University of Potsdam
Hasso Plattner Institute
Information Systems Group

Knowledge Base Construction with Machine Learning Methods

Dissertation
zur Erlangung des akademischen Grades
“Doktor der Ingenieurwissenschaften”
(Dr.-Ing.)
in der Wissenschaftsdisziplin
“Informationssysteme”

eingereicht an der
Digital Engineering Fakultät
der Universität Potsdam

von
Michael Loster

Potsdam, den 20. August 2020

Reviewers

Professor Dr. Felix Naumann
Hasso Plattner Institute for Digital Engineering, University of Potsdam

Professor Dr. Myra Spiliopoulou
Faculty of Computer Science, Otto-von-Guericke-University Magdeburg

Professor Dr. Heiko Paulheim
School of Business Informatics and Mathematics, University of Mannheim

Published online on the
Publication Server of the University of Potsdam:
https://doi.org/10.25932/publishup-50145
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-501459

Abstract

Modern knowledge bases contain and organize knowledge from many different topic areas. Apart from specific entity information, they also store information about their relationships amongst each other. Combining this information results in a knowledge graph that can be particularly helpful in cases where relationships are of central importance. Among other applications, modern risk assessment in the financial sector can benefit from the inherent network structure of such knowledge graphs by assessing the consequences and risks of certain events, such as corporate insolvencies or fraudulent behavior, based on the underlying network structure. As public knowledge bases often do not contain the necessary information for the analysis of such scenarios, the need arises to create and maintain dedicated domain-specific knowledge bases.

This thesis investigates the process of creating domain-specific knowledge bases from structured and unstructured data sources. In particular, it addresses the topics of *named entity recognition* (NER), *duplicate detection*, and *knowledge validation*, which represent essential steps in the construction of knowledge bases.

As such, we present a novel method for duplicate detection based on a Siamese neural network that is able to learn a dataset-specific similarity measure which is used to identify duplicates. Using the specialized network architecture, we design and implement a knowledge transfer between two deduplication networks, which leads to significant performance improvements and a reduction of required training data.

Furthermore, we propose a named entity recognition approach that is able to identify company names by integrating external knowledge in the form of dictionaries into the training process of a conditional random field classifier. In this context, we study the effects of different dictionaries on the performance of the NER classifier. We show that both the inclusion of domain knowledge as well as the generation and use of alias names results in significant performance improvements.

For the validation of knowledge represented in a knowledge base, we introduce Colt, a framework for knowledge validation based on the interactive quality assessment of logical rules. In its most expressive implementation, we combine Gaussian processes with neural networks to create Colt-GP, an interactive algorithm for learning rule models. Unlike other approaches, Colt-GP uses knowledge graph embeddings and user feedback to cope with

data quality issues of knowledge bases. The learned rule model can be used to conditionally apply a rule and assess its quality.

Finally, we present CurEx, a prototypical system for building domain-specific knowledge bases from structured and unstructured data sources. Its modular design is based on scalable technologies, which, in addition to processing large datasets, ensures that the modules can be easily exchanged or extended. CurEx offers multiple user interfaces, each tailored to the individual needs of a specific user group and is fully compatible with the Colt framework, which can be used as part of the system.

We conduct a wide range of experiments with different datasets to determine the strengths and weaknesses of the proposed methods. To ensure the validity of our results, we compare the proposed methods with competing approaches.

Zusammenfassung

Moderne Wissensbasen enthalten und organisieren das Wissen vieler unterschiedlicher Themengebiete. So speichern sie neben bestimmten Entitätsinformationen auch Informationen über deren Beziehungen untereinander. Kombiniert man diese Informationen, ergibt sich ein Wissensgraph, der besonders in Anwendungsfällen hilfreich sein kann, in denen Entitätsbeziehungen von zentraler Bedeutung sind. Neben anderen Anwendungen, kann die moderne Risikobewertung im Finanzsektor von der inhärenten Netzwerkstruktur solcher Wissensgraphen profitieren, indem Folgen und Risiken bestimmter Ereignisse, wie z.B. Unternehmensinsolvenzen oder betrügerisches Verhalten, auf Grundlage des zugrundeliegenden Netzwerks bewertet werden. Da öffentliche Wissensbasen oft nicht die notwendigen Informationen zur Analyse solcher Szenarien enthalten, entsteht die Notwendigkeit, spezielle domänenspezifische Wissensbasen zu erstellen und zu pflegen.

Diese Arbeit untersucht den Erstellungsprozess von domänenspezifischen Wissensdatenbanken aus strukturierten und unstrukturierten Datenquellen. Im speziellen befasst sie sich mit den Bereichen *Named Entity Recognition* (NER), *Duplikaterkennung* sowie *Wissensvalidierung*, die wesentliche Prozessschritte beim Aufbau von Wissensbasen darstellen.

Wir stellen eine neuartige Methode zur Duplikaterkennung vor, die auf Siamesischen Neuronalen Netzwerken basiert und in der Lage ist, ein datensatzspezifisches Ähnlichkeitsmaß zu erlernen, welches wir zur Identifikation von Duplikaten verwenden. Unter Verwendung einer speziellen Netzwerkarchitektur entwerfen und setzen wir einen Wissenstransfer zwischen Deduplizierungsnetzwerken um, der zu erheblichen Leistungsverbesserungen und einer Reduktion der benötigten Trainingsdaten führt.

Weiterhin schlagen wir einen Ansatz zur Erkennung benannter Entitäten (Named Entity Recognition, NER) vor, der in der Lage ist, Firmennamen zu identifizieren, indem externes Wissen in Form von Wörterbüchern in den Trainingsprozess eines Conditional Random Field Klassifizierers integriert wird. In diesem Zusammenhang untersuchen wir die Auswirkungen verschiedener Wörterbücher auf die Leistungsfähigkeit des NER-Klassifikators und zeigen, dass sowohl die Einbeziehung von Domänenwissen als auch die Generierung und Verwendung von Alias-Namen zu einer signifikanten Leistungssteigerung führt.

Zur Validierung der in einer Wissensbasis enthaltenen Fakten stellen wir mit Colt ein Framework zur Wissensvalidierung vor, dass auf der interaktiven

Qualitätsbewertung von logischen Regeln basiert. In seiner ausdrucksstärksten Implementierung kombinieren wir Gauß'sche Prozesse mit neuronalen Netzen, um so Colt-GP, einen interaktiven Algorithmus zum Erlernen von Regelmodellen, zu erzeugen. Im Gegensatz zu anderen Ansätzen verwendet Colt-GP Knowledge Graph Embeddings und Nutzer-Feedback, um Datenqualitätsprobleme des zugrunde liegenden Wissensgraphen zu behandeln. Das von Colt-GP erlernte Regelmodell kann sowohl zur bedingten Anwendung einer Regel als auch zur Bewertung ihrer Qualität verwendet werden.

Schließlich stellen wir mit CurEx, ein prototypisches System zum Aufbau domänenspezifischer Wissensbasen aus strukturierten und unstrukturierten Datenquellen, vor. Sein modularer Aufbau basiert auf skalierbaren Technologien, die neben der Verarbeitung großer Datenmengen auch die einfache Austausch- und Erweiterbarkeit einzelner Module gewährleisten. CurEx bietet mehrere Benutzeroberflächen, die jeweils auf die individuellen Bedürfnisse bestimmter Benutzergruppen zugeschnitten sind. Darüber hinaus ist es vollständig kompatibel zum COLT-Framework, was als Teil des Systems verwendet werden kann.

Wir führen eine Vielzahl von Experimenten mit unterschiedlichen Datensätzen durch, um die Stärken und Schwächen der vorgeschlagenen Methoden zu ermitteln. Zudem vergleichen wir die vorgeschlagenen Methoden mit konkurrierenden Ansätzen, um die Validität unserer Ergebnisse sicherzustellen.

Acknowledgements

First of all, I would like to thank Prof. Dr. Felix Naumann for his supervision, support and the opportunity to pursue my Ph.D. in his group. His mentorship and insights had not only a significant impact on this work but also on my personal development and growth. I would also like to thank Dirk Thomas and Dr. Oliver Maspfuhl for their ongoing commitment and dedication, without which this work would not have been possible. I am also grateful for the collaboration with Prof. Dr. Davide Mottin and Prof. Dr. Paolo Papotti, whose commitment, insights, and support have contributed to the progress of my research.

I was fortunate enough to meet and work with many extremely talented researchers throughout my time at the Hasso Plattner Institute, some of whom I can now, fortunately, call friends. My special thanks go to Hazar, Ioannis, Toni, Zuo, and Tim, who not only supported me in the final stage of my studies but also gave me valuable advice and support throughout the years, it was an honor to graduate at your side.

I would also like to thank my talented students, Benjamin, Jan, Marvin, Patrick, Danijar, Tanja, and Daniel, for their valuable support and contributions to my research. Thanks are also extended to all the participants of my teaching activities, with whom I enjoyed working.

Last but not least, I would like to thank my family and friends for their continuous support and encouragement in difficult times.

Contents

Chapter 1

From Raw Data to Knowledge

Modern knowledge bases contain and organize factual information on various topics, making it accessible in machine-readable form. By providing information about real-world entities, such as people, places, and organizations, they have become a cornerstone of many advanced information systems. A central aspect that distinguishes knowledge bases from other forms of knowledge organization is that, in addition to entity information, they also contain information about relationships. Together with the entities, these relationships form a knowledge graph, that can be used to relate the entities to each other and thus recognize relationship patterns. This property makes the use of knowledge graphs particularly useful when insights into relationships play a central role in problem solving.

With DBpedia [Auer et al., 2007], Wikidata [Vrandečić and Krötzsch, 2014], and YAGO [Suchanek et al., 2007], several publicly available knowledge bases have emerged, that cover millions of facts from various domains. Apart from knowledge bases that cover a wide range of different subjects, specialized knowledge bases with a focus on specific domains also exist. For example, the *GeneOntology* [Consortium, 2004] knowledge base collects information about gene functions, while *WordNet* [Fellbaum, 1998] specializes in covering lexical associations between word groups. Although the existing knowledge bases consist of millions of different entities, they are by no means complete. Despite their enormous size, they contain only part of all available real-world information. This lack of information becomes particularly relevant if the implementation of certain use cases hinges on the existence of information usually not available in publicly accessible knowledge bases. It is therefore not surprising that large companies, such as Google [Singhal, 2012], LinkedIn [He, 2016], or Walmart [Deshpande et al., 2013] are making significant efforts to build their own customized knowledge bases that contain the information needed to support their business-relevant use cases. These include enriching search results with additional knowledge, improving ad placements, or enhancing friendship, job, or shopping recommendations.

Apart from these well-known use cases, others, such as analyzing risk factors within the financial sector, can also benefit from domain-specific knowledge bases. The assessment of potential risk factors poses an extremely complex challenge that involves the use of a large number of public as well as proprietary data sources. However, to assess

such risks with sufficient precision, it is important to consider not only individual market participants but also their relationships. Here, knowledge bases with their inherent network structure can be used to obtain a holistic view of the economic situation or to identify systemic risk factors at an early stage so that appropriate measures can be taken. Moreover, exploiting such networks allows for anticipating the effects of corporate bankruptcies on other market participants based on their relationships. This makes it possible to estimate which market participants could be affected by ripple effects of insolvencies and to what extent. Given its challenges and implications, the analysis of risk factors in the financial sector represents one motivation for this thesis, which is why a more detailed discussion of this use case is provided in the following section.

Since such use cases are difficult to address without connectivity information of the involved entities, creating a domain-specific knowledge base and its associated knowledge graph represents a necessary first step. To this end, either a customized knowledge base can be created from scratch or a publicly available knowledge base can be extended by adding additional information. In both cases, however, it should be borne in mind that the required information may be spread across many different data sources, which, depending on their nature, require special treatment. As such, data sources can be divided into structured data sources, such as database tables or CSV files, and unstructured data sources, which, in addition to image, audio, and video files, also include textual data. While structured data sources can be accessed by using traditional means, such as relational database systems or software libraries, accessing the information contained in unstructured data sources requires considerably more effort. With unstructured data sources, the contained information must first be extracted and converted into a machine-readable format before it can be processed by downstream applications. Besides separating data sources into structured and unstructured sources, they can also be divided along a second axis into publicly accessible and proprietary data sources, whose processing can present additional challenges.

To obtain a unified view of the information provided by the individual data sources, it is necessary to integrate them into a common knowledge base. This can be achieved in two ways, both of which have their advantages and disadvantages. The first approach consists of manually creating and maintaining a domain-specific knowledge base. Pursuing this approach usually leads to knowledge bases of high data quality, but at the same time limits their scope due to the high manual effort that is required for their construction. This effort is exacerbated by the fact that manually created knowledge bases tend to be maintained manually, demanding continuous effort to keep the contained information up to date. Additionally, considering the continually growing amount of information and the rate at which it changes, we can conclude that manual knowledge base creation is only feasible for very small amounts of data. Alternatively, knowledge bases can be created automatically or semi-automatically. In this scenario, a system usually takes over the creation of the knowledge base by incorporating information from structured and unstructured data sources. Unfortunately, creating extensive, domain-specific knowledge bases in a fully automated fashion still poses a major challenge, as it requires the interaction of many different techniques, such as schema matching, entity linking, or relationship extraction, that are still subject of active research. Thus,

the integration of structured data sources entails the reconciliation of different source schemata, the discovery and elimination of redundant information, and the consolidation of complementary information into a uniform representation. In addition, a wide range of text mining methods are needed to detect entities in text documents, match them with entities already present in the knowledge base, and extract relationships between them. Since neither approach poses a good solution in and of itself, an optimal solution should aim to combine the high data quality of manual approaches with the scalability of automated approaches.

To automate these tasks, machine learning methods are often used, as they are able to learn and perform tasks that otherwise have to be performed manually. As such, the use of various machine learning techniques, such as support vector machines or neural networks, enables the automatic alignment of data schemata, the detection of duplicate entries, as well as the extraction of named entities and relationships from text documents. In this way, the manual effort for creating domain-specific knowledge bases can be significantly reduced, so that processing large amounts of data no longer poses an obstacle for creating large knowledge bases. Despite their advantages, the application of machine learning approaches suffers from the disadvantage that they usually produce knowledge bases of lower data quality. Often this deterioration in data quality can be traced back to data errors introduced by the employed machine learning methods. Such data errors propagate and amplify throughout the generation process until they finally manifest and accumulate as incorrect data entries in the generated knowledge base. Driven by this fact, there is a need to not only improve the techniques used in the construction of knowledge bases but also to develop methods capable of identifying incorrect entries, the elimination of which leads to an improvement in data quality of the generated knowledge base.

This thesis focuses on constructing domain-specific knowledge bases from structured and unstructured data sources by using machine learning methods. It contributes to the state-of-the-art in the areas of *named entity recognition*, *duplicate detection*, and *knowledge validation*, addressing several of the challenges mentioned above. A central point of this thesis is to point out how machine learning methods can be used to support and simplify the construction process of domain-specific knowledge bases. It aims to advance the field towards a fully automated process for building domain-specific knowledge bases. The next section introduces both the background and use cases that served as one motivation for this work. The challenges involved in constructing domain-specific knowledge bases are outlined in Section 1.2. Section 1.3 concludes the introduction by giving an overview of the thesis structure and a summary of its main contributions.

1.1 Use cases for integrated knowledge bases

This thesis represents the result of a long-term research cooperation with one of Germany's largest financial institutions. The starting point of this cooperation was to investigate how information systems can be used to address use cases arising from the 2008 financial crisis. Based on the close collaboration with financial experts, a number of

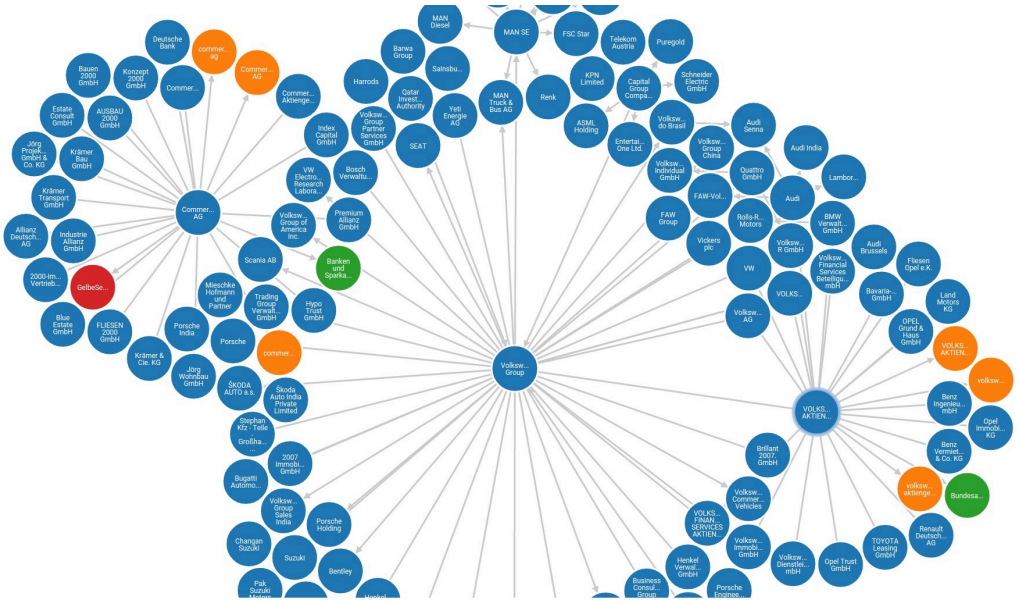

Figure 1.1: An example of a company graph in which each node represents a company that is related to other companies via various relations which are represented as edges.

use cases were identified, which henceforth served as a reference point for the conducted research activities and are presented below.

Risk management. Among the many possible applications that could benefit from domain-specific knowledge bases is modern risk analysis as performed by various financial institutions. In this context, the use of a domain-specific knowledge base that captures the economic landscape and consists of market participants and their relationships plays a central role in assessing financial risks [Amini et al., 2016]. Such risks may arise from contracts between two parties, e.g., when a bank (creditor) grants a loan to a private company (debtor). When granting loans, the risk is that the debtor cannot meet its repayment obligations, which is why the assessment of a debtor's creditworthiness is of central importance to its creditors. Traditionally, public and proprietary data sources used to assess credit risk contain only information about individual customers, but no information about their relationships. With the effects of the financial crisis of 2008/2009 at the latest, it has become clear that dependencies between individual market participants are of decisive importance when assessing financial risks. To this end, specialized knowledge bases, containing not only information about individual market participants but also information about their relationships, can be used to assess financial risk factors. Consequently, modern risk assessment should take into account network structures, such as those shown in Figure 1.1.

Other applications. Beyond the analysis of systemic risks within the financial system or the general exploration of network data, other use cases exist for which the use

of domain-specific knowledge bases is almost indispensable. As such, the "*know your customer*" scenario is concerned with developing a better understanding of a customer before entering into a business relationship with him. The core of this use case is the prevention of illegal business activities that could be carried out by criminally inclined customers - without the knowledge of the respective financial institution. Here, the network structure of the knowledge graph emerging from the underlying knowledge base can be used to examine (i) the identity of customers, (ii) their suitability to conduct business, and (iii) the risk of potential business transactions. Thus, this use case is closely related to *fraud detection*, which seeks to uncover cases of fraudulent behavior, such as money laundering or bribery. As fraud cases usually involve several parties, it is not sufficient to investigate the behavior of individual customers, but rather it is necessary to analyze the interaction patterns of multiple customers to uncover cases of fraud.

Another potential use case is that of *supply chain analysis*. As the effects of the coronavirus pandemic of 2020 have shown, the success of many companies depends directly on the functioning of their supply chains. If they fail, the consequences can be devastating and even result in the affected company no longer being able to produce or operate. Many companies around the world were surprised by the pandemic-related failures in their supply chains, which confronted them with unprecedented challenges. In light of this event, it will become increasingly important for every company to analyze and assess existing supply chain structures for potential risk factors to make business-critical supply chains more resilient to unforeseen events, such as a global pandemic. In this scenario, domain-specific knowledge bases can be used to identify hidden relationships and take them into account when estimating risk. Thus, the impact of identified risk factors on existing supply chains can be reduced or eliminated by shortening, relocating, or establishing redundancies in the affected supply chains. This practice can help to ensure a company's operational capability in the face of unforeseeable events.

1.2 Creating, maintaining, and exploring domain-specific knowledge bases

Building knowledge bases from structured and unstructured data sources has a long history [Miller, 2018; Weikum and Theobald, 2010], during which many systems with different specializations emerged. Some of the more recent systems include Knowledge Vault [Dong et al., 2014], Deep Dive [Sa et al., 2017], and Elementary [Niu et al., 2012]. Driven by the use cases outlined in the previous section, the prototypical CurEx system shown in Figure 1.2 emerged as part of this work. Designed in collaboration with financial experts, it enables the creation, curation, and exploration of domain-specific knowledge bases from structured and unstructured data sources. By consolidating information from many different data sources, it eliminates data fragmentation across multiple data sources, creating a holistic view of the existing data.

As shown in Figure 1.2, the CurEx system consists of components for text mining, structured data integration, knowledge validation (Colt), and several user interfaces that allow various user groups to interact with different parts of the system. The com-

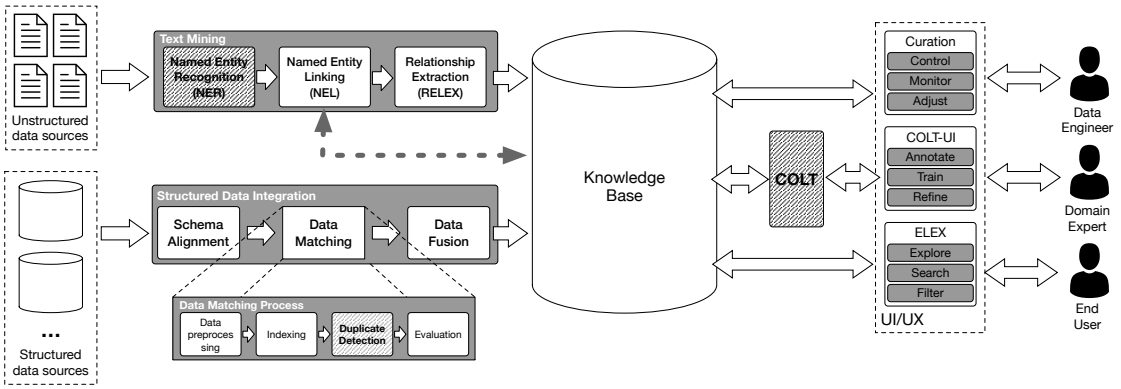

Figure 1.2: Overview of the CurEx system architecture

ponents for *text mining* and *structured data integration* form the core of the system and enable the creation of a knowledge base from structured and unstructured data sources. They consist of several sub-components, which will be covered in more detail in Chapters 2 and 3. *Knowledge validation* is realized through the COLT framework presented in Chapter 4. It enables domain experts to address data quality issues in the generated knowledge base by validating facts interactively via the COLT-UI, thereby improving the data quality of the underlying knowledge base. While the Curation Interface enables data engineers to monitor and control the individual components of the knowledge base construction process, the Entity Landscape Explorer, also called ELEX, is aimed at the end-user and simplifies knowledge base exploration. Chapter 5 focuses on the overall system and examines the interplay of the individual components in more detail.

As each part of the system presents its own challenges, we begin by outlining the challenges that arise from the integration of structured data sources, continue with the challenges posed by the integration of unstructured data sources and conclude this section with an overview of the challenges encountered when improving the data quality of the generated knowledge base.

1.2.1 Structured data

As introduced by Christen [2012] and Naumann et al. [2006], the integration process of structured data sources, such as relational database tables, CSV or TSV files typically consists of the three processing steps; *schema alignment*, *data matching* and *data fusion* as shown in Figure 1.3. Each processing step addresses a specific challenge that arises during the integration process. In the following, we discuss both the challenges associated with the integration of structured data as well as the components intended to meet these challenges.

(I) – Schema alignment. When integrating structured data sources, each data source adheres to a specific schema that defines the structure of the stored entities. A schema consists of several attributes which, in addition to a particular data type (i.e., *string*, *float*, *integer*), also express a semantic concept, such as a street name or a product description.

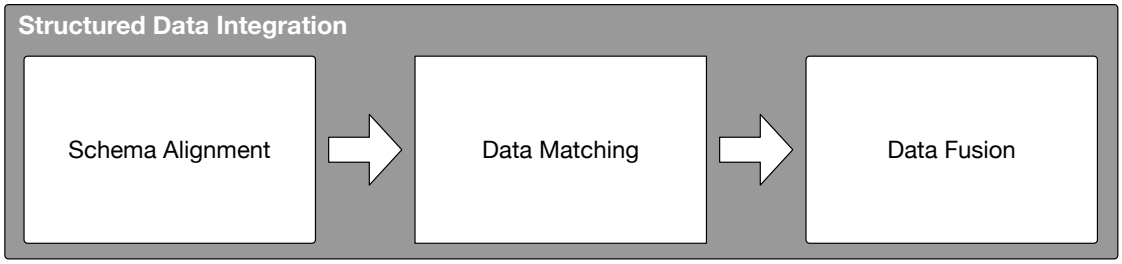

Figure 1.3: Overview of the integration process for structured data sources

For example, an attribute *prod_name* can be of data type *string* and describe a product name, while the attribute *ex_rate* is of type *float* and represents an exchange rate. The main challenge is that the schemata of different data sources are by no means identical, even if they describe the same entity. This heterogeneity poses a major obstacle to the integration of different data sources. Thus, attribute names that describe identical semantic concepts can differ greatly from one data source to another. While in data source A the exchange rate attribute is named *ex_rate*; data source B refers to the same concept by the name *exchange_rate*.

The goal of the *schema alignment* component is to map the individual data source schemata to a unified *global schema*, so that attribute names describing the same semantic concept are assigned to a unique attribute name in the global schema. In this way, the use of the global schema ensures that the individual data sources can be handled uniformly, which forms the basis for all subsequent steps. Over the years, many different schema alignment techniques have been developed; a comprehensive overview is given by [Bellahsene et al., 2011]. As *schema alignment* is not in the main focus of this work, it is assumed that a corresponding schema alignment already exists.

(II) – Data matching. After the previous *schema alignment* component has created a *global schema*, the next step is to cope with the fact that the individual data sources are likely to contain redundant information in the form of duplicate entries. In this context, the term *duplicate* refers to data records that describe the same real-world entity. The intention of the *data matching* step is to identify these duplicates so that they can be consolidated and thus eliminated by the subsequent *data fusion* step. As illustrated in Figure 1.2, the *data matching* component consists of the subcomponents *data preprocessing*, *indexing*, *duplicate detection,* and *evaluation*, which in turn address different challenges that arise during the data matching process. Because the *data matching* component is covered in Chapter 2, we only describe its main functionality and postpone a detailed discussion until later.

In addition to the challenges pointed out by Christen [2012], such as the lack of global identifiers, demanding runtime complexity, and data privacy issues, a major challenge of the data matching component is to determine the similarity of two entities. Although a wide variety of similarity measures have been proposed over the years, many of them have been designed with specific use cases in mind and are therefore not equally suitable for all kinds of domains. Overcoming the associated limitations requires either the manual

design of a customized similarity measure or learning it by means of machine learning techniques. Deciding on the machine learning option presents additional challenges, one of them being the need for large amounts of training data.

(III) – Data fusion. The final step in the integration process of structured data sources is called *data fusion*. In this step, the duplicates identified by the preceding *data matching* step are merged into a single, unified representation. Despite the previously established *global schema*, it still occurs that different data sources provide conflicting values for certain attributes. For example, a person may have a different family name in data source A than in data source B. The objective of the *data fusion* component is to resolve such conflicting values as correctly as possible when consolidating duplicate entries. As the topic of *data fusion* is outside the scope of this thesis, we refer to the work of Bleiholder and Naumann [2008] for a detailed discussion.

1.2.2 Unstructured data

To obtain a holistic picture of specific scenarios, it is not enough to merely consider the information coming from structured data sources. Much valuable information is available only in the form of unstructured data sources. Unstructured data sources are data sources whose information cannot be accessed by conventional means, such as relational database systems or simple software libraries. In addition to image, video, and audio files, this also includes, written texts. To access the information expressed in textual form, a number of techniques are required to extract the information contained in the texts and provide access to it. This task is performed by the *text mining* component shown in Figure 1.2. It is the subject of Section 3 and consists of the sub-components *named entity recognition* (NER), *named entity linking* (NEL), and *relation extraction* (RELEX), which will be discussed in more detail later.

In general, the challenges in this area are to cope with the unstructured nature of the processed documents. As such, the written word is often ambiguous and leaves room for interpretation of the expressed facts. Real-world entities frequently appear in texts under different names, where it is often unclear to which real-world entity a textual mention refers. Furthermore, relationships in texts can be expressed in many ways, which proves to be particularly challenging in cases where the underlying language offers much expressive freedom, as is the case, for example, with German texts. The challenges outlined above represent only a small selection of challenges associated with the extraction of information from written documents.

1.2.3 Knowledge assessment and curation

To cope with the ever-growing amounts of data and their complexity, machine learning methods are increasingly being used for the construction of knowledge bases. Since the applied methods are by no means infallible, errors can propagate and amplify across many processing steps, ultimately leading to incorrect entries in the generated knowledge base. Such data errors can be corrected by applying logical rules, which can be generated

by rule learning systems, such as AMIE [Galárraga et al., 2015, 2013] or RuDiK [Ortona et al., 2018]. When applied to the knowledge base, such rules add new facts or remove incorrect ones. As these rules are derived from the facts contained in a knowledge base, their generation is negatively impacted by noisy or missing facts. Although the generating systems provide statistical quality estimates for the generated rules, their calculation ultimately depends on the data quality of the knowledge base. Thus, low data quality of the underlying knowledge base inevitably leads to considerable inaccuracies in the quality estimation of the generated rules. The challenge is to assess the quality of the generated rules without relying exclusively on the information provided by the knowledge base. To address this problem, access to external information is needed, which enables the verification of knowledge base facts. To this end, we consult experts on the topics covered by the knowledge base and aim to radically reduce the manual effort by efficiently using existing domain knowledge.

1.3 Structure and contributions

The remainder of this thesis is organized into three main chapters that cover the core components shown in Figure 1.2 and one chapter that focuses on the overall system describing the interplay of the individual components. In addition to a short introduction to the most important concepts of the respective topic area, they contain the contributions of this thesis. This work contributes to the current state-of-the-art in the fields of *duplicate detection*, *named entity recognition* and *knowledge validation*, aiming at advancing the field towards a fully automated process for building domain-specific knowledge bases from structured and unstructured data sources. Complementary to the core contributions, this work is also intended to provide a holistic picture of the knowledge base construction process, which is why an overview of the current state-of-the-art of adjacent processing steps is given whenever necessary. As highlighted in Figure 1.2, the contributions of this work are distributed across different areas of knowledge base construction, resulting in the following structure of this thesis:

Chapter 2 is devoted to the integration process of structured datasets. It first introduces the individual process steps, while its main focus lies on a novel deduplication method. The proposed deduplication method is based on Siamese neural networks, which are used to learn a dataset-specific similarity measure that is then used for the identification of duplicate entries. By eliminating the need for manual feature engineering, the effort for model creation can be significantly reduced. Moreover, the network design is conceived in a way that enables a knowledge transfer between deduplication networks. Our experiments show that the knowledge accumulated in a trained network can be transferred to an untrained network by copying the weight matrices of selected attributes, which led to an overall improvement in F-measure of +4.7 and +4.6 percent. The chapter is based on a publication by Loster et al. [2020a], which is joint work with Ioannis Koumarelas, who helped with data preprocessing and the generation of non-duplicate pairs.

Chapter 3 addresses the extraction and integration of information from unstructured data sources. Concretely, a method for recognizing named entities (here company names)

is presented, which allows the integration of dictionaries into the training process of a conditional random field classifier. The objective of this procedure is, on the one hand, to reduce the manual effort required to generate training data and, on the other hand, to cover as many name variations as possible, which helps the classifier to generalize better. By carefully incorporating domain knowledge in the form of dictionaries into the training process, we were able to increase the recall and F-measure by an average of 6.57 and 3.85 percentage points compared to our baseline, proving that performance improvements can be achieved by integrating additional knowledge. This chapter is based on the publication of Loster et al. [2017], in which Oliver Maspfuhl and Dirk Thomas provided their professional expertise in advising on the requirements of the financial sector.

Chapter 4 is dedicated to the assessment of the information quality of generated knowledge bases. It introduces the Colt framework - a rule-based knowledge validation framework that enables the interactive quality assessment of logical rules. Unlike existing methods such as AMIE or RuDik, Colt is based on active learning principles and uses both Gaussian processes and neural networks to interactively learn labeling functions on the basis of rules. The presented approach benefits from knowledge graph embeddings and takes into account user feedback to address data quality issues of the underlying knowledge base. With its most expressive implementation Colt-GP, it achieves a 10% error in the confidence estimate of facts, for which it requires only 20 user-validated facts. In addition, rules can be validated with a prediction quality of 75%, requiring as little as 5% of the rule instances to be annotated. The work by Loster et al. [2020b], which was produced in collaboration with Davide Mottin, Paolo Papotti, Jan Ehmueller, and Benjamin Feldmann, constitutes the basis of this chapter. Davide Mottin contributed to the design and formalization of the approach, while Paolo Paotti assisted in creating the rules and the use of RuDik. Jan Ehmueller and Benjamin Feldmann contributed to the implementation of the system, in particular to data preprocessing, the design of a user interface for instance annotation, and the subsequent instance annotation itself.

Chapter 5 introduces the prototypical CurEx system shown in Figure 1.2. It is based on the publication of Loster et al. [2018b] and enables the construction of domain-specific knowledge bases from structured and unstructured data sources. In addition to providing different user interfaces, CurEx enables the selective generation of multiple knowledge graphs and, thanks to its modular architecture, supports the step-by-step improvement of individual system components. During the development of the system, the experiences and insights gained in the previous chapters were taken into account. The system covers the areas of data integration, -curation, and -exploration. Within the scope of this work, Jan Ehmueller and Benjamin Feldmann assisted in the implementation of various system components.

Chapter 6 concludes this thesis by summarizing its contributions. It provides concrete suggestions for the extension of the presented approaches as well as an outlook on promising directions for future research and general issues that need to be addressed.

Chapter 2

Integrating Structured Information

A large number of applications rely on data stored in multiple data sources, making their integration a topic of central importance. Especially when building knowledge bases, the integration of different structured data sources can be regarded as a first necessary step. As discussed in Chapter 1, the associated integration process can be divided into the steps *schema alignment*, *data matching,* and *data fusion*. While these three components have already been outlined in the previous chapter, this chapter focuses on the *data matching* process and particularly on the *duplicate detection* step. In general, the term *data matching* describes the process of discovering digital objects that refer to the same real-world entity, also known as *duplicates*. Research typically distinguishes two types of duplicate detection: *duplicate detection*, also called *deduplication*, which refers to the task of identifying duplicates within a *single* data source, and *record linkage*, which aims to identify matching records across *multiple* data sources.

The *data matching* process introduced by Christen [2012] can further be subdivided into a series of discrete steps that address specific subproblems arising during the identification of duplicate entities. While Christen defines a process consisting of five steps, we simplify that pipeline by combining the *record pair comparison* and *classification* steps of the original pipeline definition into a single *duplicate detection* step. This reduces the *data matching* process to the four steps shown in Figure 2.1. To put this chapter's contribution into context, we briefly introduce the individual steps of the *data matching* process:

(I) – Data preprocessing. The entities that pass through the *data matching* process are composed of attributes and their values that describe certain aspects of the respective entities. A common problem encountered during the consolidation of entities from different data sources is that they can vary considerably in terms of data format and content. These differences are often caused by different formatting rules of the individual data sources, e.g., different street name formats, or by data entry and processing errors, such as spelling mistakes or poor quality of optical character recognition systems. To ensure

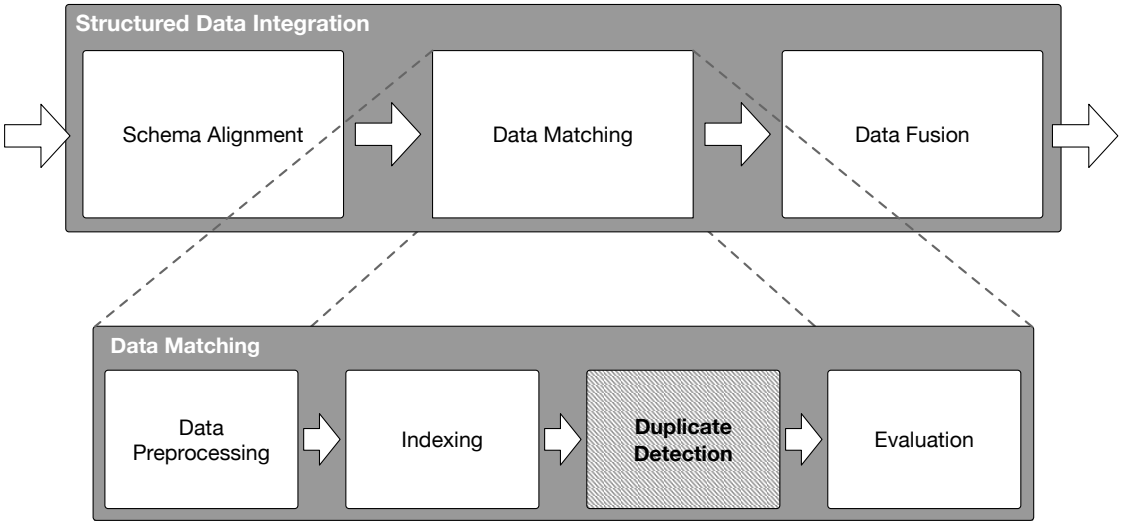

Figure 2.1: Overview of the *data matching* process

high-quality data matching, it is beneficial to first eliminate these data-related variations. The *data preprocessing* step, also known as *data preparation*, addresses this problem by first cleansing the entity attributes and then converting them into a common format that takes into account the characteristics of the involved data sources. These measures ensure that data related differences do not have to be considered in later processing steps. Typically, this phase involves removing characters and words, correcting spelling errors, replacing abbreviations, regrouping attributes, or validating data against external data sources. On top of that, it is possible to derive new attributes from existing ones, which can be considered as a form of feature engineering for downstream operations.

(II) – Indexing. To detect duplicates, it is theoretically necessary to compare each entity with every other entity. As an exhaustive pairwise comparison results in a quadratic runtime complexity in the number of records, this approach may be acceptable for small datasets but does not represent a viable solution for large datasets. Based on the observation that most comparisons are conducted between entities that are unlikely to be duplicates, the total number of comparisons can be drastically reduced. The core idea is to perform entity comparisons only if two entities already exhibit a certain degree of similarity. Implementing this idea is the objective of the *indexing* component. In the case of duplicate detection, the *indexing* component receives as input all records of a single relation, while in the case of record linkage, the records of all considered data sources are combined and passed in as one relation, in which a new attribute specifies from which data source each record originates. While many different indexing techniques have been developed over time, one of the most widely used is the "blocking" technique [Christen, 2012], which we also employ in our proposed approach. Based on certain attribute combinations, this technique divides the entities into individual groups, allowing for an exhaustive comparison of all entities within each group. Thus, the *indexing* component reduces the number of necessary comparisons to the extent that duplicate detection becomes feasible for even large datasets.

(III) – Duplicate detection. The *duplicate detection* step represents the central topic of this chapter. It is responsible for identifying entities that refer to the same unambiguous representation of a real-world object within one (*duplicate detection*) or across multiple (*record linkage*) data sources. As duplicate detection represents a fundamental task in data quality and data cleansing, a variety of deduplication techniques have been proposed over the years [Elmagarmid et al., 2007]. These techniques range from the application of traditional similarity measures over simple machine learning methods to the training of complex neural networks. We contribute to the research on *duplicate detection* by proposing a Siamese neural network (SNN), capable of learning a similarity measure tailored to the characteristics of a particular dataset. Using this learned similarity measure, we are able to classify entities with high accuracy as "duplicates" or "non-duplicates".

(IV) – Evaluation. Finally, the evaluation step is responsible for assessing the performance of the entire *data matching* process. This involves checking how many duplicates are actually duplicates and how many non-duplicates were incorrectly identified as duplicates. To this end, we rely on the traditional performance metrics: precision, recall, and F_1-measure.

Traditionally, *duplicate detection* involves using similarity measures to identify and merge multiple representations of the same entity – duplicates – into an extensive homogeneous data collection. Often, traditional similarity measures do not cope well with the heterogeneity of the underlying datasets. The extent to which similarity measures differ can be shown by comparing the phrases "loans and accounts" and "accounts and loans" using different similarity measures. While the Jaro-Winkler measure [Winkler and Thibaudeau, 1987], which is known to perform well on names, yields a value of 0 (no similarity), a token-based measure, such as Jaccard [Jaccard, 1901], results in a value of 1 (absolute similarity) for the same example. This illustrates that choosing a similarity measure for a domain requires a high level of expertise, and depending on the domain, it may even be necessary to manually design a custom similarity measure, which is both time consuming and requires extensive domain knowledge.

To mitigate these issues, we contribute to Step (III) of the *data matching* process, by introducing a deep Siamese neural network (SNN) capable of learning a similarity measure tailored to the idiosyncrasies of a particular dataset. Using the properties of deep learning, we are able to eliminate the manual feature engineering process and thus considerably reduce the effort required for model construction. In addition, we show that it is possible to transfer knowledge acquired during the deduplication of one dataset to another, and thus significantly reduce the amount of data required to train a similarity measure. We evaluated our method on multiple datasets and compare our approach to state-of-the-art deduplication methods. Our approach outperforms competitors by up to +26 percent F-measure, depending on task and dataset. In addition, we show that knowledge transfer is not only feasible but in our experiments led to an improvement in F-measure of up to +4.7 percent. For the remaining steps of the *data matching* pipeline, we proceed as described in Sections 2.3 and 2.5.

In particular, we make the following contributions:

- We present an SNN architecture for duplicate detection that facilitates knowledge transfer. To achieve both competitive deduplication performance and knowledge transfer, it combines character embeddings as studied by DeepMatcher [Mudgal et al., 2018] with the architectural design of DeepER [Ebraheem et al., 2018].
- We show how to transfer previously learned knowledge to support the deduplication of other datasets.
- We analyze the impact of transfer learning strategies on the performance of the neural network.
- We compare our system with the state-of-the-art of related work.

This chapter is based on the work of Loster et al. [2020a] and is organized as follows: Section 2.1 gives a brief introduction to duplicate detection and introduces the problem we are addressing. Section 2.2 discusses related work. Section 2.3 introduces SNNs and presents the overall structure of our neural network, including its unique characteristics. We introduce the concept of knowledge transfer and how it relates to our experiments in Section 2.4. Section 2.5 is dedicated to introducing the datasets and gold standards for our experiments. In Section 2.6, we present our experimental results, and Section 2.7 concludes this chapter.

2.1 On the detection of duplicates

The need to integrate multiple data sources into a single dataset is present in many application areas. A major challenge that arises during the integration process emerges from the fact that records from different data sources often contain duplicate entries, i.e., several entries that refer to the same real-world entity. These duplicates, implying poor data quality, can directly affect downstream operations, e.g., causing low customer satisfaction in customer-relationship-management, incorrect stock-keeping, or overfitting of machine learning methods.

Resolving this issue requires the detection of such duplicates, which is well studied, also under the terms *entity resolution*, *record linkage,* and several others [Christen, 2012; Elmagarmid et al., 2007; Naumann and Herschel, 2010]. As *duplicate detection* and *record linkage* are conceptually very similar, we conduct experiments for both tasks. Unless explicitly stated otherwise, we refer to both tasks, for ease of exposition, under the term *duplicate detection* for the remainder of this chapter. As such, *duplicate detection* is commonly divided into two subtasks: **(i)** the identification of duplicate candidates, and **(ii)** the classification of these candidates as duplicates and non-duplicates. The problem of subsequently eliminating these duplicates, known as *data fusion* [Bleiholder and Naumann, 2008], is not addressed in this thesis. Since most deduplication approaches depend heavily on the performance of the similarity measure used to assess the similarity

between processed entities, its selection or creation plays a crucial role and must be performed with great care.

So far, a wide variety of similarity measures have been proposed [Christen, 2012], each with its strengths and weaknesses. As each of these proposed similarity measures has its own individual approach to quantifying the similarity of two entity representations, they are often very domain-specific and hence not a good choice for determining similarities in unintended domains. This difficulty becomes evident upon a closer look at the circumstances in which these measures are used. For example, a similarity measure that is designed to compare first names is clearly unsuitable for comparing timestamps, but also other string-types, such as street-addresses, should be compared differently. Moreover, similarity measures are use-case driven. For instance, it strongly depends on the particular use case, whether it makes more sense to compare two timestamps regarding their syntactic or chronological similarity. Even within the values of a particular domain, the chosen similarity measure may not be equally suitable for all encountered values. As such, an address column might contain both street and postbox addresses for which a case-based similarity measure might work best. Finally, it is often quite difficult even for experts to find a good weighting of the various attribute similarities to determine an overall similarity score of two input-records. It is, for example, not always clear whether first names are generally more important than city names when comparing customers. Given these observations, it becomes evident that it is difficult to perform deduplication by solely relying on a single similarity measure. Thus, several similarity measures, such as Levenshtein [Levenshtein, 1966] or Jaro-Winkler [Winkler and Thibaudeau, 1987], are often combined to capture different aspects of the characteristics underlying the processed entities. While this combination offers more flexibility, it is often difficult to sufficiently adapt traditional similarity measures to the entity characteristics of the processed data sources.

To address this issue, there are essentially two possible solutions: either a customized similarity measure is designed manually, or it can be learned by applying machine learning techniques. The main disadvantage of manually designing a similarity measure is that it requires extensive domain expertise, making it difficult and time-consuming. The alternative approach is to leverage machine learning techniques to learn a new similarity measure that is tailored specifically to the entities of a specific domain.

As with most traditional machine learning methods for duplicate detection, such as SVMs [Bilenko and Mooney, 2003; Christen, 2007] or decision trees [Elfeky et al., 2002; Tejada et al., 2002], their main bottleneck lies in their dependence on manual feature engineering. Due to the costs associated with extensive manual feature engineering, it is, in practice, often limited to the creation of feature vectors consisting of various traditional similarity measures, which are ultimately used to train a selected classifier. Although good results can be achieved by optimally combining several similarity measures, this approach is still bound by the limitations of traditional similarity measures developed for specific domains or data types. Overall, the process of manual feature engineering, as well as the manual creation of similarity measures, both involve a considerable amount of time and effort, which makes them undesirable. In addition, it should be noted that most machine learning techniques require large amounts of training data to train

a classifier of sufficient quality. It can, therefore, be stated that both approaches have their disadvantages with regard to manual labor costs. While the bulk of the time for manually creating custom similarity measures is spent on iterating different design ideas, the majority of time for learning a custom similarity measure is spent on annotating training data and designing a set of useful features.

To alleviate both issues, we propose SNNDedupe, a deep neural network that is based on the architecture of a Siamese neural network (SNN) [Bromley et al., 1993] and designed to facilitate transfer learning. Using this architecture, we learn a similarity measure for a given dataset by letting the neural network automatically adapt to the idiosyncrasies of the respective dataset and capture its characteristics. Thus, we overcome the limitations of traditional similarity measures, which often lack sufficient adaptability when applied to unintended data. At its core, the proposed SNN architecture consists of two identical subnetworks that are connected via an energy function. We selected this architecture as the basis for our approach, as it has been successfully applied to learn similarity functions in various areas [Koch et al., 2015; Mueller and Thyagarajan, 2016]. By using deep learning techniques, we are able to automatically discover promising features, thus eliminating both the costly manual feature engineering and the required domain knowledge needed for this task. In our experiments, we compare SNNDedupe with the SVM-based approach of Christen [Christen, 2008] and the DeepMatcher [Mudgal et al., 2018] system. We show that we are able to learn a competitive similarity measure that enables a state-of-the-art duplicate classification.

As annotating vast amounts of training data is in practice often infeasible, we evaluate the behavior of SNNDedupe under different amounts of training data. To reduce the amount of training data and, thus, the associated effort for its generation, we focus on the design and implementation of a knowledge transfer [Pan and Yang, 2010] between deduplication networks. In doing so, we address what is often perceived as the Achilles' heel of many machine learning approaches, namely, their dependence on large amounts of training data. This objective is reflected primarily in the design of the network, as it is tailored to support knowledge transfer at the attribute level. In a nutshell, the core idea that makes this knowledge transfer possible is to process the entities at their attribute level. This decision provides the ability to learn and transfer the properties of each attribute's semantic domain (e.g., band names, book titles). By combining character embeddings and BLSTM layers, we create a dedicated embedding for each attribute. Thus, representation learning occurs at the attribute level, resulting in embeddings for each attribute domain, which can informally be interpreted as knowledge about the processed attribute domain. This compressed domain knowledge accumulates in the weight matrices of the network layers and can then be transferred by initializing the attribute weights of a deduplication network operating on a different dataset. This transfer can be carried out whenever the source and target domains of the corresponding attribute are either the same or sufficiently similar.

Our experiments show not only how a knowledge transfer affects the training of our network, but in particular, that it is possible to increase classification performance or to reduce the amount of necessary new training data by transferring parts of the already acquired knowledge. In addition, we investigate how much improvement can be achieved

over learning without prior knowledge and to what extent the training data can be reduced while maintaining a good result.

2.2 Related work

Related work can be classified according to the degree of supervision and whether transfer learning represents a key element of the respective approach. In the following, we, therefore, discuss the related work that does not involve transfer learning according to its supervision approach, which is divided into the categories: *supervised*, *semi-supervised*, and *unsupervised* approaches. Thereafter, we separately discuss work that supports the concept of *transfer learning*.

Supervised approaches. Apart from traditional similarity measures, efforts have been made to learn similarity measures using a range of supervised machine learning techniques. Systems based on those techniques require tagged training data from which they learn to perform a specific, well-defined task. For duplicate detection, this task is to classify a particular entity pair as either duplicate or non-duplicate.

Both Christen [2007] as well as Bilenko and Mooney [2003] address this problem by using support vector machines (SVM). Bilenko and Mooney propose two similarity measures that are learned using two different machine learning techniques [Bilenko and Mooney, 2003]. The first uses the Expectation-Maximization (EM) algorithm to learn a variant of the edit distance measure with affine gaps; the second is based on the vector space model and trained using an SVM. They were able to show that the learned similarity measures can be adapted to the underlying dataset, resulting in better system performance.

The approach developed by Christen [2008] also uses SVMs to train a classifier for detecting duplicates. Their feature engineering step consists of choosing and calculating different traditional similarity measures between the attribute values of each record pair. The resulting values are then combined into a feature vector, which is used to train the classifier. In addition to SVM-based systems, other studies have investigated the use of decision trees [Elfeky et al., 2002; Tejada et al., 2002].

More recently, DeepER [Ebraheem et al., 2018] and DeepMatcher [Mudgal et al., 2018] have been introduced. Both systems use deep neural networks specifically developed for the entity matching task. The core idea of DeepER is to generate a vector representation for each of the compared tuples, projecting them from a symbolic into a high-dimensional embedding space. To this end, DeepER first uses word embeddings to translate the individual attribute values of the processed tuple into their corresponding vector representation. Subsequently, these attribute vectors are concatenated and form the final vector-based tuple representation. Using this representation, the network is trained so that similar tuples are drawn to each other, while unequal tuples repel each other. An interesting architectural characteristic of DeepER's network is that vector representations are established for both the entire tuple and each of its attributes. Generating the tuple representation has the advantage that the generated tuple embeddings can be used by downstream applications, such as clustering. Unfortunately, a major

disadvantage of this approach is the use of word embeddings as the smallest unit for constructing the attribute and, in turn, tuple embeddings. This design decision makes the system susceptible to out-of-vocabulary (OOV) words, that is, words that are missing in the used word embeddings. By operating at the word level, it becomes significantly more difficult to adequately handle subword structures, such as misspellings or common substrings.

In addition to conducting a design space exploration, DeepMatcher [Mudgal et al., 2018] extends the architecture of DeepER and adds various attribute summarization techniques. In contrast to DeepER, DeepMatcher operates by calculating similarity measures between the attribute pairs of the compared tuples. The resulting similarities are then aggregated into one similarity vector, which constitutes the basis of their classification. In their work, Mudgal et al. show that character embeddings can lead to significant performance improvements, especially when dealing with uncommon words or dirty data.

Despite its similar architecture, our proposed network differs in some aspects from the existing work. While our basic idea of calculating an embedded representation for the processed tuples is similar to that of DeepER, our approach differs in the processing of the individual attribute values. While DeepER uses *word embeddings* to compute each attribute's vector representation, our architecture operates by splitting each attribute value into its characters, which are then mapped to a vector representation by using *character embeddings*. As discussed by Mudgal et al. [2018], the choice of word versus character-level embeddings is critical, due to the implications of out-of-vocabulary (OOV) tokens. The use of word-level embeddings reduces the mapping of a token to its vector representation to a lookup in an embedding matrix. If this lookup fails because the processed token is unknown, a predefined "unknown vector" is usually used instead of a pre-trained token embedding. Because values occurring in tables can often not be matched to pre-trained word embeddings, such as Word2Vec [Mikolov et al., 2013] or GloVe [Pennington et al., 2014], the use of word-level embeddings is not optimal for the deduplication task. To mitigate this issue, retrofitting techniques, such as those employed by DeepER, can be used to generate approximations of OOV words. This can be implemented, for example, by calculating an average embedding of co-occurring word embeddings. However, the calculation of such approximations causes the meaning of multiple word embeddings to be mixed so that the meaning of the generated interpolation of an OOV word embedding is, at best, a diffuse representation of the missing word. By using the suggested character embeddings, the need for retrofitting techniques is eliminated, as any word can be created as a combination of individually learned character embeddings.

Another effort to reduce the OOV problem is made by the *fastText* approach introduced in Bojanowski et al. [2017]. It complements any word embedding by additionally including all n-gram embeddings that can be generated from the processed word. When compared to the lookup procedure of word embeddings, this approach drastically reduces out-of-vocabulary (OOV) words as it is able to represent each OOV word as the combination of its n-gram embeddings. However, as stated by the authors, *fastText* usually utilizes a range of 3- to 5-grams to create word vector representations. Depending on

the granularity of the used n-grams, some of them may not be known during the generation of the individual word vector representations, so that for each unknown n-gram, an "unknown vector" is used instead.

With our model, we take an even more generic approach by training an embedding for each character in the ASCII table, further reducing the granularity from n-grams to character embeddings. By replacing n-grams with character embeddings, we further mitigate the OOV problem, as we are able to construct any word embedding from its character embeddings. In addition to reducing the number of lookup errors, the size of the embedding matrix is also significantly reduced. Drawing on the findings of Mudgal et al. [2018], the use of character embeddings allows us to overcome DeepER's limitations by being able to handle character-level issues, such as misspellings and common substrings. Looking, for example, at the duplicate pair ("`hibrid theory`", "`hybrid theory`"), which differs only in a single character, a lookup of the word "`hibrid`" will likely return the "unknown vector", as it has never been encountered before.

We aim to learn an embedding for each attribute by combining the character embeddings into a vector representation of the underlying entity (much like the original DeepER). The compressed entity representation is then altered during the training process to the effect that similar entities are located closer together than dissimilar ones. Unlike the other systems, we are not trying to classify record pairs directly into duplicates and non-duplicates but concentrate on learning an entity representation that allows us to classify based on simple distance measures, such as the Euclidean distance. As a consequence, this method could also be used as a distance measure in other applications, such as clustering.

With Termite, the authors of [Fernandez and Madden, 2019] propose a system that allows the execution of queries across multiple structured and unstructured data sources. In a nutshell, the system transforms structured and unstructured data into a common embedding space where it can be used to retrieve related data points regardless of their origin and schema. Because the authors are mainly interested in identifying *related* entities across heterogeneous data sources, they address a problem different from ours. Although they also use an SNN architecture, they focus on learning a concept of relatedness between entities, which can be interpreted as a more relaxed case of deduplication, where it is sufficient for the entities to be similar, but do not necessarily correspond to the same real-world entities. Another difference to our work is that the authors consider both structured and unstructured data sources and do not explore the concept of transfer learning.

Semi-supervised approaches. Algorithms that follow active learning principles form a subset of semi-supervised learning methods. The core of this principle is to enable the algorithm to query an external information source to resolve particularly challenging or hard-to-decide training cases. Sarawagi and Bhamidipaty [2002] designed a duplicate detection system that follows this principle. The system works by having difficult-to-classify duplicate pairs evaluated by a user. The information gained is then fed back into the training process, creating a new and enhanced model. The iteration of this procedure allows the repeated improvement of the model until it can finally identify all entities as

either duplicates or non-duplicates. A similar system was created by Tejada et al. [2002], which relies on decision trees to detect duplicate entity pairs.

Over the years, many crowd-based systems have been proposed, which also fall into the category of semi-supervised systems [Abboura et al., 2015; Demartini et al., 2012; Firmani et al., 2016; Verroios and Garcia-Molina, 2015]. One of these systems, CrowdER [Wang et al., 2012], pursues a hybrid approach to entity resolution, seeking to narrow the gap between purely machine-based and purely human approaches. This is achieved through a two-step heuristic approach using a machine-based approach to reduce the search space and then present possible matches to a human audience for evaluation. Another recently introduced system is CloudMatcher [Govind et al., 2018], which was introduced as a fast, easy-to-use, scalable, and highly available service for entity matching. These active learning approaches represent a fundamentally different approach compared to our proposed method. While our approach assumes that an annotated dataset already exists, active learning approaches are designed to create an annotated dataset from scratch, creating annotations by consulting external information sources, such as domain experts. Although these approaches do not require a fully annotated dataset, one drawback is that a sampling bias [Settles, 2012] can occur where only elements from a specific subset of the dataset are presented for annotation. If not prevented, this sampling bias leads to biased datasets and, in turn, to biased machine learning models, which can be harmful when applied in production.

Unsupervised approaches. In contrast to supervised methods, unsupervised techniques simplify duplicate detection by not requiring labeled training data to achieve the desired classification. In this case, clustering algorithms, such as k-means or farthest-first clustering, were used to group duplicate and non-duplicate entity pairs into clusters [Elfeky et al., 2002; Goiser and Christen, 2006; Gu and Baxter, 2006]. To this end, the required parameter k, which determines the number of clusters to be created, is often set to a value of two or three. A value of two clusters (k=2) corresponds to the categories "duplicate" and "non-duplicate", while a value of three (k=3) considers an additional category with the name "possible duplicates". Although cluster-based approaches can often be less accurate than their supervised counterparts [Goiser and Christen, 2006], they are still useful for real-world applications, as sufficient training data is usually not available and has to be created with great effort. In a more recent approach, the authors of [Koumarelas et al., 2020] implement a two-phase process that first uses annotated datasets to train a machine learning model on matching dependency rules capable of identifying duplicates. In a second step, the trained model is applied to a new dataset detecting matching dependencies that, when applied, are able to identify duplicates in the absence of annotations.

Transfer learning. In addition to the methods discussed so far, there are approaches in which transfer learning plays an essential role. In their approach, the authors of [Negahban et al., 2012] focus on reducing labeling costs through adaptive sharing of learned structures between scoring problems, involving more than two data sources. To this end, they define the problem of *multi-source similarity learning*, which explicitly focuses on entity resolution with more than two data sources. Furthermore, their approach is based

on similarity vectors composed of traditional similarity metrics that are used to train an entity resolution that is confined to linear classification models. In contrast, we concentrate on one (*deduplication*) or two (*record linkage*) data sources, learn a similarity measure specifically tailored to the underlying dataset, and use a non-linear classification methodology by means of a neural network.

The deep learning-based approach of Kasai et al. [2019] focuses on low resource consumption and, to this end, combines transfer learning with active learning components. Like earlier approaches, the authors turned to *fastText* embeddings, which contain subword information, to translate the processed tuples into a common embedding space, thus reducing the OOV problem. While this approach produces fewer OOV cases, our approach employs character embeddings, which are even more fine-grained, and almost completely avoid the OOV problem. Furthermore, the authors base their classification on similarity vectors and use a negative log-likelihood loss in conjunction with a softmax function. In contrast, our architecture is based on SNNs and employs a contrastive loss function between the vector representations of the processed entities. For knowledge transfer, they train all network parameters on the source dataset and reuse them to classify the entities of the target dataset. We follow the same training approach but transfer only the learned attribute representations. We then fine-tune the upper network layers on the target dataset to learn the composition of the new entities.

Instead of an SNN architecture, the authors of [Zhao and He, 2019] suggest the use of a *hierarchical neural network* that combines character and word-level representations as well as attention mechanisms. While they use the same intuition, their approach differs in some aspects from ours. In particular, they extensively use type information from a knowledge base to pre-train the attribute-level EM and type matching models in an offline process. Thus, the use of a knowledge base becomes an essential part of their approach, which is not a prerequisite in ours. For training, they employ a log loss function that, in contrast to our contrastive loss, does not account for class imbalances in the training data.

2.3 Proposed approach

This section presents the *architecture*, *loss function*, and *training details* of our approach, which is based on two key concepts: the application of Siamese neural network (SNN) to recognize similarities between various entities and the trend of automatic feature learning, as pioneered by the deep learning community. Due to its design, this type of network is particularly suitable for learning distance measures between two entities, which is why we decided to exploit its features for duplicate detection.

We start by giving a short introduction to similarity measures in Section 2.3.1 and continue by introducing the general intuition and overall architecture of Siamese neural networks in Section 2.3.2. Section 2.3.3, gives a detailed description of the subnet architecture and discusses the loss function and its mode of operation in Section 2.3.4. We conclude this section by presenting the training procedure and its parameters in Section 2.3.5.

2.3.1 Similarity measures

Since similarity measures are an integral part of the deduplication process, we briefly introduce the most important concepts. Traditionally, similarity measures are chosen w.r.t. the underlying attribute's type [Cohen et al., 2003]. For instance, if the field is of type Date, then the measure should consider similarity in years to be more important than just in months or days. In our experiments, we treat all attributes as alphanumerics, as it is the most general type. Thus, we consider only string similarity measures. These can be subdivided into (a) *edit-based*, (b) *token-based*, (c) *hybrid*, and (d) *phonetic measures*. Measures of type (a) are based on edit-operations, such as Levenshtein [Levenshtein, 1966] or Jaro-Winkler [Winkler and Thibaudeau, 1987], whereas measures of type (b) tokenize the strings and compare the sets of tokens – Dice [1945] and Jaccard [1901] being two examples. Type (c) measures are typically a combination of (a) and (b), with MongeElkan [Monge and Elkan, 1996] serving as a good example. Lastly, Type (d) measures, such as Soundex [Odell and Russell, 1918], are ideal for words that sound similar as they transform syllables into characters representing that sound.

2.3.2 Siamese neural network

The SNN architecture was first introduced by Bromley et al. [1993] to verify signatures on credit-cards. Since then, it has been used in many different areas, such as one-shot learning [Koch et al., 2015] as well as recognizing textual [Neculoiu et al., 2016] and facial similarities [Chopra et al., 2005]. As Siamese networks have already been shown to work well in areas where similarities between different entities need to be evaluated [Melekhov et al., 2016; Neculoiu et al., 2016], we chose this type of network to address the task of duplicate detection. Due to the twin architecture of these networks, they are particularly well suited to identify differences and similarities between the considered entities.

We intend to learn a similarity measure directly from the raw data, thus skipping the traditional feature engineering step required in other machine learning approaches. For this purpose, the network is trained by exposing it to both positive and negative example pairs. In our case, a training example consists of two tuples t_1 and t_2, and a corresponding binary label Y, which indicates whether the tuples refer to the same ($Y = 1$) or different real-world entities ($Y = 0$). In turn, a tuple consists of several attribute values $a_1, ..., a_n$, which are individually preprocessed and passed to the subnets. The architectural details of the subnets are explained in more detail in Section 2.3.3.

In training the network, we aim to learn a function capable of mapping the input values into a low-dimensional embedding space, where a simple distance measure, such as the Euclidean distance, can be used to estimate the semantic similarity between two tuples. The learned function should be able to position similar tuples close to each other within the embedding space, while different tuples should be positioned further apart. During training, we use the positive examples to reduce the distance between known duplicates as much as possible, while using the negative examples to maximize the distance between known non-duplicates. According to our label definition Y, the final function should yield a small distance value if two tuples are duplicates, and a large distance value otherwise.

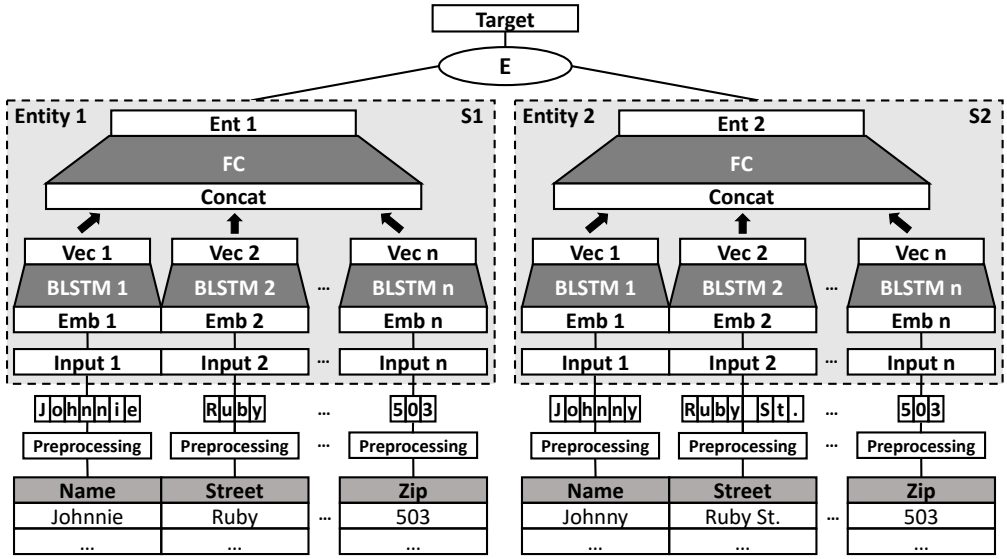

Figure 2.2: Overview of the SNN architecture

An SNN consists of two identical subnetworks S_1 and S_2, interconnected via a common energy function E. Since both subnets are identical in all aspects, this also means that they share a common weight or parameter matrix W. By sharing the model parameters between the subnets, both networks calculate the same function, thus making the learned similarity measure symmetric.

Figure 2.2 provides a conceptual overview of the network structure and illustrates all key components. As input, the network expects an entity pair in the form of two tuples and the corresponding label, which serves as the optimization target. More specifically, each tuple is provided to the network in an attribute-by-attribute fashion. Each of the two subnets thus processes one tuple of a given entity pair. Viewed from a conceptual level (see Figure 2.2), both input tuples pass through three network layers and eventually end up in the last layer as two reduced vectors of fixed dimensionality (Ent_1, Ent_2). The two resulting vectors serve as inputs to the energy function, which calculates a scalar value that can be interpreted as a similarity estimate. Next, we explain the structure of the two subnets (S_1, S_2) in detail.

2.3.3 Neural network model

This section contains a detailed description of the subnet architecture and an explanation of our design decisions. We experimented with different network architectures during the development phase, exploring different layer depths, pooling layers, and attention mechanisms. In the following, we describe the architecture that achieved the best results.

As input, each subnetwork receives one of the tuples t_1 or t_2. These tuples consist of attribute values $\mathcal{A} = (t[a_1], \ldots, t[a_n])$ that belong to a specific domain $\mathcal{D} = (d_1, \ldots, d_n)$. Since we intend to transfer knowledge between two networks, the tuples are fed to the network attribute by attribute. The reasons for this approach are explained in Sec-

tion 2.4. Although a domain can be categorized by its data type, for example, d_1 can be of type *integer* or *string*, we are more interested in the semantic content of each domain. For instance, attributes belonging to a specific domain represent *street names*, while attributes from another domain represent *movie titles*. The core idea is to use neural networks to learn the characteristic data distribution of each domain, using all domain attributes as training data. Once the weight matrices of the individual domains have been formed as a result of the network training, they can be transferred to another network. As such, the weight matrix for a *book title* attribute of a network N_1 can be used to initialize the weight matrix for a *book title* attribute of another network N_2 operating on a different dataset, thus implementing a knowledge transfer for book titles. Moreover, assuming that movies are often based on books, it is plausible to use the weight matrix of the *book title* attribute in N_1 to initialize the weight matrix of a *movie title* attribute in N_2, thereby performing a knowledge transfer from book to movie titles.

Examining the attributes of a particular domain, we find that they are often very different in length. For example, the street name "A Street" is relatively short, while "Newport Pagnell Motorway Services Areajunction 14 15 M1" is rather long. Since the network design has to cope with these varying attribute lengths, this affects both the structure as well as the building blocks used to construct the network.

Although it is possible to use 1D Convolutional Neural Networks (CNNs) [Goodfellow et al., 2016] to process sequential data streams of varying lengths $(x_1, \ldots, x_t)$, our architecture uses Recurrent Neural Networks (RNNs) [Goodfellow et al., 2016] for the sequential processing of the attribute values. Unfortunately, simple RNNs suffer from the vanishing [Bengio et al., 1994] or exploding [Pascanu et al., 2012] gradient problem and are therefore difficult to train. Due to this limitation, we decided to use Long Short-Term Memory (LSTM) networks [Hochreiter and Schmidhuber, 1997], an extension of RNNs that mitigate these problems. We arrange two LSTM networks according to the Bidirectional RNN architecture described by Schuster and Paliwal [1997], forming a *bidirectional LSTM network* (BLSTM). A BLSTM network consists of two LSTM networks that process the transferred character embeddings of a given attribute in opposite directions. The first network processes the input sequence from left to right, while the second processes the input sequence from right to left. Processing the input sequence from both sides allows us to consider context information from future and past states. The final results of both networks are then concatenated and forwarded to the subsequent network layer.

As shown in Figure 2.2, we use an embedding layer ($Emb_{1\ldots|\mathcal{A}|}$) as the first layer of each subnet. As discussed in Section 2.2, we feed each attribute value character-by-character into the network, building a character embedding for each character used within the attributes. We decided to process the attributes on a character basis, hoping that this enables the network to learn how to cope with character level issues, such as transposed numbers, misspellings, or shared character sequences. Such effects are particularly prevalent in attribute domains with many name variants and ambiguities, such as last names, company names, product names, and brands. These embeddings are utilized both by the subsequent BLSTM layer and during our knowledge transfer experiments, where they are used to transfer the underlying character distribution of

the attribute domains. We employ the BLSTM architecture described above as the next network layer. Since we use one BLSTM network per attribute, the network structure ultimately depends on the number of existing attributes in a tuple. This decomposition allows us to automatically learn a condensed representation of the underlying attribute domain for each attribute separately.

As discussed in more detail in Section 2.4, the knowledge accumulated in both the weight matrices of the attribute embeddings ($Emb_{1...|\mathcal{A}|}$) and BLSTM layers ($BLSTM_{1...|\mathcal{A}|}$) shall be transferred to another deduplication network, operating on a different dataset. Each BLSTM network generates a fixed-length vector ($Vec_{1...|\mathcal{A}|}$), representing a compressed version of the currently processed attribute. These vectors are then concatenated by a *Concat* layer, forming a compressed representation of the entire tuple.

The resulting tuple representation is then passed through a fully connected layer (FC), enabling the network to learn relationships between the constructed tuple's attributes. The noise that occurs in entities of structured data sources is often caused by typos or spelling variations in their attribute values. By leveraging character embeddings as well as end-to-end network training, we are able to learn that even though two entities differ in some characters, they still represent the same entity. This is achieved by adjusting the weights of the network during backpropagation so that the FC layer causes different weights to be assigned to character embeddings of common characters than to those that differ. In this way, the FC layer generates an entity representation that reduces or increases the distance of duplicate or non-duplicate entity pairs in the latent space despite the presence of noise. Finally, the resulting tuple embeddings (Ent_1, Ent_2) are compared by an energy function, whose result is then passed to the loss function, which in turn is used to calculate the gradients for the backpropagation step.

The individual dimensions of our model are as follows: The dimensionality of the character embeddings is 32. Our LSTM layer has 128 dimensions, and the fully connected layer (FC) has 64 dimensions. To determine optimal dimension values for the individual layers, it is possible to either experiment with different dimension values for each layer or determine more optimal values in a systematic way by leveraging hyperparameter optimization techniques [Bergstra et al., 2011; Snoek et al., 2012]. The details of our loss function are discussed in the next section.

2.3.4 Loss function

Our system's general intuition is to train a neural network $N_W(X)$ that is able to project the given tuples into a low-dimensional embedding space. In this space, duplicate and non-duplicate tuple pairs can be distinguished by using simple distance measures, such as the Euclidean distance or other more complex measures [Cha, 2007]. Learning in this context means adapting the weights of N_W so that the calculated loss is minimized. The goal of this mapping is to place similar tuples close to each other in the embedding space while dissimilar tuples are placed further apart. To learn a function with these properties, we use the contrastive, energy-based loss function presented in Hadsell et al. [2006]. As described in the last section, both tuples are passed through the network yielding one embedding vector (Ent_1, Ent_2) for each tuple. For the Euclidean distance

as energy function $E_W(Ent_1, Ent_2) = \|Ent_1 - Ent_2\|$, the general definition of the loss function is:

$$\mathcal{L}(E_W, Y) = Y L_D(E_W(Ent_1, Ent_2)) + (1 - Y) L_{ND}(E_W(Ent_1, Ent_2)) \tag{2.1}$$

The dependence of the energy function on the parameter values of the network is expressed by the W in the definition. At its core, the loss function consists of the terms $L_D(E_W)$ and $L_{ND}(E_W)$, which ensure that similar inputs receive a low energy value, whereas unequal inputs receive a higher one. In their work, Hadsell et al. motivate the intuition behind these terms with the behavior of mechanical springs, where L_D brings similar embedding vectors closer together, and L_{ND} repels different vectors [Hadsell et al., 2006]. If we define the two terms as follows:

$$L_D(E_W) = \frac{1}{2}(E_W)^2 \tag{2.2}$$

$$L_{ND}(E_W) = \begin{cases} \frac{1}{2}(m - E_W)^2 & \text{if } E_W < m \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

The result is the following loss function:

$$\mathcal{L}(E_W, Y) = Y \frac{1}{2}(E_W)^2 + (1 - Y)(\frac{1}{2}\max(0, m - E_W))^2 \tag{2.4}$$

The value m in this definition is a margin value, which controls that only tuple pairs within the specified margin contribute to the loss function. In addition, Hadsell et al. emphasize the meaning of the L_{ND} term, which makes it possible to take dissimilar tuples as well as similar tuples into account when minimizing E_W, thus enabling a more optimized solution.

Duplicate detection often involves highly skewed datasets containing many more non-duplicate pairs than duplicate pairs. We introduce the weights W_D and W_{ND}, which allow us to weigh the loss terms L_D and L_{ND} differently. In this way, we can control how strong duplicates or non-duplicates contribute to the calculated loss. Extending Equation (2.4) with the weight parameters W_D and W_{ND} yields our final loss function:

$$\mathcal{L}(E_W, W_D, W_{ND}, Y) = W_D Y \frac{1}{2}(E_W)^2 + W_{ND}(1 - Y)(\frac{1}{2}\max(0, m - E_W))^2 \tag{2.5}$$

We used the duplicate to non-duplicate ratio of the respective training set to determine appropriate weights (W_D, W_{ND}) for the loss function. Because we generated our datasets with a duplicate, non-duplicate ratio of 1 to 10 (described in Section 2.5.3), we also adopt these settings for the weight parameters W_D and W_{ND}. In cases where it is difficult to estimate the parameter values for W_D and W_{ND}, they can be determined using hyperparameter optimization [Bergstra et al., 2011; Snoek et al., 2012].

2.3.5 Training details

We use backpropagation to determine the gradient of the loss function with respect to the weights W. We update the weights by employing Adaptive Moment Estimation (Adam) [Kingma and Ba, 2015], which is capable of computing adaptive learning rates for each parameter. We kept the default settings for the optimizer at $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10^{-8}$. Due to the weight sharing between S_1 and S_2, the gradients of both networks behave additively, so that we use the summed gradient contributions of both subnets while updating the weights. For training, we use batch sizes of 16 and 32. The margin on the loss function was set to a value of 1. To initialize the weights of each layer, we use the Xavier initialization scheme as presented in [Glorot and Bengio, 2010]. To counteract overfitting of the network, we employ dropout regularization for each time step of the LSTM by setting the dropout value to 0.1 [Srivastava et al., 2014].

We trained the network until convergence, which in our case, required between 10-15 epochs. For the implementation of the model, we used the Keras Framework using Tensorflow as its backend. All models were trained using an Nvidia Titan X Pascal GPU. Training a model took between 30 minutes and 24 hours, depending on the dataset size. For all datasets that were categorized as *small* in Table 2.1, we performed a 10-fold cross-validation. We initially divided the entire dataset into a training and a test set at a ratio of 70 to 30 percent, using the training set to perform a 10-fold cross-validation. For each fold, we selected the model from the epoch with the best F-measure on the cross-validation's test data to make a prediction on the previously separated test set. We use these predictions to report on the average precision, recall, and F-measure over all 10 folds.

Since the datasets categorized as *large* in Table 2.1 were too large for a full cross-validation, we employed a different evaluation scheme. We divided the corresponding datasets according to a ratio of 3:1:1 into training, validation, and test sets. We then trained on the training set and used the validation set to select the model from the epoch with the best F-measure, which we used to make a prediction on the test set. In this scenario, we report the best precision, recall, and F-measure on the test set. For the entire model, we used rectified linear units (RELUs) as our activation function.

2.4 Knowledge transfer

To reduce the amount of data needed to train neural networks, our goal is to transfer already learned knowledge between networks. As pointed out by Pan and Yang [2010], transfer learning addresses this by allowing domains, tasks, and distributions used in training and testing to be different. According to their proposed categorization, we identify our case of knowledge transfer as "Transductive Transfer Learning". While for us, the task of duplicate detection always remains unchanged, we change the underlying domain by switching from detecting duplicates in one dataset (e.g., Movies) to detecting duplicates in a different dataset (e.g., CDs). Thus, knowledge transfer makes it possible to train models for which this would not have been possible due to insufficient training data.

As in other areas, the available datasets for duplicate detection differ significantly in terms of labeled training data. On the one hand, some datasets contain a large number of already labeled duplicate pairs, while others contain almost no labeled pairs. Our central idea is to exploit this imbalance by training a neural network on a dataset with a large number of duplicate pairs and transferring the acquired knowledge to the classification of duplicates, where the dataset contains significantly fewer training examples.

As described in Section 2.3.3, a crucial step in our deduplication approach is to learn the characteristic data distribution for each attribute domain individually. This is achieved by the network design shown in Figure 2.2. Here, the critical part is making use of individual embedding and BLSTM layers for each attribute. By deliberately separating the processing of the individual attributes, the domain knowledge gained during training accumulates in the weight matrices of the dedicated embedding and BLSTM layers. This effect allows us to transfer the domain knowledge for specific attributes individually. For example, we can choose to transfer only the knowledge of specifically selected attribute domains, which can give us an advantage in the target domain. Formally, knowledge transfer can be defined as follows:

> **Knowledge transfer:** Given two neural networks – a source network N_1 and a destination network N_2 – as well as their associated training data T_1 and T_2, we define knowledge transfer as the transfer of selected weight matrices from N_1 to N_2. To this end, we train N_1 using T_1 and transfer the gained knowledge to N_2 before training it on T_2.

> **Domain compatibility:** We define that two attribute domains – a source attribute domain D_S and a destination attribute domain D_T – are *domain-compatible* to each other if they are either equal ($D_S = D_T$) (e.g., both street names) or structurally similar ($D_S \approx D_T$) to each other (e.g., book title and movie title).

Although we assign compatible attribute domains manually in our experiments, systems, such as Sherlock [Hulsebos et al., 2019], can be used to determine the semantic type of data columns, according to which an assignment of compatible attribute domains can be determined. Alternatively, systems like DeepAlignment [Kolyvakis et al., 2018] can also be used to automatically determine an appropriate domain assignment. This aspect is outside the scope of this thesis.

In our case, there are essentially three possible configuration options for conducting a knowledge transfer: (i) transferring only the weight matrices of the embedding layer, (ii) transferring only the weight matrices of the BLSTM layer, or (iii) transferring both weight matrices together. As part of our experiments in Section 2.6.2, we investigate the effects of each of the three configurations on the network's performance. Technically, we perform the knowledge transfer by using the weight matrices of the embedding and BLSTM layers in N_1 to initialize the weight matrices of compatible attributes in N_2. As a result, we are able to train N_2 even if T_2 contains much fewer training examples than T_1 ($|T_2| \ll |T_1|$). This method is often referred to as "fine-tuning" a network. Note that the knowledge accumulating in the weight matrices of the FC layer during the

training process is specific to the entities being compared (e.g., books or movies), not the attributes (e.g., name, zip code). Thus, the weights of the FC layer are not easily transferable for the deduplication of other entities, such as cars or hotels. While the learned attribute representations can be reused for the deduplication of other entities, the knowledge of how to combine the individual attribute representations into a vector-based representation of the processed entities must be relearned for the deduplicated entities. Due to this fact, we cannot avoid the retraining of the network on a new dataset. To adapt the FC layer's weights to the characteristics of a new dataset, at least a small amount of training data is needed. To transfer the maximum available knowledge, this transfer can be performed for multiple attribute domains coming from different source networks. For instance, assuming two previously trained source networks N_1 and N_2, it is possible to transfer a weight matrix for *art titles* from N_1 and a weight matrix for a *location* domain from N_2 to a target network N_3. The elegance of attribute-based knowledge transfer lies in the fact that a steadily growing repository of weight matrices for a wide range of attribute-domains accumulates over time. In case only some attributes are domain-compatible during knowledge transfer, the remaining attribute values could be initialized with weights from the repository. In addition, one could try to learn the weight matrices for a particular domain completely unsupervised through the use of encoder-decoder architectures, which would allow the repository to be easily extended. By following this protocol, it is possible to combine knowledge from multiple different source networks to initialize as many attributes of the target network as possible with pre-trained weights. Finally, it should be noted that the transferred weight matrices can be used to initialize multiple attribute domains of the target network if a domain meets the requirements mentioned above.

2.5 Data & Gold-standard

This section provides a brief overview of the used datasets (Section 2.5.1), their preprocessing (Section 2.5.2), and a description of the process used for creating non-duplicate pairs (Section 2.5.3).

2.5.1 Datasets

The identification and evaluation of real-world solutions for duplicate detection also requires the use of real-world data sets. Since the proposed method can be used to find duplicates both within the same dataset (*duplicate detection*) and between different datasets (*record linkage*), we test it on datasets for both tasks. The main difference between the two scenarios is that duplicate detection aims to find all duplicates within a single dataset, whereas record linkage seeks to uncover duplicates across multiple datasets. Given two relations $\mathcal{A}$ and $\mathcal{B}$ of sizes n and m respectively, the process of duplicate detection produces candidate pairs in $\mathcal{O}((n+m)^2)$, whereas, in a record linkage scenario, the number of pairs that need to be evaluated is in $\mathcal{O}(n \cdot m)$. Therefore, the two scenarios differ in that deduplication creates more candidate pairs, which can result in

more duplicates being misclassified, which in turn can negatively impact the classification results. Although the record linkage scenario can also be mapped to a deduplication scenario, we have deliberately refrained from doing so to reflect the evaluation setup of the competitor approaches.

For comparison on the deduplication task, we use the same datasets as Christen, which can be found on the project website[1]. The details of the individual datasets can be found in Table 2.1 and the brief descriptions below.

Dataset	#dpl	#ndpl	#records	class
Deduplication				
Restaurants	112	1,120	864	small
Census	376	3,760	841	small
CD	300	3,000	9,763	small
Cora	64,578	179,125	1,879	large
Movies	14,190	141,900	39,180	large
Record linkage				
BeerAdvo-RateBeer	68	382	544	small
iTunes-AMA	132	407	933	small
DBLP-Scholar	5,473	54,730	66,879	large
DBLP-ACM	2,224	22,240	4,910	large
WMT-AMA	1,157	11,570	24,628	large
AMA-GOOG	1,300	13,000	4,589	large

Table 2.1: Datasets used for our experiments

- **CD**: Entries of audio CDs with descriptive attributes, such as artist, title, tracks, genre, and year.
- **Census**: Based on real data generated by the U.S. Census Bureau, it contains a single attribute with a record value, called "text". This dataset contains two relations "A" and "B" and could also be used for record linkage, i.e., linking the two relations. Like Christen, we treat it as a typical single-relation dataset and try to find duplicates instead of linking matches.
- **Cora**: Bibliographic records of publications for machine learning. It includes reference information, such as authors, title, and year.
- **Movies**: Result of merging two different datasets, such that real-world duplicates are available. The information provided is limited to actors and movie titles.
- **Restaurants**: Mixed dataset of two relations, based on the Fodor's and Zagat's restaurant guides. This corresponds to the Fodor-Zagat dataset in [Mudgal et al., 2018].

[1] `https://hpi.de/naumann/projects/repeatability/duplicate-detection/knowledge-transfer-for-duplicate-detection.html`

For the task of duplicate detection, we compare our approach with that of Christen [2008], which is based on the classification of feature vectors created from the compared entities. To this end, they use an SVM classifier with various hyperparameters.

For the record linkage task, we compare our approach with the reported performance metrics of the DeepMatcher system and use the same datasets as Mudgal et al. [2018] (see Table 2.1). Though DeepMatcher can be configured with several different architectural variants, their comparison shows that, apart from the hybrid variant, the RNN variant is superior to the other architectural variants in terms of performance. When comparing the RNN with the hybrid variant, it turns out that the former is, on average, only 1.5% F-measure points worse, which does not constitute a clear superiority of the hybrid variant. For this reason, and because the RNN architecture of DeepMatcher comes closest to our approach, we focused on comparing our approach to the RNN variant of DeepMatcher. We do not compare our approach to their *Home*, *Electronics*, and *Tools* datasets, as they are not publicly available. All other datasets can be obtained from the Magellan Data Repository [Das et al., 2018], which contains not only statistics about each dataset but also a detailed description of how they were created.

It should be noted that the BeerAdvo-RateBeer and iTunes-AMA datasets were the only datasets that provided both duplicate, as well as non-duplicate pairs, and are thus considered complete. All other datasets are considered incomplete, as they only provide duplicate pairs. To overcome this issue, we created non-duplicate pairs for all incomplete datasets according to the procedure described in Section 2.5.3. Statistics about the number of duplicate ($\#dpl$) and non-duplicate ($\#ndpl$) pairs, the total number of records ($\#records$) as well as a categorization into small and large datasets ($class$) can be found in Table 2.1. We categorized datasets that contain fewer than 4,500 labeled tuple pairs ($\#dpl + \#ndpl$) as small datasets and those with more than 4,500 labeled tuple pairs as large datasets.

2.5.2 Data preprocessing

We kept preprocessing to a minimum to not alter the input data too drastically from its original raw state. As such, we first lower-cased all input characters, which represents a typical preprocessing step that is not specifically related to the used datasets. This not only helped the network to generalize better but also reduced the size of the embedding layers.

We then normalized any Unicode characters so that, e.g., accents and umlauts are translated into their respective ASCII characters, using the *normalize* function of the Python *unicodedata* package. For example, the German *ö* was replaced by the combination *oe*, whereas other characters, such as *é* or *ç*, were simply replaced with the letters *e* and *c*. During this process, we also replaced existing NULL values with an empty string. This normalization step further reduced the size of the embedding layers.

In a final step, we translated each ASCII character into its corresponding ASCII numeric representation, which served as input for the network. To speed up network training, we truncated the values of the description attribute of the AMA-GOOG dataset after 1,000 characters.

2.5.3 Selecting duplicate and non-duplicate pairs

The gold standards that are available for most of the datasets in Section 2.5.1 each contain a list of record pairs that uniquely identify *duplicate record pairs*. In case this set is not transitively closed, we additionally create all transitive pairs and call this extended set *DPL*.

For most datasets, the number of existing duplicates represents only a small subset of all possible tuples. To train and test a classifier, we also require a number of negative examples, i.e., non-duplicate pairs. Unfortunately, a random pairing of tuples most likely results in non-duplicate tuple pairs that exhibit a very low similarity in their attribute values, which makes them easily distinguishable from true duplicates. This setup does not reflect the challenges of a real-world matching process; hence it is necessary to create non-duplicate pairs of greater similarity, making it harder for the model to classify them. Since training and testing with hard to distinguish non-duplicates comes much closer to a realistic matching scenario, we rely on blocking techniques [Christen, 2012; Elmagarmid et al., 2007] to generate non-duplicate pairs that are harder to distinguish from real duplicates. In practice, to speed up duplicate detection, blocking methods are typically used to divide datasets into disjoint subsets, called blocks, according to a predefined partitioning key. To avoid false negatives, usually, multiple partition keys are defined. It is important to select the partition key with great care, as it controls not only the size and number of blocks created but also how similar the individual data records within each block are.

We perform a simple blocking by using each attribute of the currently processed dataset as a partitioning key. In case an attribute is multi-valued, we first divide it into its individual values and use those for blocking. Attributes exhibiting a high level of uniqueness, which would lead to many single-record blocks, are split into n-grams of 6 to 10 characters, depending on the length of the attribute, and then used as partition keys. As a result, we obtain numerous blocks, within which we create the cross-product of all contained data records, and thus form data record pairs that are then combined into a common dataset. After removing all known duplicate pairs from the resulting dataset, we refer to it as *NDPL*.

The NDPL dataset is, therefore, the set of all record pairs that can be formed within all blocks and does not include duplicate pairs. To achieve a more realistic training set, we ensured that the ratio of DPL to NDPL is 1:10 by randomly selecting pairs from the NDPL dataset. An exception to this is the Cora dataset, where we were unable to maintain the ratio of 1:10 without drastically reducing the number of n-grams used in the blocking strategy.

2.6 Experiments

This section presents the results of our experiments. In Section 2.6.1, we discuss the results achieved by training the SNN from scratch, while in Section 2.6.2, we present how transfer learning helped to further improve the results. For details about the training process and its parameters, please see Section 2.3.5.

2.6.1 Learning from scratch

As a first experiment, we train the network on each of the datasets listed in Table 2.1 and report the achieved precision, recall, and F-measure values. To see how our approach performs against deduplication systems that use manual feature engineering, we compare ourselves to the SVM-based system of Christen [2008]. As pointed out by Ebraheem et al., DeepMatcher is an extension of DeepER [Ebraheem et al., 2018], hence in the record linkage case, we compare SNNDedupe with DeepMatcher.

Duplicate detection

As seen in Table 2.2, we surpass the SVM-based approach in all cases, sometimes even dramatically, as in the case of the Cora dataset, where we reach an F-measure of 99.24 percent. On average, we manage to exceed the SVM approach by +5.9 and +23.7 percentage points in precision and recall, corresponding to an average improvement of +15.8 percentage points in F-measure. We attribute the increase in performance to the possibility of training the network in an end-to-end fashion. For example, the character embeddings are trained together with the deduplication task, and, in contrast to manually created features, can be adapted according to the propagated error. In this way, the entire network is tuned to the deduplication task, which is difficult to accomplish with an SVM approach that works with a number of predefined features.

Dataset	**SNNDedupe**			**SVM**		
	P	**R**	$\mathbf{F_1}$	**P**	**R**	$\mathbf{F_1}$
Restaurants	96.5	100	**98.2**	97.3	75.8	84.1
Census	88.7	94.6	**91.5**	87.5	75.1	80.8
Cora	99.3	99.2	**99.2**	82.2	71.8	76.4
CD	91.8	84.0	87.4	–	–	–
Movies	93.1	85.7	89.2	–	–	–

Table 2.2: Performance comparison for deduplication datasets as reported in [Christen, 2008]

An interesting observation during the evaluation was the analysis of the causes for misclassification: We found that when the network generated classification errors, they were often accompanied by missing data in at least one of the entity's attributes. This is a well-known problem in the area of duplicate detection, and we noticed that it negatively affects the performance of the neural network. In short, if we decide to replace an empty attribute field with a specific value, we change the entities so that they become either more similar or dissimilar. In the worst case, this process makes the correct classification considerably more difficult. To avoid this problem, we decided to exclude attributes with missing values from the entity representation that is passed to the network. Technically, we achieve this by replacing both attribute values in their respective entities with a fixed value, e.g., zero, and use a corresponding masking layer within the network to make

these specific values (zeros) invisible to the network. Effectively, this causes the replaced attribute values to be ignored by the network.

Record linkage

In the case of record linkage, Table 2.3 shows that we were able to outperform our competitor on four out of six datasets. The largest improvement was achieved on the AMA-GOOG dataset, where we measured an improvement of +26.2 percentage points in F-measure over DeepMatcher. We suspect this performance increase to be caused by the truncation of the description attribute during the preprocessing phase, making it easier to learn a better representation of the description attribute. In most other cases, the performance increase is not as significant as with the deduplication task, however, we were also able to measure an improvement of +4.4 percentage points for the iTunes-AMA dataset.

Dataset	**SNNDedupe**			**DeepMatcher**		
	P	**R**	**F_1**	**P**	**R**	**F_1**
BeerAdvo-RateBeer	74.9	74.3	**73.1**	59.1	92.9	72.2
iTunes-AMA	93.8	93.0	**92.9**	92.0	85.2	88.5
DBLP-Scholar	94.8	91.9	**93.3**	93.2	92.7	93.0
DBLP-ACM	98.4	98.0	98.2	97.1	99.5	**98.3**
WMT-AMA	61.6	60.8	61.2	70.9	64.6	**67.6**
AMA-GOOG	90.6	82.1	**86.1**	69.5	52.6	59.9

Table 2.3: Performance comparison for record linkage datasets as reported in [Mudgal et al., 2018]

In the case of the WMT-AMA dataset, our approach is 6.4 percentage points worse than DeepMatcher. As mentioned by Mudgal et al., it can sometimes occur that due to inaccurate extraction methods, some attribute values are not assigned to the intended columns, but to columns of other attributes [Mudgal et al., 2018]. Since the WMT-AMA records contain such impurities, our network architecture struggles to find a good representation for dirty attribute columns, which is reflected in weaker performance. To address this issue, we could proceed similarly to Deep Machter's dirty entity matching case, in which a representation of the entire entity is first created by concatenating its attribute values before the deduplication takes place. Using this approach, the network would be able to determine the similarity of two entities in their entirety, eliminating the issue of incorrect attribute assignments because entities are no longer deduplicated on attribute but on entity level. Unfortunately, leaving the attribute level would prohibit the formation of attribute embeddings during the deduplication process, which would render the envisioned attribute-level knowledge transfer impossible. We have, therefore, decided not to proceed accordingly.

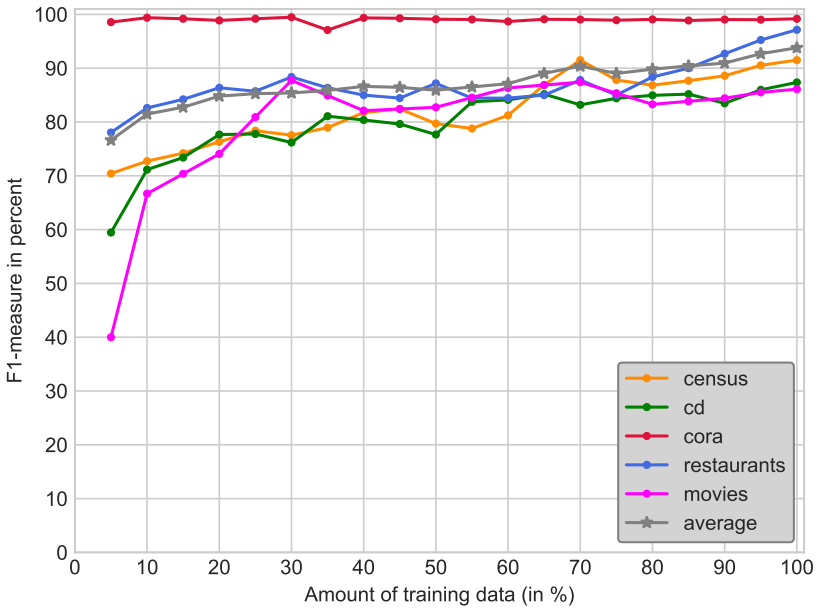

Figure 2.3: Performance of SNNDedupe on deduplication datasets for increasing amount of training data (of a 3:1:1 split).

In summary, our results are on par with those of DeepMatcher, justifying our design decisions, but in addition, allowing us to perform knowledge transfer. In Section 2.6.2, we examine the effects of transfer learning on the performance of the network.

Learning with increasing training data

Due to the large number of record pairs that can arise during duplicate detection, a complete annotation of the entire dataset is usually very cost-intensive. This raises the question of how SNNDedupe behaves for different amounts of training data. To answer this question, we train SNNDedupe with an increasing amount of training data, starting at 5% of the corresponding dataset, and gradually increasing it by 5% increments until we arrive at utilizing the entire amount of training data. We do this for both the duplicate detection and the record linkage datasets, with results shown in Figures 2.3 and 2.4. As both figures show, the performance of SNNDedupe depends on the complexity of the dataset. While with relatively simple datasets, such as Cora or DBLP-ACM, an F-measure of 98.6% and 94.89% can be achieved with only 5-10% of the total training data, it is more difficult to achieve a satisfactory F-measure with more demanding datasets, such as WMT-AMA, where we obtain an F-measure of 61.2% even after utilizing all training data.

Looking at the average performance, it can be seen that SNNDedupe is capable of achieving F-measure values between 77-81% for deduplication and 59-66% in case of record linkage using just 5-10% of training data. In addition, we observe that for every additional 5% of training data, the performance increase is significantly stronger in the beginning than later on. Thus, the average increase in F-Measure for adding 5%

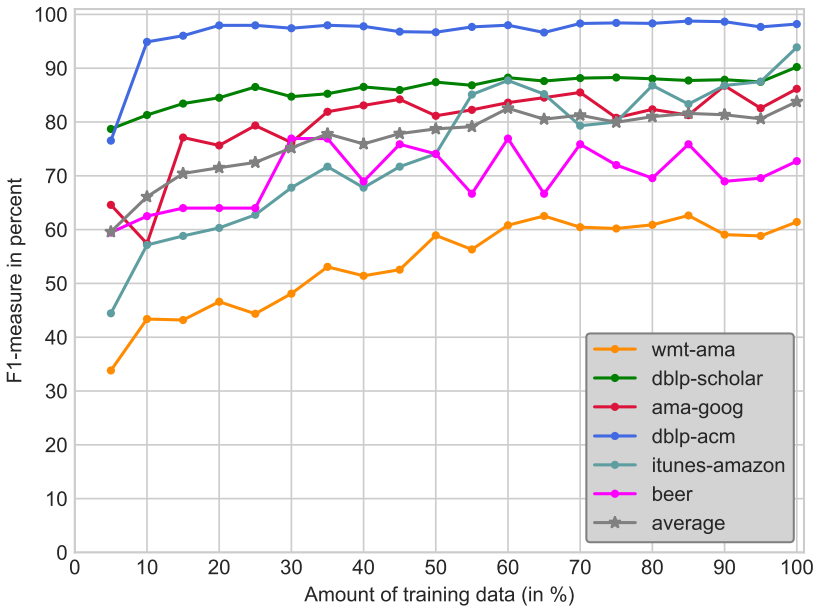

Figure 2.4: Performance of SNNDedupe on record linkage datasets for increasing amount of training data (of a 3:1:1 split).

additional training data within the first 25% of the dataset amounts to 3.23% for record linkage and 2.16% for deduplication. In contrast, for every 5% increase in training data within the remaining 75% of the dataset, the average increase in F-measure measures only 0.56% (deduplication) and 0.75% (record linkage). This indicates that SNNDedupe is able to provide a good classification performance after using up to 25% of all available training data. In practice, however, this depends on the individual case as well as on the complexity of the underlying dataset, which can be seen in the case of iTunes-AMA, where a significant performance increase is also observed later during the transition from 50 to 55% of training data. Another interesting finding is that the amount of training data used also influences the stability of predictions. As such, the prediction behavior is relatively stable for datasets with many training examples, whereas in the case of the Beer dataset, which consists of only 450 training examples, strong fluctuations in the predictions of SNNDedupe can be observed. In general, SNNDedupe is able to achieve average F-values of 72.49% (record linkage) and 85.26% (deduplication) by using up to 25% of the respective training data.

It should be noted that for this experiment, the increase in training data is achieved by adding randomly selected training examples. The selection of informative training examples is at the core of active learning approaches and outside the scope of this thesis.

2.6.2 Transfer learning

We examine the effects of transferring selected weight matrices, and thus the knowledge they contain, from one network to another. Our goal is to explore how knowledge transfer affects the training process and, thus, the performance of the network.

Experimental setup

As explained in Section 2.3, the network architecture is structured so that each attribute is processed by a dedicated subnetwork. This allows us to carry out even partial knowledge transfers by transferring only the weight matrices for specific attributes. The general experimental setup can be described as a two-step process. First, we take a neural network (N_1), which we previously trained on a source dataset and transfer the knowledge contained in the weight matrices for specific attributes to another untrained network (N_2). After the weight transfer is complete, we train the network N_2, which already contains the knowledge of N_1 for the transferred attributes, on a destination dataset. During our experiments, we transfer knowledge between two network pairs, which each operate on one dataset pair. The first transfer takes place from a network that has been trained on the Movies dataset to a network that shall be trained on the CD dataset. In a second transfer, we transmit knowledge between a network trained on the WMT-AMA dataset and a network that we then train on the AMA-GOOG dataset.

As discussed in Section 2.4, the general assumption made for these transfers is that the data distribution of the transferred source attribute resembles the data distribution of the destination attribute and is therefore well suited for knowledge transfer. With this in mind, we decided for the first transfer to map the weight matrices of the *actor* and *title* attributes of the Movie dataset to the *artist* and *title* attributes of the CD dataset, respectively. This concrete transfer is reflected in the notation used in Table 2.4: $dataset_{src}.attribute_{src} \longrightarrow dataset_{dst}.attribute_{dst}$ describes from which source attribute ($attribute_{src}$) of the source dataset ($dataset_{src}$) to which destination attribute ($attribute_{dst}$) of the destination dataset ($dataset_{dst}$) the knowledge transfer is performed. Although we manually perform the mapping, schema matching research has produced a number of techniques to automate this step, both heuristically and using machine learning approaches [Bellahsene et al., 2011; Kolyvakis et al., 2018]. The exploration of these techniques lies outside the scope of this thesis.

What to transfer?

The next step is to determine which of the weights are transferred from one network to the other. As shown in Figure 2.2, each subnetwork consists of two layers, an embedding layer ($Emb_{1...|\mathcal{A}|}$) and a BLSTM layer ($BLSTM_{1...|\mathcal{A}|}$). For our experiment, this results in three possible transfer configurations, which are listed in Table 2.4. First, we transfer only the weight matrices of the embedding layers, in a second step only the matrices of the BLSTM layers, and finally, both matrices of the embedding, as well as those of the BLSTM layers, are transferred. Once the weight transfer is complete, we train the network on the destination dataset, as described in the previous section. Table 2.4 reports on precision, recall, F-measure, and the relative improvements over the baseline, which was trained from scratch in the previous section. The baseline values can thus be found in Table 2.2 and Table 2.3. The relative improvements over the baseline are shown in parentheses below their corresponding precision, recall, and F-measure values.

Transfer of Embeddings	P	R	F_1
Movies.actors ⟶ CD.artists Movies.title ⟶ CD.album	87.1 (-4.7)	94.5 (+10.5)	90.4 (+3.0)
WMT-AMA.title ⟶ AMA-GOOG.title WMT-AMA.techdetails ⟶ AMA-GOOG.description WMT-AMA.brand ⟶ AMA-GOOG.manufacturer	90.2 (-0.4)	86.1 (+4.0)	88.1 (+2.0)
Transfer of BLSTMs			
Movies.actors ⟶ CD.artists Movies.title ⟶ CD.album	88.3 (−3.5)	87.5 (+3.5)	87.6 (+0.2)
WMT-AMA.title ⟶ AMA-GOOG.title WMT-AMA.techdetails ⟶ AMA-GOOG.description WMT-AMA.brand ⟶ AMA-GOOG.manufacturer	89.8 (-0.8)	84.1 (+2.0)	86.8 (+0.7)
Transfer of Embeddings & BLSTMs			
Movies.actors ⟶ CD.artists Movies.title ⟶ CD.album	91.9 (+0.1)	92.5 (+8.5)	92.1 **(+4.7)**
WMT-AMA.title ⟶ AMA-GOOG.title WMT-AMA.techdetails ⟶ AMA-GOOG.description WMT-AMA.brand ⟶ AMA-GOOG.manufacturer	91.7 (+1.1)	89.8 (+7.7)	90.7 **(+4.6)**

Table 2.4: Results of transferring different weight matrices from one network to another

Transfer embeddings. Regarding the first experiment, in which only the embeddings are transferred, it is noticeable that an improvement of +3.0 and +2.0 percent compared to the respective baseline can be observed for both datasets. Upon closer inspection, it can be seen that this improvement is mainly driven by an improvement in recall (+10.5 / +4.0). These results support our initial assumption that the network, due to its high flexibility, is capable of learning a similarity measure that correctly identifies more duplicate pairs. It also shows that the knowledge about certain attribute domains contained within the embeddings can be transferred to other attributes that possess similar domain properties. Attention should also be paid to the circumstance that, in both cases, the recall values increase significantly without a severe decrease in precision. This interaction ultimately leads to the aforementioned improvement in F-measure.

Transfer BLSTM weights. In a second experiment, we transfer only the weight matrices of the BLSTM layers. Compared to the transfer of the embeddings, there is only a relatively small improvement of +3.5 and +2.0 percent in the recall values, while the precision values decrease in both cases. On careful consideration, this was to be expected, as the BLSTM layers were trained together with the embedding layers and therefore the values of their weight matrices are conditioned on the specific embeddings that were jointly trained. If the embeddings are not transferred, but initialized randomly, the weight matrices of the BLSTM layer also lose a lot of their significance. Ultimately, this causes the network to perform hardly better than the baseline with +0.2 and +0.7 percent improvement in F-measure.

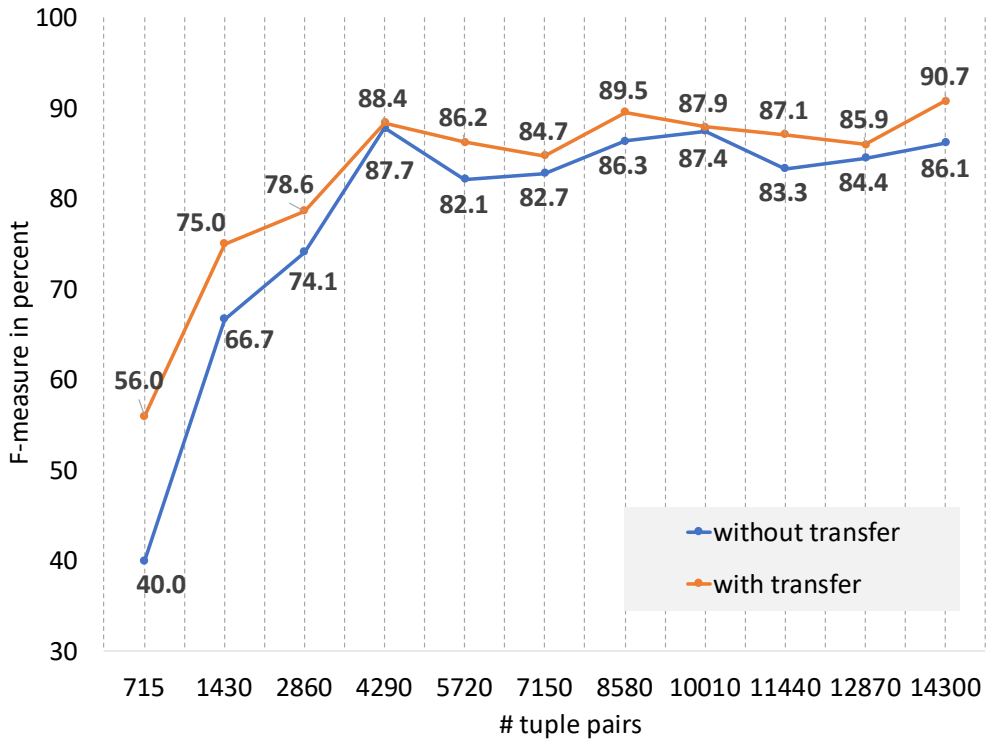

Figure 2.5: Development of the F-measure with and without knowledge transfer.

Transfer embeddings & BLSTM weights. In the last transfer experiment, we measure the network's performance after transferring both the weight matrices of the embedding and the BLSTM layers. With an improvement of +8.5 and +7.7 percent, the recall values show the greatest improvement in this experiment. In contrast to the experiment in which only the embeddings were transferred, however, an improvement in the precision values of +0.1 and +1.1 percent can also be observed. Even though the improvements in the precision values are small, in combination with the recall improvements, they ensure that the F-measure values improved by +4.7 and +4.6 percent compared to the baseline. These results lead us to conclude that the best results can be achieved when the weight matrices of both the embedding and the BLSTM layers are transferred jointly.

Overall, the transfer enabled us to improve the total network performance by +4.7 percent from 87.4 to 92.6 percent F-measure for the CD dataset and by +4.6 percent F-measure from 86.1 to 90.7 percent for the AMA-GOOG dataset.

Data reduction through knowledge transfer

In a final experiment, we investigate the extent to which transfer learning can be used to reduce training data. To this end, we trained the network on an increasing number of tuple pairs from the AMA-GOOG dataset and recorded the F-scores achieved with and without knowledge transfer. The knowledge transfer was carried out in the same way

as in the previous transfer experiment (WMT-AMA $\longrightarrow$ AMA-GOOG). We transferred both the weights of the embedding layers as well as those of the BLSTM layers. Figure 2.5 shows that knowledge transfer significantly improves performance when training with a smaller number of tuple pairs. This performance increase can be observed up to a number of 4,290 tuple pairs and is less prominent afterward. The transfer values always remain above the values measured without transfer. Considering the graph, it can be observed that an F-score of 75.0 percent can be achieved when training with 1,430 tuple pairs if a knowledge transfer has been carried out beforehand. To achieve the same F-score without knowledge transfer, one would have to use more than twice as many tuple pairs for training: the required amount of training data can be significantly reduced through a well-executed knowledge transfer.

2.7 Summary

In this chapter, we introduced a Siamese neural network capable of learning a similarity measure between tuple pairs of specific datasets, that can then be used to detect duplicates. In doing so, we eliminate the manual feature engineering process and significantly reduce the effort required for model building. We compare our approach with two competitors, a more traditional, SVM-based duplicate detection approach and DeepMatcher, a neural network for entity linking. In duplicate detection, we were able to outperform our competitor in all cases, with performance improvements of up to 22 percent. We managed to outperform the DeepMatcher system on four out of six entity-linking datasets and achieved improvements of up to 26 percent F-measure.

We conceived and implemented a knowledge transfer between two deduplication networks, which shows that knowledge which accumulates during the training in the weight matrices of one network can be transferred to another deduplication network. By transferring the matrices of selected attributes, we succeeded in increasing the overall network performance by +4.7 and +4.6 percentage points. This marks a significant performance increase when compared to the respective baselines. In addition, we showed in a subsequent experiment that it is possible to reduce the amount of training data by performing a knowledge transfer.

Chapter 3

Extracting Knowledge from Unstructured Data

As outlined in Section 1, the information contained in unstructured data sources has tremendous value and can be a business-critical asset for the success of many corporations. Text documents, such as newspaper articles or financial credit reports, may contain information about a merger of two companies or a company's solvency. Such information is often only available in text form, which is not easily accessible via traditional systems, such as relational or non-relational database systems. Making this information accessible in a structured and machine-readable way is at the core of the *text mining* component. It extracts the information required for a specific use case from textual documents and integrates it into a knowledge base that can then be used by other applications.

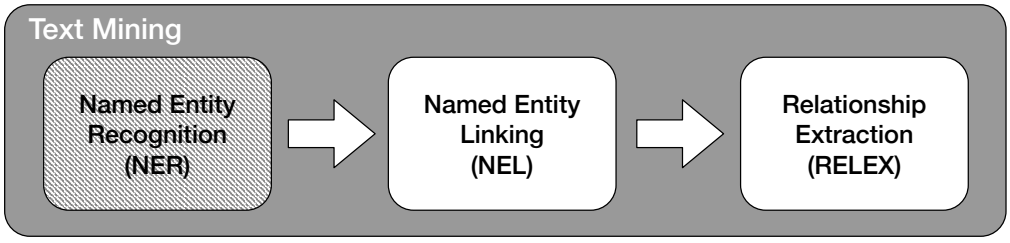

Figure 3.1: Overview of the text mining pipeline and its subcomponents

As shown in Figure 3.1, the *text mining* component itself is divided into the three subcomponents *named entity recognition* (NER), *named entity linking* (NEL), and *relationship extraction* (RELEX), which are the subject of the following sections.

In this chapter, we present a conditional random field (CRF) [Lafferty et al., 2001] based machine learning approach capable of reliably recognizing organizations in German texts. In particular, we address the problem of recognizing company names from textual data by incorporating dictionary matches into the training process of a CRF. The resulting NER can then be applied as the first subcomponent of the *text mining* pipeline. While named entity recognition is a much-addressed research topic, recognizing companies in texts is of particular difficulty. Company names are extremely heterogeneous in

structure. In many cases, they are referenced by different names made up of various constituent parts, such as personal names, locations, acronyms, numbers, and other unusual characters. For example, the company name "da Jobst Getränke und mehr . . . e.K. Inh. K. Kubitscheck" contains, in addition to the company name itself (da Jobst), it's business area (Getränke), punctuation marks (. . .), its legal form (e.K.), the name of a person (K. Kubitscheck), the role that the person plays within the company (Inh.) and some parts that cannot be clearly assigned to any specific category (und mehr). Further, instead of using the official company name, quite different colloquial names are frequently used by the general public. Thus, the car manufacturer "Daimler AG" is popularly known under a multitude of other names such as "Mercedes", "Daimler", "Mercedes-Benz", "die Singelfinger Autobauer", "Daimler-Benz", or simply "Benz".

Besides using features, such as regular expressions and the entity context, our idea is to capture external knowledge in the form of a feature that indicates whether a token is part of a known entity, such as a company name. To this end, we transform various dictionaries into token tries that enable us to efficiently determine whether the analyzed text contains company mentions that are included in the dictionary. Our evaluation focuses on analyzing the impact of using a perfect dictionary and different real-world dictionaries, as well as the effects of different ways to integrate the knowledge contained in the dictionaries on the performance of the NER system. Using our system, we were able to extract 263,846 company mentions from a corpus of 141,970 newspaper articles. In particular, we make the following contributions:

- Creation of an NER system capable of recognizing companies in German texts with a precision of 91.11% and a recall of 78.82%.
- Analysis of the impact of various dictionary-based feature strategies on the performance of the NER.
- A public dataset consisting of 1 000 manually annotated documents containing 2 351 company mentions.

The remainder of this chapter is structured as follows: Section 3.1 introduces the task of named entity recognition with emphasis on the extraction of company names. Section 3.2 discusses related work, while Section 3.3 presents the baseline configuration for the proposed CRF. In Section 3.4, we give an overview of the text corpus and the dictionaries we used. We describe the key data structures and technical aspects of the approach in Section 3.5. We then present our experimental results in Section 3.6. Sections 3.7 and 3.8 provide an overview of the areas *named entity linking* and *relationship extraction* as well as a summarization of the current state-of-the-art in both fields. Both sections are intended to complete the picture of the *text mining* component. We conclude this chapter by giving a summary in Section 3.9. Except for Sections 3.7 and 3.8, the content of this chapter is based on [Loster et al., 2017].

3.1 Named entity recognition for company names

In our context, named entity recognition (NER) defines the task of not only recognizing named entities in unstructured texts but also classifying them according to a predefined set of entity types. The NER task was first defined during the MUC-6 conference [Grishman and Sundheim, 1996], where the objective was to discover general entity types, such as persons, locations, and organizations as well as time, currency, and percentage expressions in unstructured texts. Subsequent tasks, such as entity linking, question answering, or relationship extraction, rely heavily on the performance of NER systems, which often serve as a foundation for these procedures. Motivated by the use case presented in Section 1.2, we highlight the particular difficulties of finding business entities in (German) texts.

Although there is a large body of work on recognizing entities starting from persons and organizations to entities like gene mentions or chemical compounds, research often neglects the detection of more fine-grained sub-categories, such as person roles or commercial companies. In many cases, the "standard" entity classes turn out to be too coarse-grained to be useful in subsequent tasks, such as automatic enterprise valuation, identifying the sentiment towards a particular company, or discovering political and company networks from textual data.

What makes recognizing company names particularly difficult is that in contrast to person names, they are more heterogeneous in their structure. As such, they can be referenced in a multitude of ways and are often composed of many constituent parts, including person and country names, locations, industry sectors, acronyms, numbers, and other tokens, making them particularly hard to recognize. This heterogeneity is expected to be true, particularly for the range of medium-sized to small companies. Regarding examples like "Simon Kucher & Partner Strategy & Marketing Consultants GmbH", "Loni GmbH", or "Klaus Traeger", which all are official names of German companies, one can easily see that they vary not only in length and types of their constituent parts but also in the position where specific name components appear. In the example "Clean-Star GmbH & Co Autowaschanlage Leipzig KG", the legal form "GmbH & Co KG" is interleaved with information about the type of the company (carwash) and location information (Leipzig, a city in Germany). What is more, company names are not required to contain specific constituent parts: the example "Klaus Traeger" from above is simply the name of a person. It does not provide any additional information apart from the name itself, which leads to ambiguous names that are difficult to identify in practice.

Additionally, and in contrast to recognizing named entities in English texts, detecting them in German texts presents itself as an even greater challenge. As pointed out by Faruqui and Padó [2010], this difficulty is due to the high morphological complexity of the German language, making tasks such as lemmatization much harder to solve. Hence, features that are highly effective for English often lose their predictive power for German. Capitalization is a prime example of such a feature. Compared to English, where capitalization of common nouns serves as a useful indicator for named entities, in German *all* nouns are capitalized, which drastically lowers the predictive power of this feature.

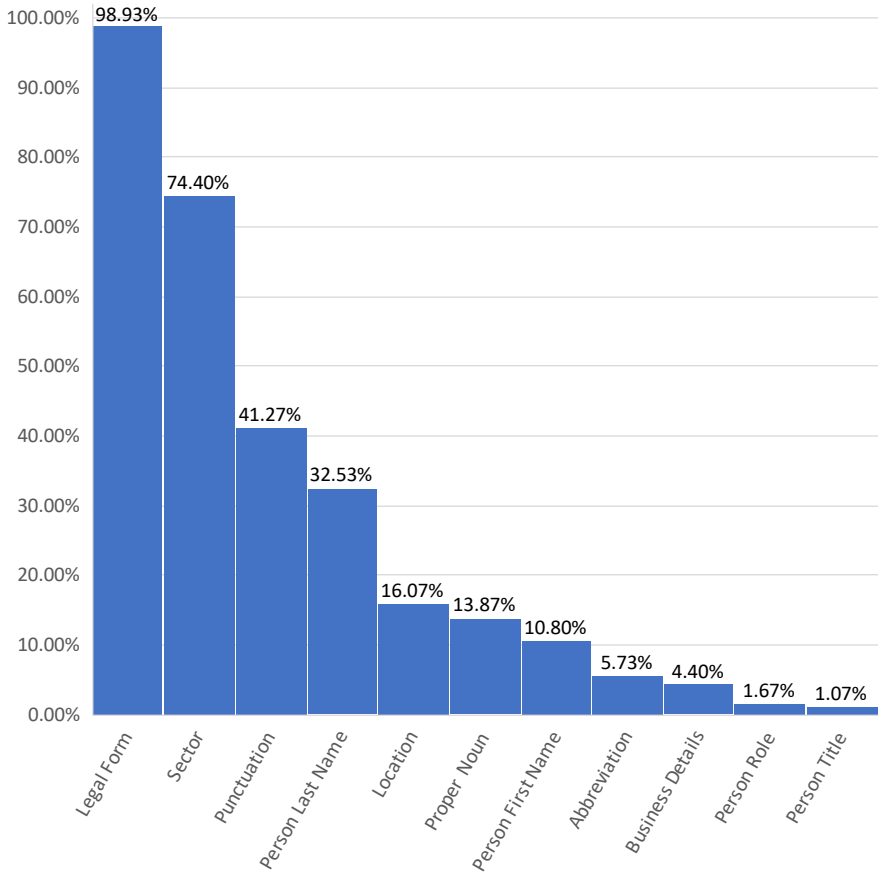

Figure 3.2: Proportion of constituent parts in 1500 manually annotated company names

As part of another research project [Loster et al., 2018a], we investigated the distribution of the constituent parts of company names in more detail. Figure 3.2 represents the result of this investigation as it shows the distribution of the constituent parts of 1500 manually annotated company names. As can be seen, the annotated company names contain information about their legal form in almost all cases, while information about business sectors, punctuation marks, or the last name of persons are rapidly decreasing and generally occur in significantly fewer cases. Looking at the other end of the spectrum, information about a person's title, role, and first name, as well as information on business details and abbreviations, was very rarely available, in fact, well below 10% for almost all the aforementioned parts. In general, the name constituents follow the characteristics of a power-law distribution, where a few common constituents occur very often while others occur only a few times.

3.2 Related work

Since its first appearance on the MUC-6 conference [Grishman and Sundheim, 1996], the problem of named entity recognition (NER) has become a well-established task leading

to many systems and methods that have been developed over time [Nadeau and Sekine, 2007]. Before discussing the differences of our approach to the most related approaches, we give an overview of the related work in general.

Most existing NER systems can be classified into *rule-based* [Chiticariu et al., 2010; Sekine and Nobata, 2004], *machine learning-based* [McCallum and Li, 2003; Zhou and Su, 2002], or *hybrid* systems [Hermann et al., 2014; Srihari, 2000]. While rule-based systems make use of carefully hand-crafted rules, machine learning approaches tend to train statistical models, such as Hidden Markov Models (HMM) [Zhou and Su, 2002] or Conditional Random Fields (CRF) [Lafferty et al., 2001], to identify named entities. Hybrid systems combine different methods to compensate for their shortcomings. They try to incorporate the best parts of the applied methods to reach a high system performance.

Many approaches to the NER problem are based on CRFs. [Faruqui and Padó, 2010; Krishnan and Manning, 2006; McCallum and Li, 2003]. One of the most popular and freely available NER systems for English texts is the Stanford NER system [Finkel et al., 2005]. It recognizes named entities by employing a linear-chain CRF to predict the most likely sequence of named entity labels. While this system shows good performance on English texts, it's performance values decrease when applied to German texts. This effect has also been pointed out by Benikova et al. [2014], who argue that German NER systems are not on the same level as their English counterparts even though German belongs to the group of well-studied languages. This difficulty arises from the fact that the German language has a very rich morphology, making it especially challenging to identify named entities. Besides the already mentioned problem of capitalization, the German language is capable of creating complex noun compounds like "Vermögensverwaltungsgesellschaft" (asset management company) or "Industrieversicherungsmakler" (industry insurance broker), which make the application of traditional NLP methods even harder.

Nonetheless, German NER systems exist, and some were presented at the CoNLL-2003 Shared Task [Sang and Meulder, 2003]. With the participating systems achieving F_1-scores between 48% and 73%, the winning system by Florian et al. [2003] obtained an overall F_1-measure of 72.41% on German texts and 64.62% on recognizing organizational entities. Since the inception of the CoNLL-2003 Shared Task, one of the most successful NER systems for the German language was introduced by Faruqui and Padó [2010]. It reaches overall F_1-scores between 77.2% and 79.8% by using distributional similarity features and the Stanford NER system. Even more recently, additional German NER systems were presented at the GermEval-2014 Shared Task [Benikova et al., 2014]. The GermEval Shared Task specifically focuses on the German language and represents an extension of the CoNLL-2003 Shared Task. The three best competing systems were ExB [Hänig et al., 2014], UKP [Reimers et al., 2014], and MoSTNER [Schüller, 2014]. All of them apply machine learning methods, such as CRFs or neural networks, which leverage dependencies between the utilized features. Additionally, they use semantic generalization features, such as word embeddings or distributional similarity, to alleviate the problem of limited lexical coverage, which, according to Watrin et al. [2014], is triggered by the often insufficient corpus size used in the training phase of statistical models. To summarize the performance of these systems, they operate in the range of 73% to 79% F_1-measure.

Considering the role of dictionaries in building NER systems, Ratinov and Roth [2009] argue that they are crucial for achieving high system performance. The process of automatically or semi-automatically creating such dictionaries from various information sources has been addressed by Kazama and Torisawa [2007] and Toral and Muñoz [2006]. Both works focus on the creation of large dictionaries, also known as *gazetteers*, from open and freely available data sources, such as Wikipedia. The general idea is to establish and assign category labels for each word sequence representing a viable entity by using the information contained in corresponding Wikipedia articles. According to Toral and Muñoz [2006], dictionaries can be separated into two different classes, so-called *trigger dictionaries*, which contain keywords that are indicative for a particular type of entity, and *entity dictionaries*, which are composed of the entire entities. For example, a trigger dictionary for companies would most likely contain tokens for legal-forms, such as "GmbH" (LLC) or "OHG" (general partnership), whereas an entity dictionary would contain the entire representation of the entity itself, e.g., "BMW Vertriebs GmbH". For our approach, we decided to employ entity dictionaries, since there are many openly available data sources from which they can be constructed. Similar to semantic generalization features, features generated from dictionaries aim to mitigate the out-of-vocabulary (OOV) problem resulting from the low lexical coverage of statistically learned models.

Besides our proposed system, other systems also make use of dictionaries to increase their performance. As such, all systems mentioned above use dictionaries at some point in their process [Hänig et al., 2014; Reimers et al., 2014; Schüller, 2014]. Most of the currently existing systems integrate the knowledge contained in dictionaries by constructing features that represent a dictionary lookup. Since each dictionary accounts for a particular type of entity, the constructed feature encodes to which dictionary the word currently under classification belongs and, therefore, implicitly provides evidence for its correct classification. These features are subsequently used in the training process of statistical models, such as CRFs or HMMs.

Cohen and Sarawagi [2004] describe another way of integrating dictionary knowledge into the training process of an NER system. They present a semi-Markov extraction process capable of classifying entire word sequences instead of single words. In doing so, they effectively bridge the gap between NER methods that sequentially classify words and record linkage metrics that apply similarity measures to compare entire candidate names.

While the previously mentioned systems focus on detecting entities belonging to the entity class "organization", which, apart from companies, includes sports teams, universities, political groups, etc., our system, driven by the use cases in Section 1.1, specifically excludes such entities and solely focuses on detecting commercial companies. As dictionaries are often incomplete, they are unable to identify unknown companies, which, under realistic conditions, makes it challenging to recognize company names by the sole use of dictionaries. However, the inclusion of a dictionary in the training process of a CRF classifier combines both, the knowledge contained in the dictionary and the generalization capabilities of the classifier to enable the recognition of unknown company names. In particular, we use a dictionary to annotate already known companies in a preprocessing step, which in turn allows us to construct a feature that can be used in

the training of a CRF classifier. We use dictionaries from different sources and examine their impact on the overall system performance. Additionally, we report on strategies to integrate the domain knowledge provided by the dictionaries into the training process.

Since 2016 much has changed in natural language processing (NLP), which is why I briefly summarize the current state of research. Driven by the success of deep learning techniques in the NLP field, approaches for named entity recognition have also adapted to this trend. In contrast to traditional entity recognition, most modern methods leverage the properties of deep neural networks while also using some form of distributed word representation. Distributed word representations aim to capture the semantic meaning of words based on their surrounding context and have the advantage that they can be trained without supervision on large text corpora. The most common approaches for creating such word representations, also known as word embeddings, are Word2Vec [Mikolov et al., 2013], GloVe [Pennington et al., 2014], and fastText [Bojanowski et al., 2017]. However, a weakness of these models is their inability to correctly capture different word senses. This limitation results from the fact that these models ignore the position of the context words during model training. Thus, each word is mapped to exactly one word vector, which represents a mixture of all word senses for the corresponding word. For example, it becomes impossible to distinguish between the different senses of the word *bowl* as in *super bowl* or *soup bowl*. Recently, this circumstance has led to the replacement of word embeddings created by using one of the above techniques with embeddings created by complex neural language models such as BERT [Devlin et al., 2019], which uses a bidirectional arrangement of the Transformer architecture first introduced in [Vaswani et al., 2017b]. Unlike the previous approaches, these models are designed to take into account the full context of each word, including the position of context words, which enables them to capture the different senses of each word. Although these models can also be trained in an unsupervised manner, the large number of trainable parameters makes their training quite expensive. For this reason, pre-trained models are usually used. Currently, state-of-the-art models for named entity recognition often consist of a combination of neural language models and task-specific neural networks, in our case, a network for named entity recognition. While the accumulated knowledge of the pre-trained neural language models is used to generate more sophisticated word representations, the purpose of the task-specific network is to focus on solving a specific task.

A widely used task-specific network architecture for entity recognition is the BiLSTM-CRF architecture of Huang et al. [2015]. Their architecture combines a bidirectional LSTM (BiLSTM) network with a network layer implementing a conditional random field (CRF). Besides recognizing named entities, this architecture can also be used for other sequence tagging tasks, such as part-of-speech (POS) tagging. The network expects a vector representation of each word in the processed text segment as its input. To learn the sequence tagging, the BiLSTM layer processes the input sequence in both directions from start to end and vice versa. This type of processing allows the network to consider not only information from past but also from future states. For each time step, both past and future states of the BiLSTM layer are passed to the CRF layer, where they are handled together with the tagging information of the processed text segment. Thus, the

CRF layer is responsible for determining the most likely tag sequence for the given text segment.

At the time of writing, one of the best performing models is the one developed by Baevski et al. [2019]. Similar to other neural language models, the training objective is to first hide each word of a given text in order to then predict it using all its contextual words. To this end, they use a self-attention mechanism that acts as a language model and is combined with the previously introduced BiLSTM-CRF architecture, which is used to predict the final tag sequence. Another NER approach uses the popular BERT language model by Devlin et al. [2019] and combines it with only a BiLSTM network omitting the additional CRF layer. Given the enormous progress in NLP, it can be expected that the previously outlined methods will most likely outperform the presented approach.

3.3 Conditional random fields as NER baseline

For the construction of our company-focused NER system, we use the CRFSuite Framework[1] to implement a conditional random field model (CRF). For the baseline configuration of the system, we used various features, such as n-grams, prefixes, and suffixes, that are based on those used in the Stanford NER system [Finkel et al., 2005]. Besides considering different window sizes for each feature, we considered a variety of additional features, for example, a token-type feature, reducing the type of a token to categories like InitUpper, AllUpper, etc., a feature that concatenates different prefix and suffix lengths for each token or features that capture some specific characteristic of German company names. However, these features did not result in additional improvements of our baseline configuration. Experimenting with different feature combinations resulted in a baseline configuration consisting of the following six feature groups for a total of 20 features:

	The	auto	maker	VW	AG	is	now...
words :	w_{-3},	w_{-2},	w_{-1},	w_0,	w_1,	w_2,	w_3,
pos-tags :		p_{-2},	p_{-1},	p_0,	p_1,	p_2,	
shape :			s_{-1},	s_0,	s_1		
prefixes :			pr_{-1},	pr_0,			
suffixes :			su_{-1},	su_0,			
n-grams :				n_0,			

Here, the w symbol encodes the word token features of a text with its subscript marking the position of the token. Thus, w_0 refers to the current token, whereas w_{-1} and w_1 refer to the previous and next tokens, respectively. The symbols p and s represent the part-of-speech and word shape features with analog subscript notation.

For the creation of POS tags, we used the Stanford log-linear part-of-speech tagger [Toutanova et al., 2003]. As the name suggests, the shape feature condenses a given word to its shape by substituting each capitalized letter with an **X** and each lowercase

[1] `http://www.chokkan.org/software/crfsuite/`

letter with an **x**. Thus, the word "**Bosch**" would be transformed into "**Xxxxx**". We also added prefix and suffix features (*pr*, *su*) for the current and previous word. These features generate all possible prefixes and suffixes for the specific word. As the last feature, we include the set n_0 of all n-grams of the current token with n between 1 and the length of the current token. This feature set yielded the best performance metrics for our baseline configuration without adding any external knowledge besides POS tags.

The baseline system achieves an F_1-measure of 80.65%. More detailed performance metrics of the baseline are presented later in Table 3.4, in the context of our overall experiments.

3.4 Corpus & Dictionaries

Before describing our approach in Section 3.5, we introduce and examine the text corpus and different information sources we used for building our dictionaries.

3.4.1 Text corpus

Our evaluation corpus consists of 141,970 documents containing approximately 3.17 million sentences and 54 million tokens. The documents were collected from five German newspaper websites, namely, Handelsblatt, Märkische Allgemeine, Hannoversche Allgemeine, Express, and Ostsee-Zeitung. We intentionally selected not only large national newspapers but also smaller regional ones; we observe that larger newspapers tend to report more about larger companies or corporations, while the regional press also mentions smaller companies due to their locality in the region. By using regional articles in our training process, we intend to increase our chances of discovering small and mid-sized companies (SMEs) in the long tail. We extract the main content from the articles by using jsoup[2] with hand-crafted selector patterns, which give us the raw text without HTML markup. Using our final NER system, we were able to extract a total of 263,846 company mentions from this corpus.

3.4.2 Dictionaries

To build our dictionaries, we used two official information sources: the Bundesanzeiger (German Federal Gazette)[3] and the Global Legal Entity Identifier Foundation (GLEIF), which hosts a freely available company dataset[4]. Additionally, we used DBpedia[5] to account for large businesses and the German Yellow Pages[6] to cover middle-tier and local businesses. To simulate the best-case scenario, we also composed a "perfect" dictionary containing all manually annotated companies from our test set. Finally, our last dictionary consists of the union of all dictionaries except the perfect one. Although the

[2] `https://jsoup.org`
[3] `https://www.bundesanzeiger.de`
[4] `https://www.gleif.org`
[5] `http://wiki.dbpedia.org`
[6] `http://www.gelbeseiten.de`

information sources discussed below contain many different attributes, we use only the company name for the creation of each dictionary.

Bundesanzeiger (BZ). The Bundesanzeiger is the official gazette for announcements made by German federal agencies. Among other things, it contains official announcements from companies of various legal forms, such as corporations, limited liability companies, and others. The role of the Bundesanzeiger and the information it provides are comparable to the U.S. Federal Register. By crawling the BZ company announcements, we obtained 793,974 company names, their addresses, and their commercial register ID.

GLEIF (GL). The Global Legal Entity Identifier Foundation (GLEIF) was founded by the International Financial Stability Board[7] in 2014. It is a non-profit organization set up to aid the implementation of the Legal Entity Identifier (LEI). The LEI is designed to be a globally unambiguous, unique identifier for entities that partake in financial transactions. In this context, the dataset of legal entities assigned with a unique LEI is made available for public use by GLEIF. An entry in the provided dataset is, among other data, comprised of the LEI number, legal name, legal form, and address of a legal entity. At the time of writing, the dataset consisted of 413,572 legal entities from all global countries that have been assigned a LEI. The subset for German legal entities (**GL.DE**) consists of 42,861 entries.

DBpedia (DBP). The DBpedia project is an effort to systematically extract information from Wikipedia and provide it to the public in a structured form [Lehmann et al., 2015]. Structuring the data contained in Wikipedia pages enables us to use query languages, such as SPARQL, to answer complex queries based on data originating from Wikipedia. We queried for the names of all companies contained in the German DBpedia database, yielding a dictionary of 41,724 entries. The resulting dataset contains all companies that have a German Wikipedia page. Thus it predominantly contains German companies as well as some international companies that also possess a German Wikipedia page, such as IBM or Microsoft. We expect that most of the collected company names in this dataset belong to larger companies, as small companies usually do not have a Wikipedia page. Since the extracted names originate from Wikipedia pages, they are very often already in their colloquial form. Also, the dataset contains some additional aliases, such as "VW" for the "Volkswagen AG", which are difficult to generate automatically.

Yellow pages (YP). As a marketing solutions provider, the German Yellow Pages maintains a large company register, which mainly contains information about small and middle-tier businesses. Using the web pages provided by the register, we were able to extract information, such as the company name, address, email address, phone number, and industrial sector for each company listed in the Yellow Pages. The dataset consists of 416,375 company entries.

[7] `http://www.fsb.org/`

	BZ	DBP	YP	GL	GL.DE	PD
BZ	796,389	-	-	-	-	-
DBP	333	41,724	-	-	-	-
YP	14,689	757	416,375	-	-	-
GL	16,420	792	2,166	413,572	-	-
GL.DE	16,370	452	2,130	42,861	42,861	-
PD	62	633	105	50	31	2,351

Table 3.1: Dictionary overlaps using exact match. For instance, 796,389 BZ entries find 333 exact matches in DBP.

	BZ	DBP	YP	GL	GL.DE	PD
BZ	796,389	4,746	114,958	122,308	119,514	4,900
DBP	2,436	41,724	2,049	3,472	1,775	857
YP	38,170	3,141	416,375	7,988	7,741	330
GL	25,419	4,569	6,546	413,572	43,838	504
GL.DE	23,372	1,907	6,128	42,861	42,861	249
PD	232	821	207	248	125	2,351

Table 3.2: Dictionary overlaps using fuzzy match (cosine, $\theta = 0.8$). For instance, 796,389 BZ entries find 2,436 similar entries in DBP.

Perfect dictionary (PD). For evaluation purposes, we manually labeled company mentions in 1,000 documents (see Sec. 3.6.1 for details). The perfect dictionary contains exactly the 2,351 manually annotated company names from our training and test set. Because of their origin, the company names contained in this dictionary are already in their colloquial form. Using this dictionary, we were indeed able to correctly identify all companies in our test set. Furthermore, this dictionary enables us to simulate the best-case scenario in which the dictionary is composed of all companies occurring in our test set.

All aforementioned dictionaries contain large sets of German company names, so we expect them to overlap. To gain a better understanding of our dictionary's coverages, we computed their mutual containment. We calculated the overlaps using exact match and a fuzzy match. The latter constitutes a more realistic matching scenario accounting for typos and other noise. For computing the matches, we applied the method described by Okazaki and Tsujii [2010]. Summarizing their approach, the authors compute the similarity between two strings by splitting them into n-grams and using similarity measures like Dice, Jaccard, or cosine similarity to determine their similarity using a threshold α. For our calculations, we chose a trigram tokenization of the strings and cosine similarity as our metric. We calculated the fuzzy overlaps using different thresholds for θ and found experimentally that a value of 0.8 performed best on our data.

The pairwise overlaps are shown in Table 3.1 for exact matches and Table 3.2 for fuzzy matches. Surprisingly, even in the case of fuzzy overlaps, the highest overlap was only 11.24%, namely, between the BZ and the GL dictionary. All other overlaps were below this value, except in cases where they were contained in each other (GL.DE$\subset$GL). The exact matching overlaps scored even lower, with a maximum overlap of 1.37%.

We identified three possible reasons for these low overlaps. The first and most obvious reason is that our quite simplistic fuzzy matching is not sufficient to recognize many correct matches. Secondly, each of the dictionaries favors a different kind of company and company size. For example, the DBpedia dictionary contains mostly colloquial names, whereas the Bundesanzeiger refers to companies using their full legal name. Finally, the dictionaries were crawled at slightly different points in time; hence some may contain companies that no longer exist and are thus missing from the other dataset. As a consequence, we created an additional dictionary that combines all the mentioned dictionaries into one as well as several dictionary versions used to evaluate the impact of different dictionaries on the training process of the classifier:

All dictionaries (ALL). This dictionary contains the union of all company names from all dictionaries, except the perfect one. When merging the individual dictionaries, similar company names were not consolidated (no fuzzy matching), so that all name variations were preserved. In total, it comprises 1,713,272 company names.

Dictionary versions (VER). To evaluate the impact of different dictionary versions on the performance of the CRF model, we also generated several dictionary versions that correspond to the rows in table 3.4. In total, we created three different dictionary versions for the Bundesanzeiger, GLEIF, GLEIF(DE), Yellow Pages, and DBpedia. The first dictionary version contains the original company names obtained from the crawled sources. The second version, marked with "+ Alias", additionally includes all aliases generated by the process described in Section 3.5.1. The last version, marked with "+ Alias + Stem", also incorporates a stemmed version of each company name and all its generated aliases. We excluded the perfect dictionary from the alias generation process since it contains the manually tagged colloquial company names. Hence, the approximation of colloquial company names through alias generation is not necessary.

3.5 Company recognition using dictionaries

Named entity recognition (NER) is a sequence labeling task that aims to sequentially classify each word in a given text as belonging to a specific class, e.g., person or company. As mentioned, we make use of the CRFSuite Framework to construct our NER system. First, we describe our alias generation process, which extends the given dictionaries, in Section 3.5.1. Then, Section 3.5.2 describes how we create the dictionaries and how we efficiently integrate the contained domain knowledge into the training process of the CRF.

3.5.1 Alias generation

Unfortunately, company names acquired from web sources contain noise, such as country names, legal forms, and other spurious terms. That is, they often differ significantly from their colloquial names. Here the "colloquial name" is to be understood as the name by which a company is commonly referred to in text. For example, while "Dr. Ing. h.c. F. Porsche AG" represents the official company name of the automobile manufacturer,

we most often refer to the company by its colloquial name, which is simply "Porsche". Assuming that articles mention companies more frequently by their colloquial name than their official name, it becomes necessary to automatically derive such alternative names, in the following referred to as *aliases*, from a company's official name.

Regarding the alias generation, special attention should be paid to the fact that one company often possesses more than one alias. Considering again the example from above, the company Porsche has at least four valid and common aliases, namely "Dr. Ing. h.c. F. Porsche AG", "Ferdinand Porsche AG", "Porsche AG", or just plain "Porsche". Furthermore, there are a number of non-trivial aliases that are particularly difficult to anticipate by using an automated process. For example, the automobile manufacturer "Volkswagen" is also referred to as "VW" or even "die Wolfsburger", referring to the city of Wolfsburg, in which Volkswagen's headquarters is located.

Our alias generation process consists of the following five steps, using the example of "TOYOTA MOTOR™USA INC.", which is depicted in Table 3.3. Each of the Steps 1–4 yields one new alias for the currently processed company name, resulting in four aliases per name. Note that some of the four aliases are identical and identical copies are removed. The fifth and final stemming step adds another five aliases by stemming the company name itself and all previously generated aliases. This means that a maximum of nine aliases could be generated by applying the five processing steps to a given company name.

Step	Action	Example
0	Original name	TOYOTA MOTOR™USA INC.
1	Removal of legal form designations	TOYOTA MOTOR™USA
2	Removal of special characters	TOYOTA MOTOR USA
3	Normalization	Toyota Motor USA
4	Country name removal	Toyota Motor
5	Stemming of company names	no change

Table 3.3: Alias name generation process exemplified by Toyota Motors USA

1 & 2: Legal form & special character cleansing. We start to infer the aliases by using a rule-based approach that relies on regular expressions to strip away a company's legal form. The regular expressions we use are derived from the description of business entity types found on Wikipedia[8]. The derivation process consists of looking at the business entity types for selected countries and manually creating regular expressions that are able to match the legal forms of the selected countries. We chose the countries based on the most frequent legal forms occurring in our datasets. For example, the business entity types we used to derive the regular expressions for Germany include "Gesellschaft bürgerlichen Rechts (GbR)", "Kommanditgesellschaft (KG)", or "Offene Handelsgesellschaft (OHG)". Step 2 further cleanses the names by removing various special characters, such as "®", "TM", and parentheses.

[8] `http://en.wikipedia.org/wiki/Types_of_business_entity`

3: Normalization of company names. In Step 3, we tokenize the company name and "normalize" each token that has a length greater than four characters and is written in all capital letters. This normalization step consists of first lowercasing and then capitalizing each token that matches the aforementioned criterion. As an example, the normalization step would transform "VOLKSWAGEN AG" into "Volkswagen AG" and "BASF INDIA LIMITED" into "BASF India Limited".

4: Country name removal. During the fourth step, we remove all country names appearing in a company's name using a list of country names and their translations to other languages[9]. Although, in general, more intricate transformation rules can be created, we found that the ones presented here are sufficient for our purposes.

5: Stemming. Using an exact matching strategy to match company names that deviate only slightly from the aliases stored in a dictionary can produce suboptimal results. For example, consider the name "Deutsche Presse Agentur", which can also occur as "Deutschen Presse Agentur", depending on the grammatical context. To mitigate these matching issues, we generate additional aliases by stemming each token in a company's name and all its generated aliases using a German Snowball Stemmer[10]. Using this strategy, we generate the alias "Deutsch Press Agentur", which can, in turn, be used to match both representations of the aforementioned name. Adding the resulting aliases to a dictionary increases the chances to match a slightly varying company name to an entity contained in the dictionary while using an exact match strategy.

3.5.2 Dictionary and feature construction

To create the dictionary, we decided to use entity dictionaries containing whole entity names and their aliases instead of trigger dictionaries that consist of simple keywords indicating the presence of an entity. Using this approach simplifies dictionary creation, as it is reduced to the simple addition of company names and their aliases. In contrast, the creation of trigger dictionaries first requires the creation of keywords out of company names, either by manually creating extraction rules or by using algorithms, such as TF–IDF. [Manning et al., 2008].

To make use of the information contained in a dictionary during the CRF training process, we create a feature that encodes whether the currently classified token is part of a company name contained in one of the dictionaries. To efficiently match token sequences in a text against a specific dictionary, we tokenize a company's official name as well as all its aliases and insert the generated tokens according to their sequence into a trie data structure. During insertion, we mark the last inserted token of each token sequence with a flag denoting the end of the inserted name. In this manner, we insert all company names into the token trie. Figure 3.3 shows an excerpt of such a token trie after inserting several company names. After its creation, the token trie functions as a

[9] https://en.wikipedia.org/wiki/List_of_country_names_in_various_languages
[10] http://snowball.tartarus.org/algorithms/german/stemmer.html

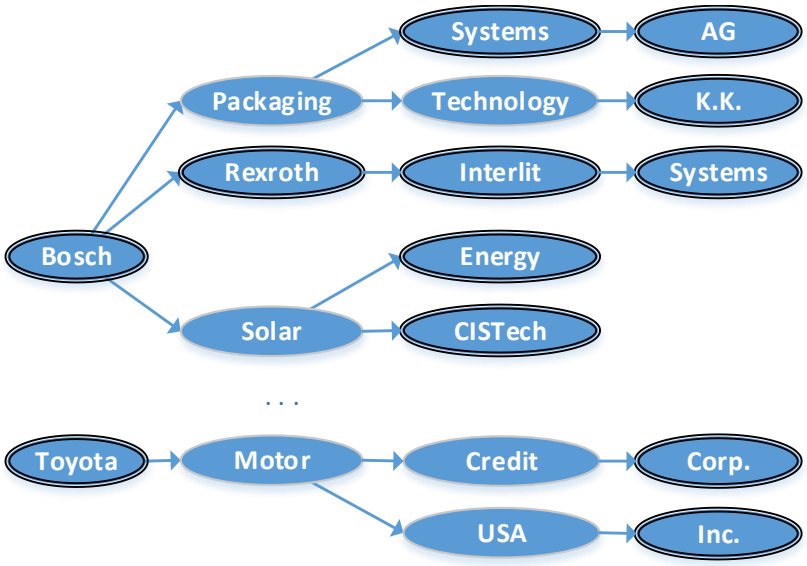

Figure 3.3: An example of a token trie. Double circles indicate final states.

finite-state automaton (FSA) for efficiently parsing and annotating token sequences in texts as companies.

We perform the matches in a greedy fashion by always choosing the longest possible match. The outlined approach is crucial when using entity dictionaries. In contrast to trigger dictionaries which contain only single tokens, entity dictionaries mark the entire token sequence representing an entity (e.g., "Volkswagen Financial Services GmbH"). They, therefore, need to keep track of their matching state to determine if a match occurred.

3.6 Experiments

In this section, we describe our experiments and present the results generated by our system. In Section 3.6.1, we present the setup of our experiments by introducing our test data, annotation policy, and the validation method used. Our goal is to evaluate the effect of using dictionaries for NER. Section 3.6.2 presents the evaluation results of our baseline system *without* the use of dictionaries, as well as a comparative evaluation against the Stanford NER system. The results of using *only* the generated dictionaries to discover companies in our test data are discussed in Section 3.6.3. Section 3.6.4 then shows and discusses the results of *integrating* the domain knowledge contained in the dictionaries into our baseline system. Finally, we discuss the case of using the *perfect* dictionary in Section 3.6.5. The performance results in terms of precision, recall, and F_1-measure for all analyzed system configurations can be found in Table 3.4.

3.6.1 Experimental setup

For the evaluation of our system, we randomly selected 1,000 articles across all sources for which we could confirm that they contain at least one company mention. We manually

annotated these articles by assigning the company-label to each token representing a company mention in the text. We used a very strict annotation policy for tagging the company names in each document; the goal of the policy is to distinguish between mentions referring to a company and mentions referring to related products, persons, or brands. To this end, we considered the context of a company mention to identify a "real" company like BMW, as opposed to a mention appearing as part of another phrase, such as BMW X6, which we did not annotate. In this case, the token X6 identifies the token BMW as part of a product mention. During the annotation process, we discovered and marked 2,351 company mentions in the chosen documents, each consisting of one or more tokens. Links to the news articles of this corpus, together with titles and labeled entities, are available at the project's website[11].

To evaluate our system's performance, we performed a ten-fold cross-validation by splitting the annotated documents into ten folds, each fold containing 900 articles for training and 100 articles for testing. For each fold, we measure precision, recall, and F_1-measure. The overall performance of the trained model is calculated by averaging the performance metrics over all folds.

We conduct a series of experiments to evaluate our system as well as the impact of different dictionary versions on the system's performance. The results of all experiments are given in Table 3.4. First, we compared the performance of our baseline system to the Stanford NER system, as described in Section 3.6.2. Subsequently, we conducted multiple experiments to evaluate the impact of different dictionary versions on the performance of the generated CRF model. To this end, we used the different dictionary versions (VER), as introduced in Section 3.4.2. We evaluated each of the generated dictionary versions in two scenarios, illustrated by the two columns "Dict only" and "CRF" in Table 3.4. In the "Dict only" scenario, described in Section 3.6.3, we use each dictionary on its own to identify the companies contained in our test set. The "CRF" scenario is discussed in Section 3.6.4, where we focused on integrating the different dictionary versions into the training process of the CRF and use the generated model to discover company names.

3.6.2 No dictionaries

We started our experiments by evaluating the baseline configuration introduced in Section 3.3. Using the basic features mentioned there, we were able to achieve a performance of F_1=80.65% without adding any additional domain knowledge to the system (see Table 3.4 for details).

We additionally compare our base system with the Stanford NER system [Finkel et al., 2005], which we used to train a new model on the same training and test documents as our system. To this end, we used the configuration suggested on their website[12]. Using the resulting model, the Stanford system achieves a slightly better F_1-score of 81.76%. This result is 1.36 percentage points below the precision and 2.68 percentage points above the recall metrics of the baseline, which is caused by slight variations in the features used.

[11] `https://hpi.de/en/naumann/projects/repeatability/datasets/corpus-comp-ner.html`
[12] `http://nlp.stanford.edu/software/crf-faq.shtml`

Dictionary	Dict only			CRF		
	P	**R**	**F_1**	**P**	**R**	**F_1**
Baseline (BL)	–	–	–	91.38%	72.25%	80.65%
Stanford NER	–	–	–	90.02%	74.93%	81.76%
BZ	**74.23**%	3.23%	6.15%	90.90%	75.79%	82.63%
BZ + Alias	16.20%	39.27%	22.91%	91.09%	75.74%	82.63%
BZ + Alias + Stem	6.38%	39.77%	10.98%	90.93%	76.03%	82.78%
GL	34.61%	2.92%	5.37%	90.91%	75.76%	82.62%
GL + Alias	41.71%	50.55%	45.67%	90.78%	77.43%	83.55%
GL + Alias + Stem	18.79%	50.77%	27.39%	90.83%	77.07%	83.36%
GL.DE	68.91%	1.17%	2.29%	90.92%	75.82%	82.66%
GL.DE + Alias	55.78%	21.58%	31.02%	90.97%	76.89%	83.30%
GL.DE + Alias + Stem	39.54%	21.58%	27.85%	90.83%	77.07%	83.36%
YP	16.11%	15.01%	15.53%	91.02%	75.88%	82.73%
YP + Alias	18.34%	21.26%	19.68%	90.92%	75.89%	82.67%
YP + Alias + Stem	7.05%	21.34%	10.58%	90.29%	75.92%	82.72%
DBP	63.13%	43.61%	**51.51**%	**91.25**%	78.54%	84.40%
DBP + Alias	44.18%	53.38%	48.29%	91.11%	**78.82**%	**84.50**%
DBP + Alias + Stem	29.79%	53.47%	38.24%	91.14%	78.76%	84.48%
ALL	20.07%	71.56%	31.33%	90.60%	77.36%	83.43%
ALL + Alias	20.11%	71.80%	31.39%	90.61%	77.33%	83.41%
ALL + Alias + Stem	8.15%	**72.16%**	14.64%	90.94%	76.93%	83.32%
PD (perfect dict.)	81.67%	100.00%	89.90%	94.68%	96.47%	95.56%
PD (perfect dict.) + Stem	81.67%	100.00%	89.90%	94.68%	96.47%	95.56%

Table 3.4: Results of including different dictionaries into the CRF training process

3.6.3 Dictionaries only

Next, we used the generated dictionaries on their own to discover the company mentions contained in our test set, as described in Section 3.5.2. The left, "Dict only" part of Table 3.4, represents the results of these experiments. The highest precision of 74.23% could be achieved by using the Bundesanzeiger dictionary in its original form. Using the DBpedia dictionary in its original form resulted in the highest F_1-measure value of 51.51%. It is worth noting that using this dictionary in combination with our baseline system and the generated aliases also yielded the best overall results, as described in the following section. Not surprisingly, the highest recall of 72.16% was achieved by combining all dictionaries (except PD) that include the generated aliases and the stemmed name versions.

To understand the impact of alias generation, we compare the average recall of all basic dictionaries, which is 22.92%, with the average recall of all extended dictionaries, which is 42.97% (data not shown). The difference of 20,06 percentage points is sufficiently high to justify the use of aliases in principle. Analogously, we analyzed stemming. The average improvement caused by using the dictionaries that include aliases as well as the stemmed names accounted for another increase of 0.21%. However, the improvements in recall are accompanied by an average decrease in precision of 13.46% from the no-aliases to the aliases version, and a further decrease by 14.44 percentage points to a total

decrease of −18.28% when including the stemmed versions. In summary, we suggest the use of aliases but refrain from including stemmed company names in a dictionary.

In addition, we experimented with a dictionary that contained only the company names and their stemmed versions, but no aliases, to assess the impact of stemming on the dictionary-only approach. Here, the precision decreased by 18.94 percentage points, while the recall increased only by 0.08 percentage points (not shown in Table 3.4). Hence, we conclude that the stemming of company names has a negative impact on the precision of the dictionary-only approach and does not significantly improve recall.

When averaging over all the different dictionary versions (without PD), we arrive at an overall performance of 32.39% precision and 36.36% recall. Considering these metrics, it becomes clear that a dictionary-only approach is not sufficient for discovering company names in textual data.

Regarding the perfect dictionary, it is interesting to see that while a recall of 100% could be achieved, the precision reached only a maximum of 81.67%, which is owed to false positives. These are mostly of the form mentioned earlier, where a company name is part of a product name or role description (the VW executive was ...). We expect such errors to be eliminated by the combination with the CRF approach, which makes use of a terms' context.

3.6.4 Combining dictionaries and CRF

We now discuss the results achieved by combining the domain knowledge contained in the dictionaries and the CRF training process. Overall, we were able to improve the performance over the no-dictionary and the dictionary-only approaches, regardless of which dictionary we used. Regarding the right column of Table 3.4, we achieved the best results in recall and F_1-measure by using the dictionary generated from DBpedia, including the generated aliases (DBP + Alias). Using this dictionary, the system was able to reach an F_1-score of 84.50% with precision and recall values of 91.11% and 78.82%, respectively. By combining the colloquial names already contained in the DBpedia dictionary with the additionally generated alias names, we are able to match more companies than with any of the other dictionaries, explaining our high recall. Interestingly, the initial intuition that combining all dictionaries into one would result in the best performance of our system turned out not to be true. A more concise dictionary, such as DBpedia, yields slightly better results.

As we have done in the previous section, we calculated the average change in precision, recall, and F_1-measure. Table 3.5 shows the average change in performance for gradually evolving our baseline system by including the different dictionary versions. We calculated these values to determine which of the extension steps described in Section 3.5.1 had the largest impact on system performance. As can be seen, the average change in performance increases significantly, moving from the baseline system to a system that uses additional domain knowledge by integrating the basic dictionary version without aliases or stemming. Using additional domain knowledge, the system's precision slightly decreased by 0.45 percentage points, whereas recall and F_1-measure improved on average by 4.28 and 2.43 percentage points, respectively.

Transition	Avg. P	Avg. R	Avg. F_1
BL $\longrightarrow$ BL + Dict	−0.45%	+4.28%	+2.43%
BL + Dict $\longrightarrow$ BL + Dict + Stem	+0.05%	−0.06%	−0.09%
BL + Dict $\longrightarrow$ BL + Dict + Alias	−0.02%	+0.49%	+0.26%
BL + Dict + Alias $\longrightarrow$ BL + Dict + Alias + Stem	−0.09%	−0.05%	−0.01%

Table 3.5: Performance change for different dictionary versions, averaged over all dictionaries except PD

Using the dictionary versions containing the generated aliases for each company name, the system gained, on average, another 0.26 percentage points in F_1-measure. With respect to average precision and recall, the recall increased by 0.49 percentage points, while precision slightly decreased by 0.02 percentage points. Due to the alias generation process that condenses a given company name according to the rules described in Section 3.5.1, we were able to increase the recall while at the same time sustaining precision: we achieved a maximum increase of 6.57 percentage points for recall while the precision decreased only slightly by 0.28% using the DBpedia dictionary including generated alias names. The largest increase of 3.85 percentage points in F_1-measure was also recorded while using the same dictionary. The results suggest that by further improving the alias generation process, it should be possible to increase the recall while sustaining high precision.

Regarding dictionaries containing the stemmed version of the original company names and their aliases, we conclude that stemming has only a limited impact; the results produced by including stemmed names are not significantly better. For a dictionary version that included only the company names and their stemmed version, the improvements were so low or even negative, that we report only on the average change of using this dictionary in Table 3.5. As it turned out, the reduction of company names to their stemmed form accounts only for a very limited number of cases. For instance, the airline Lufthansa can be referred to as "Deutsche Lufthansa" or "Deutschen Lufthansa", depending on the grammatical context. By using the common stemmed version ("Deutsch Lufthansa") of these two aliases, it is possible to match both company names. For company names, however, this approach did not translate into performance improvements.

Because the dictionary feature might add a bias towards labeling known tokens as a company, we also examined how many novel named entities we find, i.e., ones that are not already included in the dictionary. For this experiment, we used each test set in our 10 folds, each consisting of 100 documents not used during the corresponding model training. Using the DBpedia, including aliases model trained on the remaining 900 documents of each fold, we were able to discover, on average, 328 company mentions. On average, 45.85% ($\approx$ 150 companies) of the discovered companies were already included in the dictionary, whereas the remaining 54.15% ($\approx$173) were newly discovered. This shows that although the dictionary feature adds a bias towards already known companies, the model is still able to generalize to entities that are not part of the used dictionary.

3.6.5 Perfect dictionary

To simulate a scenario in which the dictionary can be used on its own to identify the company names in a given text, we use the perfect dictionary. As already mentioned in Section 3.4, the perfect dictionary consists of all manually annotated company mentions from our test and training sets.

Although using this dictionary yields the highest scores for precision, recall, and F_1-measure, the F_1-measure does not reach 100%. The reason for this behavior can be explained by our strict annotation policy. By using this annotation scheme, it becomes hard for the algorithm to avoid producing false positives. Consider recognizing the airline Boeing in the mentions "Boeing" and "Boeing 747". In both cases, "Boeing" would be recognized as a company, producing one true positive and one false positive. Hence, a drawback of our system is that the dictionary feature introduces a bias towards companies contained within the dictionary, inducing some false positives if the dictionary feature turns out to be wrong. This problem translates to all other dictionaries that we use. Therefore, we argue that even under ideal circumstances where the dictionary contains all entities that we want to discover, it is not possible to sustain a high precision value by using the dictionary on its own.

Nonetheless, as can be seen by comparing the results in Table 3.4, using dictionaries to incorporate domain knowledge into the CRF method yields superior results over using them on their own to recognize company names. Considering the average precision, recall, and F_1-measure, the combination of dictionaries and CRF performs significantly better than the pure dictionary approach described in Section 3.6.3.

3.7 Named entity linking

To give a complete picture of the text mining component, we have dedicated this section to the topic of *entity linking*. First, we give an overview of entity linking and then summarize the current state of the art. As discussed in the previous section, the NER component of the *text mining* pipeline focuses on the discovery of mentions in textual documents that have a high likelihood to refer to real-world entities. In this sense, NER often acts as a precursor for entity linking. Unfortunately, as natural language is inherently ambiguous, it is often unclear which specific entity is referred to in a given text. For example, the mention "VW" could potentially refer to either the car manufacturer "Volkswagen" or to the company "Vorwerk", a manufacturer of household appliances. The distinct assignment of ambiguous entity names to their corresponding entries in a given knowledge base is known in the literature as *named entity linking* (NEL), *named entity disambiguation* (NED), or in short *entity linking* (EL), *entity disambiguation* (ED). While some research uses the terms *entity linking* and *entity disambiguation* synonymously, in this thesis, we consider entity disambiguation as a subcomponent of an entity linking approach. A more detailed description of the building blocks of entity linking systems is given in Section 3.7.2. Considering the previous example, the mention "VW" would most likely be mapped to the knowledge base entry of the *Volkswagen AG*, since "VW" represents a commonly used acronym for the company Volkswagen and is usually

not associated with the company Vorwerk. To correctly resolve such ambiguities, entity linking approaches rely on a wide variety of features that are either derived from the entity mention itself or its context. A list of the most frequently encountered features used for entity linking is provided by Shen et al. [2015]. Entity linking plays a crucial role in many applications:

Information extraction. Information Extraction (IE) focuses on discovering relevant information in large quantities of unstructured and semi-structured data, making it accessible in a machine-readable format. When applied to the use case outlined in Section 1.2, a collection of company information is obtained together with information about their relationships. However, to make use of the discovered information, it is necessary to establish a connection to an unambiguous representation of a real-world entity. To establish this assignment, entity linking methods are used. Once an appropriate assignment has been made, additional information from the linked knowledge base entry can be used to derive a more fine-grained entity classification or build a network of co-occurring entities. A detailed discussion of information extraction and its relation to Semantic Web technologies can be found in [Martínez-Rodríguez et al., 2020].

Information retrieval. In recent years, there has been an increasing shift in the focus of Information Retrieval (IR) away from keyword-based document searches towards entity-aware semantic searches. Search engine users increasingly make queries about specific entities and expect to receive information about the entity in question, rather than just documents that mention it. This need has led to the development of *entity retrieval*, which focuses on satisfying an entity-centric information need, relying heavily on entity linking techniques [Balog, 2018; Ensan and Du, 2019; Hasibi et al., 2016]. In this context, entity linking enables both the resolution of ambiguous search terms and the identification of entities that are of interest to the user, allowing the generation of targeted responses. This facilitates not only a direct response with respect to specific entities or an improved retrieval of documents containing direct matches, but also the retrieval of documents containing related entities and concepts. Thus, entity linking represents a key component of semantically aware search engines.

Knowledge base construction. When creating new or extending existing knowledge bases, entity linking ensures that a mapping between the discovered information and the corresponding entries in the knowledge base is established. The unambiguous assignment of an entity mention to a knowledge base entry can then be used to integrate newly found information, such as an extracted fact or a discovered relationship. Based on the established assignment, this extension can be made without negatively affecting the data quality of the underlying knowledge base, enabling its structured growth. There are several works that discuss the role of entity linking in the process of knowledge base construction. [Lin et al., 2012, 2020; McNamee et al., 2012]

Question answering. Virtual assistants, such as Apple's Siri, Amazon's Alexa, and Google's Assistant heavily rely on entity linking techniques as they play a crucial role

in question answering [Dubey et al., 2018; Echegoyen et al., 2019; Li and Shi, 2016]. To correctly answer a question, it is necessary to determine the intention of the question, identify the relevant entities for the question, and establish an assignment of these entities to the corresponding entries in the used knowledge base. Only by correctly resolving the core elements of a question is it possible for the underlying question answering system to derive the correct answer.

3.7.1 Challenges

There are three major challenges that need to be addressed during the development of entity linking approaches:

Name diversity. Unlike programming languages, natural language is highly ambiguous, which leads to real-world entities being referred to in many ways. Thus, entities can be referenced by their official full name, abbreviations, colloquial names, and a variety of aliases. These name variations considerably increase the difficulty of finding the correct assignment to a corresponding knowledge base entry. As such, the names "Continental Teves AG & Co.oHG", "Continental Teves AG", "Continental Teves", "Conti-Teves", "Teves", or "Conti" all represent common name variations of the German company Continental.

Entity ambiguity. In addition to name variations, a mention can also refer to multiple different knowledge base entries. As an example, the mention "Amazon" can refer to the multinational company "Amazon.com, Inc." but also to the "Amazon River" or a "tribe of female warriors". Finding the right match amongst these potential candidates constitutes another challenge. To address this issue, many linking approaches use contextual information of the entity mention to identify the most likely mapping. Concretely, they make use of the words occurring within a specific window around the entity mention (local context) or leverage word co-occurrences at the document level (global context). Some approaches also combine local and global context information to improve linking performance.

Unlinkable entities. For some mentions, it is impossible to find an assignment as there exists no corresponding entry in the used knowledge base. It is, therefore, impossible to link a company mention, such as "MCI Inc.", to an entry in the knowledge base if no corresponding entry exists. For example, this can be the case if the processed documents are newer than the information stored in the knowledge base. In such cases, it is further important to ensure that an entity linking approach avoids a possible misassignment caused by a superficial resemblance to some knowledge base entry. It should, therefore, be avoided that a reference such as "MCI Inc." is incorrectly assigned to the entry "Medical Council of India (MCI)" instead of "Motor Coach Industries International Inc." simply because the two entries coincide regarding their acronyms.

3.7.2 Building blocks of entity linking systems

As outlined by Shen et al. [2015], entity linking approaches can be broken down into two major components: **(i)** *candidate generation (CD)* and **(ii)** *entity disambiguation (ED)*, whose functionality is described below.

Candidate generation, also referred to as *candidate selection* or *candidate classification*, defines the task of selecting a subset of entities from the knowledge base $\mathcal{E}_c \subseteq \mathcal{K}$ to which a processed mention m is most likely referring to. Since m can potentially correspond to any knowledge base entry, the main objective of this step is to reduce the number of entities that need to be examined in the following disambiguation phase to those most likely resulting in a correct match. One of the most commonly used candidate generation techniques exploits the information contained in Wikipedia and builds a dictionary that maps a variety of names occurring on Wikipedia pages to a collection of alternative names. Other approaches generate potential link candidates by extracting specific elements that are collocated with existing mentions. For example, these methods create new link candidates by extracting acronyms (Volkswagen (VW) ⇒ VW) or name expansions (TUM (Technische Universität München) ⇒ Technische Universität München) that occur in the vicinity of a mention. A more detailed overview of common approaches for candidate generation can be found in [Shen et al., 2015]. The result of this phase is a set of link candidate pairs $C = \{(m, e)\}$, consisting of the mention m and an entity $e \in \mathcal{E}_c$ to which m potentially refers.

Entity disambiguation. The entity disambiguation component is at the core of an entity linking approach. Its task is to estimate the likelihood that a particular link candidate pair $(m, e) \in C$ generated in the *candidate generation* step is the correct disambiguation of the entity mention m to the representation of the real-world entity e. For example, the pair (VW, Volkswagen AG) should receive a high score, while the pair (VW, Vorwerk) should receive a low score. The result of this estimation is a set of scored link candidates $\mathcal{S}_c = \{(m, e, s)\}$, where s is the score indicating the likelihood of e being the correct disambiguation of m. By sorting the link candidates in $\mathcal{S}_c$ according to their score, candidates that most likely correspond to a correct assignment of mention m to knowledge base entry e receive the highest score and can, therefore, be considered correct. Motivated by this approach, *entity disambiguation* is also referred to as *entity ranking*. As summarized by Shen et al. [2015], this component has been the subject of intensive research so that many different approaches have accumulated over time.

The most common methods of entity disambiguation can essentially be divided into supervised and unsupervised approaches. While supervised methods rely on annotated training data to learn a ranking function, unsupervised methods use techniques mostly from the field of information retrieval that do not require annotated training data. The proposed supervised methods range from binary over probabilistic to graph-based and ensemble approaches. In this context, a large number of different features were examined that were used to train these methods. Most of these features refer either to the discovered mention itself or additionally to the textual context in which the mention occurred. A detailed summary of the most common entity disambiguation techniques

and the features used in this context can be found in Shen et al. [2015]. As in the case of named entity recognition, the enormous impact of deep learning techniques on the field of natural language processing (NLP) has resulted in the latest entity linking approaches being largely based on neural networks and deep learning techniques. The next section discusses the current state-of-the-art in entity linking and briefly summarizes the latest developments.

3.7.3 Current state of the art

The influence of deep learning techniques on natural language processing has also transcended to most current entity linking approaches. Therefore, many of the current state-of-the-art linking approaches make use of deep learning techniques.

We start with the approach of Yamada et al. [2016], who leverage the representation learning capabilities of deep neural networks to embed words and entities into one unified vector space. Their method is based on the skip-gram model of Mikolov et al. [2013], which learns the semantic representation of a particular word by predicting its surrounding context. Yamada et al. extend the skip-gram model by including a *knowledge graph* and an *anchor model*. While the knowledge graph model is essentially used to learn the relationship degree between two knowledge graph entities, the anchor model's goal is to predict the context words of a linked entity mention occurring in an article. The combination of these individual models results in a vector space in which words and entities that semantically resemble each other are projected in close proximity to each other. Apart from largely eliminating the need for manual feature engineering, the resulting embedding vectors can be used to (i) construct context vectors of entity mentions and (ii) calculate a coherence score of the entities to be linked at the document level. The idea of the coherence score is based on the assumption that a linking decision concerning a particular mention should be carried out according to other linking decisions taken in the same document. All the techniques described below utilize embedding techniques whose rationale is in many aspects similar to that introduced by Yamada et al. [2016].

The disambiguation strategy proposed by Yamada et al. [2016] uses the previously learned embeddings to generate a local context vector from the context words surrounding an entity mention as well as a globally calculated coherence score for each possible link candidate. Both the local context vector as well as the global coherence score are then used as features in training a *gradient boosted regression tree* (GBRT) that learns to arrange the candidate pairs consisting of entity mention and knowledge base entry according to their likelihood of being the correct linking decision. In a subsequent paper, Yamada et al. [2017] create the embeddings based on full texts instead of single words and replace the GBRT with a multilayer perceptron.

Like Yamada et al. [2016], Ganea and Hofmann [2017] also project words and entities into a common vector space by training entity embeddings. To this end, they start with pre-trained Word2vec [Mikolov et al., 2013] embeddings and extend them by adding an additional vector for each entity mention. Driven by the assumption that only certain context words play a role in the disambiguation of an entity, they use a neural attention mechanism to get a neural network to focus on the words most relevant for the disam-

biguation. Their proposed linking approach consists of a combination of a local neural attention model and a conditional random field (CRF) [Lafferty et al., 2001] model operating at the document level. They train both models in an end-to-end fashion using loopy belief propagation (LBP) as presented in [Domke, 2013].

The work of Le and Titov [2018] revolves around the idea of leveraging relationships between the entity mentions in a document to improve the linking quality. They point out that besides coreferences, other relationship types can be exploited during the linking process. Their approach is based on the work of Ganea and Hofmann [2017] and extends it by including relationships between the individual entity mentions of a document into the global CRF-based coherence model. To avoid the manual annotation of relationships between the entity mentions of a given text, they model them as latent variables that are learned during model training in an end-to-end fashion.

One of the most recent and most successful entity linking approaches is OpenAI's DeepType system by Raiman and Raiman [2018]. The core idea behind DeepType is to support the linking process by providing type information about the linked entities to the disambiguation process. To this end, DeepType is able to automatically derive a type system from a given knowledge base, such as YAGO or Wikidata, which is then used during the training and inference phase of a neural network. With an F-measure of up to 94.88%, DeepType is one of the best performing entity linking systems available to date.

3.8 Relationship extraction

Similar to the previous section, we dedicated this section to relation extraction, which completes the picture of the text mining component. We proceed analogically to the previous section by first giving an overview of relation extraction and then summarizing the current state-of-the-art.

Besides the discovery (NER) and identification (NEL) of entities in text, the construction of a comprehensive knowledge base also requires the discovery of relationships between them. The discovery of relationships forms the basis for many promising applications, ranging from the assessment of various corporate risk factors to the analysis of protein-protein interactions in biomedical applications. Even though with Wikidata [Vrandečić and Krötzsch, 2014], Freebase [Bollacker et al., 2008], and YAGO [Suchanek et al., 2007], some publicly accessible knowledge bases already exist that cover a significant number of relations, they are far from complete. Thus, it is not unusual for new applications to impose specific requirements on the existence of entities and relationships. Regarding, for example, the use cases outlined in Section 1.2, a comprehensive assessment of a bank's systemic risk entails the analysis of interactions between a large number of market participants. To this end, domain-specific relationships, such as `isInLitigationWith`, `divestsDivisionTo`, and `isPartnerOf`, are needed, which are, in many cases, either not at all or insufficiently covered by publicly accessible knowledge bases. As an additional challenge, the real world is constantly changing both in terms of

relationships and their participants, which requires continuous monitoring and updating of the facts in the knowledge base.

As a branch of natural language processing, relationship extraction is concerned with the discovery of relationships that exist between two or more entities in a given text document. As before, the core of this task is to convert the information contained in free-form texts into a machine-readable format. Driven by the importance of the problem, many different approaches have emerged over the years, which can be divided along different dimensions. At a high abstraction level, relation extraction approaches can be divided into *open* and *closed* extraction approaches. While the goal of closed relationship extraction approaches is to identify relationships from a predefined set of relations, open relationship extraction approaches aim to extract all existing relationships without adding any restrictions.

Along a second dimension, relation extraction approaches can be divided into *unsupervised*, *supervised*, and *semi-supervised* approaches. Motivated by the goal of discovering previously unknown, new relationships, most approaches that implement the *open relationship extraction* paradigm are based on unsupervised learning techniques. Like any other procedure, the unsupervised approach has its strengths and weaknesses. While unsupervised approaches are not limited to a predefined number of relationships nor require manually generated training data, their results are often difficult to interpret and cannot easily be mapped to the set of relationships defined by a given knowledge base. Especially the last point leads to a scenario similar to the one in Section 3.7, where a linking needs to be established between the discovered relationships and the relationships defined by the knowledge base. Since a relationship can be expressed in many different ways, it is necessary for integration purposes to either map the extracted relationship to the set of relationships defined by the used knowledge base or to extend it by the newly discovered relationship. As explained in the previous section, finding such an assignment is not trivial.

In contrast to open extraction approaches, most *closed extraction approaches* are implemented using either supervised or semi-supervised machine learning methods. Since the relationship types to be identified in a closed relationship extraction scenario are defined in advance, many of the supervised extraction techniques formulate the problem of relationship extraction as a classification problem. To this end, they use a selection of features to determine whether a text segment contains one of the predefined relationships, effectively turning the extraction into a multi-class classification problem. Although supervised extraction approaches usually outperform their unsupervised competitors in terms of extraction quality, they have the disadvantage of requiring a large amount of training data. Since adding new relationships to be extracted involves annotating another large amount of training data, any extension of a supervised approach becomes labor-intensive.

To mitigate this issue, Mintz et al. [2009] proposed *distant supervision*, a technique that leverages the information contained in existing knowledge bases to generate training data for supervised relationship extraction methods automatically. To this end, they operate under the assumption that a sentence expresses a relation defined in a knowledge base once it contains all entities involved in the relation. The identified sentences can then

be used to train a supervised relation extraction model. Although this process facilitates the automatic generation of training data, the underlying assumption often produces inaccurate annotations and, thus, noisy training data. In addition, distant supervision can only be performed for relationships already contained in the knowledge base, but not for discovering new relationships. A comprehensive discussion of distant supervision techniques for relation extraction is provided by Smirnova and Cudré-Mauroux [2019]

When dealing with relation extraction approaches, it is also important to consider their ability to extract relationships connecting a certain number of entities. Extraction approaches that focus on extracting relationships between exactly two occurring entities are referred to as *binary relation extraction* approaches, whereas approaches that consider more than two entities are commonly known as *n-nary* or *multi-way extraction approaches*. To illustrate the difference, consider the following fictitious example:

> "HP announced that it has signed an agreement with Lenovo to sell its cybersecurity business unit."

Where a binary relation extraction approach would yield the relationship `[HP, Lenovo, divestsDivisionTo]`, a multi-way approach would also take into account the divested entity, thus extracting the four-tuple `[HP, Lenovo, cybersecurity business unit, divestsDivisionTo]`. Especially in the area of event detection, n-ary relation extraction approaches are often used.

As can be seen in the overview of the *text mining* component in Figure 3.1, the relationship extraction module connects directly to the entity linking module, allowing it to take advantage of already linked entities. In addition, an appropriate solution to this problem requires considering a variety of aspects, such as the relevance of a particular entity for the considered relationship, the context of the relevant entities, and the grammatical structure of a given sentence.

In the following section, we provide an overview of some of the most prominent use cases for relationship extraction, while Section 3.8.2 highlights some of the most important challenges. We briefly summarize the current state-of-the-art in Section 3.8.3.

3.8.1 Applications

Relation extraction is an essential component of many applications. In the following, we provide a brief overview of several selected application areas:

Biomedical applications. In the biomedical domain, relation extraction techniques are used to extract knowledge from millions of biomedical texts provided by sources, such as PubMed [Zhou et al., 2014]. A concrete application is the extraction of textual evidence for molecular interactions between protein molecules, commonly referred to as protein-protein interactions. The extracted relationships can then be used to create a network graph of molecular interactions. To this end, Hsieh et al. [2017] make use of RNNs to recognize long-range relationships between proteins, while Peng and Lu [2017] integrate syntactic sentence structure into a dependency-based convolutional neural network for extracting protein-protein interactions.

Question answering. To answer a specific question, question answering systems often rely on relationships obtained by running a relation extraction system. Based on these relationships, the system is able to recognize and reason about the connections between entities that are relevant for answering a question. For example, Xu et al. [2016] use a combination of entity linking and relationship extraction to find possible answers to a question. They evaluate which of the answers is most likely correct by using Wikipedia articles as a source of evidence.

Sentiment analysis. Emotions often occur in product reviews, where they are expressed as positive and negative comments, for example, on parts of a mobile phone such as its camera or display quality. Here, relationship extraction can establish a connection between entities or parts of entities and the expressed emotions mentioned in a given text. In relation to the example above, a `partOf` relationship could be used to determine that an expressed emotion refers to the camera of a mobile phone and not to its display or the mobile phone itself. Another application of relation extraction techniques in the context of sentiment analysis can also be seen in the work of Qu et al. [2014]. Here, the authors employ relation extraction techniques to comprehend comparative statements, such as the claim that the camera of phone A is better than the camera of phone B.

3.8.2 Challenges

As with the recognition and linking of named entities, the extraction of relationships from free-form text comes with its own challenges, some of which are presented below:

Hyponymic & homonymic relationships. In addition to the fact that a multitude of different relationships can exist between the same two entities, the relationships can also be of homonymic or hyponymic character. In a homonymic relationship, the same verb can express different meanings depending on its context. For example, the verb "runs" in the phrase "Tim Cook runs Apple." expresses the meaning of Tim Cook being the leader of the company Apple, whereas in the phrase "Tim Cook runs a marathon", it expresses participation in a sports event. A hyponymic relationship, on the other hand, constitutes a general relationship which can be expressed by several other, often more specific, terms. For example, the `buys` relationship can also be expressed by using the verbs `acquire`, `purchase,` or `get`. The challenge with homonymic relations lies in distinguishing their different semantic meanings, whereas, with hyponymic relations, a problem similar to entity linking arises, where different expressions of a relationship must be mapped to a unique relational representation.

Long-range relationships. While many of the current extraction approaches focus on finding relationships within the bounds of a sentence, relationships can also be expressed across sentence boundaries. Relations expressed in this way tend to use pronouns to refer to other entities mentioned in the text. To capture such relations, it is necessary to resolve all references referring to other entities. Coreference resolution [Sukthanker et al., 2020], a subtask in the field of natural language processing, addresses exactly

this problem. After resolving all references, it is possible to also extract long-distance relationships that span several sentences.

Asymmetric relationships. As pointed out by Zuo et al. [2017], handling asymmetric relationships poses another challenge. In contrast to symmetric relationships that are semantically valid in both directions, asymmetric relationships such as the `ownershipOf` relationship can only be read in one direction. Identifying the direction of asymmetric relationships is not trivial and directly related to the participating entity types. If the entities participating in a relation are of different semantic types (e.g., person and location), the direction of a relationship (e.g., `bornIn`) can be derived from the type information. However, if both entities are of the same semantic type, for example, both companies, it is no longer evident in which direction a `supplierOf` relationship is meant to be read.

3.8.3 Current state of the art

At the time of writing, most state-of-the-art relation extraction techniques follow the supervised learning paradigm. As such, they use machine learning methods to learn the relation extraction task from a set of training documents and then apply what they have learned to extract relationships from previously unseen texts. While traditional extraction systems rely on either manually creating lexical and semantical features or the design of complex kernels, many of the newer approaches leverage representation learning, a practice pioneered by the deep learning community. Representation learning leverages the hierarchical structure of deep neural networks to learn distributed vector representations that aim to capture the semantic meaning of discrete entities such as words in a text or nodes in a graph. Apart from this practice, deep learning techniques have, in general, led to a dramatic change in the way most extraction systems address the extraction problem. As such, all the relation extraction systems presented below adopt deep learning concepts and neural networks. In the following, we give a brief overview of the current state-of-the-art in relation extraction.

The relation extraction approach proposed by Lee et al. [2019] is based on a recurrent neural network (RNN) and employs multiple attention layers to identify the most relevant text elements for relationship classification. They begin by translating the individual words of a given text into their respective embedding vectors by using pre-trained GloVe word embeddings [Pennington et al., 2014]. To determine the semantic meaning of each word in relation to its context, they use the self-attention mechanism of Vaswani et al. [2017a]. They feed the word representations generated by the underlying attention layer into an LSTM network, which aims to recognize more abstract syntactical structures. A subsequent second attention layer focuses on determining the meaning of the occurring words with respect to the occurring entity pairs. To this end, they consider both the relative position of the individual words to the occurring entities and their type information. Apart from the two-stage attention mechanism, a big advantage of their approach is that they model the type information of the occurring entities as latent

variables. This allows them to learn the types of individual entities in a fully automatic way during model training.

Similar to the previous approach, Wang et al. [2016] use multiple attention layers to identify the most relevant syntactic patterns for the relation extraction task. However, in contrast to the RNN-based approach presented by Lee et al. [2019], Wang's model is built using convolutional neural networks (CNN). Instead of relying exclusively on the translation of individual words through the use of pre-trained Word2Vec [Mikolov et al., 2013] or GloVe [Pennington et al., 2014] embeddings, they expand their word representations by combining them with *position embeddings* that encode the relative position of a word to the entity mentions in the text. These word representations are then transferred to a first attention layer, which determines the relevance of each word in relation to the occurring entities. Next, the representation generated by the attention layer is passed through the convolution layer, which constructs more abstract features by taking into account the relationships of several adjacent words. The second attention mechanism is performed during a pooling step that is applied to the feature outputs of the convolutional layer. By focusing on the relationships between these higher-level features, the network tries to estimate each feature's relevance with regard to the correct classification of a given relation. The result of the second attention layer is then used to determine the most likely relationship.

In their approach to relationship extraction, Cai et al. [2016] focus on making use of the information encoded in the *shortest dependency path* (SDP) between two entities of a given text block. They propose a *recurrent convolutional neural network* (RCNN) that combines the strengths of convolutional and recurrent neural networks as they represent the two most widely used network architectures for relationship extraction. After both the words and the dependency relations have been translated into a vector representation, they are passed through a recurrent LSTM network, which processes them in two individual channels. As a result, the word embeddings are processed separately from the dependency embeddings in their own channel. This part of the network is designed to capture global dependency patterns for a given SDP. The outputs of both channels are then passed on to a convolutional layer whose task is to identify local features between the outputs of both LSTM channels. The final BRCNN model arranges two RCNN networks bidirectionally, enabling the network to process the SDP from both directions. Apart from predicting the relationship itself, this network arrangement makes it possible to determine the direction of the relationship. The outputs of both RCNNs are then combined in a pooling layer, the output of which ultimately forms the basis for the subsequent relation classification.

The current best-performing approach to relationship extraction was introduced by [Soares et al., 2019] and is based on the BERT model architecture [Devlin et al., 2019] a state-of-the-art language model. The proposed model is capable of learning relation embeddings from text segments containing at least two entities that are already mapped to an unambiguous identity, e.g., by using one of the entity linking methods outlined in Section 3.7. This procedure is based on the assumption that if two entities are mentioned in the same text segment, a relationship is expressed between them. To avoid the replication of the used entity linking system, the linked entities are replaced

by placeholders before the training. Starting with a pre-trained BERT model, they optimize a loss function that reduces the distance between text segments that express similar semantic relationships, otherwise increases the distance between them. As part of their work, they examine the extent to which architectural model variations are suited for extracting the learned relationship embeddings from the BERT model. Finally, the extracted relationship embeddings are used to classify previously unseen text segments with respect to their expressed relationship. During their experiments, they achieved F-scores of up to 89.5 %, depending on the dataset.

3.9 Summary

In this chapter, we introduced and discussed the individual subcomponents of the *text mining* component, shown in Figure 3.1.

First, we introduced a named entity recognition system that recognizes companies in textual data with high lexical complexity, achieving a precision of up to 91.11% at a recall of 78.82%. Besides creating an NER system, we analyze the impact of different dictionaries containing company names on the performance of the NER system. Our investigation showed that significant performance improvements can be made by carefully including domain knowledge in the form of dictionaries into the training process of an NER system. On average, we were able to increase recall and F_1-measure by 6.57 and 3.85 percentage points, respectively, over our baseline that did not use any external knowledge.

Furthermore, we were able to show that the application of an alias generation process leads to an increase in recall while sustaining a high precision. Given the positive impact of the generated aliases on system performance, developing a more sophisticated alias generation process carries the potential for further performance improvements and the ability to better address the inherent complexities of company names. To complete the picture of the *text mining* component, we presented challenges, applications, and current state-of-the-art for both *entity linking* and *relationship extraction* in Sections 3.7 and 3.8.

Chapter 4

Few-Shot Knowledge Validation using Rules

Knowledge bases and their resulting knowledge graphs (KGs) form the basis of many modern information systems – their inherent network structure is used to enable semantic reasoning and the interpretation of complex tasks. By combining the methods presented in the previous chapters, it has become possible to automatically create large knowledge bases and hence large KGs from structured and unstructured data sources. During this endeavor, two main error sources can be identified. The first error source arises from the highly dynamic structure of KGs, which are subject to constant change as facts are updated, added, and removed, allowing incorrect information to find its way into the KG. The second error source, despite advances in the field of automatic knowledge base construction, consists in the fact that the methods presented in Chapters 2 and 3 are not flawless. Thus, it still happens that incorrect facts, caused by deduplication, extraction, or linking errors, find their way into the generated knowledge base, from where they manifest as data quality issues in the generated KG.

A typical approach to ensure data quality in the presence of continuous changes lies in the application of logical rules. Through the application of such rules, it becomes possible to increase the data quality of a KG by adding missing or removing incorrect facts. Usually, systems such as AMIE [Galárraga et al., 2015, 2013] or RuDiK [Ortona et al., 2018] are used to derive such rules by applying frequency-based approaches to the data of the considered KG. As a result, these approaches also depend on the data quality of the underlying KG and are therefore susceptible to errors and incompleteness themselves.

To address these issues, we propose Colt, an interactive, rule-based knowledge validation framework that combines knowledge graph embeddings with a few user interactions, also known as "a few shots", to create a classifier that is used to assess the quality and confidence of logical rules. We formalize the problem of *rule-driven knowledge validation* as learning a validation function over the rule's outcomes and establish a connection to the generalized maximum coverage problem and quality-preserving Gaussian processes. Further, we show that the presented interactive learning approach can improve classification performance by exploiting knowledge graph embeddings while

keeping the required user effort at a minimum. To this end, a domain expert is asked to validate a small percentage of rule-generated facts of a KG. Our model obtains (i) an accurate estimate of the quality of a rule with fewer than 20 user interactions and (ii) 75% quality (F_1-measure) with 5% annotations in the task of validating facts entailed by any rule. Furthermore, we publish our dataset, which consists of 26 manually annotated rules and contains more than 23 000 annotated facts. Moreover, we would like to point out that in addition to determining quality and confidence, the trained classifier can also be used for the conditional application of logical rules.

The content of this chapter is based on the work of Loster et al. [2020b] and is organized as follows: In Section 4.1, we give a comprehensive introduction to the topic of this chapter. We discuss related work in Section 4.2 and formally define the problem in Section 4.3. Our solution is presented in Section 4.4. In Section 4.5, we discuss the role of similarity functions in our framework. Section 4.6 reports on our experimental results, while Section 4.7 concludes this chapter.

4.1 Data quality and knowledge graphs

In recent years, systems, such as DeepDive [Shin et al., 2015], Knowledge Vault [Dong et al., 2014], and CurEx [Loster et al., 2018b], made it increasingly easy to automatically construct vast knowledge graphs (KGs). These systems often consist of many different components designed to extract and integrate facts from numerous different sources. As shown in the previous chapters, many of these components are based on machine learning techniques, which are rarely error-free. Thus, despite dramatic improvements through the use of state-of-the-art deep learning techniques, it is not possible to ensure the correct extraction [Akbik et al., 2018], linking [Raiman and Raiman, 2018] and discovery of relationships between textual entities [Wang et al., 2016], that form the facts of a knowledge graph. In addition, KGs suffer not only from incorrect but also from missing facts, which in their sum negatively affect its data quality and thus all downstream applications. Correcting these errors by adding missing or removing incorrect facts can be approached from two different directions, each with its own advantages and disadvantages.

The first approach, often used to add missing facts, leverages *knowledge graph embeddings* (KGE) [Wang et al., 2017], such as TransE [Bordes et al., 2013] or HolE [Nickel et al., 2016]. These models can be used to predict new facts based on existing ones, reducing the number of missing facts. To this end, they rely on the correctness and completeness of the underlying KG without taking additional data sources into consideration. While widely studied, these models have several shortcomings. First, although they correctly capture large parts of the graph, they fail to model entities and relationships that contain noisy information or are underrepresented due to missing data [Pujara et al., 2017]. Second, KGEs are hard to interpret as they cannot provide a clear *explanation* as to why a fact should be included in the KG.

The second approach is the application of *logical rules*. They can be derived automatically by rule learning systems, such as AMIE [Galárraga et al., 2015, 2013] and

RuDiK [Ortona et al., 2018]. By executing such rules on a KG, new facts can be added, and incorrect facts can be removed. For example, the rule

$$\texttt{isMarriedTo}(\texttt{a}, \texttt{c}) \wedge \texttt{livesIn}(\texttt{c}, \texttt{b}) \Rightarrow \texttt{livesIn}(\texttt{a}, \texttt{b})$$

states that if a person `a` is married to another person `c`, they most likely live in the same place `b`. By applying this rule, i.e., when the body of the rule is satisfied, a new `livesIn` fact between a person and a place can be added to the KG. Rules are much easier to understand and interpret than embeddings, but their discovery also suffers from noisy and missing data in the KG. As such, data quality issues of the underlying KG lead to rules being derived from factually incorrect examples, which implies that the generated rules model an incorrect state. In addition, rules are rarely completely correct or completely wrong. It is common for a rule to be applicable only to a certain percentage of KG facts [Galárraga et al., 2015; Ortona et al., 2018], however, the rule itself does not contain any information about the cases in which it can be safely applied. Ideally, a rule should be applied only if it contributes to the quality improvement of a KG. Although rule learning systems provide a statistical measure for the support of a discovered rule, this computation assumes the correctness of the underlying facts, which in practice inevitably leads to an unreliable confidence estimate.

A process aimed at addressing such data quality issues must, therefore, have access to *external information* in order to make necessary corrections. The natural way to verify facts is to consult experts on the topics covered by the KG at hand. Unfortunately, KGs often consist of millions of facts, making a comprehensive manual verification impossible. Therefore, we aim to radically reduce the manual effort by efficiently utilizing existing domain knowledge.

Because it is desirable to retain the understandability of rule-based approaches, we focus on a solution that (i) computes reliable rule confidence measures and (ii) complements the rules by a probabilistic model that validates their resulting facts. In a nutshell, our framework learns the characteristics of a particular rule from humans, who annotate *a small number* of its generated facts.

To address the data quality issues of KGs, we model the problem as learning a classifier for each rule that balances the exploration of new facts against the exploitation of knowledge accumulated in the classifier. This allows us to effectively reduce the number of required annotations to a few interactions. In contrast to the logical rules, the classifiers contain additional knowledge in terms of (i) information contained in the knowledge graph embeddings, and (ii) user feedback, allowing for better predictions than the original rule. The learned classifiers enable the conditional application of each rule so that depending on its prediction, it can be decided whether the fact identified by the associated rule can be trusted. Compared to a data-driven estimation approach, these predictions enable us to calculate a rule's confidence with far greater accuracy. As verifying the correctness of a large number of facts is cost-intensive, we reduce the amount of work by utilizing a sampling strategy. This strategy limits the effort of manual fact-checking to an adjustable budget, while simultaneously maximizing the benefit of the verified facts.

4.2 Related work

Since the proposed approach involves many different research areas, this section is divided into the topics *rule discovery* and its *interactive application*, *knowledge graph embeddings*, *hybrid methods* using a combination of rules and knowledge graph embeddings, and *active learning* approaches. Our proposed approach leverages *logical rules*, *knowledge graph embeddings*, and *active learning* concepts.

Rule discovery. The derivation of logical rules from KGs has been investigated for many years [Dehaspe and Toivonen, 1999]. Recent systems, such as AMIE [Galárraga et al., 2015, 2013] and RuDiK [Ortona et al., 2018], can derive rules from large KGs by using structural information, such as frequently occurring graph patterns. The mined rules can be used to derive new facts, find errors, derive conclusions, and better understand the underlying data. However, the completeness and coverage of KGs can become problematic for automated rule generation systems, which derive rules from their occurrence frequencies in the underlying KG. Although both systems report statistical measures, such as the standard and PCA confidence, for each generated rule, these metrics are based on pattern frequencies that are affected by data quality issues of the KG. This circumstance ties the quality of a generated rule directly to the quality of the underlying KG. We improve the confidence assessment of a rule by including user feedback as an additional source of knowledge. As such, the quality assessment of a rule does not depend solely on the KG data.

Interactive rule execution. Different approaches have investigated how KG rules for a specific task can be executed by taking user feedback into account. Arioua and Bonifati [2018] propose the improvement of KGs through the use of update-based repairs. To this end, they derive possible KG corrections from a set of rules, which are then proposed to a user [Arioua and Bonifati, 2018; Fan et al., 2019]. The user validates the suggested corrections and thus improves the underlying KG until it is transformed into a consistent state. This line of work assumes that the given rules are correct and reliable, such as those manually written by experts.

In contrast, our approach handles automatically generated rules, which can be approximate or incorrect. Consequently, it can be considered as a preliminary step towards understanding the rules, which can be included in the data repair step. Interactive error detection was also investigated for relational data [Heidari et al., 2019; Mahdavi et al., 2019]. However, such methods cannot be directly applied to graph data. While a rule can be materialized into a relational model by flattening a portion of the KG, the resulting "view" disregards other graph information.

Knowledge graph embeddings. The creation of knowledge graph embeddings (KGEs) has attracted widespread attention in recent years and led to improvements in many research areas that make use of these embeddings [Wang et al., 2017]. As such, KGEs provide a compact representation of a KG by projecting its entities and relations into

an n-dimensional vector space while retaining its structural information. These embeddings are used to improve tasks, such as relation extraction [Xu and Barbosa, 2019], and link prediction [Liben-Nowell and Kleinberg, 2007]. Since KGEs are capable of capturing structural and semantic information of the underlying knowledge graph, we use this information to learn a model for the correctness of rule-generated facts presented in Section 4.4.3.

Besides semantic matching models [Nickel et al., 2011], translational models, such as TransE [Bordes et al., 2013] and its extensions [Ji et al., 2015; Lin et al., 2015; Wang et al., 2014b] are amongst the most widely used methods for generating KGEs. These methods maximize a score associated with each fact. Often informally interpreted as the plausibility of a fact, the score defines a distance measure between the entities of the evaluated predicate. As these methods use KG facts to generate a condensed representation of the graph, data quality issues, such as incorrect facts, directly affect the quality of the generated embeddings [Pujara et al., 2017]. It follows that poor data quality of the KG leads to poor results in downstream tasks, such as link prediction.

We approach these quality issues by including human knowledge into classifier training without relying exclusively on KG data. By incorporating user feedback in the training process, the classifier learns to counteract poor embeddings and thus mitigate their negative effects. This limits the impact of noisy and missing KG information.

Hybrid methods. Although all methods use the KG facts for computing KGEs, some approaches can consume additional information, such as textual descriptions [Wang et al., 2014a; Xie et al., 2016], literals [Gesese et al., 2019], or logical rules [Guo et al., 2016; Zhang et al., 2019]. We focus on the approaches that rely on rules during the learning process as they are closest to our work.

One approach jointly models logical rules and fact triples in a unified framework, representing triples as atomic and rules as complex formulae [Guo et al., 2016]. The learning takes place by minimizing a global loss function that spans both formulae. Another system creates high-quality rules from noisy KGs by using KGEs to control the rule generation [Ho et al., 2018]. Their system represents a complementary approach as it can be used to create a better starting point for our framework. A third approach uses rules to generate new triples for the training of KGEs, which are, in turn, used to derive new rules [Zhang et al., 2019]. By using rule-induced triples in KGE training, they improved both the quality of the resulting KGEs and the results of their link prediction. These works address a problem that is different from ours: we train a classifier on existing KGEs and rules while counteracting poor embedding quality by incorporating user feedback into the learning process. Our goal is to interactively learn a model that can both assess the quality of a given rule and predict when the rule should be applied.

Active learning. Methods that actively sample data points to be labeled belong to the group of active learning approaches [Settles, 2012]. They take advantage of both model and data properties and are well suited for human assessors and exploratory tasks [Dimitriadou et al., 2014]. Our work builds on the general idea of active learning, but estab-

lishes important connections to the weighted coverage and GP-UCB algorithm [Srinivas et al., 2010] in order to obtain quality-preserving solutions.

4.3 Background and problem definition

A *knowledge graph* (KG) contains entities, such as *Max Planck* and *Germany*, connected via relationships, such as *isCitizenOf*. We are given a set of *entities* $\mathcal{E}$ and a set of *relationships* $\mathcal{R}$. A triple (e_1, r, e_2) with $e_1, e_2 \in \mathcal{E}$, and $r \in \mathcal{R}$ is called a *fact*; as r is a relationship in the pair (e_1, e_2), each fact can be equivalently represented as an *atom* $r(e_1, e_2)$. A set of facts constitutes a *knowledge graph*, also known as knowledge base [Deshpande et al., 2013], information graph [Lissandrini et al., 2014], or heterogeneous information network [Shi et al., 2017]. YAGO [Suchanek et al., 2007] and DBpedia [Auer et al., 2007] are examples of large KGs for which facts are automatically constructed from Wikipedia, GeoNames [GeoNames, 2018], and WordNet [Fellbaum, 1998].

Definition 4.1. A *knowledge graph* is a triple $G : \langle \mathcal{E}, \mathcal{R}, \mathcal{F} \rangle$, where $\mathcal{E}$ is a set of entities, $\mathcal{R}$ a set of relationships, and $\mathcal{F} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ a set of facts.

A knowledge graph might contain errors, such as *(Max Planck, isCitizenOf, China)*. The set of all true facts (knowledge) is denoted by $\mathcal{K} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, and we assume that at least a part of the knowledge graph is correct, i.e., $\mathcal{F} \cap \mathcal{K} \neq \emptyset$. Partial correctness constitutes a reasonable assumption regarding the soundness of the KG construction process.

An *atom* is the smallest logic statement composed by facts, including variables. For example, the atom *hasChild(x, John)* denotes everything (by the variable x) that has a child named *John*.

Definition 4.2. A *rule* $A_1 \wedge A_2 \wedge ... \wedge A_n \Rightarrow C$ is a logical implication consisting of a head and a body. The head is a single atom C, and the body is a conjunction of atoms $A_1 \wedge ... \wedge A_n$.

An example of such a rule is:

$$\rho_0 : \mathit{hasChild}(a, c) \wedge \mathit{isCitizenOf}(a, b) \Rightarrow \mathit{isCitizenOf}(c, b)$$

which conveys that if a person a has a child c and is a citizen of the country b, the child must also be a citizen of b. This is an example of a *positive* rule, that asserts the existence of specific facts in the head, given the body. In addition to positive rules, we also consider *negative* rules, which identify sets of facts that lead to contradictions. For example, the negative rule $\mathit{hasChild}(a, b) \Rightarrow \mathit{isMarried}(a, b)$ defines that if b is the child of a, b is also married to a. In this respect, the rule describes a state that should never occur, whereby all facts that comply with this rule are automatically regarded as contradictory and thus incorrect. Negative rules can be used to identify inconsistencies in the data, which are eventually resolved via user intervention [Arioua and Bonifati, 2018]. In our setting, we consider the case in which the KG data can be incomplete and incorrect.

The *instances* $\mathcal{I}_G(\rho) \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ of a rule ρ in the knowledge graph G are the set of facts expressed by the right-hand-side of a rule, assuming the left-hand-side is true. For example, the instances of rule ρ_0 are the facts produced by *isCitizenOf*(c, b), given that both *hasChild*(a, c) and *isCitizenOf*(a, b) are true.

Ideally, if we knew all true facts $\mathcal{K}$, we could validate the *confidence* of the rule on the knowledge graph G by computing the instances in $\mathcal{K}$ that are correctly captured by the rule ρ via the ratio $\frac{|\mathcal{I}_\mathcal{K}(\rho) \cap \mathcal{I}_G(\rho)|}{|\mathcal{I}_\mathcal{K}(\rho)|}$. As $\mathcal{K}$ is, in reality, unavailable, we model the validation as a labeling function $f : \mathcal{I}_G(\rho) \to \{0, 1\}$ over the instances $x \in \mathcal{I}_G(\rho)$:

$$f(x) = \begin{cases} 1 & \text{if } (e_1, r, e_2) \in \mathcal{K} \\ 0 & \text{otherwise} \end{cases}$$

It returns 1 if an instance $x \in \mathcal{I}_G(\rho)$ is correct (i.e., $(e_1, r, e_2) \in \mathcal{K}$) and 0 otherwise. The access to $\mathcal{K}$ can be implemented by an external oracle, such as a domain expert, who validates whether a fact is true or not. As assessing the validity of facts by consulting an oracle can be costly, we limit the number of evaluations to a budget B, thus bounding the evaluation effort.

The *rule-driven knowledge validation* problem aims at finding a labeling function f within the budget B:

Problem 1. *Given a knowledge graph $G : \langle \mathcal{E}, \mathcal{R}, \mathcal{F} \rangle$, a rule ρ with instances $\mathcal{I}_G(\rho) \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, and a budget B, find*

$$L^* = \arg\max_{L \subseteq \mathcal{I}_G(\rho)} \sum_{x \in L} f(x) \qquad \text{subject to } C(L) \leq B$$

where C represents the sum of costs of all the facts in L.

In many practical scenarios, the cost associated with each fact is 1, as the budget quantifies the number of questions to the oracle. In our setting we make this assumption but note that even if the function f is known, the problem is **NP**-hard with non-constant cost.

4.4 The Colt framework

To learn a model for a rule ρ, we execute ρ, which yields a set of rule-compliant facts $\mathcal{I}_G(\rho)$. Each fact $x \in \mathcal{I}_G(\rho)$ corresponds to an *edge* in the knowledge graph and is composed of a *subject*, a *predicate*, and an *object* $\langle s, p, o \rangle$. As such, a fact implies that an edge of type *predicate* (p) exists between entities *subject* (s) and *object* (o). For example, the edge $\langle$`Albert Einstein, bornIn, Ulm`$\rangle$ encodes the fact "*Albert Einstein was born in Ulm*".

A fact's validity is ensured by the labeling function f, which is at the core of our solution and needs to be designed with great care. A negligent design could lead to either inefficient or impractical solutions. In particular, since the function is unknown,

the direct optimization of Problem 1 is not possible. To circumvent this impasse, we make the following assumptions. First, the labeling function can be equivalently represented by a *classifier* $\widetilde{f}$ on the facts, which is trained on the evaluations seen so far. Second, since facts are related to one another, we assume the existence of *a similarity* $\kappa : (\mathcal{E} \times \mathcal{R} \times \mathcal{E}) \times (\mathcal{E} \times \mathcal{R} \times \mathcal{E}) \mapsto [0, 1]$ between pairs of facts. Such similarities have been studied extensively in the past few years. We provide more information on how to compute expressive similarities in Section 4.5.

Our framework, COLT, presented in Algorithm 1, proceeds iteratively by using the information in the classifier and asking the user to evaluate facts until a specified budget B is depleted. In the beginning, the classifier is initialized (Line 2) using the instances and the similarity function. At every iteration t, the algorithm selects a new fact x_t using the classifier's knowledge (Line 5). The user then validates the fact (Line 6), providing a value $y_t = 1$ if the fact is true and $y_t = 0$ otherwise. Finally, the algorithm updates its beliefs by incorporating the true value y_t for the fact x_t. The classifier serves as a proxy for the unknown function f, which encodes the true facts $\mathcal{K}$ for the instances $\mathcal{I}_G(\rho)$ of rule ρ. This classifier is then used to **(i)** estimate a confidence value for the current rule and **(ii)** determine for which facts a rule is likely to be correct.

The classifier $\widetilde{f}$ is of crucial importance, as it should guarantee a high classification accuracy within a few evaluated facts. Section 4.4.1 describes an algorithm based on maximum set coverage that fully relies on the correctness of the similarity measure. In Section 4.4.2, we propose a model that combines neural networks and probabilistic Gaussian processes to overcome the rigidity of maximum coverage.

Algorithm 1 The COLT framework

Input: knowledge graph G; rule instances $\mathcal{I}_G(\rho)$
Input: similarity κ; budget B
Output: A classifier $\widetilde{f}$

1: $U \leftarrow \mathcal{I}_G(\rho)$ ▷ Unlabeled facts
2: $\widetilde{f} \leftarrow \textsc{Initialize}(U, \kappa)$ ▷ Initialize the classifier
3: $L \leftarrow \emptyset$ ▷ Initialize set of labeled facts
4: **for** $t = 1 \dots B$ **do**
5: $\quad x_t \leftarrow \textsc{Select}(\widetilde{f}, U, \kappa)$ ▷ Select the next fact
6: $\quad y_t \leftarrow f(x_t)$ ▷ User validates x_t
7: $\quad L \leftarrow L \cup \{x_t\}$ ▷ Add selected fact to L
8: $\quad \textsc{Update}(\widetilde{f}, \kappa, x_t, y_t)$ ▷ Update the classifier
9: $\quad U \leftarrow U \setminus \{x_t\}$ ▷ Update U by removing x_t
10: **return** $\widetilde{f}$

4.4.1 COLT-MC: A maximum coverage solution

We devise our first algorithm based on a variation of the greedy algorithm for the *maximum coverage* (MAXCOVER) problem [Nemhauser et al., 1978]. Given some sets over a universe of elements and a fixed number B, the maximum coverage problem finds B sets that overall contain the maximum number of elements from the universe. Although

the problem is **NP**-hard, the greedy algorithm that selects the set and adds additional elements achieves an $\mathcal{O}(1 - 1/e)$ approximation.

Note that Problem 1 shares some traits with MaxCover, as every fact x identifies a set $\kappa(x, \cdot)$ in which each of its elements has a different similarity weight. As such, Problem 1 translates into finding a set of elements that *maximize the weighted coverage* identified by each fact's similarity set $\kappa(x, \cdot)$. The variant of MaxCover in which every element's weight depends on the set containing the element is called *generalized maximum coverage* (GenMaxCover) [Cohen and Katzir, 2008]. Generalized maximum coverage admits a greedy $\left(1-\frac{1}{e}\right)$-approximate solution.

The generalized version takes into account different costs for each element. However, in our case, this cost is fixed and determined by the number of facts evaluated by a user. We relax the GenMaxCover assumptions and devise a greedy solution for the problem with a fixed cost. Such a greedy scheme selects at each iteration the fact that maximizes the *weighted marginal gain*,

$$\Delta(x|L) = \sum_{x' \in \mathcal{I}_G(\rho)} \kappa(x, x') - \max_{x'' \in L} \kappa(x', x'') \tag{4.1}$$

which quantifies the increment in similarity if the element x is added to the set L. As soon as the set L contains B elements, the algorithm stops. We refer to this greedy algorithm as Colt-MC.

Algorithm 2 Colt-MC

```
1: function Initialize(U, κ)
2:     f̃ ← 0                                  ▷ Initialize uniform classification
3: function Select(f, U, κ)
4:     return arg max_{x∈U} Δ(x|L)            ▷ See Eq. 4.1
5: function Update(f̃, κ, x_t, y_t)
6:     f̃(x_t) ← y_t
```

Algorithm 2 shows the Initialize, Select, and Update function for the greedy algorithm. The classifier $\widetilde{f}$ is a one-nearest-neighbor (1-NN) classifier that provides a binary label for each fact. The evaluation of $\widetilde{f}$ on unlabeled facts returns the label of the most similar fact among those evaluated so far. In the end, the approximation of the labeling function f provided by the classifier $\widetilde{f}$ is

$$\widetilde{f}(x) = \begin{cases} f(x) & \text{if } x \in L \\ f(\arg\max_{x' \in L} \kappa(x, x')) & \text{otherwise} \end{cases}$$

One convenient property of Colt-MC is that it ensures maximum quality if all facts generated by a rule are evaluated. We also observe this property empirically during the experiments in Section 4.6.3.

4.4.2 COLT-GP: A learning-based solution

The COLT-MC algorithm, as well as its corresponding analysis, implicitly assumes that both the similarity function κ on the facts and the user's validations are correct and reliable. To overcome this obstacle, we propose a probabilistic method based on deep kernel learning (DKL), which combines the advantages of neural networks and Gaussian processes (GPs) (Section 4.4.3).

To estimate the value and uncertainty of $\widetilde{f}$, our proxy of the labeling function, we assume that the labeling function at step t is a Bernoulli distributed random variable sampled from a logit-transformed Gaussian distribution. The Gaussian random variable is represented by a Gaussian process [Bishop, 2007], which models points in space as individual Gaussians related through a kernel function representing the similarity κ among facts.

A GP is completely determined by its mean μ and covariance matrix $\mathbf{C}$, in which each element $\mathbf{C}_{ij} = \kappa(x_i, x_j)$ represents the similarity between a triple pair x_i and x_j. GPs define a distribution over functions where the *prior* of such distribution is a Gaussian $\mathcal{N}(0, \mathbf{C})$ with mean 0 and covariance $\mathbf{C}(\mathcal{I}_G(\rho), \mathcal{I}_G(\rho))$ among all instances in $\mathcal{I}_G(\rho)$. In practice, the prior is never computed. Instead, we are interested in the *posterior* probability $\Pr(x \in \mathcal{K}|L, x)$ of a fact x of belonging to the true facts $\mathcal{K}$ knowing the labeled facts L. The posterior of a GP is Gaussian; however, in order to classify the facts, the function sampled from the posterior needs to be transformed to return a value between 0 and 1 with high probability. This transformation is done by a *logistic sigmoid* function $s(x) = \frac{1}{1+e^{-x}}$. The final posterior is the expectation of the predictions over the sample of logistic transformed Gaussian functions:

$$\Pr(x \in \mathcal{K}|L, x) = \int s(f_*) \Pr(f_*|L, x)\ df_* \tag{4.2}$$

The integral in Eq. 4.2 is analytically intractable but can be approximated with sampling methods or analytical approximations, such as the stochastic variational inference (SVI) used in Section 4.4.3.

The choice of GPs for classification has two significant implications. First, a GP is a Bayesian model that allows sampling using the predictions from the posterior. As the choice of the sampling strategy is critical for such models, we analyze multiple choices for different sampling strategies in Section 4.4.4. Second, GPs are non-parametric models; thus, they can easily update their beliefs by using Bayes' theorem on the posterior computation.

We are now ready to describe COLT-GP (Algorithm 3), represented in Figure 4.1. For the sake of completeness, Algorithm 3 (Line 2) reports on the random variable's prior initialization even though it is never explicitly computed. The model parameters $(\mu, \mathbf{C})$ are updated using the posterior inference in Eq. 4.2. Each iteration step samples an unlabeled fact using the SELECT function, which retrieves an element according to one of the strategies in Section 4.4.4. The sampled fact is then presented to a user who evaluates its correctness. For example, given the fact ⟨`Donald Trump, bornIn, New`

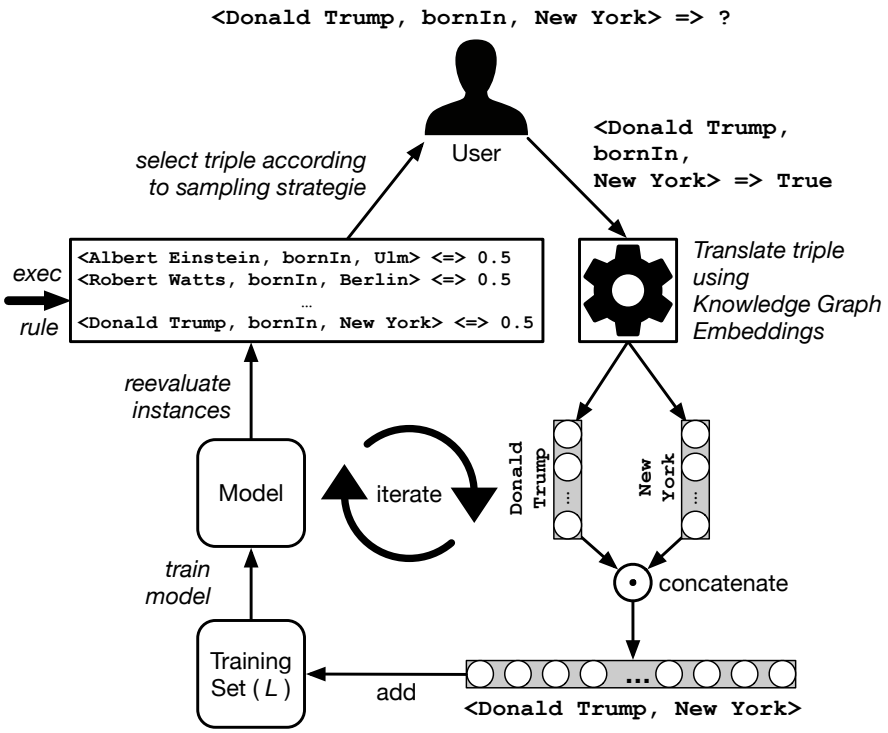

Figure 4.1: Overview of the interactive model-building process

York⟩, a user verifies whether or not *Donald Trump* was born in *New York*. The labeled triple is then added to the training set L.

Algorithm 3 Colt-GP

1: **function** Initialize($\widetilde{f}, U, \kappa$)
2: $\quad\widetilde{f} \sim s(\mathcal{N}(0, \mathbf{C}))$ ▷ Gaussian prior, not explicitly computed
3: **function** Select(f, U, κ)
4: $\quad$**return** x_t sampled with one strategy in Sec. 4.4.4
5: **function** Update($\widetilde{f}, \kappa, x_t, y_t$)
6: $\quad\widetilde{f} \leftarrow$ Update the posterior using SVI [Wilson et al., 2016b] for Eq. 4.2

The posterior calculates the probability of a fact being true, which is estimated by Eq. 4.2. To convert this probability into a label of 1 or 0 for each fact x, $\widetilde{f}(x)$ returns 1 if the probability ≥ 0.5, otherwise zero[1]. Based on the investigations in Section 4.6.1, this choice of probability threshold led to an overall good performance.

$$\widetilde{f}(x) = \begin{cases} 1 & \text{if } \Pr(x \in \mathcal{K} | L, x) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

[1]With a little abuse of notation $\widetilde{f}$ indicates both the posterior and the classifier.

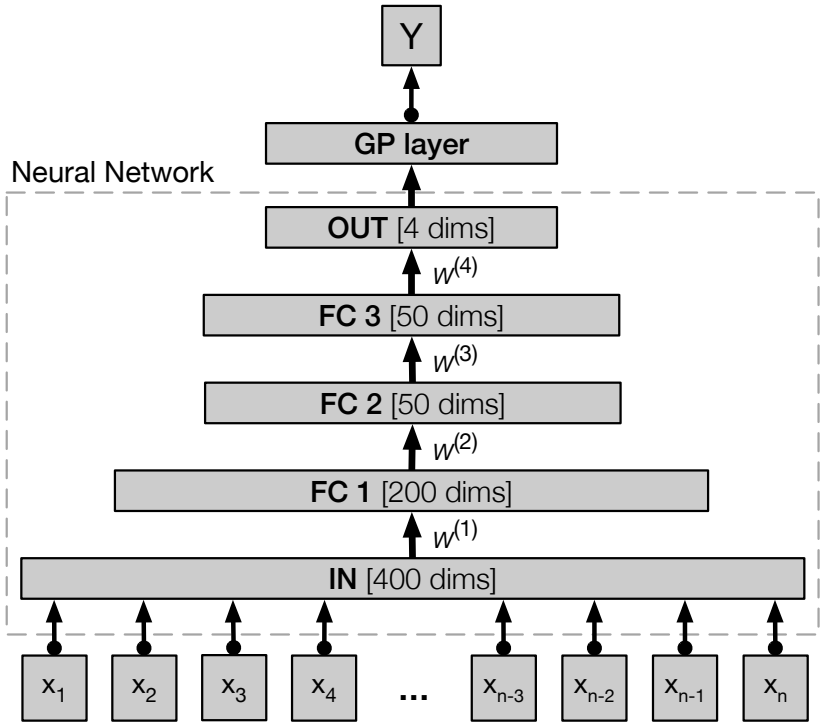

Figure 4.2: Overview of the model architecture.

4.4.3 Learning with deep kernels

The inference step in Gaussian processes requires the inversion of the covariance matrix, which grows with the number of evaluated facts. The numerical instability of this matrix inversion coupled with modest performance on high dimensional data [Djolonga et al., 2013] is detrimental to the final classification quality. By their nature, knowledge graphs have a high number of dimensions as one should consider at least all relationships, entity attributes, and connections.

To circumvent the GPs deficiencies, we employ deep kernel learning (DKL) [Wilson et al., 2016a], a combination of neural networks and Gaussian processes. Deep kernel learning aims at learning a flexible similarity function, while at the same time incorporating the user's evaluations on the facts. It scales linearly with the number of evaluated facts and combines the strengths of both neural networks and GPs in one unified model. To learn the model parameters and perform posterior inference, we use the fast stochastic variational inference (SVI) procedure [Wilson et al., 2016b], enabling the use of non-Gaussian likelihoods.

Figure 4.2 shows the general model architecture. It consists of two main components: a set of neural network layers and a Gaussian process layer (GP layer) applied to the network's output vector. The combination results in a deep probabilistic neural network that meets our requirements.

In our adaptation of the model, the neural network consists of three fully connected layers, each equipped with a ReLU activation function [Hahnloser et al., 2000].

This model part produces a condensed lower-dimensional representation of the high-dimensional input data, thus effectively reducing the dimensionality. To this end, the n-dimensional input vectors $X = (\mathbf{x}_1 ... \mathbf{x}_n)$ are transferred to the input layer (IN), from where they successively pass through three network layers (FC1-FC3) and finally form the desired low-dimensional data representation at the output layer (OUT).

The second part of the model consists of a GP layer that is connected to the output layer of the network and receives the low-dimensional data representation as its input. The GP layer itself consists of j Gaussian Processes $g_1 \dots g_j$, each having their own corresponding kernel $k_1 \dots k_j$ that operates on subsets of the vector coming from the network. When selecting the number of GPs in this layer, we follow Wilson et al. [2016b] and use one GP for each dimension in the output vector ($j = 4$). To obtain the final result, first, the individual GPs are additively combined and then transformed by an observation model, which in our case is a Bernoulli likelihood, into the final result Y.

Training. The model is trained via stochastic variational inference (SVI) and gradient descent. We jointly train the weights of the neural network and the parameters of the Gaussian processes in an end-to-end fashion. We use the marginal log-likelihood as our loss function. To determine the gradient of the loss function with respect to the weights, we use the backpropagation algorithm, while performing weight updates by using the Adaptive Moment Estimation optimizer (ADAM) [Kingma and Ba, 2015].

4.4.4 Sampling strategies

As stated in Section 4.4.2, the sampling strategy is at the heart of our Colt-GP algorithm, as it is responsible for the incremental selection of new facts. A good sampling strategy aims to select training examples of high information density that are likely to improve classifier performance when used for training. In the following, we present the sampling strategies considered in our experiments.

Random. Random sampling is the simplest sampling technique: it randomly selects training examples from the data and adds them to the training dataset, ignoring any prior knowledge. Therefore, random sampling constitutes the weakest strategy in terms of harnessing the model's beliefs during the sampling process.

Maximum uncertainty. To improve the random sampling strategy, maximum uncertainty sampling (MAXU) uses the model itself to assess how valuable the labeling of each data point is in terms of confidence gain [Lewis and Gale, 1994]. This sampling is done by using the model to estimate its uncertainty in assigning each unlabeled data point to a particular class. The data points for which the classifier is most uncertain are those closest to its decision boundary. The incremental labeling of the data points with maximum uncertainty leads to the refinement of the decision boundary, as the most critical data points are exposed to the classifier. Because we are performing binary classification, distinguishing only between correct and incorrect facts, we use the binary entropy

measure to determine the uncertainty of the classifier with respect to its classification:

$$H(X) = -p\log_2(p) - (1-p)\log_2(1-p)$$

As mentioned in [Roy and McCallum, 2001; Tang et al., 2002], a disadvantage of this strategy can be its sensitivity to outliers, which is addressed by GP-UCB.

GP-UCB. The *Gaussian process upper confidence bound* algorithm (GP-UCB) introduced by Srinivas et al. [2010] defines a sampling criterion that aims at finding a trade-off between exploring the function space and exploiting maximal function areas. To this end, they propose the following sampling strategy:

$$x_t = \arg\max_{x \in U} \mu_{t-1}(x) + \sqrt{\beta_t}\sigma_{t-1}(x)$$

In this formula, U denotes the unlabeled facts as defined in Algorithm 1. The exclusive selection of elements $x \in U$ where the model has either a high variance (σ_{t-1}) or a high expected reward (μ_{t-1}) cannot be regarded as the optimal strategy, as it leads on the one hand to the neglect of useful information and on the other hand to poor generalization capabilities of the model. To overcome this issue, the proposed GP-UCB sampling strategy strikes a balance between *exploration* by selecting elements where the model has a certain degree of uncertainty (large σ_{t-1}), and *exploitation* by choosing elements with high belief (large μ_{t-1}). By simultaneously optimizing both criteria, the strategy seeks to achieve an equilibrium between exploration and exploitation.

The parameter β_t trades-off exploration and exploitation. A large β_t corresponds to more exploration, while a small β_t leads to more exploitation. The GP-UCB strategy ensures a logarithmic growth of the regret, the deviation in quality with respect to the optimal sampling, by setting $\beta_t = 2\log(|\mathfrak{I}_G(\rho)|t^2\pi^2/6\delta)$ with $\delta \in (0,1)$. In other words, the regret on the evaluations tends to 0 when the number of validated facts tends to infinity.

According to the above formula, β_t starts at a relatively small value, which encourages it to be more exploitative. Along with the increasing number of sampling steps t, β_t also increases, making it more exploratory. We use this policy to update β_t during our experiments.

4.5 Computing similarities

The choice of an adequate similarity measure κ depends on the specifics of the data and the application domain. We avoid the need to design an ad-hoc similarity measure by proposing a flexible one based on *knowledge graph embeddings* [Wang et al., 2017]. Knowledge graph embeddings refer to a family of methods that learn to represent a graph in a low-dimensional space. These methods implicitly capture important structural and semantic information and are agnostic to the application domain. Although errors and missing data in the knowledge graph affect the quality of the generated embeddings,

they constitute a solid starting point. The design of such embedding methods is outside of the scope of this work. We refer to Wang et al. [2017] for a comprehensive survey on these methods.

For our purposes, we employ HypER [Balazevic et al., 2019], an expressive and fast embedding method inspired by ConvE [Dettmers et al., 2018]. For each fact $\langle s, p, o\rangle$ with subject s, object o, and predicate p, HypER computes a 200-dimensional vector $\mathbf{s}$, $\mathbf{o}$, and $\mathbf{p}$. Similar to previous approaches, HypER maximizes the likelihood for the existence of a fact in the knowledge graph. Additionally, the method aggregates information from the subject and predicate employing convolution operators. The object $\mathbf{o}$ in the vector space is then computed as a non-linear transformation of the subject and predicate vector.

These embeddings are calculated for all subjects, predicates, and objects in a knowledge graph before any rule is processed. The final embedding $\mathbf{x}$ of a fact $x = \langle s, p, o\rangle$ is the 400-dimensional vector concatenation $\mathbf{x} = \mathbf{s} \circ \mathbf{o}$ of the subject and object vectors. Note that since we consider one rule at a time, the predicate p in the rule's body always remains the same and can thus be omitted.

As HypER achieves state-of-the-art performance in predicting missing facts, it represents an ideal starting point and a solid baseline for our approach. The Colt framework builds on top of HypER by integrating user validation.

Colt-MC: The similarity score between fact $x = \langle s, p, o\rangle$ and fact $x' = \langle s', p', o'\rangle$ is computed as the standard cosine similarity between the fact vectors:

$$k_{cos}(x, x') = \frac{\mathbf{x}\cdot\mathbf{x}'}{\|\mathbf{x}\|\|\mathbf{x}'\|} = \frac{(\mathbf{s} \circ \mathbf{o})\cdot(\mathbf{s}' \circ \mathbf{o}')}{\|\mathbf{s} \circ \mathbf{o}\|\|\mathbf{s}' \circ \mathbf{o}'\|}$$

As a result, Colt-MC augments embeddings by enabling them to integrate user feedback.

Colt-GP: The Gaussian processes included in Colt-GP are kernel methods, which naturally incorporate similarity scores. The default kernel for Gaussian processes is the RBF kernel:

$$k_{RBF}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||}{2\sigma^2}\right)$$

The RBF kernel provides smoothness as the similarity exponentially decreases with an increase of the Euclidean distance. The parameter σ is learned through maximum likelihood estimation. GPs with RBF kernels implicitly project the points into an infinite dimensional space so that the GP layer in Figure 4.2 represents a hidden layer with an infinite number of neurons, thereby significantly increasing the model's flexibility. Finally, the transformation induced by deep kernel learning (DKL) in Section 4.4.3 further modifies the embedding space through a parametric non-linear function g. The final kernel learned by the architecture in Figure 4.2 is then

$$k_{DKL}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||g(\mathbf{x}) - g(\mathbf{x}')||}{2\sigma^2}\right)$$

4.6 Experiments

This section presents and discusses our experimental results, starting with a description of the dataset, competing algorithms, parameter settings, and running time. In Section 4.6.1, we show that COLT-GP is capable of learning a rule's behavior from the triples it generates. How different sampling strategies affect the training process is explored in Section 4.6.2. In Section 4.6.3, we investigate how COLT-GP compares to the COLT-MC algorithm (Section 4.4.1) and two active learning baselines. Section 4.6.4 concludes the experiments by examining the relationship between a given number of interactions and the estimated confidence value for the rules.

Datasets. We use YAGO [Suchanek et al., 2007], an open-source knowledge base, for the creation of knowledge graph embeddings and the derivation of logical rules. At the time of writing, the YAGO dump[2] comprised 948 358 triples, 36 relationship types, and 470 485 different entities. To obtain positive and negative rules, we ran RuDiK [Ortona et al., 2018] and AMIE [Galárraga et al., 2015] with their standard parameters on YAGO. Out of 1 517 mined rules, 928 produced more than 5 500 triples. The execution of a rule can be understood as a query, where a check is made for all triples of a KG to see whether they fulfill the condition specified on the left side of the executed rule. Whenever a checked triple evaluates to true, this triple belongs to the triple set generated by this rule.

We manually annotated the instances of 26 rules (22 positive; 4 negative) out of the 589 rules that produced fewer than 5 500 instances. The decisive criteria for the selection of these rules included the number of triples that had to be annotated as well as the variety of predicates involved. Also, we selected a majority of rules with three atoms in their body to focus on use cases where analyzing and validating is harder for users. For negative rules, where the execution in most cases yielded a very high number of triples, we randomly selected 1,000 triples for annotation. To cover a range of different output sizes, we annotated rules with a small output of just 40 triples, up to rules with 5 269 triples.

Similarly, we annotated rules that are mostly right, rules that are nearly always wrong, and rules with mixed confidence. Table 4.1 reports the selected rules with their statistics. For the listed 26 rules, we manually annotated a total of 23 324 triples, of which 5 524 corrected errors and missing facts in the underlying KG. To our knowledge, this is the largest available dataset of annotated rules.

Algorithms. In addition to COLT-MC (Section 4.4.1), we evaluate the COLT-GP algorithm (Section 4.4.2) against two active learning baselines, one based on logistic regression (AL-LogReg) and the other on an SVM (AL-SVM) [Schohn and Cohn, 2000]. Following Section 4.6.2, we choose *maximum uncertainty* as the selection strategy for

[2] `http://resources.mpi-inf.mpg.de/yago-naga/amie/data/yago2/yago2core_facts.clean.notypes.tsv.7z`

both active learning approaches. In our experiments, we use scikit-learn implementations for the logistic regression and SVM classifiers and keep their default parameter settings.

Parameter settings. To train the model presented in Section 4.4.3, we used the ADAM optimizer [Kingma and Ba, 2015] and kept its default parameter settings at $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. We train the model with a batch size of 32 until it converges or up to a maximum of 1000 epochs. For the neural network implementation, we use the PyTorch framework, while using the GPyTorch library to implement the GP layer. We train the models on a Linux machine with 2×2.20GHz CPUs, each with 10 cores, 251 GB RAM, and one Nvidia Titan X Pascal GPU.

Runtime. Recall that the main contribution of this chapter is to investigate methods that enable the quality and confidence assessment of logical rules by using only a few humanly validated rule instances. While a thorough evaluation of the runtime deviates from the main focus of this work, we observe that our methods run in less than 10 seconds on 20–100 evaluated instances and thus in real-time. This result, achieved on a commodity machine, confirms the ability to use the Colt framework and its most expressive Colt-GP model in production systems.

4.6.1 Learning rule characteristics

The goal of our first experiment is to show that Colt-GP can learn the characteristics of a rule from the instances it generates. To this end, we divide the previously annotated instances into training and test sets using a ratio of 70 to 30. We train Colt-GP on the training set and show that we are able to predict whether the relationship implied by the processed rule holds for the never seen instances from the test set. This scenario represents an out-of-sample evaluation, as it is often used in the evaluation of many machine learning approaches. It primarily aims to test the learned model's generalization capabilities and, at the same time, represents the most difficult classification scenario.

Figure 4.3 summarizes the results by showing the ROC curves for the annotated rules as well as the mean and variance of each of the rule's ROC curves. The ROC curves of the individual rules and, hence, also the average ROC curve are all significantly above the dashed line, which represents the theoretical behavior of a random classifier. Colt-GP attains 89.78% average AUC, which testifies the correctness of the classifier in predicting unknown facts. Even the lower end of the variance line attains 79.38% AUC, which is significantly above the performance of a random classifier. We conclude that Colt-GP is able to learn a meaningful model by using individual facts.

4.6.2 Selecting the sampling strategy

Different sampling strategies can significantly impact model performance, as they are responsible for the composition of the training data. In this experiment, we investigate

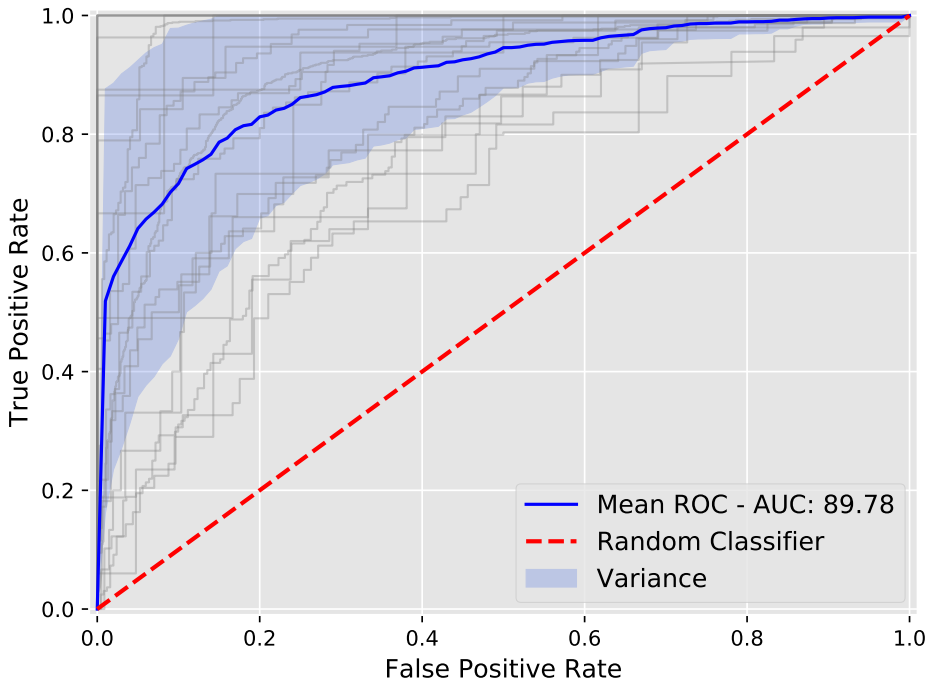

Figure 4.3: ROC curve for classifying relations

how and to what extent our approach can benefit from the sampling strategies presented in Section 4.4.4

Experimental setup. With $\mathcal{I}_G(\rho)$ being the set of all instances created by a rule ρ and L being the set of all labeled instances used to train a model for rule ρ, we start with an empty training set ($L = \emptyset$) and incrementally extend it with new instances $j \in \mathcal{I}_G(\rho)$. After adding j to L, we use L to train a new model. To test it, we make a prediction on the full set of instances $\mathcal{I}_G(\rho)$, which includes the instances added to the training set L. In this way, we create a realistic evaluation scenario by imitating the application of our system to a KG (where the instances used for training remain in the graph). Running this evaluation scheme results in L eventually containing the same elements as $\mathcal{I}_G(\rho)$, so that the last model should reach an F_1-score of 100%.

If a model cannot achieve an F_1-score of 100% when training and predicting on identical data, it is likely that its capacity is not enough to capture the information to perfectly separate the classification space. We examine this scenario in more detail in Section 4.6.3.

Evaluation. Figure 4.4 shows how the different sampling strategies affect precision, recall, and F-measure. As expected, the random selection of training instances proves to be the worst strategy. Although its precision initially increases to 78%, it then decreases and remains, on average, 3.93% and 6.22% below the precision value of GP-UCB and maximum uncertainty (MAXU), respectively. In terms of F-measure, the random

sampling strategy lags 4.42 and 6.72 percentage points behind the GP-UCB and MAXU strategies, regardless of the percentage of training instances.

GP-UCB exceeds MAXU in recall between 5–12% by an average of 2.22 percentage points. To achieve a competitive precision value of 77.44%, GP-UCB requires up to 6% of the training instances but remains below MAXU's precision value. While the average difference of 2.31% to GP-UCB is rather small, the MAXU strategy provides the best results in terms of F-measure, outperforming the other two strategies at all times.

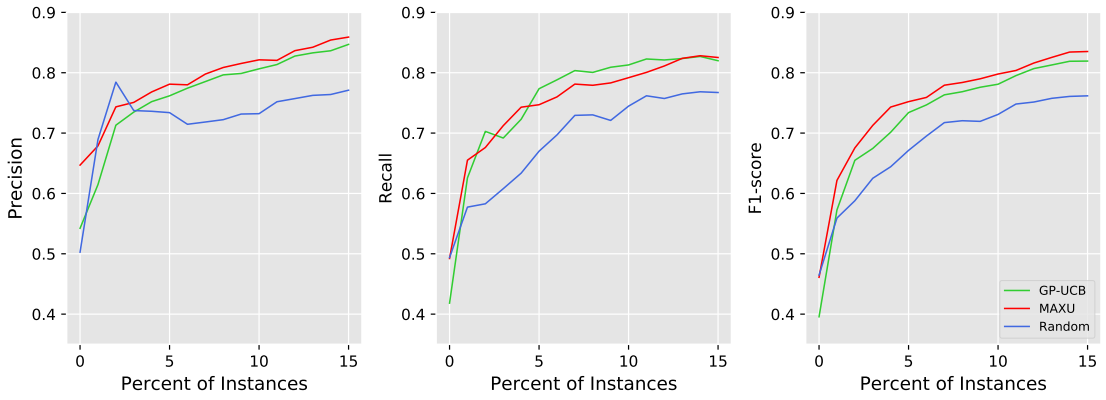

Figure 4.4: Impact of sampling strategies on precision, recall and F_1-score

GP-UCB's inability to achieve a better performance than MAXU can be attributed to the β_t parameter choice discussed in Section 4.4.4. While the formula presented by Srinivas et al. [2010] for selecting the β_t parameter provides a theoretically good starting point, it appears to be too conservative for our problem. As a result, the trade-off between exploration and exploitation is not ideal, causing GP-UCB to make overly conservative decisions, which prevents it from exceeding the MAXU strategy in terms of F-measure. Since the MAXU strategy achieves an F-measure of 75% with only 5% of the training instances, we apply this strategy in all subsequent experiments.

4.6.3 Model performance

This experiment first analyzes the behavioral change of Colt-GP w.r.t. the amount of training data, and, second, compares its performance to the Colt-MC algorithm presented in Section 4.4.1 as well as two active learning baselines. We aim to answer whether the validation problem can already be solved by simple active learning methods or by purely relying on the similarities between knowledge graph embeddings, as done by Colt-MC.

Figure 4.5 shows precision, recall, and F-measure curves of all algorithms for an increasing amount of training data. As the plots show, AL-LogReg and AL-SVM quickly fall behind the Colt algorithms. Although AL-SVM provides better results than AL-LogReg, which is due to its non-linear RBF kernel, neither approach fully captures the complexity of the problem. Thus, after reaching 15% (AL-SVM) and 20% (AL-LogReg) of the training data, improvements start to stagnate, so that the respective F-measures

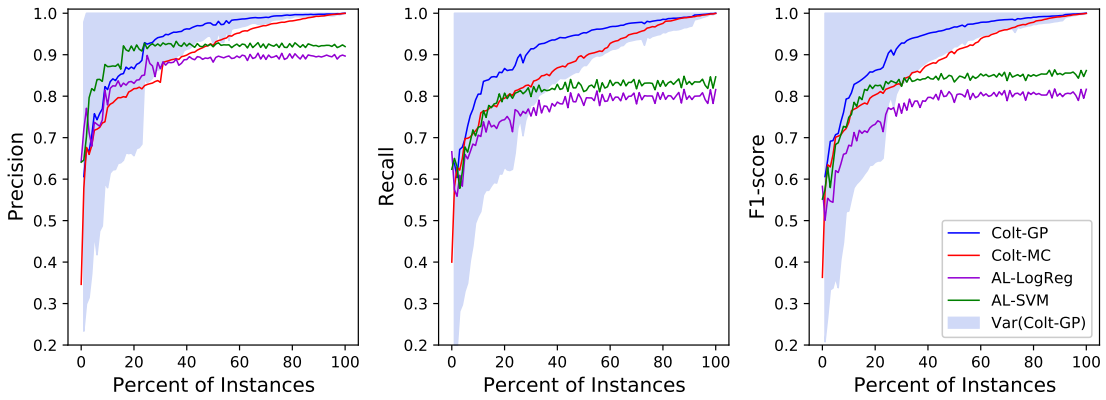

Figure 4.5: Performance of Colt-GP, Colt-MC and active learning baselines

fluctuate around 84% and 80%. In contrast, Colt-GP and Colt-MC outperform the active learning approaches and continue to improve as training data increases.

While Colt-GP exceeds Colt-MC in all plots, it shows, in particular, a faster increase than Colt-MC. Regarding the F-measure, the performance of Colt-GP grows, on average, 1.54% faster than Colt-MC in the ranges between 5-8% of used training instances and slows down to 0.85% in the range 22-25%. In contrast, the F-curve of Colt-MC increases almost linearly between 12% and 100%.

As expected, we observe that the variance of Colt-GP's predictions decreases with the amount of training data, as the model becomes increasingly confident in its prediction. In terms of F-measure, Colt-GP achieves 100% when all training data is used, proving that the model is expressive enough to capture the full complexity of the problem. Using only 5% and 10% of the training data, Colt-GP achieves 75% and 80% F-measure. We measured the most significant improvement over Colt-MC when using 31% of the training data, resulting in an F-measure of 92.8% compared to 80.2% produced by Colt-MC. This represents an improvement of 12.6% in F-measure. Looking at the evaluation of all four approaches, we conclude that it is neither sufficient to solely rely on the similarities between knowledge graph embeddings nor to apply simple active learning techniques to solve the problem adequately.

4.6.4 Rule confidence estimation

We compare the rule confidence of Colt-GP with the data-driven standard confidence measure (*SCF*) [Galárraga et al., 2015], which is defined as $\frac{\#KG}{\#total}$ in Table 4.1. To this end, we first determine the correct confidence value of each rule based on its annotated facts and use it to calculate the average estimation error across all rules.

As shown by the red line in Figure 4.6, the average *SCF* estimate deviates by 19.6% from the correct confidence value. To calculate the average estimation error of Colt-GP, we train it with an increasing number of instances and calculate the average estimation error for each trained model.

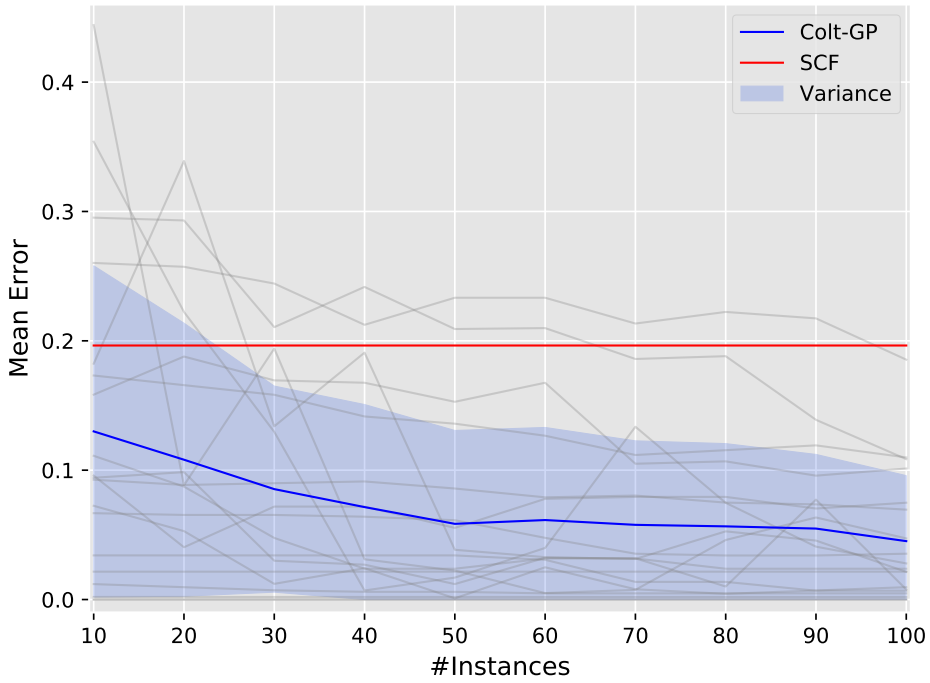

Figure 4.6: Error reduction in confidence estimation using an increasing number of training instances

For this calculation, we use only rules with at least 100 instances, which can be identified in Table 4.1. In addition, we added a light grey line to Figure 4.6, indicating the model's prediction error for each of the rules. The average estimation error for Colt-GP is shown as a blue line in Figure 4.6. The figure shows that, from the very beginning, Colt-GP estimates the average rule confidence more accurately than the *SCF*. As a result, 10 to 20 training instances are sufficient to achieve an average estimated confidence error of 12.9 to 10.8 percent, which corresponds to an improvement of 6.7 and 8.8 percentage points over the *SCF*. As the number of training data increases, the estimation error decreases and reaches its lowest value of 4.5 percent when using 100 training instances. Figure 4.6 also shows that, although the estimated error between 50-100 training instances improves only by 1.3 percentage points, the variance decreases by 3.5%, indicating that Colt-GP gains more confidence in the predictions.

4.7 Summary

This chapter introduced Colt, a few-shot rule-based knowledge validation framework that enables the interactive quality assessment of logical rules. We devised Colt-GP, an algorithm based on Gaussian processes and deep kernel learning, which interactively learns labeling functions on rules. Colt-GP benefits from *knowledge graph embeddings* and integrates *user feedback* to overcome data quality issues of the underlying KG. We compare Colt-GP with two active learning approaches and a baseline built on maximum

coverage and knowledge graph embeddings (Colt-MC). We show that state-of-the-art methods are not sufficient to adequately address data quality issues of the underlying KG. The proposed method achieves a 10% error in the confidence estimation of facts, which requires only 20 user-validated facts. For the task of rule validation, a prediction quality of 75% can be achieved when using only 5% of the annotated rule instances. Finally, we release our dataset of 26 manually annotated rules and 23 000 facts. We note that Colt's focus on rule-driven validation does not prevent its adoption to triples returned by other query mechanisms.

ID	Rule	#KG	#correct	#total	type
1	dealsWith(a, v_0) ∧ isLeaderOf(v_1, b) ∧ wasBornIn(v_1, v_0) ⇒ dealsWith(a, b)	9	24	99	pos
2	dealsWith(v_0, b) ∧ isCitizenOf(v_1, a) ∧ isLeaderOf(v_1, v_0) ⇒ dealsWith(a, b)	42	42	42	pos
3	dealsWith(v_0, b) ∧ isCitizenOf(v_1, a) ∧ isLocatedIn(v_1, v_0) ⇒ dealsWith(a, b)	108	134	134	pos
4	isCitizenOf(v_0, b) ∧ livesIn(v_0, a) ⇒ dealsWith(a, b)	18	664	734	pos
5	diedIn(a, v_0) ∧ isKnownFor(v_1, v_0) ∧ livesIn(v_1, b) ⇒ diedIn(a, b)	808	815	838	pos
6	diedIn(a, v_0) ∧ isLeaderOf(v_1, v_0) ∧ livesIn(v_1, b) ⇒ diedIn(a, b)	2,143	3,135	4,292	pos
7	actedIn(a, b) ∧ created(a, b) ⇒ directed(a, b)	384	388	1,003	pos
8	actedIn(v_0, b) ∧ created(a, b) ∧ directed(v_0, b) ⇒ directed(a, b)	421	414	804	pos
9	hasWonPrize(a, v_0) ∧ hasWonPrize(b, v_0) ∧ hasAcademicAdvisor(b, a) ⇒ hasAcademicAdvisor(a, b)	85	85	85	neg
10	influences(a, b) ⇒ hasAcademicAdvisor(a, b)	1,000	1,000	1,000	neg
11	isCitizenOf(v_0, a) ∧ livesIn(v_0, b) ⇒ hasCapital(a, b)	43	50	734	pos
12	dealsWith(a, v_0) ∧ hasCurrency(v_0, b) ⇒ hasCurrency(a, b)	7	10	293	pos
13	hasCapital(v_0, v_1) ∧ hasCurrency(v_0, b) ∧ isLocatedIn(v_1, a) ⇒ hasCurrency(a, b)	18	46	49	pos
14	dealsWith(v_0, a) ∧ hasOfficialLanguage(v_0, b) ⇒ hasOfficialLanguage(a, b)	52	72	668	pos
15	hasOfficialLanguage(v_0, b) ∧ isLocatedIn(v_0, a) ⇒ hasOfficialLanguage(a, b)	18	62	126	pos
16	hasOfficialLanguage(v_0, b) ∧ isLocatedIn(v_1, a) ∧ livesIn(v_1, v_0) ⇒ hasOfficialLanguage(a, b)	20	52	65	pos
17	hasOfficialLanguage(v_0, b) ∧ isLocatedIn(v_1, v_0) ∧ livesIn(v_1, a) ⇒ hasOfficialLanguage(a, b)	20	16	40	pos
18	influences(a, v_0) ∧ isCitizenOf(v_0, b) ⇒ isCitizenOf(a, b)	146	547	1,382	pos
19	hasChild(a, b) ⇒ isMarriedTo(a, b)	991	990	1,000	neg
20	isLocatedIn(v_0, b) ∧ livesIn(a, v_0) ⇒ isPoliticianOf(a, b)	130	2,257	5,269	pos
21	isCitizenOf(v_0, b) ∧ isLeaderOf(v_0, v_1) ∧ livesIn(a, v_1) ⇒ livesIn(a, b)	416	863	882	pos
22	isMarriedTo(a, v_0) ∧ livesIn(v_0, b) ⇒ livesIn(a, b)	181	442	537	pos
23	actedIn(a, b) ∧ created(a, b) ⇒ produced(a, b)	207	261	1,003	pos
23	actedIn(v_0, b) ∧ directed(a, b) ∧ produced(v_0, b) ⇒ produced(a, b)	246	281	702	pos
25	hasAcademicAdvisor(v_0, a) ∧ worksAt(v_0, b) ⇒ worksAt(a, b)	47	163	543	pos
26	actedIn(a, b) ⇒ wroteMusicFor(a, b)	1000	998	1,000	neg

Table 4.1: Generated rules and their statistics: number of KG facts that satisfy the rule (#KG), number of correct facts, total number of facts, type of the rule (colors are for ease of reading).

Chapter 5

CurEx – Extracting, Curating, and Exploring Knowledge Graphs

As stated in the previous chapters, the generation of domain-specific knowledge bases from structured and unstructured data sources plays an important role in use cases, such as supply chain analysis or the assessment of financial risk factors. Both the creation and continuous maintenance of knowledge bases entails many complex tasks and their challenges.

To meet these challenges, we present CurEx, a prototypical, modular system that allows structured and unstructured data sources to be integrated into a domain-specific knowledge base. The system consists of several components that allow, amongst other things, the extraction of information from newspaper articles and their combination with information from other data sources, such as Wikidata and DBpedia. In doing so, CurEx is not only able to integrate structured data sources, but also to recognize named entities in unstructured sources, link them to entries in the knowledge base, and extract relationships between the identified entities in the text. Each of the system's components addresses a specific subproblem of the knowledge base construction process and can be easily exchanged due to its modular system architecture. This design decision ensures that future innovations in the respective research areas can be easily be adopted. The constructed knowledge base can then be used to generate domain-specific knowledge graphs. Using CurEx,[1] we were able to create an integrated knowledge base that comprises almost 2.1 million entities and roughly 18 million relationships, including co-occurrences.

In particular, the CurEx system **(i)** enables the incremental improvement of each integration component; **(ii)** enables the selective generation of multiple knowledge graphs from the information contained in the generated knowledge base; and **(iii)** provides two distinct user interfaces tailored to the needs of data engineers and end-users respectively. The former has curation capabilities and controls the integration process, whereas the latter focuses on the exploration of the generated knowledge graph.

[1] `https://github.com/bpn1/ingestion`

The content of this chapter is based on [Loster et al., 2018b] and is structured as follows: In Section 5.1, we give an overview of the CurEx system architecture and discuss the concrete implementations of both the structured and unstructured integration components. We introduce the different user interfaces and how they can be used to interact with the system in Section 5.2. Section 5.3 outlines typical use cases that are covered by CurEx, and Section 5.4 concludes this chapter.

5.1 System architecture

Since the goal of CurEx is to integrate information from structured and unstructured data sources, it is divided into two main components that are specifically tailored to the integration of the respective data sources. Figure 5.1, which is repeated for convenience, shows a general overview of the system. CurEx is entirely based on scalable technologies, such as Apache Spark and Cassandra, and, therefore, able to process large amounts of data. Another essential aspect of the system is its modular architecture. This modularity is achieved by implementing all components as Spark-jobs, making them easy to extend and replace. While the component for *structured data integration* (Chapter 1&2) focuses on integrating structured data sources, the *text mining* component (Chapter 3) handles the information extraction from unstructured data sources and their subsequent integration into the knowledge base.

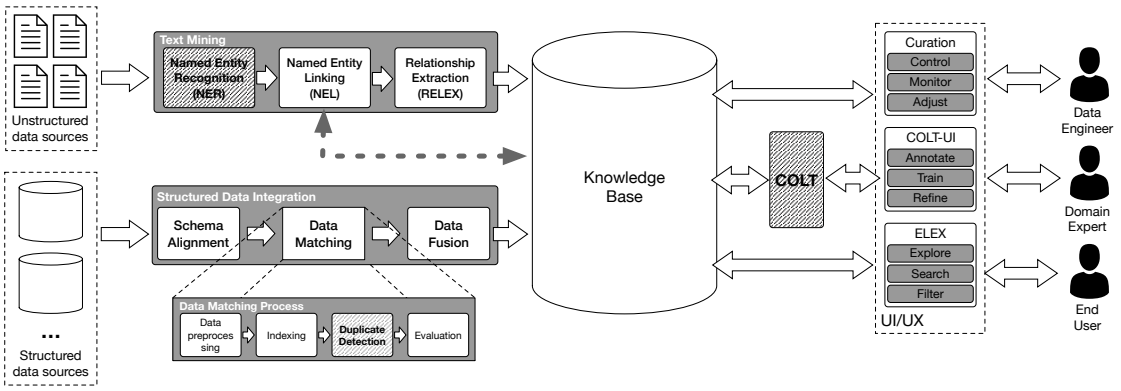

Figure 5.1: Overview of the CurEx system architecture

As shown in Figure 5.1, the *text mining* component consists of the sub-components *named entity recognition* (NER), *named entity linking* (NEL) and *relation extraction* (RELEX), while the component for integrating structured data sources includes components for *schema alignment*, *data matching* and *data fusion*. Through the coordinated execution of these subcomponents, CurEx is able to build up large knowledge bases that can then be used to create a knowledge graph consisting of entities and their relations. To efficiently access the information in this knowledge graph, it is exported to a graph database, such as Neo4j. This export allows the selective creation of multiple knowledge graphs that cover only certain parts of the knowledge base. This requirement may be necessary due to security clearances or other restrictive criteria.

CurEx also introduces two distinct user interfaces customized for the needs of end-users and data engineers. End-users can use the *Entity Landscape Explorer* (ELEX) to explore the resulting knowledge graph and provide valuable feedback to data engineers. Data engineers can use the *Curation Interface* to not only control and monitor many steps of the integration process but also to directly make changes to the generated knowledge base. In addition, CurEx is fully compatible with the COLT framework introduced in Chapter 4, which provides its own user interface and integrates seamlessly with CurEx.

5.1.1 Structured data integration

The purpose of integrating structured data is to merge multiple structured data sources into one while consolidating all entities that refer to the same real-world entity. This section describes the concrete implementation of the individual components comprising the structured integration process as it is implemented by CurEx. For the development of an initial knowledge base, we integrate the following structured data sources: the German Wikidata, the German DBpedia, Implisense[2], and Kompass[3]. The latter two data sources are manually curated commercially available company datasets. We choose the Implisense data source as the starting point for our integration efforts since it is manually maintained by an industry partner and thus of high data quality. We then continue to integrate the other data sources by executing the structured data integration process whose individual steps are described below.

Schema alignment

As the schemata of the individual data sources to be integrated can differ significantly from each other, it becomes necessary to consolidate them into a common *global schema* before the integration can take place. This necessity arises from the fact that the individual data sources often use different column names to refer to semantically equivalent concepts. Thus, some data sources refer to the concept of geo-coordinates as "coordinate_location", others as "geo.coords" or "lat, long". To address this issue, the *schema alignment* step seeks to unify the distinctions between the entity schemata of the different data sources. For this purpose, columns containing semantically similar information are not only grouped under a common name but can also be merged into new columns if required. While the CurEx system expects a manually created schema mapping that both specifies a *global schema* and defines how each data source is mapped to this schema, the modular design of the system allows this component to be replaced by any schema matching approach [Bellahsene et al., 2011; Euzenat and Shvaiko, 2013].

Data matching

As can be seen in Figure 5.1, CurEx implements the *data matching* process presented in Chapter 2. Thus, the pipeline consists of the steps *data preprocessing*, *indexing*, *duplicate detection*, and *evaluation*.

[2] `https://implisense.com`
[3] `https://de.kompass.com`

Data preprocessing. Entities that pass through the *data matching* process frequently differ in various aspects, depending on their origin. The *data preprocessing phase* takes care of these variations by eliminating the differences between the values of the integrated data sources. To give an example: the values of the *legal form* column vary considerably across different data sources – some containing full company suffixes such as "Limited Liability Company", while others use abbreviations such as "LLC" or "l.l.c.". Before inserting these data values into the previously constructed *global schema*, they are converted into a standardized format. To this end, the CurEx preprocessing step involves a range of operations, such as the deletion, shortening or expansion of specific phrases, the resolution of measurement units, the application of regular expressions as well as more sophisticated methods such as the decomposition of company names into their constituent parts [Loster et al., 2018a]. A concrete example is provided by Table 5.1, which shows different data formats for geographic coordinates listed by data source. During the preprocessing phase, the different coordinate representations get normalized into the common format `[latitude];[longitude]`. As can be seen in the example, the normalization can even cause values from multiple columns to be merged into one new value (e.g., lat, long ⇒ normalized). Note that many of these preprocessing steps are created manually so that CurEx should not be considered a tool for data preprocessing.

Data Source	Coordinate column values
WikiData	52.3906;13.0645
DBpedia	lat: 52.3906^^xsd:float long: 13.0645^^xsd:float
Implisense	lat: 52.3906 long: 13.0645
Kompass	*n.A.*
normalized	52.3906;13.0645

Table 5.1: Geographic coordinates values

Indexing. To create a duplicate-free knowledge base, entities that refer to the same real-world entity must be merged during the deduplication process. As each data source in our examples contains several hundred thousand entities, a complete pairwise comparison of all entities is not feasible. To minimize comparisons, CurEx utilizes a standard blocking technique that essentially groups similar objects into smaller blocks according to a specific blocking criterion (e.g., zip code). In this way, a quadratic comparison can be carried out within each block. Although the blocking criterion can be customized to fit specific requirements, by default, CurEx uses a prefix-based blocking criterion. Specifically, the first three to five characters of the individual values are used to assign all entities with a common prefix to the same block.

Duplicate detection. With the entities grouped into smaller blocks, CurEx performs an exhaustive duplicate comparison within each block to find duplicates. To this end,

the component for duplicate detection is capable of comparing any attribute subset, such as names, unique identifiers, or URLs. When determining the similarity of two entities, we employ different similarity measures, such as Jaro-Winkler [Winkler and Thibaudeau, 1987], MongeElkan [Monge and Elkan, 1996], or the Euclidean distance between coordinate vectors. We first determine the similarity of each attribute pair and then combine the resulting similarity values by means of a simple linear combination. Finally, we normalize the calculated value to the range [0,1] and use the resulting value to decide whether two entities are duplicates or not. While the current version of CurEx implements this simple deduplication approach, its modular design allows us to use more advanced deduplication strategies such as SNNDedupe (Chapter 2) in future versions.

Evaluation. To evaluate the performance of the data matching pipeline, CurEx uses the standard performance metrics precision, recall, and F_1-measure, which are determined for each deduplication run and can be accessed via the Curation Interface.

Data fusion

As conflicting values can occur during the consolidation of individual data sources, the *data fusion* component decides how these values are to be merged. To this end, CurEx uses a relatively simple fusion policy that relies on consolidating conflicting values according to a manually specified sequence of participating data sources. That is, if there are conflicting values, they are merged in the following order: Implisense, Kompass, Wikidata, DBpedia. This sequence can be specified manually and determines which values from which data source are more trustworthy and thus receive a higher priority during integration. If there are no conflicts, the values of the different data sources are simply combined into one entity, which then contains all values. If a field allows multiple values, as is the case for enumerations, they are simply appended to the corresponding field. Given the use case presented in Section 1.1, we used this fusion policy to create a knowledge base of 2.1 million companies, of which about 29,000 companies occur in exactly two and 3,700 companies in exactly three data sources.

5.1.2 Text mining

After a first version of the knowledge base has been created by integrating the structured data sources, we systematically extend it by including entities and relations extracted from unstructured data sources. To this end, we use various text mining techniques to recognize and link named entities and extract relationships between them. This task is carried out by the *text mining* component, as presented in Chapter 3. As the text mining component is divided into the three sub-modules *named entity recognition* (NER), *named entity linking* (NEL), and *relation extraction* (RELEX), we briefly describe their concrete implementation in the following.

Named entity recognition. It is the task of the NER component to discover named entities in unstructured texts. To do this, we use a technique similar to the one described

in Section 3.1. We first create large dictionaries of externally available knowledge, which can then be integrated into the training process of a conditional random field classifier used to discover the entities of interest.

Named entity linking. As discussed in Section 3.7, the entities extracted by the NER component can refer to many possible entries in the knowledge base. Therefore, it requires a linking step to establish a unique assignment to a corresponding knowledge base entry. The current implementation of this component is based on a fuzzy matching approach, which uses different string similarities to match identical entities. However, to discover links between the knowledge base entries and German Wikipedia articles, we use the more sophisticated CohEEL approach [Grütze et al., 2016]. Since this approach also considers the context of a company mention, it allows us to find the best of several possible knowledge base matches. This turns out to be particularly useful for linking companies operating in different sectors and have very similar names, as their textual context can often be used as an indicator for a precise match.

Relation extraction. Finally, it is the focus of the relation extraction (RELEX) component to detect relationships between the previously discovered entities. The component currently in use extracts co-occurrence relationships between entities found within the same sentence. Due to its modular architecture, future CurEx versions can employ techniques, such as the one presented by Zuo et al. [2017] or one of the state-of-the-art approaches outlined in Section 3.8. In particular, the approach proposed by Zuo et al. is promising, as it is able to extract directional relationships where the arguments are of the same type, for example, company-to-company relationships. Since the entities were already linked to the knowledge base in the previous step, the extracted relationships can easily be integrated into the knowledge base.

5.2 Interface & Interactions

Every real integration system needs to adhere to the requirements of different user groups. As such, data engineers must be able to control the individual steps of the integration process and directly adopt the knowledge base, while end-users of the integrated data are more interested in methods for efficient data exploration. To this end, CurEx provides two separate user interfaces, specifically tailored to the needs of two user groups: data engineers and end-users. The Entity Landscape Explorer (ELEX) is designed for the end-user and allows him/her to efficiently explore and inspect the knowledge base by browsing through the generated knowledge graph. The Curation Interface, on the other hand, is aimed at data engineers and enables them to control the integration process and to curate the knowledge base. Since an essential aspect of generating knowledge bases is to determine their data quality and to improve it if necessary, the Colt framework presented in Chapter 4 provides a dedicated user interface (Colt-UI) for domain experts, which is also covered in this section. This interface allows domain experts to annotate rule-generated facts required by the Colt framework to improve the data quality of the underlying knowledge base.

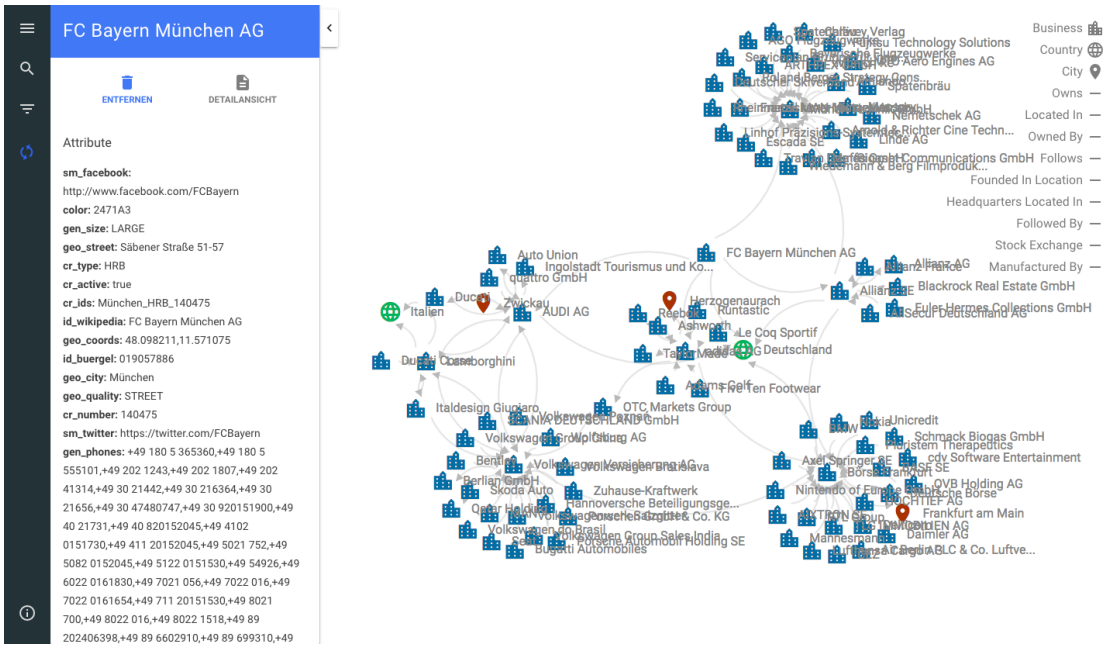

Figure 5.2: The Entity Landscape Explorer (ELEX) enables end-users to graphically explore the created knowledge base. As such, a user can examine not only entities and their relationships, but also their attributes.

5.2.1 Entity landscape explorer

The Entity Landscape Explorer (ELEX[4]) shown in Figure 5.2 is designed to meet the end-user's needs. Since a knowledge graph can easily contain thousands of nodes and edges, a user must be able to examine the graph and its associated knowledge base efficiently. ELEX provides this functionality by allowing the user to explore the graph using appropriate filtering, layout, and search methods.

The starting point for such a focused exploration is an initial node that the user can select through a search interface. To focus the exploration even more, the user can limit the exploration to nodes of a particular entity type as well as to certain edge types. When examining a specific subgraph, nodes and edges can be filtered out dynamically, revealing structures that were difficult to recognize beforehand. For a better overview, the currently displayed nodes can also be arranged in a tree structure, which is particularly useful when analyzing customer-supplier relationships. To thin out the graph even further, it is possible to show or hide nodes and edges of certain types completely. Another key feature of ELEX is the ability to examine the individual attributes of each node and edge. These attributes provide access to the information in the knowledge base and can be used to gain a better understanding of the displayed relationships and guide the exploration of the knowledge graph. Since the information stored in the knowledge base is not necessarily error-free, it is an integral part of ELEX to allow the user to report incorrect entries through a feedback mechanism. The corrections made are then

[4] `https://github.com/HPI-Information-Systems/ELEX`

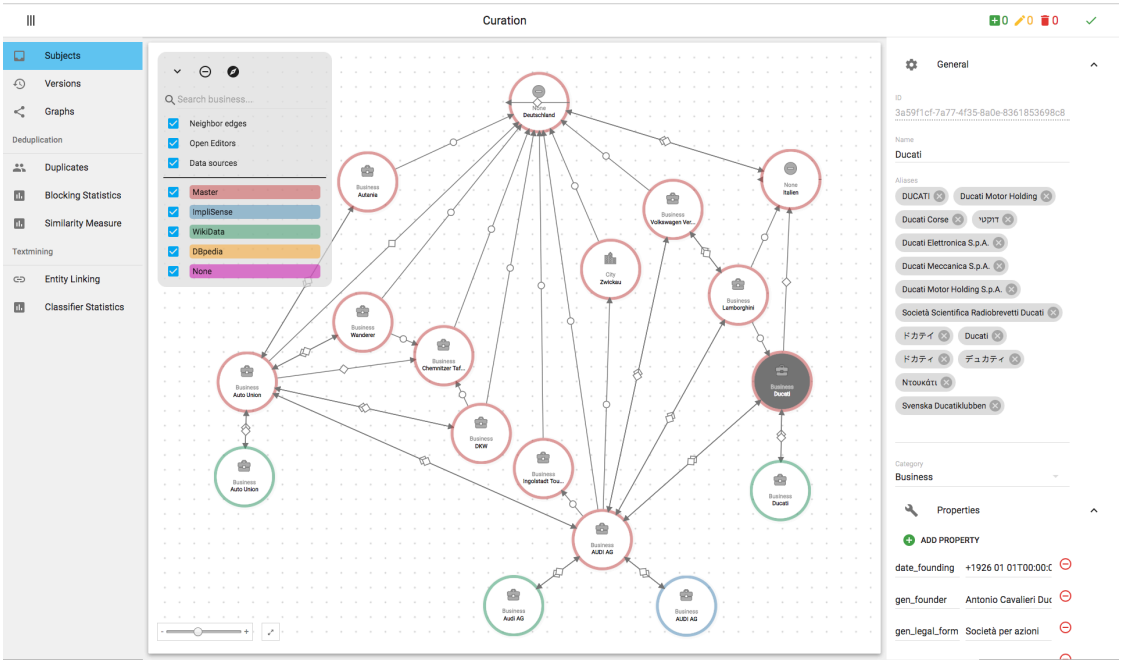

Figure 5.3: The Curation Interface is directed to the data engineer and allows to control the knowledge base creation. It offers functionality for monitoring individual processing steps, making changes to the knowledge base and displaying the statistics of certain components.

forwarded to the data engineer as suggested corrections, which can then be incorporated into the knowledge base via the Curation Interface.

5.2.2 Curation interface

The functionality of the Curation Interface[5] shown in Figure 5.3 is tailored to the needs of data engineers. This user group is interested in controlling the actual integration process, inspecting the resulting knowledge base, and, if necessary, making changes to improve information quality. First, it must be possible to minimize the errors caused by successive data transformation steps, and, second, it is important to enable the correction of errors that were already present in the original data sources. The Curation Interface was developed with these two aspects in mind and offers appropriate functionalities to address them. As shown in Figure 5.3, the interface provides several tabs, which are divided into different groups. The "Subjects", "Versions", and "Graphs" tabs are used to curate the knowledge base. Among other things, they allow the user to make changes directly in the knowledge base, roll back the knowledge base to an earlier state, and edit the entities in a graph view, simplifying the handling of relationships. The "Duplicates", "Blocking Statistics", and "Similarity Measure" tabs, on the other hand, are used to display and evaluate the results of the deduplication component. These tabs show the results of a duplication run between the knowledge base and a new data source, provide

[5] `https://github.com/bpn1/curation`

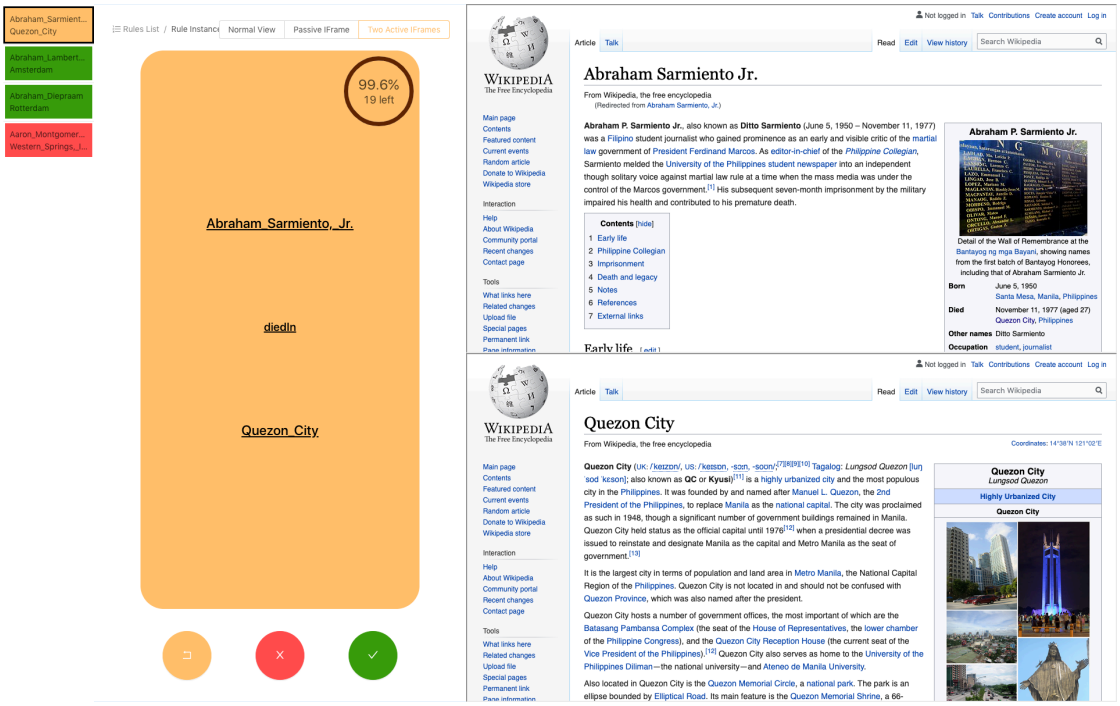

Figure 5.4: The COLT user interface allows domain experts to verify facts which are used by the COLT framework to improve the data quality of the knowledge base.

blocking statistics to estimate the cluster utilization, and show precision, recall, and F-measure of the currently used similarity measure in case a gold standard is available. Finally, the text mining group consists of the "Entity Linking" and "Classifier Statistics" tabs, which make it possible to view the results of the entity linking subcomponent and to evaluate various classification models.

5.2.3 COLT user interface

To address the fact that errors are likely to occur when machine learning models are used to construct integrated knowledge bases, we introduced the COLT framework in Chapter 4. This framework can be used to correct some of these errors in order to improve the overall data quality of the created knowledge base. As the application of this framework requires the annotation of facts, the COLT-UI shown in Figure 5.4 provides the necessary annotation interface. According to the process presented in Section 4.4, the interface first requires the selection of a specific rule for which a model is to be learned. The user is then presented with facts that are generated by the execution of the selected rule. Next, the user has to verify the correctness of the individual facts and record the result by using the UI. To assist the user in his decision-making process, the COLT-UI displays the participating entities' linked websites. The intention of providing this information is to make it easier for the user to verify the correctness of the presented facts. In cases where the user is unable to verify a fact based on the provided information, external information sources, such as Google searches, need to be consulted. Driven

by the constantly increasing number of annotated facts, a continuously improving rule model emerges, whose application can be used to improve the information quality of the underlying knowledge base. While developed separately from the CurEx system, the technology stack used to implement the COLT-UI is fully compatible with the one used for the Curation Interface allowing a seamless integration of the COLT-UI into the Curation Interface.

5.3 Typical use cases

This section outlines some of the core use cases that CurEx is designed to cover. Typical use cases include the maintenance of the knowledge base, monitoring the individual data processing steps, and the subsequent data exploration using the ELEX graph exploration tool.

Data inspection. The first use case is to give the user an overview of the entities contained in the knowledge base. To this end, a user can search for specific entities and is presented with various views on the available information. As shown in Figure 5.3, the graph view can be used to explore a subgraph of selected entities, revealing relationships between and information about different entities.

Data curation. In this use case, the user can use the graph view's curation capabilities to change the attribute values of a selected entity. Besides updating existing entities, it is possible to add new entities or delete existing ones. Using the status view, the user can inspect the changes made. By accepting the listed operations, the user initiates their execution, thereby altering the knowledge base. As described in the previous section, the user may also use the COLT-UI to select a rule and annotate several generated facts. The resulting rule model can then be used to curate the knowledge base by adding or removing missing or incorrect facts.

History examination. In case of an error, the user can inspect the history of the different data processing steps applied to the knowledge base. The user can thus rewind the state of the knowledge base to a certain point in time, for example, to undo the effects of erroneous changes. This functionality can be used to ensure the integrity of the knowledge base.

Process monitoring. In addition to the operations mentioned above, monitoring tools are provided to assist the user in inspecting the intermediate steps of the deduplication and text mining components. Thus, he/she can examine duplicates discovered in previously performed deduplication runs, evaluate the effects of different blocking keys on the deduplication process, and set different thresholds to be used during duplicate detection. In addition, the entity linking view can be used to understand the relationship between text documents and the linked entities from the knowledge base. Using the evaluation

tool, a user can analyze different classification models as well as the parameter settings used by the text mining component.

Exploration. After completing the curation process, a common requirement is to explore the integrated knowledge base. The Entity Landscape Explorer, in short ELEX, was designed precisely for this purpose. Thus, a user can use ELEX to explore and access information about the entities stored in the knowledge base. By using the powerful display and exploration capabilities of ELEX, a user can inspect single entities, display the knowledge graph in different layouts, or point out errors in specific nodes and edges. For example, the user can report an error in a data field or an incorrect relationship. In contrast to the data inspection use case, the data exploration via ELEX is primarily focused on the end-user and does not support direct modifications of the knowledge base.

5.4 Summary

In this chapter, we introduced CurEx, a prototypical, modular system to integrate structured and unstructured data sources into a domain-specific knowledge base that can be used to create explorable knowledge graphs for specific domains. The system is based on scalable technologies and can process large amounts of data, making it suitable for real-world scenarios. It consists of two main components specifically designed for integrating information from structured and unstructured data sources. Since all subcomponents are implemented as Spark-jobs, they are easily replaced or extended. It provides two distinct user interfaces, each addressing the individual needs of a specific user group. As such, a data engineer can control and manage the integration process using the Curation Interface, whereas a typical end-user uses ELEX to explore the knowledge graph and submit feedback. Furthermore, CurEx is fully compatible with the Colt framework, which can thus be used in conjunction with CurEx to improve the data quality of the generated knowledge base.

Chapter 6

Conclusion and Outlook

Modern knowledge bases store and organize knowledge from many different topic areas. They contain not only information about entities, but also information about their relationships, allowing the creation of a knowledge graph. This property makes them particularly valuable in cases where relationships play a central role, such as improving ad placements or recommendations. In addition to these well-known use cases, modern risk assessment in financial institutions can also benefit greatly. In this case, the inherent network structure of knowledge bases can be used to assess both the impact and the risk potential of various events, such as corporate insolvencies or fraudulent behavior. As public knowledge bases often lack the information needed to analyze the effects of such events, it becomes necessary to create and maintain special domain-specific knowledge bases.

This thesis investigates the process of creating domain-specific knowledge bases from structured and unstructured data sources. It contributes to the areas of *named entity recognition*, *duplicate detection*, and *knowledge validation*, which represent essential building blocks for the construction of knowledge bases.

We began this thesis with an introduction to knowledge base construction and presented the most important use cases that served as motivation for this work. We then highlighted the challenges associated with creating knowledge bases from structured and unstructured data sources. We continued by giving an overview of the thesis structure and its contributions. In detail, the contributions of this work can be summarized as follows:

Integrating structured data. In Chapter 2, we focused on the integration process of structured data sources, where a novel method for detecting duplicate entities formed the core of this chapter. We proposed a Siamese neural network called SNNDedupe, capable of learning a dataset-specific similarity measure that can be used to identify duplicates. The properties of the model made it possible to eliminate manual feature engineering and reduce the manual effort needed for model training. When comparing SNNDedupe with a traditional SVM approach and an approach based on neural networks, we were able to show that SNNDedupe significantly outperforms the traditional approach while remaining on par with its neural network competitor. Next, we conceived and implemented

a knowledge transfer between two deduplication networks, which made it possible to transfer knowledge from one network to another. The specialized network architecture allowed us to transfer weight matrices of selected attributes, which led to significant performance improvements. In a final experiment, we showed that the amount of required training data can be reduced by carrying out a knowledge transfer in advance.

Future work. Although we were able to show that weight matrices of attributes can be transferred from one network to another, attributes for which no pre-trained weight matrices are available need to be initialized randomly, which is not ideal. This problem becomes particularly severe if the target dataset possesses a large number of attributes, of which only a few can be initialized with pre-trained weights. In this regard, it would be interesting to investigate whether the knowledge for uninitialized attributes can be acquired in an unsupervised manner, e.g., by applying sequence-to-sequence learning techniques [Sutskever et al., 2014]. By training an encoder-decoder network on a set of attribute values, a feature vector forms between encoder and decoder, representing the condensed knowledge of the attribute values. This vectorized knowledge could then be used to initialize uninitialized attributes of the target dataset to improve the deduplication performance even further.

Another interesting aspect that could be investigated is how multi-task learning can be applied in the context of duplicate detection [Caruana, 1997]. To this end, it would be conceivable to extend SNNDedupe so that multiple loss functions are taken into account, thus providing the possibility to learn specialized similarity measures between attribute pairs.

The adaptation of the mechanisms presented in Chapter 4 should also be considered to further improve SNNDedupe. In detail, the last layer of the presented deduplication network could be replaced by a layer consisting of Gaussian processes. This change would result in a network that provides confidence estimates in addition to its predictions. As described in Chapter 4, these confidence estimates could then be used by an active learning framework to ask for the annotation of duplicates that are particularly valuable for the network's learning process. Moreover, by combining neural networks with Gaussian processes, the strengths of both models can be exploited.

Integrating unstructured data. Chapter 3, discussed the integration process of unstructured data sources and presented a method for named entity recognition, capable of identifying company names in textual data with high lexical complexity. The distinguishing aspect of the designed approach is that it is able to integrate external knowledge in the form of dictionaries into the training process of a conditional random field (CRF) classifier. Apart from designing and creating the actual NER system, we also studied the impact of different dictionaries on the performance of the NER classifier. Our experiments have shown that significant performance improvements can be achieved by incorporating additional domain knowledge. Furthermore, we were able to show that the generation and use of alias names led to an increase in recall while maintaining precision.

Future work. Although enormous progress has been made in natural language processing since 2017, there are still opportunities to further develop the proposed method. As such, future work on this topic should consider nested named entity recognition (NNER)

for dictionary preparation. By adding additional semantic knowledge of the name components, the dictionary quality could be improved. To this end, the method presented by Loster et al. [2018a] could be used to decompose names into their constituent parts.

Furthermore, the token trie could be extended to include different entity types, such as brands or products. If this trie is used as a blacklist, it becomes much easier to decide whether a token sequence should be marked as valid entity or not. The fact that the best results were obtained while using the smallest dictionary suggests that the dictionaries' characteristics should match those of the text corpus. Thus, the creation and use of dictionaries adapted to the characteristics of the corpus under consideration seems promising, as they have the potential to further improve the results.

Knowledge validation. The subject of knowledge validation was covered in Chapter 4. Here, we introduced Colt, a rule-based framework for knowledge validation based on the interactive quality assessment of logical rules. By combining Gaussian processes with neural networks, we created Colt-GP, an interactive algorithm for learning labeling functions from rules. To this end, Colt-GP uses knowledge graph embeddings and user feedback to cope with data quality issues of the underlying knowledge graph. The rule model learned by Colt-GP can be used for both the conditional application as well as the quality assessing of the rule. By comparing Colt-GP with a baseline of maximum coverage (Colt-MC) and two active learning approaches, we were able to show that the latest approaches are unable to adequately address data quality issues in the underlying knowledge graph. In our experiments, we demonstrated that Colt-GP is capable of estimating a rule's confidence with less than 20 user interactions. For rule validation, our experiments have shown that Colt-GP makes good predictions despite the availability of only a small number of annotated rule instances.

Future work. While Colt-GP focuses on learning the properties of each rule individually, the next logical step should be to extend the framework so that the characteristics of multiple rules can be learned simultaneously, thereby reducing annotation costs even further. Thus, a promising extension of the current model would be to equip it with meta-learning capabilities, such as those provided by neural processes [Kim et al., 2019].

Another way to improve the framework is to extend the feedback loop so that the generated models are directly used to improve the knowledge base. The improved knowledge base could then be used to create a new set of improved rules, which, by reapplying the framework, would allow the creation of even better rule models. This extension would facilitate an iterative improvement of the knowledge base. Finally, it could be promising to integrate the presented framework with rule generation approaches, such as AMIE or RuDiK. This integration would allow the learned models to be used for creating more precise and reliable rules.

CurEx – A system for building knowledge bases. In Chapter 5, we presented CurEx, a prototypical system for constructing domain-specific knowledge bases from structured and unstructured data sources. The experiences gained in the course of this thesis were taken into account when designing the system. Thus, it has a modular design and is based on scalable technologies, so that the processing of large datasets, as

they occur in real scenarios, represents no obstacle. It essentially consists of two main components designed specifically for the integration of information from structured and unstructured data sources. Since all subcomponents are implemented as Spark-jobs, they can easily be replaced or extended. CurEx offers multiple user interfaces, each tailored to the individual needs of a specific user group. Thus, a data engineer can control, monitor, and manage the integration process using the *Curation Interface*, while a typical end-user can use the *Entity Landscape Explorer* (ELEX) to explore the knowledge graph and provide feedback regarding its entities and relations. In addition, CurEx is fully compatible with the Colt framework presented in Chapter 4, so that it can also be used as part of the system to improve the information quality of the generated knowledge base.

Future work. In the future, the system could be extended by additional components. For example, a component for profiling structured datasets or a component for preprocessing data coming from structured and unstructured data sources could be developed. Each component could define its own user interface, which could then be centrally accessible via the Curation Interface. Although the system is modular, the current system design still requires extensions to be written in Python or, better yet, Java or Scala. This can be a major limitation, as cross-language functionalities are often challenging to implement. Thus, it would be useful to bring the system to a higher abstraction level by implementing the individual modules based on container technologies, such as Docker. This approach would also simplify the deployment of CurEx in corporate environments.

Apart from these concrete contributions and the ensuing further research, there are more general developments and trends that are likely to impact the area of knowledge base construction in the future, some of which are presented below:

Human in the loop. The use of machine learning systems requires constant adaptation either by designing tailor-made models or by retraining or continued training of models on constantly growing datasets. This also applies to machine learning-based systems for knowledge base construction. In this scenario, human interactions can be used to dynamically train and adapt the machine learning models used. As shown in Chapter 4, active learning or human-in-the-loop approaches can create useful models with only a small number of interactions. Considering this property, it is expected that the use of human feedback will become increasingly important for industrial adaptation and, thus, for the productive operation of machine learning-based systems. As active learning approaches become more popular, it is getting increasingly easier to collect additional training data, which will ultimately lead to the continuous adaptation and improvement of the applied models. Following this trend, the models presented in Chapters 2 and 3 could be extended by an active learning approach.

Joint model training. Looking at current developments in designing machine learning models (e.g., for information extraction), it becomes increasingly clear that, at least for network-based models, there is a trend towards joint, end-to-end model training, which is likely to gain more importance in the future. This approach makes it possible to counteract the error propagation, as it occurs in traditional model pipelines. For this

purpose, each model is first pretrained separately on a specific task, such as NER or NEL, before they are combined to form a model that addresses the target problem, for instance RELEX. This composite model is then trained, with errors propagating throughout the network, so that each of the individual model components adapts to the target task in dependence of the error. This represents an advantage over conventional model pipelines, which, unlike the jointly trained models, are difficult to adapt to a specific task. This lack of adaptability, in which errors on the target task do not effect other model components, leads to errors that can hardly be eliminated without joint training. Using this approach, the pipeline presented in Chapter 3 could be merged into one integrated model, which would most likely lead to additional performance improvements.

Explainability. Another point that should be the subject of intensive future research efforts is the explainability of complex models. This point is mainly driven by the increasing complexity of machine learning models. Given their ever-increasing number of parameters, these models often behave like a black box, making predictions incomprehensible for humans. Consequently, they are difficult to use in high-risk environments, such as medical or financial applications, where the explainability of the applied models is often required by law. Thus, in the context of knowledge base creation, it would be desirable to understand why certain facts ended up in the knowledge base. Therefore, the use of complex models requires the development of methods to explain their behavior. Although with LIME [Ribeiro et al., 2016] or SHAP [Lundberg and Lee, 2017], there are some promising approaches that explain the complexities of a model, they mostly focus on explaining specific model aspects, leaving room for future research.

While this work adds to several areas of knowledge base construction, other challenges remain that require further research. Finally, we hope that this work contributes to advancing the current state-of-the-art towards the goal of a fully automated process for the construction of comprehensive knowledge bases.

References

Asma Abboura, Soror Sahri, Mourad Ouziri, and Salima Benbernou. Crowdmd: Crowdsourcing-based approach for deduplication. In *IEEE International Conference on Big Data (BigData)*, pages 2621–2627, 2015.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 1638–1649, 2018.

Hamed Amini, Rama Cont, and Andreea Minca. Resilience to contagion in financial networks. *Mathematical Finance*, 26(2):329–365, 2016.

Abdallah Arioua and Angela Bonifati. User-guided repairing of inconsistent knowledge bases. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 133–144, 2018.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. DBpedia: A nucleus for a web of open data. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 722–735, 2007.

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. Cloze-driven pretraining of self-attention networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5359–5368, 2019.

Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. Hypernetwork knowledge graph embeddings. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 553–565, 2019.

Krisztian Balog. *Entity-Oriented Search*, volume 39 of *The Information Retrieval Series*. Springer, 2018.

Zohra Bellahsene, Angela Bonifati, and Erhard Rahm, editors. *Schema Matching and Mapping*. Data-Centric Systems and Applications. Springer, 2011.

Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

Darina Benikova, Chris Biemann, Max Kisselew, and Sebastian Padó. Germeval 2014 named entity recognition shared task: Companion paper. In *Proceedings of the KONVENS Workshop GermEval Shared Task on Named Entity Recognition*, 2014.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2546–2554, 2011.

Mikhail Bilenko and Raymond J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 39–48, 2003.

Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. Springer, 2007.

Jens Bleiholder and Felix Naumann. Data fusion. *ACM Computing Surveys*, 41(1):1–41, 2008.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics (TACL)*, 5:135–146, 2017.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 1247–1250, 2008.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2787–2795, 2013.

Jane Bromley, James W. Bentz, Leon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using siamese time delay neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):669–688, 1993.

Rui Cai, Xiaodong Zhang, and Houfeng Wang. Bidirectional recurrent convolutional neural network for relation classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 756–765, 2016.

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300–307, 2007.

Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Frederick Reiss, and Shivakumar Vaithyanathan. Domain adaptation of rule-based annotators for named-entity recognition tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1002–1012, 2010.

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 539–546, 2005.

Peter Christen. A two-step classification approach to unsupervised record linkage. *Proceedings of the Australasian Conference on Data Mining and Analytics (AusDM)*, 70: 111–119, 2007.

Peter Christen. Automatic record linkage using seeded nearest neighbour and support vector machine classification. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 151–159, 2008.

Peter Christen. *Data Matching – Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection.* Data-Centric Systems and Applications. Springer, 2012.

Reuven Cohen and Liran Katzir. The generalized maximum coverage problem. *Information Processing Letters*, 108(1):15–22, 2008.

William W. Cohen and Sunita Sarawagi. Exploiting dictionaries in named entity extraction: combining semi-markov extraction processes and data integration methods. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 89–98, 2004.

William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the International Workshop on Information Integration on the Web (IIWeb)*, pages 73–78, 2003.

Gene Ontology Consortium. The gene ontology (go) database and informatics resource. *Nucleic Acids Research*, 32(1):258–261, 2004.

Sanjib Das, AnHai Doan, Suganthan G. C. Paul, Chaitanya Gokhale, and Pradap Konda. The magellan data repository. `https://sites.google.com/site/anhaidgroup/useful-stuff/data`, 2018.

Luc Dehaspe and Hannu Toivonen. Discovery of frequent DATALOG patterns. *Data Mining and Knowledge Discovery*, 3(1):7–36, 1999.

Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 469–478, 2012.

Omkar Deshpande, Digvijay S. Lamba, Michel Tourn, Sanjib Das, Sri Subramaniam, Anand Rajaraman, Venky Harinarayan, and AnHai Doan. Building, maintaining, and using knowledge bases: A report from the trenches. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 1209–1220, 2013.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 1811–1818, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186, 2019.

Lee R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. Explore-by-example: An automatic query steering framework for interactive data exploration. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 517–528, 2014.

Josip Djolonga, Andreas Krause, and Volkan Cevher. High-dimensional gaussian process bandits. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1025–1033, 2013.

Justin Domke. Learning graphical model parameters with approximate marginal inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35 (10):2454–2467, 2013.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 601–610, 2014.

Mohnish Dubey, Debayan Banerjee, Debanjan Chaudhuri, and Jens Lehmann. EARL: Joint entity and relation linking for question answering over knowledge graphs. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 108–126, 2018.

Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq R. Joty, Mourad Ouzzani, and Nan Tang. Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment*, 11(11):1454–1467, 2018.

Guillermo Echegoyen, Álvaro Rodrigo, and Anselmo Peñas. Benchmarking entity linking for question answering over knowledge graphs. *Procesamiento del Lenguaje Natural*, 63(0):121–128, 2019.

Mohamed G. Elfeky, Vassilios Verykios, and Ahmed Elmagarmid. TAILOR: A record linkage tool box. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 17–28, 2002.

Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 19(1):1–16, 2007.

Faezeh Ensan and Weichang Du. Ad hoc retrieval via entity linking and semantic similarity. *Knowledge and Information Systems (KAIS)*, 58(3):551–583, 2019.

Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer, 2013.

Wenfei Fan, Ping Lu, Chao Tian, and Jingren Zhou. Deducing certain fixes to graphs. *Proceedings of the VLDB Endowment*, 12(7):752–765, 2019.

Manaal Faruqui and Sebastian Padó. Training and evaluating a german named entity recognizer with semantic generalization. In *Proceedings of the Conference on Natural Language Processing (KONVENS)*, pages 129–133, 2010.

Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

Raul Castro Fernandez and Samuel Madden. Termite: A system for tunneling through heterogeneous data. In *Proceedings of the International Workshop on Exploiting Artificial Intelligence Techniques for Data Management (aiDM)*, pages 1–8, 2019.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 363–370, 2005.

Donatella Firmani, Barna Saha, and Divesh Srivastava. Online entity resolution using an oracle. *Proceedings of the VLDB Endowment*, 9(5):384–395, 2016.

Radu Florian, Abraham Ittycheriah, Hongyan Jing, and Tong Zhang. Named entity recognition through classifier combination. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, pages 168–171, 2003.

Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB Journal*, 24(6):707–730, 2015.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 413–422, 2013.

Octavian-Eugen Ganea and Thomas Hofmann. Deep joint entity disambiguation with local neural attention. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2619–2629, 2017.

GeoNames, 2018. The GeoNames geographical database. `http://www.geonames.org`, 2018.

Genet Asefa Gesese, Russa Biswas, and Harald Sack. A comprehensive survey of knowledge graph embeddings with literals: Techniques and applications. In *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG)*, pages 31–40, 2019.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 249–256, 2010.

Karl Goiser and Peter Christen. Towards automated record linkage. In *Proceedings of the Australasian Conference on Data Mining and Analytics (AusDM)*, pages 23–31, 2006.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Yash Govind, Erik Paulson, Palaniappan Nagarajan, Paul Suganthan G. C., AnHai Doan, Youngchoon Park, Glenn Fung, Devin Conathan, Marshall Carter, and Mingju Sun. Cloudmatcher: A hands-off cloud/crowd service for entity matching. *Proceedings of the VLDB Endowment*, 11(12):2042–2045, 2018.

Ralph Grishman and Beth Sundheim. Message understanding conference – 6: A brief history. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 466–471, 1996.

Toni Grütze, Gjergji Kasneci, Zhe Zuo, and Felix Naumann. CohEEL: Coherent and efficient named entity linking through random walks. *Journal of Web Semantics*, 37–38:75–89, 2016.

Lifang Gu and Rohan A. Baxter. Decision models for record linkage. In *Data Mining - Theory, Methodology, Techniques, and Applications*, pages 146–160. Springer, 2006.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Jointly embedding knowledge graphs and logical rules. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 192–202, 2016.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1735–1742, 2006.

Richard H. R. Hahnloser, Rahul Sarpeshkar, Misha A. Mahowald, Rodney J. Douglas, and H. Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405:947–951, 2000.

Christian Hänig, Stefan Bordag, and Stefan Thomas. Modular classifier ensemble architecture for named entity recognition on low resource systems. In *Proceedings of the KONVENS Workshop GermEval Shared Task on Named Entity Recognition*, 2014.

Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. Exploiting entity linking in queries for entity retrieval. In *Proceedings of the International Conference on the Theory of Information Retrieval, (ICTIR)*, pages 209–218, 2016.

Qi He. Building the linkedin knowledge graph. *LinkedIn Engineering Blog*, 2016. URL `https://engineering.linkedin.com/blog/2016/10/building-the-linkedin-knowledge-graph`.

Alireza Heidari, Joshua McGrath, Ihab F. Ilyas, and Theodoros Rekatsinas. Holodetect: Few-shot learning for error detection. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 829–846, 2019.

Martin Hermann, Michael Hochleitner, Sarah Kellner, Simon Preissner, and Desislava Zhekova. Nessy: A hybrid approach to named entity recognition for german. In *Proceedings of the KONVENS Workshop GermEval Shared Task on Named Entity Recognition*, 2014.

Vinh Thinh Ho, Daria Stepanova, Mohamed H. Gad-Elrab, Evgeny Kharlamov, and Gerhard Weikum. Rule learning from knowledge graphs guided by embedding models. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 72–90, 2018.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Yu-Lun Hsieh, Yung-Chun Chang, Nai-Wen Chang, and Wen-Lian Hsu. Identifying protein-protein interactions in biomedical literature using recurrent neural networks with long short-term memory. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, pages 240–245, 2017.

Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991, 2015.

Madelon Hulsebos, Kevin Zeng Hu, Michiel A. Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çagatay Demiralp, and César A. Hidalgo. Sherlock: A deep learning approach to semantic data type detection. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1500–1508, 2019.

Paul Jaccard. Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:241–272, 1901.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 687–696, 2015.

Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. Low-resource deep entity resolution with transfer and active learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5851–5861, 2019.

Jun'ichi Kazama and Kentaro Torisawa. Exploiting wikipedia as external knowledge for named entity recognition. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 698–707, 2007.

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, S. M. Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *International Conference on Learning Representations (ICLR)*, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *International Conference on Machine Learning (ICML) Deep Learning Workshop*, 2015.

Prodromos Kolyvakis, Alexandros Kalousis, and Dimitris Kiritsis. Deepalignment: Unsupervised ontology matching with refined word vectors. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 787–798, 2018.

Ioannis Koumarelas, Thorsten Papenbrock, and Felix Naumann. Mdedup: Duplicate detection with matching dependencies. *Proceedings of the VLDB Endowment*, 13(5): 712–725, 2020.

Vijay Krishnan and Christopher D. Manning. An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1121—-1128, 2006.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 282–289, 2001.

Phong Le and Ivan Titov. Improving entity linking by modeling latent relations between mentions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1595–1604, 2018.

Joohong Lee, Sangwoo Seo, and Yong Suk Choi. Semantic relation classification via bidirectional LSTM networks with entity-aware attention using latent entity typing. *Symmetry*, 11(6):785–794, 2019.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the International Conference on Information Retrieval (SIGIR)*, pages 3–12, 1994.

Huiying Li and Jing Shi. Linking named entity in a question with dbpedia knowledge base. In *Joint International Semantic Technology Conference (JIST)*, pages 263–270, 2016.

David Liben-Nowell and Jon M. Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology (JASIST)*, 58(7):1019–1031, 2007.

Thomas Lin, Mausam, and Oren Etzioni. Entity linking at web scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 84–88, 2012.

Xueling Lin, Haoyang Li, Hao Xin, Zijian Li, and Lei Chen. Kbpearl: A knowledge base population system supported by joint entity and relation linking. *Proceedings of the VLDB Endowment*, 13(7):1035–1049, 2020.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 2181–2187, 2015.

Matteo Lissandrini, Davide Mottin, Themis Palpanas, Dimitra Papadimitriou, and Yannis Velegrakis. Unleashing the power of information graphs. *SIGMOD Record*, 43(4): 21–26, 2014.

Michael Loster, Zhe Zuo, Felix Naumann, Oliver Maspfuhl, and Dirk Thomas. Improving company recognition from unstructured text by using dictionaries. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 610–619, 2017.

Michael Loster, Manuel Hegner, Felix Naumann, and Ulf Leser. Dissecting company names using sequence labeling. In *Proceedings of the Conference on "Lernen, Wissen, Daten, Analysen" (LWDA)*, pages 227–238, 2018a.

Michael Loster, Felix Naumann, Jan Ehmueller, and Benjamin Feldmann. Curex: A system for extracting, curating, and exploring domain-specific knowledge graphs from text. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 1883–1886, 2018b.

Michael Loster, Ioannis Koumarelas, and Felix Naumann. Knowledge transfer for entity resolution with siamese neural networks. In *Journal of Data and Information Quality*, 2020a.

Michael Loster, Davide Mottin, Paolo Papotti, Jan Ehmüller, Benjamin Feldmann, and Felix Naumann. Few-shot knowledge validation using rules. In *under submission*, 2020b.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4765–4774, 2017.

Mohammad Mahdavi, Ziawasch Abedjan, Raul Castro Fernandez, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. Raha: A configuration-free error detection system. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 865–882, 2019.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

José-Lázaro Martínez-Rodríguez, Aidan Hogan, and Ivan López-Arévalo. Information extraction meets the semantic web: A survey. *Semantic Web*, 11(2):255–335, 2020.

Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, pages 188–191, 2003.

Paul McNamee, Veselin Stoyanov, James Mayfield, Tim Finin, Tim Oates, Tan Xu, Douglas W. Oard, and Dawn J. Lawrie. HLTCOE participation at TAC 2012: Entity linking and cold start knowledge base construction. In *Proceedings of the Text Analysis Conference (TAC)*, 2012.

Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. Siamese network features for image matching. In *International Conference on Pattern Recognition (ICPR)*, pages 378–383, 2016.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR)*, 2013.

Renée J. Miller. Open data integration. *Proceedings of the VLDB Endowment*, 11(12): 2130–2139, 2018.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1003–1011, 2009.

Alvaro E. Monge and Charles Elkan. The field matching problem: Algorithms and applications. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 267–270, 1996.

Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 19–34, 2018.

Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 2786–2792, 2016.

David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

Felix Naumann and Melanie Herschel. *An Introduction to Duplicate Detection*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.

Felix Naumann, Alexander Bilke, Jens Bleiholder, and Melanie Weis. Data fusion in three steps: Resolving schema, tuple, and value inconsistencies. *IEEE Data Engineering Bulletin*, 29(2):21–31, 2006.

Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. Learning text similarity with siamese recurrent networks. In *Proceedings of the Workshop on Representation Learning for NLP (Rep4NLP)*, pages 148–157, 2016.

Sahand Negahban, Benjamin I. P. Rubinstein, and Jim Gemmell. Scaling multiple-source entity resolution using statistically efficient transfer learning. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 2224–2228, 2012.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, 14(1):265–294, 1978.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 809–816, 2011.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. Holographic embeddings of knowledge graphs. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 1955–1961, 2016.

Feng Niu, Ce Zhang, Christopher Ré, and Jude W. Shavlik. Elementary: Large-scale knowledge-base construction via machine learning and statistical inference. *International Journal on Semantic Web and Information Systems*, 8(3):42–73, 2012.

Margaret Odell and Robert Russell. The soundex coding system. *US Patents*, no. 1261167, 1918.

Naoaki Okazaki and Jun'ichi Tsujii. Simple and efficient algorithm for approximate dictionary matching. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 851–859, 2010.

Stefano Ortona, Venkata Vamsikrishna Meduri, and Paolo Papotti. Robust discovery of positive and negative rules in knowledge bases. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 1168–1179, 2018.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 22(10):1345–1359, 2010.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *Computing Research Repository (CoRR)*, abs/1211.5063, 2012.

Yifan Peng and Zhiyong Lu. Deep learning for extracting protein-protein interactions from biomedical literature. In *Workshop on Biomedical Natural Language Processing (BioNLP)*, pages 29–38, 2017.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

Jay Pujara, Eriq Augustine, and Lise Getoor. Sparsity and noise: Where knowledge graph embeddings fall short. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1751–1756, 2017.

Lizhen Qu, Yi Zhang, Rui Wang, Lili Jiang, Rainer Gemulla, and Gerhard Weikum. Senti-lssvm: Sentiment-oriented multi-relation extraction with latent structural SVM. *Transactions of the Association for Computational Linguistics (TACL)*, 2:155–168, 2014.

Jonathan Raiman and Olivier Raiman. Deeptype: Multilingual entity linking by neural type system evolution. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 5406–5413, 2018.

Lev-Arie Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, pages 147–155, 2009.

Nils Reimers, Judith Eckle-Kohler, Carsten Schnober, Jungi Kim, and Iryna Gurevych. GermEval-2014: Nested named entity recognition with neural networks. *Proceedings of the KONVENS Workshop GermEval Shared Task on Named Entity Recognition*, 2014.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1135–1144, 2016.

Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 441–448, 2001.

Christopher De Sa, Alexander Ratner, Christopher Ré, Jaeho Shin, Feiran Wang, Sen Wu, and Ce Zhang. Incremental knowledge base construction using deepdive. *Proceedings of the International Conference on Very Large Databases (VLDB)*, 26(1):81–105, 2017.

Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 Shared Task: Language-independent named entity recognition. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, pages 142–147, 2003.

Sunita Sarawagi and Anuradha Bhamidipaty. Interactive deduplication using active learning. *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 269–278, 2002.

Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 839–846, 2000.

Peter Schüller. MoSTNER: Morphology-aware split-tag german ner with factorie. *Proceedings of the KONVENS Workshop GermEval Shared Task on Named Entity Recognition*, 2014.

Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing (TSP)*, 45(11):2673–2681, 1997.

Satoshi Sekine and Chikashi Nobata. Definition, dictionaries and tagger for extended named entity hierarchy. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2004.

Burr Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

Wei Shen, Jianyong Wang, and Jiawei Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 27(2):443–460, 2015.

Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and Philip S. Yu. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 29(1):17–37, 2017.

Jaeho Shin, Sen Wu, Feiran Wang, Christopher De Sa, Ce Zhang, and Christopher Ré. Incremental knowledge base construction using DeepDive. *Proceedings of the VLDB Endowment*, 8(11):1310–1321, 2015.

Amit Singhal. Introducing the knowledge graph: things, not strings. *Official Google Blog*, 2012. URL `https://www.blog.google/products/search/introducing-knowledge-graph-things-not`.

Alisa Smirnova and Philippe Cudré-Mauroux. Relation extraction using distant supervision: A survey. *ACM Computing Surveys*, 51(5):1–35, 2019.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2960–2968, 2012.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2895–2905, 2019.

Rohini K. Srihari. A hybrid approach for named entity and sub-type tagging. In *Proceedings of the Applied Natural Language Processing Conference (ANLP)*, pages 247–254, 2000.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias W. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1015–1022, 2010.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 697–706, 2007.

Rhea Sukthanker, Soujanya Poria, Erik Cambria, and Ramkumar Thirunavukarasu. Anaphora and coreference resolution: A review. *Information Fusion*, 59:139–162, 2020.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112, 2014.

Min Tang, Xiaoqiang Luo, and Salim Roukos. Active learning for statistical natural language parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 120–127, 2002.

Sheila Tejada, Craig A. Knoblock, and Steven Minton. Learning domain-independent string transformation weights for high accuracy object identification. *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 350–359, 2002.

Antonio Toral and Rafael Muñoz. A proposal to automatically build and maintain gazetteers for named entity recognition by using Wikipedia. *Proceedings of the Workshop on NEW TEXT Wikis and blogs and other dynamic text sources*, 2006.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, page 173–180, 2003.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017a.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017b.

Vasilis Verroios and Hector Garcia-Molina. Entity resolution with crowd errors. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 219–230, 2015.

Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

Jiannan Wang, Tim Kraska, Michael J. Franklin, and Jianhua Feng. Crowder: Crowdsourcing entity resolution. *Proceedings of the VLDB Endowment*, 5(11):1483–1494, 2012.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. Relation classification via multi-level attention cnns. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1298–1307, 2016.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 29(12):2724–2743, 2017.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1591–1601, 2014a.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 1112–1119, 2014b.

Patrick Watrin, Louis De Viron, Denis Lebailly, Mathieu Constant, and Stéphanie Weiser. Named entity recognition for german using conditional random fields and linguistic resources. *Proceedings of the KONVENS Workshop GermEval Shared Task on Named Entity Recognition*, 2014.

Gerhard Weikum and Martin Theobald. From information to knowledge: harvesting entities and relationships from web sources. In *Proceedings of the Symposium on Principles of Database Systems (PODS)*, pages 65–76, 2010.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 370–378, 2016a.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2586–2594, 2016b.

William E. Winkler and Yves Thibaudeau. An application of the Fellegi-Sunter model of record linkage to the 1990 u.s. decennial census. *US Bureau of the Census*, pages 1–22, 1987.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 2659–2665, 2016.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. Question answering on freebase via relation extraction and textual evidence. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2326—-2336, 2016.

Peng Xu and Denilson Barbosa. Connecting language and knowledge with heterogeneous representations for neural relation extraction. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 3201–3206, 2019.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, pages 250–259, 2016.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. Learning distributed representations of texts and entities from knowledge base. *Transactions of the Association for Computational Linguistics (TACL)*, 5:397–411, 2017.

Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein, and Huajun Chen. Iteratively learning embeddings and rules for knowledge graph reasoning. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 2366–2377, 2019.

Chen Zhao and Yeye He. Auto-em: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 2413–2424, 2019.

Deyu Zhou, Dayou Zhong, and Yulan He. Biomedical relation extraction: From binary to complex. *Computational and Mathematical Methods in Medicine*, 2014:1–18, 2014.

Guodong Zhou and Jian Su. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 473–480, 2002.

Zhe Zuo, Michael Loster, Ralf Krestel, and Felix Naumann. Uncovering business relationships: Context-sensitive relationship extraction for difficult relationship types. In *Proceedings of the Conference on "Lernen, Wissen, Daten, Analysen" (LWDA)*, pages 271–283, 2017.

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Doktorarbeit mit dem Thema:

Knowledge Base Construction with Machine Learning Methods

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Potsdam, den 20. August 2020

Michael Loster