZATION$_1$ announced intention to acquire ORGANIZATION$_2$] extracted many correct mentions, as the acquisition in fact happened at a later stage. Nevertheless, filtering out these cases is correct from a semantic point of view, even if many of the reported plans or attempts concerning acquisitions later become a reality. A decrease in recall in such cases may well be a side effect of the way we measure. If our news corpus

Relation	**Base pattern**	**Precision**	**Recall**	**F1**
award-award_honor-ceremony	win at	0.58 (0.31)	0.73 (1.00)	**0.65** (0.47)
base-locations-continents-countries_within	country in	0.67 (0.57)	0.84 (1.00)	**0.75** (0.59)
education-education-major_field_of_study	degree in	1.00 (0.47)	0.49 (1.00)	**0.65** (0.64)
film-...-film_regional_debut_venue	premiere at	0.73 (0.13)	0.89 (1.00)	**0.80** (0.23)
film-performance-film	star in	0.97 (0.57)	0.53 (1.00)	0.69 (**0.72**)
organization-place_founded	founded in	0.83 (0.16)	0.83 (1.00)	**0.83** (0.27)
people-marriage-location_of_ceremony	marry in	0.80 (0.07)	1.00 (1.00)	**0.89** (0.12)
people-marriage-spouse	marry	1.00 (0.15)	0.58 (1.00)	**0.73** (0.26)
people-person-place_of_birth	born in	1.00 (0.41)	0.84 (1.00)	**0.91** (0.58)
people-place_of_burial	buried at	1.00 (0.33)	0.71 (1.00)	**0.83** (0.50)

TABLE 4.6: Entailment graph based pattern selection (vs. baseline) for relations in FB15k-237 corpus.

stems from a certain time interval we may not have the reports on the completed take-over from which we could correctly extract the fact.

With the *acquisition* relation, there was also a notable increase in recall when using the adapted version of *MultiAlign*. Here, as with the *marriage* relation, the adaptations led to additional alignments, for example between *takeover* and *acquisition*, resulting in additional patterns being recognized as entailing the base pattern.

Precision of the gold-standard patterns achieved on the *acquisition* relation was much lower than for the *marriage* relation, due to patterns annotated as entailing in the TEG-REP corpus, which extracted comparably many incorrect instances. One such pattern is [ORGANIZATION$_1$ <takeover of> ORGANIZATION$_2$], which yields low precision values because it often occurs in sentences expressing irrealis moods, such as *the proposed Microsoft takeover of Yahoo* or *is a Pfizer takeover of BMS realistic?*, and because of its generality often extracts non-company entities, e.g., *Republican takeover of Congress*. Detecting the embedding of correct patterns in irrealis contexts is a largely unsolved problem and calls for the development of general methods for recognizing nonfactual modalities along the lines of the NegEx algorithm for detecting negations in medical texts [Chapman et al., 2001] and its later extensions.

4.5.2.2 FB15k-237

The results of my experiments on the FB15k-237 corpus are presented in Table 4.6, showing the performance of the pattern selection method based on entailment graphs (with the adapted *MultiAlign* implementation) compared to a simple baseline (all patterns). The results show that, even using an RTE model trained on a completely different semantic relation (*marriage*), our method achieves decent performance on selecting meaningful patterns for a wide range of relations. The figures in the table were produced with the simple graph optimization strategy, but the global graph optimizer performed very similar on this dataset, achieving the same results for eight out of the ten relations. It performed worse on the *award-award_honor-ceremony* relation (F1: 0.48), and better for the *base-locations-continents-countries_within* relation (F1: 0.91). Nevertheless, when dealing with larger numbers of patterns, the global graph optimizer should be the method of choice, as it is less prone to semantic drift.

Note that the selection of the base pattern is crucial for the overall result: For example, whereas selecting [PERSON$_1$ <win> AWARD$_1$] as base pattern for the award relation achieves very high precision, choosing [PERSON$_1$ <receive> AWARD$_1$] instead yields much better recall. This exhibits a major advantage of the hierarchical structure: It allows the user to influence the trade-off between precision and recall depending on the task to be solved by selecting base patterns accordingly.

4.6 Summary and discussion

I presented an approach for structuring relation extraction patterns using entailment graphs and evaluated the usefulness of both manually and automatically generated graphs for pattern selection. I also showed how existing RTE technology can be applied and adapted to generate pattern-based entailment graphs automatically. In particular, I employed and improved an existing alignment-based entailment classifier from the EXCITEMENT Open Platform, which makes use of external knowledge resources, and experimented with different graph optimization strategies. The classifier was trained on a manageable amount of annotated patterns for a single semantic relation, resulting in a generic model that was shown to produce valid entailment decisions for a wide range of other semantic relations. The experimental results exhibit the benefits of structuring and selecting patterns

based on entailment graphs: They suggest that meaningful pattern-based entailment graphs can be constructed automatically and that the derived knowledge is in fact valuable for selecting useful relation extraction patterns. In particular, entailment graph based filtering can help achieve higher precision than methods which do not take into account the asymmetric nature of semantic relations. Future work includes further improvements of recall through integration of knowledge that goes beyond lexical entailments (e.g., syntactic information and semantic roles), which may be essential for achieving a high level of performance on other relations. Another interesting direction for future work is the generation of entailment graphs across several semantic relations.

Chapter 5

Entailment Graphs for Email Categorization

5.1 Introduction

The second NLP task addressed in this thesis is the task of categorizing customer emails. In customer support, the main task is to provide an informative reply to the request sent by a customer. Reducing the time to produce this reply is essential in order to save resources.

As customer requests often contain issues that have been described by other customers before, an effective way of reducing the time needed to produce the reply is to re-use text blocks formulated for a similar request. This approach requires creating a database of previously described problem cases, associated to corresponding replies. The main task to be automatized then is to find, for an incoming email, a matching problem case in the database. With an email being a text document, this task is commonly approached as a text categorization task, where the goal is to assign input documents (here: customer emails) to predefined categories (here: problem cases).

In this chapter, I investigate the usage of entailment graphs in the context of categorizing customer emails. I start out with an analysis of a dataset of manually categorized German customer emails, in which the role of textual inference for assigning matching categories is investigated. Based on the analysis, I present an approach that uses entailment graphs for email categorization. The general

idea is to build entailment graphs representing semantic relationships between text expressions extracted from emails and category descriptions, and use the generated knowledge as input when building a classifier model for categorizing new emails. For evaluating the proposed approach, I create entailment graphs based on the outcome of the data analysis and measure the effect of using the semantic knowledge expressed by the graph for categorization. In addition, I construct entailment graphs automatically, making use of entailment engines provided by the EXCITEMENT Open Platform and the technology for creating entailment graphs described in chapter 3, and evaluate their usefulness for the email categorization task. A particular focus in this evaluation is on how the usage of different types of inference knowledge affects categorization performance.

5.2 Data analysis

In this section, I describe the analysis of a real-world dataset of manually categorized customer emails written in the German language. Investigating the nature of textual inference in this data, I lay the ground for developing an inference-based email categorization system. This is the first analysis of this kind on German data. I describe the details of the analysis, compare the results to previous analyses on English data and present major findings. A summary of this analysis was published in Eichler et al. [2014].

5.2.1 Motivation

Human language allows us to express the same idea in various ways. Recognizing that the meaning of one text can be inferred from the meaning of another can be of help in many natural language processing applications. One such application is the assignment of emails to predefined categories. A typical situation in customer support is that many customers send requests describing the same issue. Recognizing that two different customer emails refer to the same problem can help save resources, but can turn out to be a difficult task. Customer requests are usually written in the form of unstructured text. When automatically processing unstructured natural language text, we are often faced with the issue of variability: Different speakers of a language express the same meanings using different linguistic forms. These differences can depend on factors such as the purpose in

communication, the relationship between speaker and hearer, the production circumstances, or the speaker's demographic affiliations [Reppen et al., 2002]. While some differences, like pronunciation, are only relevant in spoken language, most of the linguistic choices, including morphology, word choice, and grammar, also affect the way a speaker produces written texts. Due to this, customer requests decribing the same issue are characterized by large differences in wording across different customers. There are, in fact, cases where two sentences expressing almost exactly the same meaning do not share a single word. An example, taken from a real-world dataset of customer emails sent to the support center of a German multi-media company, is given in the following:

1. *Wenn ich Daten im .mpg-Format öffne, stimmt die Tonspur nicht mit dem Film überein.* [When opening data in .mpg format, the audio track does not match the film.]

2. *Bild und Ton einer Videodatei sind asynchron.* [Image and sound of the video file are asynchronous.]

Detecting the semantic equivalence or relatedness of two texts may require textual inference steps at various levels. At the level of individual tokens, differences in wording can occur with regard to inflection, derivation, or lexical semantics. In German language text, an additional problem for automatic processing is the usage of compounds written as single words, e.g., *Tonspur* [audio track] in the example above, which needs to be split into its components in order to automatically recognize its semantic relatedness to the word *Ton* [sound].

However, our example shows that matching the two sentences requires more than comparing the semantics of individual words. For example, even if we assume the availability of a lexical resource containing relationships between lexical items such as *Film* [film] and *Video* [video], this information will not be enough to determine that the expression *Daten im .mpg-Format* [data in .mpg format] refers to a *Videodatei* [video file], and that the expression *stimmt die Tonspur nicht mit dem Film überein* [audio track does not match the film] is equivalent to *Bild und Ton [...] sind asynchron* [image and sound are asynchronous]. This requires the semantic comparison of and inferencing based on larger text units, such as phrases or sentences.

In this section, I lay the ground for developing an email categorization system based on textual inference by analyzing a large set of manually categorized customer emails. The dataset, which is described in detail in section 5.2.3, consists of several hundred emails associated to categories representing issues reported by customers. Each category is linked to a textual description summarizing or generalizing the issue, which was created by a domain expert from the company's support staff.

The analysis of this dataset aims at finding answers to the following questions:

1. Which types of inference steps are involved for semantically deriving the category description from the email text and how are these inference phenomena distributed in real-world data?
2. What text representation is most appropriate for the task of categorizing German customer emails?

For answering these questions, I systematically compare each email text to the description of its associated category and investigate the nature of the inference steps involved, identifying the inference steps that would be required to semantically derive the category description from the email text. I also analyzed, which text representation would be required for the relevant inference steps. The outcome of the analysis does not only help decide which existing NLP tools and resources to integrate in an inference-based email categorization system, but also provides valuable hints as to non-existing tools and resources that may be needed in addition.

5.2.2 Related work

The decision that a textual expressions can be inferred from another may be made based on various levels of reasoning, such as lexical, syntactic, or morphological cues, or logical constraints. In connection with RTE research, several groups have analyzed existing (English) datasets in order to investigate the nature of textual inference with regard to these different linguistic levels.

Vanderwende and Dolan [2005] analyzed the test set of the first RTE Challenge [Dagan et al., 2006] to find the percentage of cases in which syntactic analysis alone

(with optional use of a thesaurus for the lexical level) suffices to decide whether or not entailment holds. Their results show that about one third of the test items can be handled by syntax (including phenomena such as argument assignment, intra-sentential pronoun anaphora resolution); roughly half of the test items can be handled by syntax plus a general purpose thesaurus. Bar-Haim [2010] extends Vanderwende and Dolan [2005]'s work by introducing two levels of entailment, lexical and lexical-syntactic, and analyzing the contribution of each level and of individual inference mechanisms within each level. The lexical level captures knowledge about lexical-semantic and morphological relations, and lexical world knowledge; the lexical-syntactic level additionally captures syntactic transformations, lexical-syntactic inference rules and co-reference. For each level, he defines an entailment model and evaluates its performance over a sample from the RTE-1 data set. Measuring the contribution of various inference mechanisms at the two defined levels of entailment, he concludes that the main contributors are paraphrases and syntactic transformations.

Garoufi [2007] proposes a scheme for manually annotating textual entailment data sets (ARTE), viewing the entailment task in relation to three levels: Alignment, Context and Coreference, according to which 23 different features for (positive) entailment annotation are extracted. Applying the ARTE scheme to a subset of the RTE-2 test set, Garoufi [2007] concludes that reasoning, which appeared in about two thirds of the cases, is the most frequent feature, indicating that a significant portion of the data involves deeper inferences. A similar conclusion is drawn by Clark et al. [2007], who analyzes 100 positive entailment pairs from the RTE-3 dataset and identifies the type of knowledge required to recognize entailment in the sample cases. Based on the frequency of the different entailment phenomena, they state that only a few entailments can be recognized using syntactic matching and basic lexical knowledge (synonyms, hypernyms), but that the vast majority of cases require significant world knowledge.

Sammons et al. [2009] model the entailment process as one of manipulating text and hypothesis to be as similar as possible. They first identify parts in T that match parts in H, and then identify connecting structure. Both positive and negative examples are tagged based on a set of pre-defined phenomena. Applying their procedure to 210 T/H pairs from the RTE-5 dataset, their statistics reveal that the most frequent phenomena are coreference, simple rewrite rules, lexical relations, and implicit relations.

Volokh and Neumann [2011] analyze a subset of the RTE-7 development data to measure the complexity of the RTE task. They divide the T/H pairs into three different classes, depending on the type of knowledge required to solve the RTE problem: In class A, the relevant information is expressed with the same words in both T and H. In class B, T contains words that are synonyms to words used in H. In class C, recognizing entailment between H and T requires the use of logical inference and/or world knowledge. They conclude that for two thirds of the data a good word-level analysis is enough, whereas in the remainder of the data the entailment decision is based on more sophisticated entailment phenomena, such as logic or world knowledge.

The results of previous analyses are difficult to compare and do not provide a clear picture, as they are based on different corpora and different definitions of entailment phenomena. However, even though the relative importance attributed to different aspects of textual inference varies depending on corpus and setup, all analyses suggest that recognizing textual entailment cannot be solved at the level of lexical inference and syntactic transformation alone. Rather, identifying semantic relationships between larger textual expressions is a central challenge when addressing textual entailment and often requires to resolve the meaning of expressions using logic and world knowledge.

As the approaches described above, my analysis aims at measuring the contribution of inference phenomena, comparing and analyzing matching parts in T and H as done by Sammons et al. [2009]. Viewing the email as the text T whose content needs to be mapped to the content of the category description H, I identify corresponding text portions in T and H and identify the phenomena relevant for the mapping. My annotation scheme is similar to the one proposed by Garoufi [2007]. However, I look at the problem from the angle of NLP tools and resources: Phenomena are defined in such a way that they correspond to mappings that can be performed by a particular tool or resource, which helps estimate their expected contribution to the entailment decision. A particular focus of the analysis is the comparison of different text representations. I therefore distinguish between phenonema that can be solved at the level of individual words and phenonema that span larger text expressions, thus calling for more sophisticated linguistic processing. My analysis also differs from previous work in that it is based on a different data type (customer request as compared to news) and a different language (German as compared to English).

5.2.3 Dataset

The data analysis was carried out on a dataset consisting of a set of customer emails and a set of associated categories. The emails contain customer requests sent to the support center of a multimedia software company and mainly report issues concerning the products offered by this company. Each email was manually assigned to a matching category by a customer support agent (a domain expert). In addition, the support agent marked the "relevant text" of the email, i.e., the portion of the text considered relevant for categorization. The categories, predefined by the data provider, represent previously identified problems reported by customers. Each category is associated to a text description of the category, summarizing the common issue reported in the emails assigned to the category. In most cases, the description of the category was formulated based on the first customer email sent describing the problem. All emails and category descriptions in the dataset are written in German. As is common for this type of data, many emails in the dataset contain spelling mistakes, grammatical errors or abbreviations, which make automatic text processing difficult. An anonymized version of the dataset is publicly available[1]. The anonymization step was performed to eliminate references to the data provider and personal data about the customers. During this step, the data was transferred into a different product domain (online auction sales).

Nevertheless, the anonymized version is very similar to the original one in terms of language style, including spelling errors, grammatical errors, anglicisms, abbreviations, and special characters. The data analysis described in the following was done on the original (confidential) dataset. However, as the original dataset is confidential, the data examples used for illustration purposes are either adapted or taken from the anonymized dataset. A data sample from the anonymized set is given in Table 5.1.

In the analysis, I manually compare the email texts to the descriptions of their associated categories in order to investigate the nature of the inference steps involved. In order to reduce the complexity of the task, the analysis is based on the subset of categories, where the category text describes a single problem (a single H, speaking in RTE terms). In addition, I removed emails for which I was not able to semantically relate the category description to the email text. The

[1] `http://www.excitement-project.eu/attachments/article/97/omq_public_email_data.zip`

Category ID	*Category description*	*Associated email*
76	Fehlercode -9 beim Start des Programmes	Habe Produkte im IT & OFFICE Katalog 12 first-class auf einen neuen Rechner kopiert. Die Software startet auf dem neuen Rechner nicht, sondern bricht den Start ab mit Angabe des Fehlercodes -9. Eine zweimalige Neuinstallation hat nicht geholfen Was tun?

TABLE 5.1: Data example from the anonymized dataset.

reduced dataset used for the analysis consists of 369 emails associated to 25 categories. The email lengths vary between 2 and 1246 tokens. Category descriptions usually consist of a single sentence or a phrase. In five cases, it's just a phrase lacking the verb, in two cases, the description consists of a sentence and a phrase.

5.2.4 Types of representation

In the analysis, I manually compare the text of each email to the text description of its associated problem category in order to identify the required textual inference steps for recognizing that email text and problem description are semantically related. For the analysis of inference steps involved, I distinguish between two levels of inference: lexical semantics and compositional semantics.

At the lexical level, I distinguish two different types of text representation: First, the *bag-of-tokens* representation, where both the email text and the category description are represented as the set of content word tokens contained in the respective text. I consider a *token* to be a string of letters separated from other tokens by space or punctuation. This representation can easily be retrieved using a tokenizer and a list of function words or a part-of-speech tagger. Second, the *bag-of-terms* representation, where a *term* can consist of one or more content tokens occurring consecutively. Deriving the bag-of-terms representation would require the usage of a tool to detect multi-token terms. At the lexical level, following Bar-Haim [2010], I assume that entailment holds between T (the email) and H (the category description) if every token (term) in H can be matched by a corresponding entailing token (term) in T.

At the level of compositional semantics, I represent each text as the set of complex expressions in which the content terms are embedded. By *complex expression* I refer to any combination of two or more words linked syntactically and semantically. At this level, I assume that entailment holds between T and H if every term in H is part of at least one complex expression that can be matched by a corresponding entailing expression in T.

Choosing the three different types of representation described above, the underlying question is: How much more content can we cover by choosing a more complex representation? For example, in the case of terms as compared to tokens: Does the email text contain a term that does not appear in the category text but has a semantic equivalent there (e.g., *MPEG-Datei* [MPEG file] vs. *MPG Video Clip* [MPG video clip]). Or, in the case of syntactic relations: Does the email text contain a syntactic relation that is not contained in the category text but has a semantic equivalent there, e.g. *ich bekomme einen Fehler* [I receive an error] vs. *eine Fehlermeldung erscheint* [an error message pops up]. For each of the three different types of representation (token, term, complex expression), I distinguish various inference steps, described in the following sections.

5.2.5 Inference steps relating to lexical semantics

At the lexical level, I distinguish among spelling, inflection, derivation, compounding, lexical semantics at the token level and lexical semantics at the term level. The distinction above was made based on the assumption that for each of these steps a different NLP tool or resource is required. For spelling, which is usually not considered when analyzing RTE phenomena, but plays an important role when dealing with user-generated data such as customer emails, a spellchecker is needed to correct misspelled words. For inflection, we need a lemmatizer to retrieve the lemma. For derivation, we need a tool or lexicon to detect derivationally related lemmata, e.g., Zeller et al. [2013] for German. For compounding, we need a compound splitter, and for lexical semantics, we need a lexical-semantic net, such as GermaNet [Hamp and Feldweg, 1997] for German. For term-level lexical semantics, we also need a tool for detecting multi-word terms, such as the system described by Eichler and Neumann [2010]. We also distinguish between token level and term level lexical semantics.

5.2.6 Inference steps relating to compositional semantics

At the level of compositional semantics, I consider inference steps involving complex expressions. These steps go beyond the lexical level and would require the usage of at least a syntactic parser for detecting word dependencies and a tool or resource for recognizing the semantic relationship between two complex expressions. At this level, I also record the frequency of particle verbs, negation, and light verb constructions. These phenomena are recorded separately, as a pre-analysis of the data showed that they occur quite frequently in the data.

Particle verbs are important when processing German because, unlike in English, they can occur both as one token or two, dependending on the syntactic construction, in which they are embedded, e.g., *aufnehmen* versus *nehme [...] auf* [(to) record]. Recognizing the scope of negation can be required in cases where negation is expressed implicitly in one of the sentences, e.g., *A und B sind nicht synchron* [A and B are not synchronous] vs. *Es kommt zu Versetzung zwischen A und B* [There is a misaligment between A and B]. By *light verbs*, a term coined by Jespersen [1965], I refer to verbs with little semantic content of their own, forming a linguistic unit with a noun or prepositional phrase, for which a single verb with a similar meaning exists, e.g., *Meldung kommt* [message appears] vs. *melden* [notify].

Obviously, this list of special cases is not exhaustive and could be extended to other interesting phenomena.

5.2.7 Setup

The data analysis was carried out by two people separately (one of them the author of this thesis), who analyzed each assignment of an email E to a category C based on predefined analysis guidelines. The two annotators first analyzed the data separately. The results were then compared, discussed and unified.

Viewing the categorization task as an RTE problem, the assumption was that the email text would usually entail the category description. Therefore, the description of C was considered to be the hypothesis (H), the text content of E to be the text (T). For each of the text representation types described above, the task of the annotators was then to find, for each expression in H, a semantically equivalent or entailing expression in E. If such an expression was found, all involved inference

steps required to derive one expression from the other were to be noted down in an annotation table.

For example, for an email text containing the sentence *Wenn ich eine CD brenne, bricht das Programm ab mit einer Fehlermeldung.* [When I burn a CD, the program breaks with an error message] and the associated category description being *Beim Brennen kommt eine Meldung* [When burning, a message appears], the relevant terms from the category description to be mapped were *Brennen* [burning], *kommt* [appears] and *Meldung* [message]. Both *Brennen* and *Meldung* are recorded as inference at the token level because *Meldung* can be derived from *Fehlermeldung* [error message] using compound splitting or lexical semantics (hyperonymy), *Brennen* can be derived from *brenne* [burn] using inflection / derivation. The word *kommt* [appears] was considered inference at the level of compositional semantics because there is no lexical-semantic relation to a word in the email. The word can thus only be matched by considering the complete expression.

If several inference steps were required at the lexical level, to derive one token or term from another, all of them were recorded. Lexical inference steps required at the level of compositional semantics were not recorded.

5.2.8 Possible effects on precision

The focus of the analysis described so far was on ways to improve recall in an email categorization system: We counted the inference steps required to increase the amount of derivable information (similar to query expansion in information retrieval). However, expanding an expression with an alternative one can also negatively affect the precision of the system. Taking a more precision-oriented view at the problem, we also counted the number of cases for which a more complex representation could be beneficial (albeit not necessary). For multi-token terms, particle verbs, and negation, the number of "helpful" cases is given in parentheses. For example, deriving the negated expression *Programm kann die DVD nicht abspielen* [Program cannot play the DVD] from *Programm kann die DVD nicht laden* [Program does not load the DVD] is possible at the lexical level, as *abspielen* [(to) play] entails *laden* [(to) load]. However, knowing that both verbal expressions are negated is expected to be beneficial to precision, in order to avoid wrongly inferring a negated from a non-negated expression, such as *Programm kann die DVD abspielen* [Program can play the DVD].

5.2.9 Distribution of inference steps

Table 5.2 summarizes the results of our analysis. For each text representation type, it lists the distribution of the different inference steps in our data, ordered by their frequency of occurrence. The results show that the most important inference step at the lexical level is lexical semantics. At the lexical level, we found 157 different word mappings. However, only 26 of them correspond to a relation in GermaNet [Hamp and Feldweg, 1997], version 7.0. 48 of the involved words had no GermaNet entry at all, due to the word being an anglicism (e.g., *Error* instead of *Fehler*), a non-lexicalized compound (e.g., *Bildschirmbereich* [screen area]) or a highly domain- or application-specific word (for only 37.5% of the words missing in GermaNet, we found an entry in Wikipedia). In 72 cases, both words had a GermaNet entry, but no relation existed, usually because the relation was too domain-specific. For example, in the given domain, the word *brennen* [burn] entails the word *erstellen* [create], as it usually refers to the creation of a CD or DVD.

For more than 30% of the words (as compared to 10.1% in Bar-Haim [2010]'s analysis on English), a morphological transformation is required, which can be explained by the high complexity of German morphology as compared to the morphology of English. Differences or mistakes in spelling, which are usually not considered in other analyses, are found in a considerable number of words, the reason being that customer emails are less well-formed than, for example, news texts.

The significance of multi-token terms was surprisingly high for German, where word combinations are usually expressed in the form of compounds (i.e., a single token). We identified three types of multi-token terms:

1. Compounds consisting of at least one anglicism, e.g., *USB Anschluss* [USB port]
2. English loan words, e.g., *DVD player* [DVD player]
3. Combinations of an appositive modifier and a proper name, e.g., *Programm X* [program X]

For the first two types, this means that compounds are not always properly replaced by single-token compounds, but rather realized as multi-token compounds

(as would be correct in English). This suggests that texts written in a domain language with a high proportion of English loan words are more difficult to process than general language texts, as multi-token terms have to be recognized.

At the level of compositional semantics, most cases involved the mapping of two complex expressions, where one of the two expressions either entailed or paraphrased the other. However, it should be noted that, in many cases, recognizing the entailment or paraphrase relation requires world or domain knowledge. Several of the mappings involved particle verbs or light verbs. Detecting negation scope is expected to be important in a precision-oriented system.

5.2.10 Interannotator agreement

As our analysis was done by two people separately, we were able to measure the reliability of the annotation for the different inference steps. The kappa coefficient for spelling, inflection, derivation and composition ranged between 0.46 and 0.67, i.e., moderate to substantial agreement according to the scale proposed by Landis and Koch Landis and Koch [1977]. For lexical semantics, the value is only fair (0.38). An analysis showed that the identification of a lexical semantic relation is often not straightforward, and may require a good knowledge of the domain. For example, the verbs *aufrufen* [(to) call] and *importieren* [(to) import], which would usually not be considered to be semantically related, may in fact be used to describe the same action in the computer domain, referring to files. Also for the more complex inference steps, we measured only fair agreement, due to the number of positive and negative cases being very skewed. For the "helpful" cases, the values ranged between 0.73 and 0.79 (substantial agreement).

5.2.11 Comparing text representations

For answering the question of which representation is most appropriate, we also had a look at the amount of information left unmapped at each level. For the lexical level, we determined for how many of the content tokens (or terms, respectively) occurring in the category descriptions, no matching expression was found in the associated emails. For the level of compositional semantics, we looked at each term left unmapped at the lexical level and tried to map a complex expression in

which the term occurred. If for none of these expressions a matching expression was found in the email, the term was counted as non-mappable at this level.

Table 5.3 shows that, even though the majority of the required inference relates to the lexical level, choosing a representation that allows us to map more complex expressions, increases the amount of mappable terms by about 10%. However, even with a more complex representation, a considerable amount of terms (15.9%) cannot be mapped at all. This is because the email text does not always contain all information specified in the category description. For example, in the request/-category pair below, the expression *Fehlercode -9* [error code -9] corresponds to *Problemcode -9* [problem code -9] in the request. However, for the expression *beim Start des Programmes* in the problem case description, there is no corresponding expression in the request.

- request: *Immer derselbe Problemcode -9.* [Always the same problem code -9]
- problem case description: *Fehlercode -9 beim Start des Programmes* [Error code -9 when starting the programme]

Thus, even if in most cases, the requests written by the customer are more specific than the text of the associated category description, there are also cases, in which the category description is more informative.

5.2.12 Summary and discussion

I examined the inference steps required to determine that the text of a category description can be inferred from the text of a particular email associated to this category. I identified major inference phenomena and determined their distribution in a German real-world dataset. The results support previous findings for English data in that a large portion of the required inference relates to the lexical level. Choosing a representation that allows us to map more complex expressions significantly increases the amount of mappable expressions, but some expressions simply cannot be mapped because the categorization was done relying on partial information in the email. The analysis extends previous work by investigating inference steps specific to the German language (such as morphology, composition,

and particle verbs). Some outcomes are unexpected for the German language, such as the high share of multi-token terms. Processing German data, specific properties of the German language, such as compounding and particle verbs should be taken into account. Working with data from highly specialized domains, as is common in customer support, general language semantic resources are expected to be of little help. Rather, the development of tools and resources to support domain-specific inference is crucial.

The following section describes how I used the results of the analysis to build entailment graphs representing domain-relevant semantic knowledge, and how this knowledge can be used when building an email classifier.

5.3 Entailment-graph-based email categorization

In the previous section, I presented a manual analysis of the dataset of categorized customer emails with regard to textual inference phenomena. For the analysis, I looked at each email/category pair in isolation and identified the inference steps that would be required to match each piece of information in the category description to a matching piece of information in the associated email text.

In this section, I study how the integration of knowledge about semantic relationships holding among textual expressions affects the categorization performance of an email classifier. To this end, I build entailment graphs based on the outcome of the analysis in section 5.2 and then provide the knowledge represented by these graphs to the classifier. In particular, I use the graph knowledge to generate different feature vector representations reflecting the semantic relationships holding among textual expressions. Carrying out various experiments, I investigate the role of the different inference types in improving classifier performance. In addition, I generate entailment graphs automatically and evaluate the usage of various RTE systems, configurations, and graph optimization algorithms for building entailment graphs for the email categorization task. First experiments evaluating the usage of equivalence classes (i.e., bidirectional entailment only) for email categorization were published in Eichler and Gabryszak [2017].

5.3.1 Formalization of the task

Customer support systems are commonly based on a database of previously received customer requests, which are assigned to known problem cases defined by a domain expert. A crucial task in this setting is to decide, for a newly incoming request, whether a matching problem case exists in the database, and if so, identify it, for example in order to retrieve a sample reply for the request that can then be adapted and sent out to the customer. Viewing a customer request as a text document, this task can be approached as a text categorization task, where the goal is to assign input documents to predefined categories. In our setting, the incoming request represents our input document, each problem case corresponds to a category, and the goal is to retrieve, for each document, the best-matching category.

Commonly, automatic text categorization is performed based on a large set of previously categorized texts. Accordingly, previous work on the automatic categorization of emails had its focus on using datasets or subsets of datasets providing "sufficient" amounts of training data. For example, Bekkerman et al. [2004] remove categories containing only few (less than 3) associated messages. However, a particular challenge in the described scenario is the correct categorization of incoming requests if hardly any training material exists, i.e., if only one or a few previously received requests have been assigned to a particular problem case so far. With a rising number of emails assigned to a particular category, the categorization task becomes easier, because, after a while, chances are that a particular problem formulation is repeated. However, for the first instances of a particular problem case, there may be little or no repetition in formulation, resulting in the correct problem case not being identified as matching. Nevertheless, identifying these problem cases is important, for example, in order to avoid the re-definition of a problem case that already exists in the database. For identifying these cases, the assumption is that using textual inferencing is particularly useful, as it can make up for a lack of training material.

In order to investigate the usefulness of entailment information for addressing this problem, I therefore define the task to be solved as follows: Given a set of m categories $C = \{c_1, ..., c_m\}$, where each element in C is represented by the description of the category provided by the domain expert, and a set of n non-categorized customer emails $T = \{t_1, ..., t_n\}$, where each element in T is represented by the

text part of the email, the task is to identify, for each element t in T, the element in C, which best expresses the request formulated in t.

Defining the task as such, I specifically address the problem of identifying the best-matching problem case (category) for a given input request solely on the basis of the request text itself and the set problem case descriptions. In particular, I do not use other emails assigned to the category as training material when building the classifier, thus simulating a scenario, in which only a single previous request has been sent about a particular problem case, and the description of the problem case was formulated based on this first request (as, in fact, was commonly done for the category descriptions in our dataset).

5.3.2 Approach

A straightforward way of using textual entailment in the context of customer request categorization would be to consider the request text to be T and each problem case (category) description to be H, create the corresponding T/H pairs and assign the request to a category if entailment holds [Eichler et al., 2014]. However, our analysis of the dataset revealed that, in many cases, the text of the request does not entail the description of its associated category completely. Rather, parts of the description are entailed by parts of the request text, possibly spread across several sentences, while other parts do not have any correspondent in the request at all (see Sections 5.2.11 and 5.5.3.3 for details on this phenomenon). Rather than looking at the complete problem case description, I therefore propose to identify semantic units in the category descriptions and in the email texts, capture the semantic relations holding among these semantic units, and use this knowledge for categorization. For example, given email A and category B below, the category description does not entail the request text because it contains additional information (*beim Start des Programmes*) not present in the email. However, we can capture the semantic relationship between *Fehlercode -9* [error code -9] and *Problemcode -9* [problem code -9], which are synonymous in the given context, and provide this knowledge to the classifier.

- A: *Immer derselbe Problemcode -9.* [Always the same problem code -9]
- B: *Fehlercode -9 beim Start des Programmes* [Error code -9 when starting the programme]

Entailment information is thus used at the level of individual textual expressions rather than complete sentences or documents. A common way of approaching the text categorization task is to use a probabilistic classifier, which assigns, to a given input text t, probabilities expressing how likely it is that t belongs to each of the predefined categories. A crucial decision to be made in this context is the representation of the text, which serves as input to the classification algorithm. The most commonly used text representation, called "bag of words", represents each text as the set of words contained in it. This representation has shown to work decently for many tasks, but is agnostic of any syntactic or semantic information present in the text. The evaluation of more sophisticated representations has been a topic of research for many years and various methods have been proposed (see Section 2.6 for details). I propose to create a more semantics-oriented feature representation by integrating knowledge represented by an entailment graph. As the nodes of an entailment graph can be any type of expression, this allows me to go beyond the level of words (as compared to integrating lexical resources such as WordNet), and, in addition, enables the usage of directional information, represented by the direction of entailment.

5.3.3 Task-specific definition of entailment graphs

Following Definition 3.2 in Section 3.2, an entailment graph is a directed graph $G = (V, E)$ consisting of a set of nodes V, which represent textual expressions, and, a set of edges E, which connect pairs of nodes and represent entailment relations holding among the associated expressions[2]. For addressing the task of categorizing customer requests, the textual expressions forming the nodes of the graph are expressions extracted from the request texts to be categorized and from the textual descriptions of the problem categories. These can be single words or more complex expressions. The nodes are connected via edges representing entailment relations holding between the expressions represented by the nodes.

For example, assuming text A to be a customer request and text B to be the description of the associated category, a sample entailment graph is depicted in Figure 5.1:

[2]Note that the graph is not necessary acyclic, as it may contain semantically equivalent textual expressions, resulting in bidirectional edges and strongly connected components. However, from a well-formed entailment graph, a directed acyclic graph can easily be created by contracting each strongly connected component of the graph into a single vertex, thus creating an equivalence class node (see Definition 3.8).

- A: *Wenn ich Daten im .mpg-Format öffne, stimmt die Tonspur nicht mit dem Film überein.* [When opening data in .mpg format, the audio track does not match the film.]
- B: *Bild und Ton einer Videodatei sind asynchron.* [Image and sound of the video file are asynchronous.]

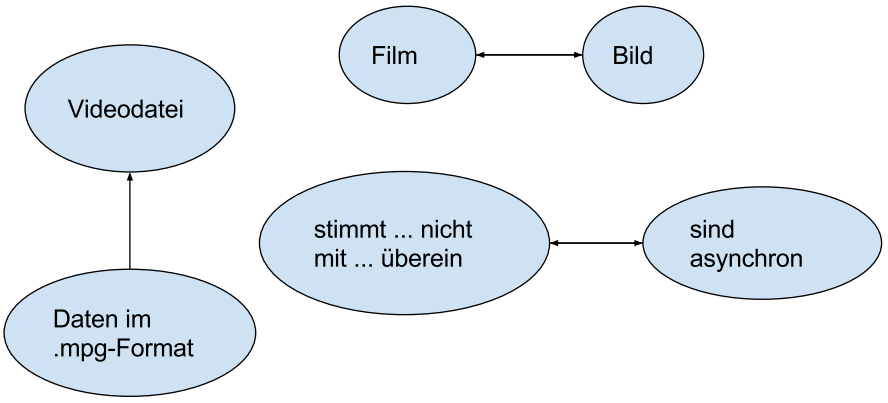


FIGURE 5.1: Entailment graph showing a subset of relevant entailment relations for the two sample sentences.

Note that I define the entailment relation with respect to the given domain context. This means that when building entailment graphs, I consider the meaning of a given expression in the context of a particular task, not the general meaning of the expression. For example, in the given example, *Bild* [image] and *Film* [film] are considered to be semantically equivalent. Even though the two words would not be considered synonymous in their most common sense, their usage in the sample sentences suggests that they express the same meaning in the given context (the visual part of a film). The same assumption can be made for the two words *Ton* (sound) and *Tonspur* (soundtrack).

5.4 Constructing entailment graphs for email categorization

5.4.1 Gold-standard entailment graphs

For creating a gold standard entailment graph to be applied for the email categorization task described in Section 5.3.1, I use the dataset described in Section 5.2.3, which contains customer emails and their associated categories. In the

data analysis described in Section 5.2, I identified textual expressions in the category descriptions of the dataset, and mapped them to matching expressions in the emails associated to the category. The identified mappings can be used to derive a gold-standard entailment graph. The textual expressions identified in the category descriptions as well as their associated expressions in the emails form the nodes of this graph. For determining the edges of the graph, we need to distinguish between mappings expressing a unidirectional entailment relation, and mappings expressing an equivalence relation (i.e., bidirectional entailment).

To this end, I categorized each mapping in the analysis based on a fine-grained set of mapping types. Table 5.4 illustrates the types I distinguish and lists the edge type for each mapping type (B=bidirectional entailment, U=unidirectional entailment), as well as the number of instances found for each mapping type in the data. Mapping types marked with (N), (V) or (Adv) refer to expression pairs with both expressions having the respective part-of-speech (N=noun, V=verb, Adv=adverb). Mapping types marked with (EN) refer to expression pairs with both expressions being English words. Mapping types marked with (M) refer to expression pairs with at least one expression being a multi-token word. Mapping types marked with (K) refer to expression pairs with at least one expression being a complex expression (consisting of at least two words with different syntactic functions). In the table, the mapping types are sorted based on their occurrence in the data.

For creating a gold-standard entailment graph, edges were added based on the decisions made during the manual analysis of the dataset: For each identified mapping between two expressions A and B, the mapping type was classified according to Table 5.4 and an edge of the respective type was added between A and B. For example, for the two expressions *fehlt die Tonspur* [soundtrack missing] and *kommt kein Ton* [no sound coming], a bidirectional entailment edge between the two corresponding nodes in the graph was added. For the two expressions *Fehlermeldung* [error message] and *Meldung* [message], a unidirectional entailment edge was added, starting at the node representing *Fehlermeldung* and ending in the node representing *Meldung*.

5.4.2 Automatically constructed entailment graphs

5.4.2.1 Procedure

For creating entailment graphs for the email categorization task automatically, I used the textual expressions that are part of the gold graph as textual input. However, rather than adding edges based on the manual decisions made by the annotators, I apply entailment decision algorithms (EDAs) to derive the decisions automatically. To this end, I first evaluate different EDAs for German on individual T/H pairs to find the EDA configurations that work best on the data at hand. Using the best setting per EDA, I then create entailment graphs automatically using the framework and technology introduced in Chapter 3 and evaluate these graphs in the context of the email categorization task. The realization of each of the steps with regard to creating entailment graphs for email categorization is summarized in the following:

Step 1: Identifying relevant expressions For identifying relevant expressions, I apply sentence-based fragment annotation (Section 3.4.4.1), considering each textual expression in the gold graph as a sentence[3].

Step 2: Identifying modifiers Modifier identification is an optional step that could be applied for generating additional shorter expressions by removing modifiers. In order to make sure that the graph nodes in both the gold and automatically created graphs are identical, I did not make use of this module.

Step 3: Building fragment graphs Fragment graphs are generated according to Section 3.4.4.3. As the optional step of modifier annotation is not performed, all fragment graphs created are single-node graphs, each representing a distinct textual expression from the gold graph.

[3]Note that the task of identifying relevant textual expressions from the raw dataset of emails and category descriptions is left for future work. This simplification of the task makes sure that the nodes in the automatically created graphs correspond to the nodes in the gold standard graph, thus allowing for a direct comparison of manually and automatically derived entailment decisions.

Step 4: Computing pair-wise entailment For recognizing entailment relations between individual T/H pairs of textual expressions, I use the all-pairs graph merger (Section 3.4.4.4), which computes entailment decisions for each possible pair. I first evaluate different EDAs for German on individual T/H pairs to find the EDA configurations that work best on the data at hand and then create entailment graphs using the best setting per EDA.

Step 5: Optimizing the graph For creating transitive graphs from the intermediate graphs created in Step 4, I apply two different strategies: The first strategy is the simple graph optimizer described in Section 3.4.4.5. The second strategy is the global graph optimizer described in Section 3.4.4.5.

5.4.2.2 Entailment decision algorithms

With most existing entailment systems being created for the English language, only a few systems can be applied on German language data. For constructing entailment graphs for the described task automatically, I applied the following two RTE engines from the EXCITEMENT platform (described in Section 3.4.1): *MultiAlign* and *MaxEntClassificationEDA*. In addition, I applied a third, rule-based RTE engine called *SEDA*, which was implemented as part of a project coordinated by the author of this thesis. The three RTE engines are described in the following.

MultiAlign *MultiAlign* combines information from various aligners and achieved state-of-the-art performance on standard RTE datasets [Noh et al., 2015], also on German data. For German, the following aligners can be made use of:

- *identicalLemmaLinker*: aligns identical lemma sequences (the longest identical lemma sequence in T and H, including at least one content word)
- *meteorParaphraseLinker*: aligns paraphrases based on the German Meteor paraphrase table [Denkowski and Lavie, 2014]
- *derivBaseLinker*: aligns lemmata from the same derivational family, according to DErivbase [Zeller et al., 2013]

- *distSimLinker*: aligns tokens based on the TransDM German resource [Utt and Padó, 2014]
- *germaNetLinker*: aligns tokens based on GermaNet [Hamp and Feldweg, 1997]

As classifier, any classification algorithm from the Weka library can be selected. The default algorithm of the EDA, which I also use in my experiments, is logistic regression.

MaxEntClassificationEDA *MaxEntClassificationEDA* is an EDA that optimizes its performance based on Maximum Entropy modeling and learns a binary classifier for deciding the entailment problem. The supervised learner receives a number of features which are provided through the interplay of different processing components and knowledge sources. It uses the following features:

- *BagOfWordsScoring*: measures bag-of-words similarity
- *BagOfLemmasScoring*: measures bag-of-lemmas similarity
- *GermaNet*: integrates knowledge about GermaNet relations
- *DErivBase*: integrates knowledge about derivational families
- *BagOfDepsScoring*: calculates dependency triple similarity (based on word forms)
- *BagOfDepsPosScoring*: calculates dependency triple similarity (based on lemmata)
- *TreeSkeletonScoring*: extracts dependency paths from dependency trees of T and H and computes feature scores based on Wang and Neumann [2007]

Unlike *MultiAlign*, *MaxEntClassificationEDA* integrates information about syntactic dependencies. In the original implementation of *MaxEntClassificationEDA*, as available from the EXCITEMENT Open Platform[4], word pairs from the same derivational family were only taken into account if they shared the same part-of-speech tag. As this is very restrictive, we adapted the EDA, allowing for word pairs with different part-of-speech tags. In my experiments, I make use of both the original and the adapted implementation.

[4] `https://github.com/hltfbk/EOP-1.2.3/wiki/MaxEntClassificationEDA`

SEDA *SEDA* is a RTE engine implemented as part of the Software Campus project ISSA[5], which was coordinated by the author of this thesis. It is based on a highly performant alignment-based algorithm, which determines entailment decisions for T/H pairs of short text fragments in German. The code of this EDA is freely available[6]. The algorithm is based on the following language resources and tools for the German language:

- TreeTagger [Schmid, 1994] for lemmatization
- GermaNet [Hamp and Feldweg, 1997], the German version of WordNet
- DErivBase [Zeller et al., 2013], a resource providing information about derivational families
- jWordSplitter [Naber, 2015], a compound splitter

The EDA returns a positive entailment decision if, for every word of the hypothesis, there is a semantically equivalent or entailed word in the text (according to any of the used resources): Words are considered semantically equivalent if they share the same lemma or the same derivational family, or are synonymous based on GermaNet. A word a is considered to entail another word b if a is a compound word containing a component b or if, according to GermaNet, any of the following relations holds from a to b: *has_hypernym*, *entails*, and *causes*.

5.4.2.3 Experiments

For identifying the best configuration per EDA, I created positive and negative T/H pair examples from the gold graph described in section 5.4.1 and compared the performance of different configurations of the three EDAs in deciding entailment for the given pairs. As positive examples, I considered all equivalence and entailment mappings identified during data analysis. For creating negative examples, I extracted all unidirectional entailment relations from the gold graph and flipped the direction of entailment. For example, from the positive entailment example *Fehlermeldung* [error message] $\rightarrow$ *Meldung* [message], I created the negative example *Meldung* $\nrightarrow$ *Fehlermeldung*. The motivation behind generating negative

[5] `https://www.dfki.de/lt/project.php?id=Project_772&l=en`

[6] `https://github.com/hltfbk/Excitement-Transduction-Layer/blob/master/tl/src/main/java/eu/excitementproject/tl/experiments/OMQ/SimpleEDA_DE.java`

examples by flipping the entailment direction of a positive T/H pair is that this makes sure that, even for the negative cases, T and H are semantically related. Producing negative examples this way, the classifier is thus trained on the actual entailment relation, rather than semantic relatedness in general.

After producing positive and negative examples, I balanced out the dataset by reducing the number of positive examples so that it is equal to the number of negative examples. For my experiments, I split the positive and negative examples, using one split for training the EDA based on a given configuration, and another one for testing the performance of the trained EDA and determine the best EDA setting.

5.4.2.4 Results

The results of the EDA evaluations are shown in Tables 5.5, 5.6, 5.7, and 5.8. For each EDA, the table lists the different configurations along with the achieved accuracy on the positive, negative and all T/H pairs, respectively.

5.4.2.5 Analysis

The best overall result is achieved with *SEDA* using all resources, which yields an overall accuracy of 70%. It achieves 100% accuracy on the negative pairs. However, it is quite cautious in assigning positive entailment, as it requires a matching counterpart for every content word in H in order to make a positive entailment decision. For *MaxEntClassificationEDA*, the results show that using the less restrictive implementation of the *DErivBase* aligner improves the results in all configurations.

5.5 Evaluation on email categorization task

This section describes an extrinsic evaluation of both gold-standard and automatically created entailment graphs on the email categorization task described in Section 5.3.1. For evaluating the usefulness of automatically generated entailment graphs for the email categorization task, I build entailment graphs based on the best setting determined for each of the three EDAs in the previous section.

5.5.1 Evaluation procedure

For evaluating how the usage of entailment information affects classification performance, I used the knowledge represented by the entailment graphs to generate a meaning-oriented representation of the input texts, as input to the classifier. In particular, I created different feature vector representations, incorporating different types of inference knowledge, and evaluated the influence of each representation on the overall performance of the classifier. Following the task description in Section 5.3.1, the classifier model was trained based on the category descriptions only. The email dataset was bisected into a development and a test split, where the first one was used for experimenting with different optimizers and confidence thresholds, and the latter one was reserved for the final evaluation. As classification algorithm, I used Naïve Bayes from the Weka library [Hall et al., 2009].

5.5.2 Feature generation and application

For generating features using entailment graph information, I first created a base graph containing, as graph nodes, all lexical words from the training data (i.e., from the category descriptions) and all multi-token words and complex expressions that were part of a mapping identified in the data analysis. To this base graph, edges were added depending on the respective configuration. For evaluations on the gold graph, edges were added per inference types based on the decisions from the manual analysis, e.g. evaluating the inference type "derivation", all edges expressing a derivation relation were added. For the evaluations on automatically created graphs, graphs were created according to the respective EDA and optimizer configuration.

Based on the graph created for the a particular configuration, I then created one feature per equivalence class. Thus, the number of features varies depending on the configuration (based on the number of entailment units combined into the same equivalence class).

In the categorization phase, information from the entailment graphs is used when creating the feature vector for the input emails to be classified. For each input email t, I identify matching features in the following way: A feature f applies whenever any of the text expressions in the graph node represented by f or any of the expressions in the graph nodes *entailed by* the graph node represented by

f is found in t. For example, if in a given configuration, the graph contains an entailment edge from the node containing the expression *Daten im .mpg-Format* [data in .mpg format] to the node containing *Videodatei* [video file], then an input text containing *Daten im .mpg-Format* would also activate the feature associated to the expression *Videodatei*. As feature weights, I use TF-IDF scores, as computed on the training data.

5.5.3 Evaluation of gold-standard entailment graphs

A major advantage of the gold standard entailment graph is that it contains information about the type of inference underlying a particular edge in the graph (from the manual analysis). Thus, the gold graph can be used to evaluate the effect of using particular types of inference knowledge.

5.5.3.1 Procedure

For evaluating the effect of adding a particular type of knowledge, the edges of the graph were added according to the inference types listed in Table 5.9. For example, when adding the feature type "synonym (N)", all edges were added that connect two synonymous nouns.

As entailment knowledge was directly drawn from the manually annotated files, transitivity of the graph was derived by applying the *SimpleGraphOptimizer* (Section 3.4.4.5), which assumes all given edges as correct and automatically adds all edges missing for transitive closure.

For evaluating the effect of incorporating different inference types, I applied a feature selection method inspired by the algorithm proposed by Surdeanu et al. [2008]: Starting with an entailment graph with no edges (baseline), I incrementally add the inference type producing the feature set that provides the highest performance improvement (in terms of accuracy) based on the development partition. The process stops when no inference types yield any improvement.

I carried out four sets of experiments, running additive feature selection for the following sets of inference types:

- single token paraphrase only

- multi-token paraphrase (including single token paraphrase)
- single token entailment (including single token paraphrase)
- multi-token entailment (all features)

Table 5.9 lists the inference types considered in each of the four experiments.

5.5.3.2 Results and analysis

Tables 5.10, 5.11, 5.12, and 5.13 show the results of the evaluation. The best results are achieved when running the experiment on the full set of feature types, resulting in an accuracy improvement of 22.81 percentage points over the baseline. Evaluating only equivalency features (i.e., bidirectional entailment), the most important features are inflection, derivation, synonymy, and spelling. Considering spelling variations and synonymy relations among multi-token expressions helps improve the results. However, the highest improvement in accuracy is achieved when integrating unidirectional entailment features. In particular, decompounding, the feature type that is selected first by the feature selection algorithm, plays an important role. Considering entailments between verbs also leads to improvements, whereas the feature type for noun entailments (hyponymy) is not selected by the algorithm. An analysis shows that many of these relations are already covered by the feature type decompounding (e.g., *Fehlermeldung* → *Meldung*). This is in line with findings by Terryn et al. [2016], who effectively use a decompounding module for hypernym detection in Dutch.

Adding multi-token entailment features, the two feature types leading to additional improvements are hypernym (M) and instance (M). None of the complex mapping types is selected by the algorithm, which suggests that all important mappings are either already covered by lower level features or that relevant mappings between complex expressions are not part of the gold standard graph.

5.5.3.3 Error analysis

The table shows that, while the features selected from the complete set of features span all the four feature groups introduced, the largest accuracy improvement (approximately 45%) can be attributed to the single token paraphrase features. The highest accuracy value achieved with gold standard entailment graphs

is 41.64%. This is a remarkable improvement over the baseline, but lower than results achieved on other email datasets, e.g., as compared to accuracies ranging between 56% and 95% on the Enron dataset [Bekkerman et al., 2004]. This can be attributed to several factors: First of all, due the definition of the task, the training material is extremely small.[7] In addition, as the categories in our dataset were defined by different users, there are large variations in terms of style and specificity of the definition as well as several category descriptions that semantically overlap and hamper the correct categorization.

In order to assess the potential of a semantics-based categorization approach, I analyzed the wrongly assigned emails und identified reasons for misclassification. In particular, I manually assigned the wrongly categorized emails along three problem classes:

- *Class A*: All information in the category text can be found in the email.
- *Class B*: Information relevant for assigning the category can be found in the email, but additional assumptions have to be made to compensate for missing information.
- *Class C*: Information for assigning the category cannot be found in the email (i.e., category assignment is either wrong or was made based on knowledge that cannot be derived from the text)

Table 5.14 shows the results of this evaluation. As the table shows, for almost half of the wrongly assigned cases, a text-based assignment is impossible. Only for a minority of cases (13%) a purely text-based assignment is possible, without additional assumptions that require world or domain knowledge.

5.5.4 Evaluation of automatically constructed entailment graphs

In the previous section, I evaluated the usage of entailment information from the gold standard entailment graph, which was generated based on hand-annotated entailment relations. These turned out useful, providing background knowledge

[7] Bekkerman et al. [2004] also report results achieved on the smaller SRI dataset, which are generally lower than those for the Enron dataset.

about semantic relationships for relevant textual expressions. However, in a real-world setting, one would like to produce the graphs with as little manual effort as possible. In this section, I therefore create entailment graphs automatically, using and adapting RTE technology, and evaluate the usefulness of these automatically created graphs for the categorization task. In particular, I evaluate how the usage of the two graph optimization algorithms described in Section 3.4.4.5 along with different confidence thresholds affects the overall result of the categorization.

Based on the EDA evaluation carried out in Section 5.4.2.3, I created graphs with the best configuration of each of the three EDAs. Tables 5.15, 5.16, and 5.17 list the results achieved on the email categorization task based on the development set. A visual summary is provided in Figure 5.2. Experiments were carried out with the simple and global graph optimizers and different confidence thresholds.

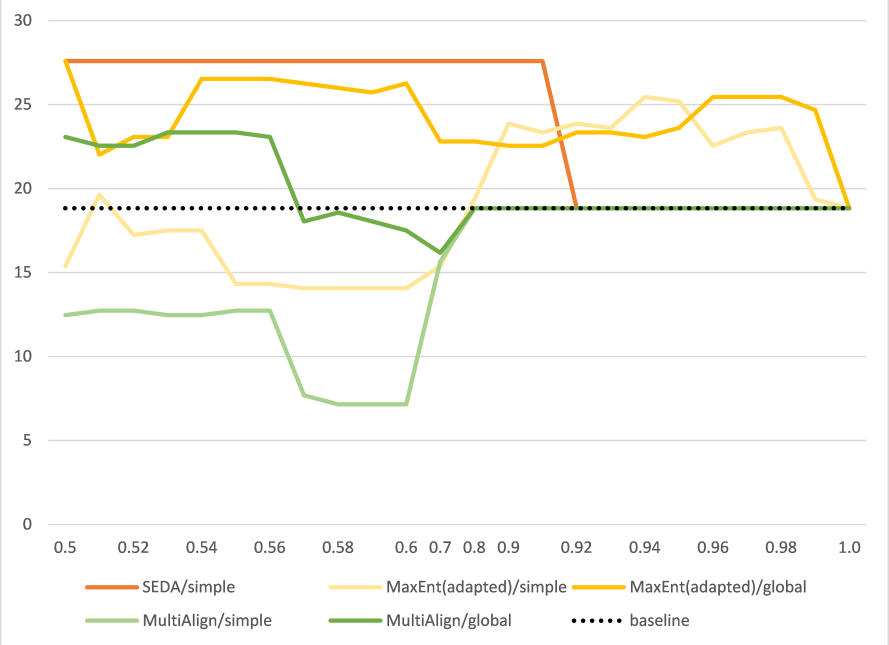


FIGURE 5.2: Accuracy (y-axis) achieved with different EDAs, graph optimizers and optimizer thresholds (x-axis).

The best overall result (27.50% as compared to the baseline of 18.83%) on the development partition of the dataset is achieved with the *SEDA* engine together with the simple graph optimizer (confidence threshold below 0.91[8]) and the adapted *MaxEntClassificationEDA* engine with global graph optimization and a confidence threshold of 0.5. The results show that the EDAs applied in the experiments are in

[8]As SEDA only a produces a single confidence score (0.91) for all decisions, all thresholds of 0.91 and below consider all positive entailment edges in the raw graph and produce the same results.

fact able to produce entailment graphs that can be used for improving performance on the categorization task.

5.5.5 Results on test set

After optimizing all parameters based on the development partition of the data, I ran a final evaluation on the test set, comparing the best setting for each EDA (based on Section 5.5.4) and the best gold graph configuration (based on Section 5.5.3) to the baseline (no entailment information). The best results on the test set (38.46 % accuracy) were achieved with the gold graph. Also, with the graph based on automatic entailment decisions by the *SEDA* algorithm, a considerable improvement over the baseline was achieved (35.54 % accuracy). With the other two EDAs, the improvement in accuracy as compared to the baseline was much lower. The main advantage of *SEDA* is that, unlike the other two EDAs, it can deal with compounds, which, as the experiments on the gold graph showed, play a major role in the dataset at hand (see Section 5.5.3.2). An analysis of the wrongly categorized emails showed that, in the given task setting, higher accuracy values can hardly be achieved due to the high proportion of email / category pairs, for which a purely semantics-based categorization is simply not possible (see Section 5.5.3.3).

5.6 Summary and discussion

In this chapter, I addressed the task of categorizing customer emails using knowledge expressed by entailment graphs. Based on an analysis of a real-world dataset of German customer emails and associated categories, I identified inference phenomena relevant for assigning emails to a given category and measured their distribution in the dataset. The analysis showed that the most important inference type for the data at hand is lexical semantics, in particular domain-specific lexical semantics. An important resource when automatizing the classification of emails is therefore a resource of word relations relevant for a particular domain. The analysis also showed the importance of multi-token words in domains with a high share of English vocabulary.

Based on the analysis, I generated entailment graphs (both a gold standard graph and automatically generated graphs) and presented a procedure for using the knowledge expressed by these graphs when building an email classifier. For creating entailment graphs for the task automatically, I applied and evaluated several RTE systems and graph optimization algorithms in order to derive the best configuration. Generating different feature representation based on various types of semantic knowledge, I evaluated the usefulness of the graphs for the email categorization task and showed that the generated knowledge can in fact improve the performance of the email classifier in a scenario where little training data is available. Investigating the effect of using different types of representation (individual words vs. more complex expressions) and semantic relations (equivalence vs. entailment), I did not see any clear positive effect using knowledge about entailment relations holding across text units more complex than individual words or multi-token terms, suggesting that, for this task, lexical entailment graphs (including multi-token terms) are sufficient.

For the experiments described in the chapter, in order to focus the evaluation on the actual entailment graphs, the text expressions from which the entailment graphs were created were manually identified during the data analysis step. An additional step to automatize would thus be to identify relevant text expressions from the input texts, as input to the entailment graph generation step.

Type of inference	**Data example**	**Total**	**Share**
Lexical semantics (Token)	"Anfang" [*start*] → "Beginn" [*beginning*]	310	20.2%
Inflection	"startet" [*starts*] → "starten" [*start*]	206	13.4%
Derivation	"Import" [*import*] → "importieren" [*(to) import*]	164	10.7%
Composition	"Fehlermeldung" [*error message*] → "Meldung" [*message*]	158	10.3%
Spelling	"Dateine" → "Dateien" [*files*]	47	3.1%
Lexical semantics (Term)	"MPEG Datei" [*MPEG file*] → "Video" [*video*]	60 [+124]	4.1% [8.6%]
Particle verbs	"spielt [...] ab" [*play*] → "abspielen" [*play*]	26 [+34]	1.8% [2.4%]
Light verbs	"Meldung kommt" [*message appears*] → "melden" [*notify*]	17	1.2%
Negation	"Brennergerät kann nicht gefunden werden" [*Burning device cannot be found*] → "Es wird kein Brenner gefunden" [*No burner is found*]	8 [+121]	0.6% [8.4%]
Other complex expressions	"Das Brennen bricht ab mit der Meldung X" [*Burning breaks with message X*] → "Beim Brennen kommt die Fehlermeldung X" [*Burning yields error message X*]	83	5.7%

Table 5.2: Distribution of inference steps in the dataset.

Representation	**Non-mappable**	**Share**
Tokens	428 / 1538	27.8%
Terms	365 /1446	25.2%
Complex expressions	229 / 1446	15.8%

TABLE 5.3: Comparing different text representations

TABLE 5.4: Overview of mappings

Type	*Edge*	*Description*	*Example*	*Count*
negation	B	negated expressions that are paraphrases	*fehlt die Tonspur* ↔ *kommt kein Ton* [soundtrack missing ↔ no sound coming]	109
hypernym (N)	U	nouns that are in a hypernymy relation	*Fehlermeldung* → *Problem* [error message → problem]	108
inflection	B	words sharing the same lemma	*abschließen* ↔ *abgeschlossen* [finish ↔ finished]	91
entailment (K)	U	complex expressions where one entails the other	*DVD hängt sich auf* → *kann DVD nicht abspielen* [DVD freezes → cannot play DVD]	74
paraphrase (K)	B	complex expressions that are paraphrases	*fehlt die Tonspur* ↔ *kommt kein Ton* [soundtrack missing ↔ no sound coming]	56
derivation	B	words from the same derivational family	*abbrechen* ↔ *Abbruch* [terminate ↔ termination]	47
spelling	B	the same word, but spelled differently	*abschließen* ↔ *abschliessen* [finish ↔ finish]	45
hypernym (M)	U	expressions that are in a hypernymy relation	*DV Kamera* → *Kamera* [DV camera → camera]	34
decompounding	U	a compound word and one of its components	*Fehlermeldung* → *Meldung* [error message → message]	31
entailment (V)	U	verbs that are in an entailment relation	*importieren* → *einfügen* [(to) import → (to) integrate]	31
synonym (N)	B	nouns that are synonymous	*Anfang* ↔ *Start* [beginning ↔ start]	30
synonym (M)	B	expressions considered synonymous	*Fehler 1601* ↔ *Fehlermeldung 1601* [error 1601 ↔ error message 1601]	25
synonym (V)	B	verbs that are synonymous	*anfangen* ↔ *beginnen* [(to) start ↔ (to) begin]	20
instance (M)	U	an instance and its class	*Quicktimeplayer* → *DVD Player* [Quicktime player → DVD player]	19
spelling (M)	B	the same expression, but spelled differently	*mov Datei* ↔ *.mov Datei* [mov file ↔ .mov file]	17
translation	B	words that are translations of each other	*Fehler* ↔ *Error* [error ↔ error]	12
translation (M)	B	expressions that are translations of each other	*Fehlercode* ↔ *Error code* [error code ↔ error code]	12
particle verb	B	different forms of the same particle verb	*abspielen* ↔ *spielt ... ab* [play ↔ play]	10
instance (N)	U	an instance and its class	*MOV* → *Dateierweiterung* [MOV → file extension]	8
inflection (M)	B	expressions sharing the same lemma	*mov Datei* ↔ *mov Dateien* [mov file ↔ mov files]	7
inflection (EN)	B	English words sharing the same lemma	*fail* ↔ *failed* [fail ↔ failed]	5
synonym (Adv)	B	adverbs that are synonymous	*ständig* ↔ *dauernd* [constantly ↔ always]	1

Configuration	**Acc pos**	**Acc neg**	**Acc all**
no resources	48.9 %	72.8 %	60.9 %
Meteor + DErivBase	52.2 %	72.8 %	62.5 %
Meteor + DistSim	50.0 %	72.8 %	61.4 %
Meteor + DErivBase + DistSim	54.3 %	72.8 %	**63.5 %**

TABLE 5.5: Results using *MultiAlign*

Configuration	**Acc pos**	**Acc neg**	**Acc all**
DErivBase	29.3 %	97.8 %	63.6 %
GermaNet	31.5 %	96.7 %	64.1 %
GermaNet + DErivBase	31.5 %	97.8 %	**64.7 %**
GermaNet + DErivBase + DistSim	31.5 %	97.8 %	**64.7 %**
GermaNet + DErivBase + DistSim + Syntax	57.6 %	70.7 %	64.1 %

TABLE 5.6: Results using *MaxEntClassificationEDA* (original)

Configuration	**Acc pos**	**Acc neg**	**Acc all**
DErivBase	34.8 %	95.7 %	65.2 %
GermaNet	33.7 %	94.6 %	64.1 %
GermaNet + DErivBase	40.2 %	93.5 %	66.8 %
GermaNet + DErivBase + DistSim	40.2 %	93.5 %	66.8 %
GermaNet + DErivBase + DistSim + Syntax	68.4 %	66.3 %	**67.4 %**

TABLE 5.7: Results using *MaxEntClassificationEDA* (adapted)

Configuration	**Acc pos**	**Acc neg**	**Acc all**
jWordSplitter	29.3 %	100 %	64.7 %
GermaNet	19.6 %	100 %	59.8 %
DErivBase	17.4 %	100 %	58.7 %
jWordSplitter + DErivBase	32.6 %	100 %	66.3 %
DErivBase + GermaNet	25.0 %	100 %	62.5 %
jWordSplitter + DErivBase + GermaNet	40.2 %	100 %	**70.1 %**

TABLE 5.8: Results using *SEDA*

Table 5.9: Mappings and configurations

Type of mapping	single token paraphrase	multi-token paraphrase	single token entailment	multi-token entailment
inflection	x	x	x	x
derivation	x	x	x	x
spelling	x	x	x	x
synonym (N)	x	x	x	x
synonym (V)	x	x	x	x
synonym (Adv)	x	x	x	x
inflection (EN)	x	x	x	x
translation	x	x	x	x
inflection (M)	-	x	-	x
spelling (M)	-	x	-	x
particle verb	-	x	-	x
synonym (M)	-	x	-	x
translation (M)	-	x	-	x
paraphrase (K)	-	x	-	x
negation	-	x	-	x
decompounding	-	-	x	x
entailment (V)	-	-	x	x
hypernym (N)	-	-	x	x
instance (N)	-	-	x	x
hypernym (M)	-	-	-	x
instance (M)	-	-	-	x
entailment (K)	-	-	-	x

Iteration	Feature mapping	Correct instances	Accuracy
0	none (baseline)	71	18.83 %
1	**+ inflection**	**78**	**20.69 %**
2	**+ derivation**	**84**	**22.28 %**
3	**+ synonym (V)**	**91**	**24.14 %**
4	**+ synonym (N)**	**96**	**25.46 %**
5	**+ spelling**	**98**	**26.00 %**
6	**+ translation**	**99**	**26.26 %**

Table 5.10: Results using **single token paraphrase** mappings

Iteration	Feature mapping	Correct instances	Accuracy
0	none (baseline)	71	18.83 %
1	+ inflection	78	20.69 %
2	+ derivation	84	22.28 %
3	+ synonym (V)	91	24.14 %
4	+ synonym (N)	96	25.46 %
5	**+ spelling (M)**	**99**	**26.26 %**
6	+ spelling	103	27.32 %
7	**+ negation**	**106**	**28.12 %**
8	**+ synonym (M)**	**108**	**28.65 %**
9	+ translation	110	29.18

TABLE 5.11: Results using single and **multi-token paraphrase** mappings

Iteration	Feature mapping	Correct instances	Accuracy
0	none (baseline)	71	18.83
1	**+ decompounding**	**92**	**24.40 %**
2	+ synonym (N)	115	30.50 %
3	+ inflection	131	34.75 %
4	**+ entailment (V)**	**139**	**36.87 %**
5	+ spelling	140	37.14 %

TABLE 5.12: Results using **single token entailment** mappings

Iteration	Feature mapping	Correct instances	Accuracy
0	none (baseline)	71	18.83 %
1	+ decompounding	92	24.40 %
2	+ synonym (N)	115	30.50 %
3	+ inflection	131	34.75 %
4	+ spelling (M)	139	36.87 %
5	+ entailment (V)	143	37.93 %
6	**+ hypernym (M)**	**147**	**38.99 %**
7	+ negation	154	40.85 %
8	+ particle verb	155	41.11 %
9	+ inflection (M)	156	41.38 %
10	**+ instance (M)**	**157**	**41.64 %**

TABLE 5.13: Results using single and **multi-token entailment** mappings

Class	Sample Category	Sample Email	Count	Percentage
A	**Fehlercode -9 beim Start des Programmes**	Habe Produkte im IT & OFFICE Katalog 12 firstclass auf einen neuen Rechner kopiert. **Die Software startet auf dem neuen Rechner nicht, sondern bricht den Start ab mit Angabe des Fehlercodes -9.** Eine zweimalige Neuinstallation hat nicht geholfen Was tun?	28	12.84 %
B	**Cwvemx Dateien stocken** in Sales firstclass 20	Ich habe einen 3,11 Ghz Quad Intel Rechner. **Beim Verarbeiten von CWVEMX Prozessen (z.B. direkte Verarbeitung) ohne voheriges Berechnen stockt die Previewanzeige in diesem Abschnitt des Ablaufs.** Was kann ich tun? Wann kommt ein SFC auf den Markt, bei dem das Berechnen der Previewanzeige automatisch im Hintergrund abläuft, so wie das andere Programme ganz selbstverständlich beherrschen?	93	42.66 %
C	Es wird kein Brenner gefunden.	bei meinem Produktorganizerprogramm kommt bei Aufruf die Meldung daß ich mich registrieren soll obwohl ich mich im April schon registriert habe. Ihr zugeschickter Democ-Code den ich eingegeben habe wird nicht angenommen weil er angeblich falsch ist. Auch nach einer erneuten registrierung wurde mir der gleiche Code zugeschickt. Da ich die Software nicht testen wollte sondern gekauft habe gehe ich davon aus, daß ich es nicht immer wieder freischalten muß. Brennen	97	44.50 %

TABLE 5.14: Distribution of problem classes for wrongly assigned categories.

Configuration	Optimizer	Threshold	Correct	Accuracy
baseline	-	-	71	18.83 %
SEDA	simple	<= 0.91	104	**27.59 %**
SEDA	simple	>= 0.92	71	18.83 %

TABLE 5.15: Results of entailment-graph-based email categorization on development data using best *SEDA* model and different graph optimizers.

Configuration	Optimizer	Threshold	Correct	Accuracy
MaxEnt (adapted)	simple	0.50	58	15.38 %
MaxEnt (adapted)	simple	0.51	74	19.63 %
MaxEnt (adapted)	simple	0.52	65	17.24 %
MaxEnt (adapted)	simple	0.53 - 0.54	66	17.51 %
MaxEnt (adapted)	simple	0.55 - 0.56	54	14.32 %
MaxEnt (adapted)	simple	0.57 - 0.60	53	14.06 %
MaxEnt (adapted)	simple	0.70	58	15.38 %
MaxEnt (adapted)	simple	0.80	73	19.36 %
MaxEnt (adapted)	simple	0.90	90	23.87 %
MaxEnt (adapted)	simple	0.91	88	23.34 %
MaxEnt (adapted)	simple	0.92	90	23.87 %
MaxEnt (adapted)	simple	0.93	89	23.61 %
MaxEnt (adapted)	simple	0.94	96	25.46 %
MaxEnt (adapted)	simple	0.95	95	25.20 %
MaxEnt (adapted)	simple	0.96	85	22.55 %
MaxEnt (adapted)	simple	0.97	88	23.34 %
MaxEnt (adapted)	simple	0.98	89	23.61 %
MaxEnt (adapted)	simple	0.99	73	19.36 %
MaxEnt (adapted)	global	0.50	**104**	**27.59 %**
MaxEnt (adapted)	global	0.51	83	22.02 %
MaxEnt (adapted)	global	0.52 - 0.53	87	23.08 %
MaxEnt (adapted)	global	0.54 - 0.56	100	26.53 %
MaxEnt (adapted)	global	0.57	99	26.26 %
MaxEnt (adapted)	global	0.58	98	25.99 %
MaxEnt (adapted)	global	0.59	97	25.73 %
MaxEnt (adapted)	global	0.60	99	26.26 %
MaxEnt (adapted)	global	0.70 - 0.80	86	22.81 %
MaxEnt (adapted)	global	0.90 - 0.91	85	22.55 %
MaxEnt (adapted)	global	0.92 0.93	88	23.34 %
MaxEnt (adapted)	global	0.94	87	23.08 %
MaxEnt (adapted)	global	0.95	89	23.61 %
MaxEnt (adapted)	global	0.96 - 0.98	96	25.46 %
MaxEnt (adapted)	global	0.99	93	24.67 %

TABLE 5.16: Results of entailment-graph-based email categorization on development data using best *MaxEntClassificationEDA* (adapted) model and different graph optimizers.

Configuration	Optimizer	Threshold	Correct	Accuracy
baseline	-	-	71	18.83 %
MultiAlign	simple	0.50	47	12.47 %
MultiAlign	simple	0.51 - 0.52	48	12.73 %
MultiAlign	simple	0.53 - 0.54	47	12.47 %
MultiAlign	simple	0.55 - 0.56	48	12.73 %
MultiAlign	simple	0.57	29	7.69 %
MultiAlign	simple	0.58 - 0.60	27	7.16 %
MultiAlign	simple	0.70	59	15.65 %
MultiAlign	simple	> 0.8	71	18.83 %
MultiAlign	global	0.50	87	23.08 %
MultiAlign	global	0.51 - 0.52	85	22.55 %
MultiAlign	global	0.53 - 0.55	88	**23.34 %**
MultiAlign	global	0.56	87	23.08 %
MultiAlign	global	0.57	68	18.04 %
MultiAlign	global	0.58	70	18.57 %
MultiAlign	global	0.59	68	18.04 %
MultiAlign	global	0.60	66	17.51 %
MultiAlign	global	0.70	61	16.18 %
MultiAlign	global	> 0.80	86	18.83 %

TABLE 5.17: Results of entailment-graph-based email categorization on development data using best *MultiAlign* model and different graph optimizers.

Configuration	Optimizer	Threshold	Correct	Accuracy
baseline (no graph information)	-	-	86	22.81 %
gold graph (best feature set)	simple	-	145	**38.46 %**
best SEDA graph	simple	0.91	134	**35.54 %**
best MaxEnt (adapted) graph	global	0.5	113	29.97 %
best MultiAlign graph	global	0.53 - 0.55	101	26.79 %

TABLE 5.18: Results of entailment-graph-based email categorization on test data (using best setting per method as optimized on the development set).

Chapter 6

Summary, Conclusions and Future Work

6.1 Summary and conclusions

The research problem addressed in this work was to investigate the applicability of entailment graphs for solving NLP tasks. I approached this problem by generating entailment graphs for two specific NLP applications, relation extraction and email categorization, and evaluating the usefulness of the generated knowledge for solving the two tasks.

To address the task of relation extraction, I used entailment graphs to exploit knowledge about semantic relationships holding among relation patterns. I introduced a new type of entailment graph, which uses automatically learned complex relation extraction patterns as its basic element and structures them hierarchically along the entailment relation. The underlying assumption made was that structuring relation extraction patterns based on the semantics expressed by the patterns can be of help in the pattern selection process. To this end, I presented and evaluated a novel approach, which uses the generated knowledge for selecting reliable patterns for the relation extraction task. For measuring the usefulness of entailment graphs in this context, I created a gold standard dataset of entailment graphs for various semantic relations. In addition, I showed how state-of-the-art RTE technology can be applied and adapted to generate pattern-based entailment graphs for the task automatically. In particular, I employed and improved an existing alignment-based entailment classifier, which makes use of external knowledge

resources, and experimented with different graph optimization strategies. The classifier was trained on a manageable amount of annotated patterns for a single semantic relation, which resulted in a generic model that can also make entailment decisions for other semantic relations. Experimental results on various relations exhibited the benefits of structuring and selecting patterns based on entailment graphs. They suggest that meaningful pattern-based entailment graphs can be constructed automatically and that the derived knowledge is in fact valuable for selecting useful relation extraction patterns. In particular, entailment graph based filtering can help achieve higher precision than methods which do not take into account the asymmetric nature of semantic relations.

Addressing the task of email categorization, I proposed and evaluated a method that makes use of knowledge about semantic relationships holding among textual expressions extracted from emails and category descriptions, represented in the form of an entailment graph. Analyzing a real-world dataset of German customer emails and associated categories, I identified inference phenomena relevant for assigning emails to a given category and measured their distribution in the dataset. Based on the analysis, I generated entailment graphs (both manually and automatically) and presented an approach that uses the knowledge expressed by these graphs when building an email classifier. In particular, I used entailment graph knowledge to derive enriched text representations and evaluated the effect of using different inference types for generating feature representations for the classifier. The experimental results showed that the generated knowledge can in fact improve the performance of the email classifier, in a scenario where little training data is available. The experiments did not reveal any clear positive effect using knowledge of entailment relations holding across more complex textual expressions, suggesting that, for this task, word- or multi-word-based entailment graphs are sufficient.

Evaluating the usage of entailment graphs in the context of the two NLP tasks, I conclude that entailment graphs are in fact an effective way of representing semantic knowledge for a particular task or domain and that current RTE technology can be successfully applied to build this type of structured knowledge resource automatically. Nevertheless, state-of-the-art technology is restricted to deciding on entailment for cases in which inferences can be made at the level of lexical semantics or syntactic transformation. More sophisticated systems would be required to deal with cases involving more complex types of inference, e.g., to properly handle

expressions extracted from sentences containing irrealis moods or to identify entailment relations that are highly domain-specific and not covered by any general language resource employed by the entailment system.

6.2 Future work

6.2.1 Recognizing relevant expressions

When building entailment graphs for the email categorization task, I assumed the text expressions forming the nodes of the graphs to be given: Relevant expressions were identified as part of the manual analysis of entailment phenomena in the data. Building entailment graphs for the task in a fully automatic way would require the development of methods to identify these relevant expressions automatically. Section 3.4.4.1 sketches first ideas how this step could be automatized for different types of expressions (sentences, tokens, keywords, syntactic dependencies). Given the results of the experiments carried out in Section 5.5, the usage of relevant multi-token terms in addition to individual words appears most promising. An approach towards identifying relevant multi-token expressions could be the extraction and ranking of phrases. A range of methods to address this task in the context of emails is described in Lahiri et al. [2017].

6.2.2 Other applications

In this thesis, the usefulness of entailment graphs was evaluated with regard to two specific NLP tasks, namely email categorization and relation extraction. Nevertheless, the underlying technology is expected to be applicable for a wide range of applications, including all tasks that can benefit from the formal representation of knowlege in a resource that is easily extendible and interpretable and allows for inferencing on natural language data. With the current limitations of state-of-the-art RTE technology, especially with respect to deciding entailment for cases that require world knowledge or logical inferences, the most suitable application areas for automatically generated entailment graphs are those, in which compromises in accuracy are acceptable or in which the amount of knowledge to be generated is manageable in a sense that the automated procedures can be followed by a manual curation phase. A possible application area for entailment graphs is the field

of education, where entailment graphs could be applied for automatically grading student answers or for developing intelligent tutoring systems. For example, in computer-assisted education, entailment graphs could be used to model knowledge about a particular topic. This knowledge could then be used to automatically validate or grade student answers based on a model solution provided by the teacher, or to present personalized follow-up questions based on answers previously given by the student.

6.2.3 Deep learning

Recently, deep learning approaches have been widely applied in many areas including natural language processing. Deep learning refers to approaches based on deep neural networks, i.e., artificial neural networks with multiple hidden layers of units between the input and output layers, which can model complex non-linear relationships. For language modeling applications, research has very successfully applied recurrent neural networks [Mikolov et al., 2010], especially LSTM [Hochreiter and Schmidhuber, 1997]. In the context of RTE, neural networks were recently applied by Bowman et al. [2015] and Rocktäschel et al. [2015]. Training neural networks requires fairly large amounts of annotated training material, which was not available and not feasible to generate for the specialized tasks addressed in this thesis. However, with the current success of these approaches, it is worthwhile considering to train an RTE classifier using neural networks, in cases where the generation of sufficient amounts of training data is feasible.

6.2.4 Performance issues

When building entailment graphs from a set of n input textual expressions, the number of possible T/H pairs to compute entailment for is $n^2 - n$. For example, given two textual expressions a and b, we can generate two T/H pairs (one with expression a being the H part of the pair, and b being the T part, and one with b being H and a being T). For 100 textual expressions, the number of possible pairs increases to 9,900. With the runtime complexity of the basic graph building algorithm being $O(n^2)$, this can easily lead to performance issues, given a large number of input expressions. This issue can be addressed in various ways. One option is to reduce the number of generated T/H pairs, either by deriving decisions

by making additional assumptions, as proposed in Section 3.3.3, or by removing a subset of possible T/H pairs based on logical considerations, as proposed in Section 4.3.3.3. Another option to reduce processing time is to parallelize the execution of the program. This could be achieved by distributing the computation of entailment decisions across several machines, for example by splitting up the set of T/H pairs and produce a subset of decisions on each machine or by splitting up the RTE engine into components for specialized tasks (e.g., into a single aligner for each available knowledge resource) and merge the individual entailment scores produced by each component to a single entailment decision.

Bibliography

Adler, M., Berant, J., and Dagan, I. (2012). Entailment-based Text Exploration with Application to the Health-care Domain. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, Jeju Island, Korea. Association for Computational Linguistics.

Agichtein, E. (2006). Confidence Estimation Methods for Partially Supervised Information Extraction. In *Proceedings of the Sixth SIAM International Conference on Data Mining*, pages 539–543, Bethesda, MD, USA.

Agichtein, E. and Gravano, L. (2000). Snowball: Extracting Relations from Large Plain-text Collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 85–94, San Antonio, TX, USA.

Agirre, E., Diab, M., Cer, D., and Gonzalez-Agirre, A. (2012). Task 6: A Pilot on Semantic Textual Similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, Montréal, Canada. Association for Computational Linguistics.

Akbik, A., Visengeriyeva, L., Kirschnick, J., and Löser, A. (2013). Effective Selectional Restrictions for Unsupervised Relation Extraction. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*, Nagoya, Japan. Asian Federation of Natural Language Processing / ACL.

Alfonseca, E., Filippova, K., Delort, J.-Y., and Garrido, G. (2012). Pattern learning for relation extraction with a hierarchical topic model. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, Jeju Island, Korea. Association for Computational Linguistics.

Angeli, G., Premkumar, M. J., and Manning, C. D. (2015). Leveraging Linguistic Structure For Open Domain Information Extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and*

the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Beijing, China. Association for Computer Linguistics.

Apté, C., Damerau, F., and Weiss, S. M. (1994). Towards language independent automated learning of text categorization models. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Dublin, Ireland. Springer, New York.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 86–90, Montréal, Canada.

Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., and Etzioni, O. (2007). Open Information Extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 2670–2676, Hyderabad, India.

Bar-Haim, R. (2010). *Semantic Inference at the Lexical-Syntactic Level*. PhD thesis, Department of Computer Science, Bar Ilan University, Ramat Gan, Israel.

Bar-Haim, R., Dagan, I., Greental, I., and Shnarch, E. (2007). Semantic Inference at the Lexical-Syntactic Level. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 871–876, Vancouver, British Columbia, Canada.

Bard, J. B. L. and Rhee, S. Y. (2004). Ontologies in biology: design, applications and future challenges. *Nature Reviews Genetics*, 5:213–222.

Bauer, S., Clark, S., Rimell, L., and Graepel, T. (2014). Learning a Theory of Marriage (and other relations) from a Web Corpus. In *Advances in Information Retrieval*, pages 591–597. Springer International Publishing.

Beckers, T., Frommholz, I., and Bönning, R. (2009). Multi-facet Classification of E-mails in a Helpdesk Scenario. In *Proceedings of the Information Retrieval Workshop at Lernen – Wissen – Adaptivität*, Darmstadt, Germany.

Bekkerman, R., McCallum, A., and Huang, G. (2004). Automatic Categorization of Email into Folders: Benchmark Experiments on Enron and SRI Corpora. *Center for Intelligent Information Retrieval, Technical Report IR*, 418.

Bentivogli, L. and Magnini, B. (2014). An Italian Dataset of Textual Entailment Graphs for Text Exploration of Customer Interactions. In *Proceedings of the first Italian Computational Linguistics Conference*, Pisa, Italy.

Berant, J., Dagan, I., Adler, M., and Goldberger, J. (2012). Efficient Tree-based Approximation for Entailment Graph Learning. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, Jeju Island, Korea. Association for Computational Linguistics.

Berant, J., Dagan, I., and Goldberger, J. (2010). Global Learning of Focused Entailment Graphs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1220–1229, Uppsala, Sweden.

Berant, J., Dagan, I., and Goldberger, J. (2011). Global Learning of Typed Entailment Rules. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 610–619, Portland, Oregon.

Bhagat, R. and Hovy, E. H. (2013). What Is a Paraphrase? *Computational Linguistics*, 39(3):463–472.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, 3:993–1022.

Bloehdorn, S., Cimiano, P., and Hotho, A. (2006). Learning Ontologies to Improve Text Clustering and Classification. In *Proceedings of the 29th Annual Conference of the German Classification Society*, pages 334–341, Magdeburg, Germany.

Bloehdorn, S. and Hotho, A. (2006). Boosting for Text Classification with Semantic Features. In Mobasher, B., Nasraoui, O., Liu, B., and Masand, B., editors, *Advances in Web Mining and Web Usage Analysis*, volume 3932 of *Lecture Notes in Computer Science*, pages 149–166. Springer.

Bollacker, K. D., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1247–1250, Vancouver, Canada.

Bos, J. and Markert, K. (2005). Recognising Textual Entailment with Logical Inference. In *Proceedings of the Human Language Technology Conference and*

Conference on Empirical Methods in Natural Language Processing, Vancouver, British Columbia, Canada.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. *Computing Research Repository*, abs/1508.05326.

Bär, D., Zesch, T., and Gurevych, I. (2011). A Reflective View on Text Similarity. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, pages 515–520, Hissar, Bulgaria.

Brin, S. (1998). Extracting Patterns and Relations from the World Wide Web. In *Proceedings of the 1st International Workshop on the Web and Databases*, pages 172–183.

Brinton, L. (2000). *The Structure of Modern English: A Linguistic Introduction*. Number Bd. 1. John Benjamins Publishing.

Bunescu, R. C. and Mooney, R. J. (2005). A Shortest Path Dependency Kernel for Relation Extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731.

Burchardt, A. (2008). *Modeling textual entailment with role-semantic information*. PhD thesis, Universität des Saarlandes.

Busemann, S., Schmeier, S., and Arens, R. G. (2000). Message Classification in the Call Center. In *Proceedings of the 6th Applied Natural Language Processing Conference*, pages 158–165, Seattle, Washington, USA.

Cai, L. and Hofmann, T. (2003). Text Categorization by Boosting Automatically Extracted Concepts. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 182–189, Toronto, Canada.

Cavnar, W. B. and Trenkle, J. M. (1994). N-Gram-Based Text Categorization. In *In Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175.

Celikyilmaz, A., Thint, M., and Huang, Z. (2009). A Graph-based Semi-supervised Learning for Question-answering. In *Proceedings of the Joint Conference of the*

47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pages 719–727, Suntec, Singapore.

Chambers, N., Cer, D., Grenager, T., Hall, D., Kiddon, C., MacCartney, B., de Marneffe, M.-C., Ramage, D., Yeh, E., and Manning, C. D. (2007). Learning Alignments and Leveraging Natural Logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 165–170, Prague, Czech Republic.

Chang, M.-W., Goldwasser, D., Roth, D., and Srikumar, V. (2010). Discriminative Learning over Constrained Latent Representations. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 429–437, Los Angeles, USA.

Chapman, W. W., Bridewell, W., Hanbury, P., Cooper, G. F., and Buchanan, B. G. (2001). A Simple Algorithm for Identifying Negated Findings and Diseases in Discharge Summaries. *Journal of Biomedical Informatics*, 34(5):301 – 310.

Chklovski, T. and Pantel, P. (2004). VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In *Proceedings of EMNLP*, pages 33–40, Barcelona, Spain.

Cimiano, P. and Völker, J. (2005). Text2Onto - A Framework for Ontology Learning and Data-driven Change Discovery. In *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems*, volume 3513, pages 227–238, Alicante, Spain.

Clark, P. and Harrison, P. (2009). An Inference-Based Approach to Recognizing Entailment. In *Proceedings of the Text Analysis Conference*.

Clark, P., Murray, W. R., Thompson, J., Harrison, P., Hobbs, J., and Fellbaum, C. (2007). On the Role of Lexical and World Knowledge in RTE3. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 54–59, Prague, Czech Republic.

Cohen, P. R. (1995). *Empirical Methods for Artificial Intelligence*. MIT Press, Cambridge, MA, USA.

Dagan, I. and Glickman, O. (2004). Probabilistic Textual Entailment: Generic Applied Modeling of Language Variability. In *Learning Methods for Text Understanding and Mining*.

Dagan, I., Glickman, O., and Magnini, B. (2006). The PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, pages 177–190, Southampton, UK. Springer.

Dagan, I., Roth, D., Sammons, M., and Zanzotto, F. M. (2013). *Recognizing Textual Entailment: Models and Applications*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

de Salvo Braz, R., Girju, R., Punyakanok, V., Roth, D., and Sammons, M. (2005). An Inference Model for Semantic Entailment in Natural Language. In *Proceedings of the 20th National Conference on Artificial Intelligence and the 17th Innovative Applications of Artificial Intelligence Conference*, pages 1043–1049, Pittsburgh, Pennsylvania, USA.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Denkowski, M. and Lavie, A. (2014). Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the Workshop on Statistical Machine Translation*.

Dolan, W. B. and Brockett, C. (2005). Automatically Constructing a Corpus of Sentential Paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing*. Asia Federation of Natural Language Processing.

Egozi, O., Gabrilovich, E., and Markovitch, S. (2008). Concept-Based Feature Generation and Selection for Information Retrieval. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, Chicago, IL, USA.

Eichler, K. (2005). *Automatic classification of Swedish email messages*. Bachelor thesis, Eberhard-Karls-Universität, Tübingen, Germany.

Eichler, K. and Gabryszak, A. (2017). Evaluating text representations for the categorization of German customer emails. In *Theorie, Semantik und Organisation von Wissen*, Passau, Germany.

Eichler, K., Gabryszak, A., and Neumann, G. (2014). An analysis of textual inference in German customer emails. In *Proceedings of the Third Joint Conference on Lexical and Computational Semantics (*SEM)*, Dublin, Ireland.

Eichler, K., Hemsen, H., and Neumann, G. (2008). Unsupervised relation extraction from web documents. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, Paris, France.

Eichler, K., Meisdrock, M., and Schmeier, S. (2012). Search and Topic Detection in Customer Requests - Optimizing a Customer Support System. *KI - Künstliche Intelligenz*, 26(4):419–422.

Eichler, K. and Neumann, G. (2010). DFKI KeyWE: Ranking Keyphrases Extracted from Scientific Articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, Los Angeles, USA.

Eichler, K., Xu, F., Uszkoreit, H., Hennig, L., and Krause, S. (2016). TEG-REP: A Corpus of Textual Entailment Graphs based on Relation Extraction Patterns. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, Portorož, Slovenia.

Eichler, K., Xu, F., Uszkoreit, H., and Krause, S. (2017). Generating pattern-based entailment graphs for relation extraction. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics*, Vancouver, Canada.

Fellbaum, C. (1998). *WordNet: an electronic lexical database*. MIT Press, Cambridge, MA, USA.

Ferrández, Ó., Spurk, C., Kouylekov, M., Dornescu, I., Ferrández, S., Negri, M., Izquierdo, R., Tomás, D., Orasan, C., Neumann, G., Magnini, B., and González, J. L. V. (2011). The QALL-ME framework: A specifiable-domain multilingual Question Answering architecture. *Journal of Web Semantics*, 9(2):137–145.

Frank, A. and Padó, S. (2012). Semantics in computational lexicons. In *Semantics. An international handbook of natural language meaning*, volume 3. Mouton de Gruyter, Berlin, New York.

Fu, R., Guo, J., Qin, B., Che, W., Wang, H., and Liu, T. (2014). Learning Semantic Hierarchies via Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1199–1209, Baltimore, USA. Association for Computational Linguistics.

Fürnkranz, J. (1998). A Study Using n-gram Features for Text Categorization. Technical Report OEFAI-TR-98-30, Austrian Research Institute for Artificial Intelligence, Wien, Austria.

Gabrilovich, E. and Markovitch, S. (2005). Feature Generation for Text Categorization Using World Knowledge. In *Proceedings of The Nineteenth International Joint Conference for Artificial Intelligence*, pages 1048–1053, Edinburgh, Scotland.

Gabrilovich, E. and Markovitch, S. (2006). Overcoming the Brittleness Bottleneck Using Wikipedia: Enhancing Text Categorization with Encyclopedic Knowledge. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1301–1306, Boston, Massachusetts. AAAI Press.

Gabrilovich, E., Ringgaard, M., and Subramanya, A. (2013). FACC1: Freebase annotation of ClueWeb corpora, Version 1 (Release date 2013-06-26, Format version 1, Correction level 0).

Garoufi, K. (2007). *Towards a Better Understanding of Applied Textual Entailment Annotation and Evaluation*. Master thesis, Saarland University, Germany.

Gonzalo, J., Verdejo, F., Chugur, I., and Cigarrán, J. M. (1998). Indexing with WordNet synsets can improve Text Retrieval. *CoRR*, cmp-lg/9808002.

Gupta, R. and Ratinov, L. (2008). Text Categorization with Knowledge Transfer from Heterogeneous Data Sources. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 842–847, Chicago, Illinois, USA.

Gurevych, I., Mühlhäuser, M., Mueller, C., Steimle, J., Weimer, M., and Zesch, T. (2007). Darmstadt Knowledge Processing Repository Based on UIMA. In *Proceedings of the First Workshop on Unstructured Information Management Architecture*.

Habash, N. and Dorr, B. (2003). A Categorial Variation Database for English. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 17–23, Edmonton, Canada. Association for Computational Linguistics.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explorations Newsletter*, 11(1):10–18.

Hamp, B. and Feldweg, H. (1997). GermaNet - a Lexical-Semantic Net for German. In *Proceedings of the ACL Workshop on Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 9–15.

Harabagiu, S. and Hickl, A. (2006). Methods for Using Textual Entailment in Open-Domain Question Answering. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 905–912, Sydney, Australia. Association for Computational Linguistics.

Harmeling, S. (2009). Inferring textual entailment with a probabilistically sound calculus. *Natural Language Engineering*, 15(4):459–477.

Harris, Z. (1954). Distributional Structure. *Word*, 10(23):146–162.

Hayes, P. J., Knecht, L. E., and Cellio, M. J. (1988). A News Story Categorization System. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pages 9–17, Austin, Texas. Association for Computational Linguistics.

Hearst, M. A. (1992). Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th Conference on Computational Linguistics*, pages 539–545, Stroudsburg, PA, USA. Association for Computational Linguistics.

Heilman, M. and Smith, N. A. (2010). Tree Edit Models for Recognizing Textual Entailments, Paraphrases, and Answers to Questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1011–1019, Stroudsburg, PA, USA. Association for Computational Linguistics.

Heylighen, F. (2001). Bootstrapping knowledge representations: from entailment meshes via semantic nets to learning webs. *Kybernetes*, 30(5/6):691–722.

Hickl, A. and Bensley, J. (2007). A Discourse Commitment-based Framework for Recognizing Textual Entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 171–176, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Hodges, D., Clark, C., Fowler, A., and Moldovan, D. (2006). Applying COGEX to Recognize Textual Entailment. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, pages 427–448, Southampton, UK. Springer.

Iftene, A. and Dobrescu, A. B. (2007). Hypothesis Transformation and Semantic Variability Rules Used in Recognizing Textual Entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 125–130, Prague. Association for Computational Linguistics.

Jespersen, O. (1965). *A modern English grammar on historical principles*. Allen & Unwin, London.

Joachims, T. (1998). Text Categorization with Suport Vector Machines: Learning with Many Relevant Features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, London, UK, UK. Springer.

Johnson, R. and Zhang, T. (2015). Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112.

Kehagias, A., Petridis, V., Kaburlasos, V. G., and Fragkou, P. (2003). A Comparison of Word- and Sense-Based Text Categorization Using Several Classification Algorithms. *Journal of Intelligent Information Systems*, 21(3):227–247.

Kipper-Schuler, K. (2005). *Verbnet: A Broad-coverage, Comprehensive Verb Lexicon*. PhD thesis, University of Pennsylvania, Philadelphia, PA, USA.

Kok, S. and Domingos, P. (2008). Extracting Semantic Networks from Text Via Relational Clustering. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 624–639, Berlin, Heidelberg. Springer.

Kolesnyk, V., Rocktäschel, T., and Riedel, S. (2016). Generating Natural Language Inference Chains. *CoRR*, abs/1606.01404.

Kotlerman, L., Dagan, I., Magnini, B., and Bentivogli, L. (2015). Textual entailment graphs. *Natural Language Engineering*, 21:699–724.

Kotlerman, L., Dagan, I., Szpektor, I., and Zhitomirsky-geffet, M. (2010). Directional Distributional Similarity for Lexical Inference. *Natural Language Engineering*, 16(4):359–389.

Kouylekov, M. and Negri, M. (2010). An Open-Source Package for Recognizing Textual Entailment. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 42–47, Uppsala, Sweden.

Krause, S., Li, H., Uszkoreit, H., and Xu, F. (2012). Large-Scale Learning of Relation-Extraction Rules with Distant Supervision from the Web. In *Proceedings of the 11th International Semantic Web Conference*. Springer.

Lahiri, S., Mihalcea, R., and Lai, P. (2017). Keyword extraction from emails. *Natural Language Engineering*, 23(2):295–317.

Landis, J. R. and Koch, G. G. (1977). The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1):159–174.

Lemur Project, T. (2009). ClueWeb09. `http://www.lemurproject.org`.

Levy, O., Dagan, I., and Goldberger, J. (2014). Focused Entailment Graphs for Open IE Propositions. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 87–97, Ann Arbor, Michigan.

Lewis, D. D. (1992). An Evaluation of Phrasal and Clustered Representations on a Text Categorization Task. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 37–50, New York, NY, USA. ACM.

Lin, D. (1998). Automatic Retrieval and Clustering of Similar Words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 768–774, Montréal, Canada. Association for Computational Linguistics.

Lin, D. and Pantel, P. (2001). DIRT - Discovery of Inference Rules from Text. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 323–328.

Lin, D. and Pantel, P. (2001). Discovery of Inference Rules for Question-answering. *Natural Language Engineering*, 7(4):343–360.

MacCartney, B. (2009). *Natural language inference*. PhD thesis, Stanford University, USA.

MacCartney, B., Grenager, T., de Marneffe, M.-C., Cer, D., and Manning, C. D. (2006). Learning to Recognize Features of Valid Textual Entailments. In *Proceedings of the Main Conference on Human Language Technology of the North American Chapter of the Association of Computational Linguistics*, pages 41–48, New York, New York. Association for Computational Linguistics.

MacCartney, B. and Manning, C. D. (2009). An Extended Model of Natural Logic. In *Proceedings of the Eighth International Conference on Computational Semantics*, pages 140–156, Stroudsburg, PA, USA. Association for Computational Linguistics.

Madhumita (2016). *Recognizing textual entailment*. Master thesis, Saarland University, Saarbrücken, Germany.

Madnani, N. and Dorr, B. J. (2010). Generating Phrasal and Sentential Paraphrases: A Survey of Data-driven Methods. *Computational Linguistics*, 36(3):341–387.

Magnini, B., Dagan, I., Neumann, G., and Padó, S. (2014). Entailment Graphs for Text Analytics in the Excitement Project. In *Text, Speech and Dialogue*, volume 8655 of *Lecture Notes in Computer Science*, pages 11–18. Springer International Publishing.

Magnini, B., Zanoli, R., Dagan, I., **Eichler, K.**, Neumann, G., Noh, T.-G., Padó, S., Stern, A., and Levy, O. (2014). The Excitement Open Platform for Textual Inferences. In *Proceedings of the 52nd Annual Meeting of the Association of Computational Linguistics*, Baltimore, USA.

Marelli, M., Bentivogli, L., Baroni, M., Bernardi, R., Menini, S., and Zamparelli, R. (2014). Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 1–8, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Mehdad, Y., Cabrio, E., Negri, M., Kouylekov, M., and Magnini, B. (2009). Using Lexical Resources in a Distance-Based Approach to RTE. In *Proceedings of the Second Text Analysis Conference*, Gaithersburg, Maryland, USA.

Mehdad, Y., Carenini, G., Ng, R. T., and Joty, S. R. (2013). Towards Topic Labeling with Phrase Entailment and Aggregation. In *Human Language Technologies: Proceedings of the Conference of the North American Chapter of the Association of Computational Linguistics*, pages 179–189, Atlanta, Georgia, USA.

Meyers, A., Macleod, C., Yangarber, R., Grishman, R., Barrett, L., and Reeves, R. (1998). Using NOMLEX to Produce Nominalization Patterns for Information Extraction. In *Proceedings of the Workshop On The Computational Treatment Of Nominals*.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association*, pages 1045–1048, Makuhari, Chiba, Japan.

Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Milne, D. N., Witten, I. H., and Nichols, D. M. (2007). A Knowledge-based Search Engine Powered by Wikipedia. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*, pages 445–454, New York, NY, USA. ACM.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant Supervision for Relation Extraction Without Labeled Data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.

Mirkin, S., Specia, L., Cancedda, N., Dagan, I., Dymetman, M., and Szpektor, I. (2009). Source-Language Entailment Modeling for Translating Unknown Terms. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 791–799, Suntec, Singapore. Association for Computational Linguistics.

Montague, R. (1970). Universal Grammar. *Theoria*, 36(3):373–398.

Moro, A., Li, H., Krause, S., Xu, F., Navigli, R., and Uszkoreit, H. (2013). Semantic Rule Filtering for Web-Scale Relation Extraction. In *Proceedings of the 12th International Semantic Web Conference*, pages 347–362. Springer.

Naber, D. (2015). jWordSplitter. `https://github.com/danielnaber/jwordsplitter`. [Online; accessed in 2015].

Nakashole, N., Weikum, G., and Suchanek, F. M. (2012). PATTY: A taxonomy of relational patterns with semantic types. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1135–1145, Jeju Island, Korea.

Nastase, V., Mihalcea, R., and Radev, D. R. (2015). A survey of graphs in natural language processing. *Natural Language Engineering*, 21(5):665–698.

Negri, M., Kouylekov, M., Magnini, B., Mehdad, Y., and Cabrio, E. (2009). Towards Extensible Textual Entailment Engines: The EDITS Package. In *Proceedings of the XIth International Conference of the Italian Association for Artificial Intelligence*, pages 314–323, Reggio Emilia, Italy.

Neumann, G. and Schmeier, S. (1999). Combining shallow text processing and machine learning in real world applications. In *Proceedings of the IJCAI Workshop on Machine Learning for Information Filtering*.

Nickel, M., Murphy, K., Tresp, V., and Gabrilovich, E. (2015). A Review of Relational Machine Learning for Knowledge Graphs: From Multi-Relational Link Prediction to Automated Knowledge Graph Construction. *CoRR*, abs/1503.00759.

Nielsen, R. D., Ward, W., and Martin, J. H. (2009). Recognizing entailment in intelligent tutoring systems. *Natural Language Engineering*, 15(4):479–501.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., and Marsi, E. (2007). Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Noh, T.-G., Padó, S., Shwartz, V., Dagan, I., Nastase, V., **Eichler, K.**, Kotlerman, L., and Adler, M. (2015). Multi-Level Alignments As An Extensible Representation Basis for Textual Entailment Algorithms. In *Proceedings of the 4th Joint Conference on Lexical and Computational Semantics*, Denver, USA.

Noh, T.-G. and Padó, S. (2013). Using UIMA to Structure An Open Platform for Textual Entailment. In *Proceedings of the 3rd Workshop on Unstructured Information Management Architecture*, pages 26–33, Darmstadt, Germany.

Padó, S., Galley, M., Jurafsky, D., and Manning, C. (2009). Robust Machine Translation Evaluation with Entailment Features. In *Proceedings of ACL-IJCNLP*, pages 297–305.

Paolo, Rosso, P., Ferretti, E., Jiménez, D., and Vidal, V. (2004). Text Categorization and Information Retrieval Using Wordnet Senses. In *In Proceedings of the 2nd Global Wordnet Conference (GWC'04*, pages 299–304. Springer.

Papka, R. and Allan, J. (1998). Document Classification Using Multiword Features. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, pages 124–131, Bethesda, Maryland, USA. ACM.

Partee, B. and Rooth, M. (1983). Generalized Conjunction and Type Ambiguity. In Bäuerle, R., Schwarze, C., and von Stechow, A., editors, *Meaning, Use and Interpretation of Language*, pages 361–383. De Gruyter, Berlin.

Pianta, E., Bentivogli, L., and Girardi, C. (2002). MultiWordNet: developing an aligned multilingual database. In *Proceedings of the First International Conference on Global WordNet*.

Ratinov, L., Roth, D., and Srikumar, V. (2008). Conceptual search and text categorization. Technical report, University of Illinois, Urbana, IL, USA.

Ravichandran, D. and Hovy, E. (2002). Learning Surface Text Patterns for a Question Answering System. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 41–47, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rennie, J. D. M. (2000). ifile: An Application of Machine Learning to E-Mail Filtering. In *Proceedings of the KDD Workshop on Text Mining*.

Reppen, R., Fitzmaurice, S. M., and Biber, D. (2002). *Using Corpora to Explore Linguistic Variation*. John Benjamins.

Riedel, S., Yao, L., McCallum, A., and Marlin, B. M. (2013). Relation Extraction with Matrix Factorization and Universal Schemas. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84.

Riloff, E. (1996). Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044–1049. AAAI Press.

Rocktäschel, T., Grefenstette, E., Hermann, K. M., Kociský, T., and Blunsom, P. (2015). Reasoning about Entailment with Neural Attention. *CoRR*, abs/1509.06664.

Romano, L., Kouylekov, M., Szpektor, I., Dagan, I., and Lavelli, A. (2006). Investigating a Generic Paraphrase-Based Approach for Relation Extraction. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy.

Roth, D., Sammons, M., and Vydiswaran, V. G. V. (2009). A Framework for Entailed Relation Recognition. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore, Short Papers*, pages 57–60.

Ruiz, M. and Srinivasan, P. (1997). Automatic Text Categorization Using Neural Networks. *Advances in Classification Research Online*, 8(1):58–68.

Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. (1998). A Bayesian Approach to Filtering Junk E-Mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin. AAAI Technical Report WS-98-05.

Sammons, M., Vydiswaran, V., Vieira, T., Johri, N., Chang, M., Goldwasser, D., Srikumar, V., Kundu, G., Tu, Y., Small, K., Rule, J., Do, Q., and Roth, D. (2009). Relation Alignment for Textual Entailment Recognition. In *Proceedings of the Text Analysis Conference*.

Schapire, R. E. and Singer, Y. (2000). BoosTexter: A Boosting-based Systemfor Text Categorization. *Machine Learning*, 39(2-3):135–168.

Schmid, H. (1994). Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, UK.

Schütze, H., Hull, D. A., and Pedersen, J. O. (1995). A Comparison of Classifiers and Document Representations for the Routing Problem. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 229–237, New York, NY, USA. ACM.

Schwarzschild, R. (1999). Givenness, Avoid F, and Other Constraints on the Placement of Accent. *Natural Language Semantics*, 7(2):141–177.

Scott, S. (1998). Feature Engineering for a Symbolic Approach to Text Classification. Technical report, School of Information Technology and Engineering, University of Ottawa, Ontario, Canada.

Scott, S. and Matwin, S. (1999). Feature engineering for text classification. In *Proceedings of the 16th International Conference on Machine Learning*, pages 379–388. Morgan Kaufmann Publishers.

Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Computational Surveys*, 34(1):1–47.

Segal, R. B. and Kephart, J. O. (1999). MailCat: An Intelligent Assistant for Organizing E-Mail. In *In Proceedings of the Third International Conference on Autonomous Agents*, pages 276–282. ACM Press.

Shinyama, Y. and Sekine, S. (2006). Preemptive Information Extraction using Unrestricted Relation Discovery. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 304–311, New York City, USA. Association for Computational Linguistics.

Slodzian, M. (2001). Wordnet: What about its linguistic relevancy? In *Proceedings of the EKAW Workshop on Ontologies and Texts*, Juan-les-Pins, France. Sunsite Aachen.

Sánchez Valencia, V. (1996). Parsing-driven Inference: Natural Logic. *Linguistic Analysis*, 3–4:258–285.

Snow, R., Jurafsky, D., and Ng, A. Y. (2004). Learning Syntactic Patterns for Automatic Hypernym Discovery. In *Advances in Neural Information Processing Systems 17*, pages 1297–1304, Vancouver, British Columbia, Canada.

Stern, A. and Dagan, I. (2011). A Confidence Model for Syntactically-Motivated Entailment Proofs. In *Proceedings of Recent Advances in Natural Language Processing*, pages 455–462, Hissar, Bulgaria.

Stern, A. and Dagan, I. (2012). BIUTEE: A modular open-source system for recognizing textual entailment. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, Stroudsburg, PA, USA.

Sukkarieh, J. Z. and Stoyanchev, S. (2009). Automating Model Building in C-rater. In *Proceedings of the 2009 Workshop on Applied Textual Inference*, pages 61–69, Stroudsburg, PA, USA. Association for Computational Linguistics.

Surdeanu, M., Ciaramita, M., and Zaragoza, H. (2008). Learning to rank answers on large online QA collections. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 719–727.

Terryn, A. R., Macken, L., and Lefever, E. (2016). Dutch Hypernym Detection: Does Decompounding Help? In *Proceedings of the Joint Workshop on Language and Ontology & Terminology and Knowledge Structures*, pages 74–78.

Thomas, P., Pietschmann, S., Solt, I., Tikk, D., and Leser, U. (2011). Not all links are equal: Exploiting Dependency Types for the Extraction of Protein-Protein Interactions from Text. In *Proceedings of BioNLP 2011 Workshop*, pages 1–9. Association for Computational Linguistics.

Toutanova, K. and Chen, D. (2015). Observed Versus Latent Features for Knowledge Base and Text Inference. In *Proceedings of the Workshop on Continuous Vector Space Models and Their Compositionality*.

Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., and Gamon, M. (2015). Representing Text for Joint Embedding of Text and Knowledge Bases. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Ureña, L. A., de Buenaga, M., and Gómez, J. M. (2001). Integrating Linguistic Resources in TC through WSD. *Computers and the Humanities*, 35(2):215–230.

Utt, J. and Padó, S. (2014). Crosslingual and Multilingual Construction of Syntax-Based Vector Space Models. *Transactions of the Association of Computational Linguistics*, 2:245–258.

Vanderwende, L. and Dolan, W. B. (2005). What Syntax Can Contribute in the Entailment Task. volume 3944 of *Lecture Notes in Computer Science*, pages 205–216. Springer.

Volokh, A. and Neumann, G. (2011). Using MT-Based Metrics for RTE. In *Proceedings of the 4th Text Analysis Conference*, Gaithersburg, Maryland, USA. National Institute of Standards and Technology.

von Fintel, K. (1999). NPI licensing, Strawson entailment, and context dependency. *Journal of Semantics*, 16(2):97–148.

Wang, B. B., Mckay, R. I. B., Abbass, H. A., and Barlow, M. (2003). A Comparative Study for Domain Ontology Guided Feature Extraction. In *Proceedings of the 26th Australasian Computer Science Conference*, pages 69–78, Adelaide, Australia. Australian Computer Society.

Wang, R. and Neumann, G. (2007). Recognizing Textual Entailment Using a Subsequence Kernel Method. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Wang, R. and Neumann, G. (2008). Relation Validation via Textual Entailment. In Adrian, B., Neumann, G., Troussov, A., and Popov, B., editors, *Proceedings of the 1st International Workshop on Ontology-based Information Extraction Systems*, volume 400 of *CEUR Workshop Proceedings Online*, pages 26–37. CEUR.

Wilcox, A. B. (2000). *Automated Classification of Medical Text Reports*. Phd thesis, Columbia University.

Yangarber, R., Grishman, R., and Tapanainen, P. (2000). Automatic Acquisition of Domain Knowledge for Information Extraction. In *Proceedings of the Conference on Computational Linguistics*, pages 940–946.

Yao, L., Haghighi, A., Riedel, S., and McCallum, A. (2011). Structured Relation Discovery Using Generative Models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1456–1466, Edinburgh, United Kingdom. Association for Computational Linguistics.

Yao, L., Riedel, S., and McCallum, A. (2012). Unsupervised Relation Discovery with Sense Disambiguation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, Jeju Island, Korea.

Yates, A. and Etzioni, O. (2009). Unsupervised Methods for Determining Object and Relation Synonyms on the Web. *Journal of Artificial Intelligence Research (JAIR)*, 34:255–296.

Young, P., Lai, A., Hodosh, M., and Hockenmaier, J. (2014). From image descriptions to visual denotations: New similarity metrics for semantic inference over

event descriptions. *Transactions of the Association of Computational Linguistics*, 2:67–78.

Yuret, D., Rimell, L., and Han, A. (2013). Parser evaluation using textual entailments. *Language Resources and Evaluation*, 47(3):639–659.

Zanzotto, F. M. and Moschitti, A. (2006). Automatic Learning of Textual Entailments with Cross-Pair Similarities. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia.

Zelenko, D., Aone, C., and Richardella, A. (2003). Kernel methods for relation extraction. *The Journal of Machine Learning Research*, 3:1083–1106.

Zeller, B., Šnajder, J., and Padó, S. (2013). DErivBase: Inducing and evaluating a derivational morphology resource for German. In *Proceedings of the Association of Computational Linguistics*, Sofia, Bulgaria.

Zhang, X., Zhao, J., and Lecun, Y. (2015). *Character-level convolutional networks for text classification*, volume 2015-January, pages 649–657. Neural information processing systems foundation.

Zhiguo Wang, Wael Hamza, R. F. (2017). Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4144–4150.