Knowledge Graph Population from Conversations

by

Michael Strobl

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

University of Alberta

Abstract

This thesis describes the design of a system that is capable of the generation of a Knowledge Graph (KG), referred to as Knowledge Graph Population (KGP), from conversations, specifically with elderly people. While this system still follows a traditional KGP approach with Entity Recognition (ER), Entity Linking (EL) and Relation Extraction (RE), we propose novel approaches for: (1) Annotating Wikipedia with Named Entities exhaustively in order to extract datasets for a variety of NLP tasks. (2) An interactive system to test and visualize the output of a KGP system based on a conversation. (3) Fast and accurate dataset creation for RE. (4) Dataset creation as well as training procedures for partially annotated ER datasets. (5) KG Question Answering (KGQA) using RE datasets for KGP instead of solely relying on dedicated KGQA datasets, not necessarily suited to the KG at hand.

A KG consists of nodes, corresponding to distinct entities, and edges, corresponding to semantic relations that link entities to each other. The main concern is composed of three topics: Family/Friends, Health, and Nutrition. We chose these since they are the main topics we think elderly people are interested in during daily conversations. The KGP follows a pipeline consisting of three systems: (1) ER, (2) EL and (3) RE. ER is concerned with finding entities of interest, mainly named entities and entities related to the topics Nutrition and Health. The EL system aims to link mentions of entities found by the ER system to their corresponding KG nodes or create a new one if not existing. The RE system then tries to find evidence in utterances that link

entities to each other. In order to test the resulting KG, a KG Question Answering (KGQA) system is used to translate natural language questions into structured queries to retrieve answers from the KG. In addition, future tasks, such as response generation, could take advantage of such a KG in order to continue a conversation by creating informative utterances using the gained knowledge from previous utterances from the current or previous conversations in a structured way.

Preface

Parts of the following chapters of this thesis have been published:

- Chapter 3: The first version of WEXEA was published at LREC 2020 [84] and the updated version including multilingual corpora at LREC 2022 [82].
- Chapter 6: This chapter was published at NLDB 2022 [85].
- Chapter 4 and 7: The main parts of these chapters have been accepted for the Knowledge and Natural Language Processing track at ACM SAC 2023 and the paper is available on Arxiv [83] under the title "FREDA: Flexible Relation Extraction Data Annotations", focusing on the creation and evaluation of manually annotated datasets for the task of Relation Extraction.

Acknowledgements

First, I would like to thank my supervisor Prof. Zaïane for his support, patience and knowledge. His guidance was indispensable for my research and the writing of this thesis.

Second, I would like to thank my supervisory committee, besides my supervisor, Prof. Goebel and Prof. Kondrak, for their time and insightful comments on my research.

My sincere thanks also goes to Prof. Langlais and Prof. Rafiei, the remaining examining committee members, as well as Prof. Ray, the committee chair, for taking their time for my defense.

Last, I would like to thank my wife for her support throughout this long journey.

Contents

List of Tables

List of Figures

List of Acronyms

CR Co-reference Resolution.

DS Distant Supervision.

EG Entity Graph.

EL Entity Linking.

EM Entity Mention.

ER Entity Recognition.

FDA Flexible Data Annotation.

FREDA Flexible Relation Extraction Data Annotation.

IE Information Extraction.

KB Knowledge Base.

KBP Knowledge Base Population.

KG Knowledge Graph.

KGP Knowledge Graph Population.

KGQA Knowledge Graph Question Answering.

MD Mention Detection.

NE Named Entity.

NER Named Entity Recognition.

NLI Natural Language Interface.

NLP Natural Language Processing.

NLU Natural Language Understanding.

NN Neural Network.

RE Relation Extraction.

RG Response Generation.

URI Unified Resource Identifier.

WEXEA Wikipedia EXhaustive Entity Annotation.

Chapter 1

Introduction

1.1 Motivation

A Knowledge Graph (KG) is a graph-based representation of knowledge and Knowledge Graph Population (KGP) is the task of creating and maintaining a KG[1]. Entities in text are represented as nodes in the KG, linked to each other with appropriate relations, as edges. Therefore, a KGP system needs to be able to extract mentions of entities from text, link them to their corresponding node in the graph (or create a new node for unseen entities) and find relations between entities, expressed in text, and add them to the KG (only if missing, otherwise nothing is done). Such a KG can be seen as a structured and compact representation of knowledge expressed in text, which can be visualized and queried, e.g. in order to answer simple questions.

There exist multiple downstream tasks, which can take advantage of such a KG, as well as practical reasons why a KG can be useful:

- Response Generation (RG) for conversations (as part of a Dialog System): When a response has to be generated in a conversation, typically previous utterances are taken into account by a sequence-to-sequence model in order to generate a new response [77]. However, the older a certain utterance is, the less likely it is that (1) it is taken into account all-together and (2) the entities in it are linked to other mentions properly. A KG with properly linked entities and their relations may help.

[1] In this thesis, we are only adding information to the KG.

- The alternative to a KG would be to keep the whole conversation as text in memory, leading to a much higher memory requirement since a graph-structure with its nodes and edges would presumably consume much less memory space. Furthermore, if external knowledge, e.g. from Wikipedia, should be included, keeping text instead of a corresponding KG, e.g. DBpedia, is even less desirable.

- KGs can be queried with query languages, such as SPARQL, whereas querying text is a much harder task to do, even if simple questions need to be answered.

Overall, a properly maintained KG can be seen as a structured long-term memory, which does not forget, can be queried with structured query languages and contains essential information expressed in a conversation.

Therefore, we propose a system that is capable of KGP for conversations. Figure 1.1 shows conceptually the system's input (text statements from dialogues) and output (KG extracted from input statements). As an example, Figure 1.2a shows a conversation of a chatbot called *Ana* with the user *Irene* with the corresponding KG in Figure 1.2b below.

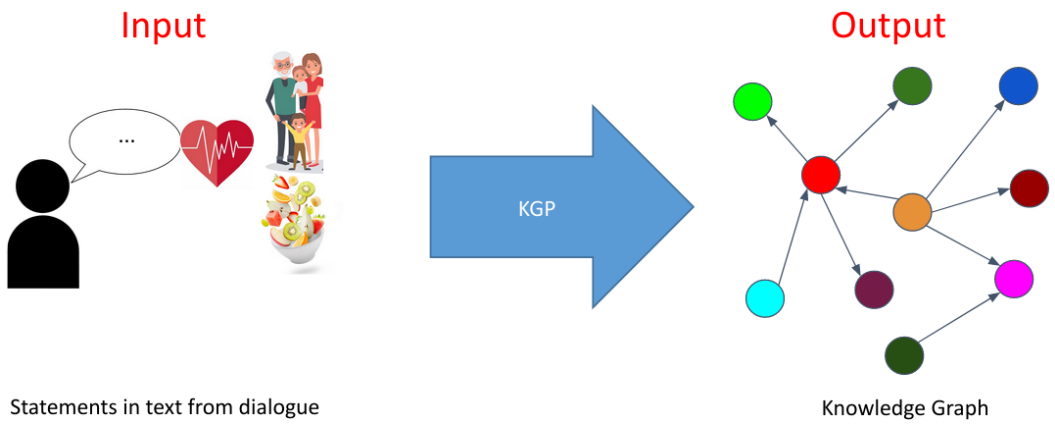
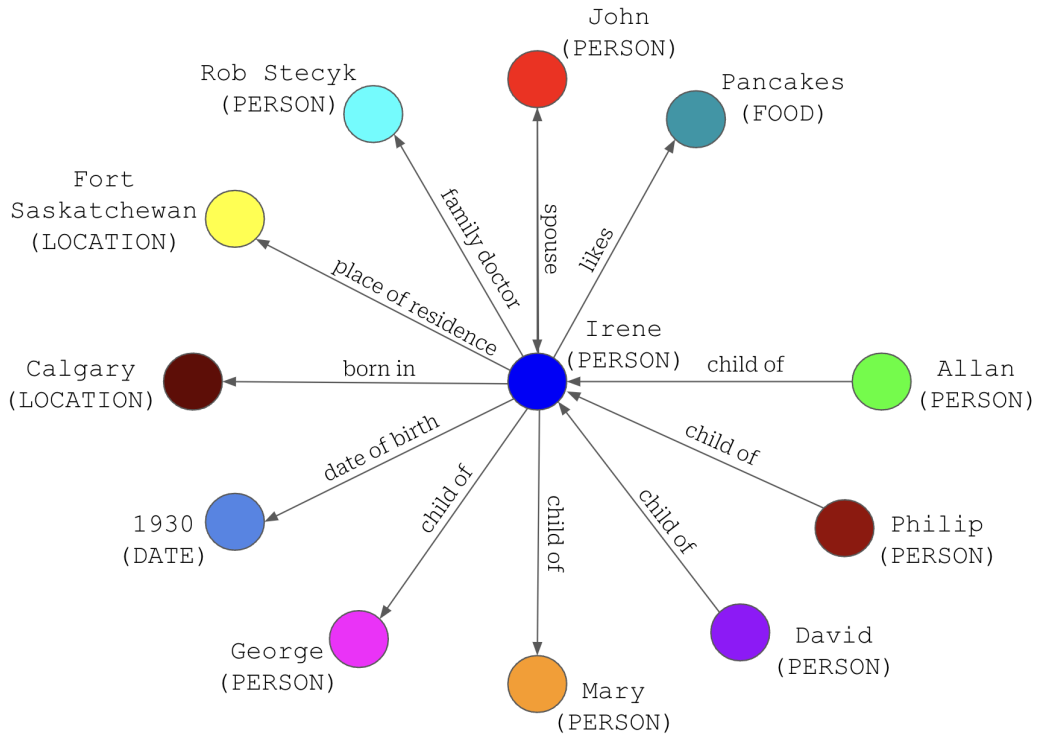


Figure 1.1: Concept of suggested system with statements from dialogue text as input and a Knowledge Graph as output with entities as nodes and relations as directed edges.

Ana: Hello, my name is Ana. What is your name?
Irene: Hi, my name is Irene.
Ana: Nice to meet you! Tell me about yourself.
Irene: Well, I was born in 1930 in Calgary, and I live in Fort Saskatchewan.
Ana: Do you have any family? Are you married?
Irene: Yes, I have three sons, David, Allan, and Philip. And my husband's name is John.
Ana: How about your parents?
Irene: My parents' names were George and Mary.
Ana: Do you have a family doctor?
Irene: Yes, my doctor is Dr. Rob Stecyk.
Ana: What's your favourite meal?
Irene: I really like pancakes for breakfast.
Ana: Thanks Irene!

(a) Conversation between Ana (chatbot) and Irene (human).



(b) Knowledge Graph, which can be extracted from Irene's utterances.

Figure 1.2: Conversation of a human (Irene) with a chatbot named Ana and corresponding Knowledge Graph.

The goal of this system is to be able to extract information from such conversations and add them to a KG[2]. In order to do so, entities have to be recognized and added to a KG as nodes as well as relations mentioned in text between these entities as edges. Entities and relations can be detected through training systems using high-quality labelled datasets. Therefore, our main contributions include:

- Due to the lack of high-quality manually annotated datasets for the task of Relation Extraction (RE), which are necessary to extract relations from text reliably, we delineate a tool for flexible data annotation for the task of RE and, in addition, Entity Recognition (ER), Co-reference Resolution (CR) and Entity Linking (EL).
- We shed some light into how semi-automatically and partially annotated datasets for ER derived from Wikipedia can be added to models trained on commonly used datasets.
- An interactive system that can be used to test subtasks of KGP and the resulting KG can be inspected.
- A KG Questions Answering system (KGQA), which can make use of existing RE datasets in order to create structured queries in order to retrieve answers from a KG.

1.2 Examples

In order to illustrate the population of a KG, consider the following simplified example, starting with the first utterance:

> "My neighbour's name is John."

Figure 1.3 shows the resulting KG after the first utterance. Two entities can be detected: The speaker is referenced through a possessive pronoun (*My*)

[2]This thesis should not be confused with providing a KG that can be used offline. Instead the goal is to provide solutions that aim to extract pieces of information from statements and unify them into a KG on demand.

and there is a mention of a person named *John*. In addition, both entities can be linked through the *neighbour*-relation.

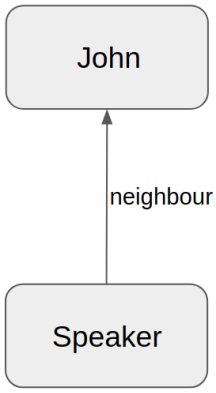


Figure 1.3: Resulting KG after first example utterance.

Second utterance:

"John's sister Sarah lives in Edmonton."

In Figure 1.4 two more entities are added: *Sarah* and the *City of Edmonton*. Both are linked to the corresponding subjects of the *sister*- and *lives_in*-relation, respectively. The entity referred to through the mention *John* is already part of the KG and therefore can be linked to an existing node in the KG.

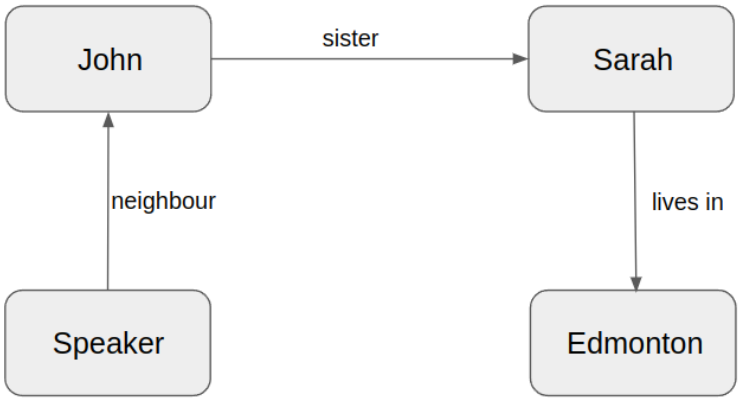


Figure 1.4: Resulting KG after second example utterance.

This example should make the general idea of KGP clear, including the three parts of such a system: (1) Entity Recognition (ER), i.e. detecting all

entity mentions (the speaker, *John*, *Sarah*, *Edmonton* and their co-references in this example), (2) Entity Linking (EL), linking these mentions to their corresponding node in the graph (create new node if it does not already exist) and (3) Relation Extraction (RE), i.e. linking entities to each other with the corresponding relations, if expressed in text (*neighbour*, *sister* and *lives_in* in this example). In reality, these tasks are not always as straightforward as it sounds and this work should shed some light into how a general KGP system for conversations can look like.

1.3 Initial Assumptions

Early work in cognitive science and language understanding dates back to the 1960s with, for example, the SHRDLU program [92] that was able to conduct a dialogue in English with a human to move blocks by the computer in a virtual box. Even in such a restricted environment and a vocabulary of only 200 words, the complexity of a system that is capable of understanding a small portion of the English language is large. Therefore, we make a few initial assumptions regarding the utterances we expect to extract information from:

1. Although intonation is an important part of human communication, we assume text as input. This is in line with previous work on Dialogue Systems, e.g. see [25].

2. The input to the system is assumed to be utterances from a single interlocutor. The phrase *my mother* contains valuable information, but it depends on who said it. Whereas, if all utterances are assumed to be from the same person, this phrase is not ambiguous.

3. We focus on conversations with elderly people. Relation Extraction is dependent on a set of pre-defined relations since an exhaustive set of relations is difficult to create and acquire data for, e.g. see [97]. Therefore, we mainly consider relations in the domains Family/Friends, Nutrition, and Health due to the assumption that these domains are the most important ones for a Dialogue System for elderly people. However, the

approaches described thereafter are kept general enough to be able to extend them to broader topics.

1.4 Problem Statement

The Resource Description Framework (RDF)[3] is a standard for triple-based RE, i.e. the extraction of facts consisting of a subject, relation and object, each. The extractions of the proposed KGP system follow this standard and are described in more detail in the following.

Our goal is to populate a KG from conversations, consisting of Literals (strings or numbers, e.g. for names or dates) or Uniform Resource Identifiers (URI) as vertices (for entities) and relations linking these together as edges. It is fed with facts extracted from conversations. Here we provide more formal details about KGs and the facts that we are aiming to extract.

The sets of entities E, Literals L and relations R form the basic building blocks of a KG[4], a specific form of Knowledge Base (KB), which itself we consider as a not necessarily structured collection of knowledge, e.g. the collection of all websites in the Internet or Wikipedia[5]. Whereas, DBpedia [7] and Freebase [9] follow a graph structure and are therefore KGs. For the rest of this document, if the structure of the KB is not important for understanding the context, the term KB is used, otherwise we use the more specific term KG, which we are aiming to create from a conversation.

Formally, given a set of n sentences (or pieces of text) $T = t_1, t_2, ..., t_n$ (with $n \in \mathbb{N}$) we are aiming to extract a set of facts $F = f_1, f_2, ..., f_m$ (with $m \in \mathbb{N}$). A fact $f_i = (s_i, r_i, o_i)$, often referred to as a subject-relation-object-triple, consists of a subject $s_i \in E$, an object $o_i \in E \cup L$ and a relation $r_i \in R$, with $1 \leq i \leq m$. Literals cannot be used as subjects as they are used for attributes, encoded as strings, numbers or dates. Therefore, the goal is to populate a Knowledge Graph $KG = (E, F, R)$.

[3] `https://www.w3.org/RDF/`

[4] Even though edges and vertices in the KG are represented as URIs, they are conceptually quite different and, therefore, we keep them separate in our notation.

[5] `https://www.wikipedia.org/`

E, L, R and even their cardinalities are difficult to specify formally, therefore we provide an informal definition here with more information later in subsequent chapters:

- The set E consists of entities, which can be part of a fact $f = (s, r, o)$ with $r \in R$, $s \in E$ and $o \in E \cup L$.
- The set L of literals can be any string or number, typically referring to certain properties of entities, e.g. dates or names.
- The set R of relations is, in general, the set of possible links between subject $s \in E$ and an object $o \in E \cup L$. Relations $r \in R$ represent directed edges in the KG[6].

1.5 Research Questions and Contributions

In the following, we describe the main research questions this thesis aims to solve and our solutions to these questions.

1.5.1 RE Dataset Creation

Each of the subtasks of KGP usually requires high-quality manually annotated datasets to train models on. However, specifically for the task of RE (the most difficult one to annotate for), creating datasets manually is a time-consuming and tedious task and it is not always clear what kind of text data is suitable for annotation since some relations are not mentioned very frequently. Therefore, approaches are needed to potentially filter text data and reduce the cognitive load for the annotator.

We delineate our tool for Flexible Relation Extraction Data Annotation (FREDA) and show that FREDA is a simple yet effective and efficient tool for manual data annotations for multiple NLP tasks. In addition, we develop and test our system Wikipedia EXhaustive Entity Annotations (WEXEA), a semi-automatic system to annotate named entities in Wikipedia for multiple

[6]Some relations, e.g. *spouse* or *sibling*, are bi-directional and therefore have to be added in both ways separately.

languages. We show how the resulting WEXEA datasets can be used for data annotation with FREDA, reducing the amount of time a user has to spend to annotate entities in sentences.

1.5.2 ER for Partially Annotated Datasets

NER models are typically trained on datasets with a specific set of entity classes, e.g. *Person*, *Location*, *Organization*, and *Miscellaneous* for the CoNLL 2003 NER dataset [72]. Whenever more classes of interest should be added, a new dataset has to be found or created containing high-quality manual annotations for all classes. However, these datasets may often not exist. This leads to the question, how is it possible to add new classes to NER systems without the need for expensive manual data annotation, if possible.

We specify a semi-automatic procedure for creating partially annotated datasets for NER, using WEXEA and the Wikipedia category hierarchy. This procedure is used to create two partially annotated dataset for the entity types *Food* and *Drugs*. In addition, we compare three training strategies for models trained and tested on these datasets as well as on two corresponding manually annotated datasets.

1.5.3 Interactive System for KGP

Testing the subtasks of KGP is often straightforward, either through testing on existing datasets with commonly used metrics as can be seen in [49] for NER, or visually, e.g. through a web-interface as done by the CoreNLP tool[7]. However, the combination of all subtasks, i.e. a KGP system, is not straightforward to test and complete systems for visual testing with replaceable modules for each subtasks do not exist.

Therefore, we describe and develop a system with a graphical user-interface showing a conversation with a chatbot and the resulting KG. Basic rule-based components for ER, RE and EL are included, which can be further extended.

[7] `https://corenlp.run/`

1.5.4 KGQA

Asking questions is a way to retrieve information stored in the KG. Especially in conversations, it is common to ask such questions in natural language. However, questions, which can be answered by a system storing the KG, are typically encoded in SPARQL, an SQL-like query language. Therefore, such a KG Question Answering (KGQA) system needs to be able to translate natural language questions to SPARQL queries for a specific KG. But this may not be working on other KGs, e.g. since the sets of relations contained by the KGs may differ, and datasets to train a model to learn this task for a new KG may be missing. This leads to the question how it is possible to still be able to train such a model on our KG, ideally without the need for new manually annotated datasets containing natural language questions and their corresponding KG-specific SPARQL queries.

We specify, develop, and test a KGQA system, which is able to translate natural language questions to SPARQL queries, partially trained on RE datasets. This is due to the fact that dedicated KGQA datasets are specific to certain KGs and may not work on others. However, we show that no or a minimal amount of dedicated KGQA data is needed to perform this translation task.

1.6 Organization

This thesis is structured as follows:

- Chapter 2: In this chapter we present details regarding commonly used related work for the task of KGP. It mainly contains descriptions of existing work and datasets on the subtasks ER, EL and RE. In addition, two approaches combining subtasks of KGP are described. Each of the following chapters provides details regarding related work more specifically.

- Chapter 3: Wikipedia EXhaustive Entity Annotations (WEXEA): A framework for semi-automatic entity annotations (in addition to exist-

ing annotations) of an entire Wikipedia dump for multiple languages. Due to Wikipedia guidelines, many entity mentions are not annotated by editors, i.e. hyperlinks to their corresponding article are missing. However, it is presumably a lot easier to add these missing annotations in Wikipedia than in open text without any annotations to start with. The resulting exhaustively annotated dataset can be used to speed up manual data annotation for the aforementioned subtasks of KGP.

- Chapter 4: Flexible Relation Extraction Data Annotation (FREDA): A flexible tool for fast data annotation for KGP subtasks. Specifically for the task of RE, manual data annotation is cumbersome since entities, their co-references and relations between two entities have to be labelled. In addition, many relations of interest are difficult to find in text data and a filtering step needs to be applied. FREDA is a tool aiming to mitigate these issues through providing an easy to use framework for manual data annotation for the task of RE as well as other KGP subtasks.

- Chapter 5: In order to be able to test the full KGP system including all subtasks, we present a description of a web-based tool with a text input and the resulting KG with real-time updates in this chapter. Rules can be added easily to the NER and RE sub-systems to be able to extend and test the system quickly. The following chapters add improvements to specific subtasks of this system.

- Chapter 6: Typically, NER systems are able to detect a specific set of classes, depending on the datasets they were trained on. Extending this set of classes is difficult as high-quality manually labeled datasets are necessary with exhaustive annotations for all classes of interest. This chapter provides details of our approach to create and use partially annotated datasets for new classes in addition to the originally used datasets without harming the NER model a lot. We manually annotated 500 sentences for the two classes *Food* and *Drugs* in order to test the system.

- Chapter 7: In this chapter we present more details on how to use our

FREDA framework, as described in chapter 4, for the task of RE. We created datasets for 19 relations with at least 500 sentences per relation and evaluated models trained on these datasets on other state-of-the-art datasets, showing promising results.

- Chapter 8: In order to ask questions to a KG in natural language, the question itself has to be translated to the SPARQL query language, which can be directly applied to the framework storing the KG. However, existing approaches are strongly tied to a specific KG, e.g. Wikidata [91] or Freebase [9], leading to not necessarily correct queries when applied to other KGs with potentially different sets of relations. This problem typically leads to the issue of a lack of datasets to train models on for this task and a specific KG. Our approach, as described in this chapter, aims to use RE datasets, which were used for populating a KG, to help with translating natural language questions to SPARQL queries without the need for new manually annotated datasets.

- Chapter 9: In this chapter, this thesis is concluded and a description of potential future works is included.

Chapter 2

Related Work

In this chapter we present commonly used related work on the subtasks in the pipeline of KGP, namely ER, EL and RE, as well as approaches aiming to combine subtasks of the KGP pipeline. Each remaining chapter of this work contains a more specific section about related work, not necessarily mentioned herein.

2.1 Related work for KGP

Since KGP is typically a pipelined approach (e.g. see [96]), this section presents related work on all relevant subtasks in order to construct a Knowledge Graph (KG) from any kind of text[1]. State-of-the-art systems in this area of NLP are typically models based on Neural Networks (NN) trained on specific datasets. These models can only be as good as the training dataset is and therefore we mainly focus on the most common English datasets used for each subtask in this section.

2.1.1 Entity Recognition

The vast majority of ER systems are NER systems that are trained and tested on the CoNLL 2003 dataset for NER [72], using the classes *Person*, *Organization*, *Location* and *Miscellaneous*, e.g. see [28], [42], [49] or [2]. These systems learn to label sequences of words with the BIO scheme (Beginning-

[1]Generally, these approaches do not focus on conversations since there is more training data available from other sources, such as news text.

Inside-Outside of an entity mention[2]). The BIO scheme encodes the first word of an entity as *B-<class>*, all subsequent words of an entity as *I-<class>* and all other words as *Outside*. Therefore, this scheme allows the system to be able to detect multiple entities in a row without separator. The CoNLL 2003 dataset is based on Reuters news stories from the years 1996 and 1997. It is split into a training set, a development set and a test set. The test set contains sentences from news articles from different months (therefore different entities) than the development and training sets, leading to a more challenging test set for neural NER systems. These neural NER systems are typically word-based models, e.g. see [49], and are trained to predict a class for each word in the input.

There are more classes in other datasets, e.g. the ACE05 corpus [23], which includes 7 types of entities. The system introduced in [73] is able to detect named entities as well as other entity mentions from the ACE05 corpus. But the scope of entities is still limited, containing only 7 types of entities: *Person*, *Organization*, *Location*, *Facility*, *Weapon*, *Vehicle* and *Geo-Political Entity*.

Ling and Weld [53] introduced a fine-grained set of 112 types named FIGER. This set was created by using Freebase's [9] type set by removing noise types (as they are created by humans) and combining overly specific types. Since there was no data available with FIGER type annotations, the authors created a dataset based on Wikipedia and its hyperlinks for training and testing their system. They compared it to the Stanford NER system [28] (FIGER types can be matched to CoNLL types) and showed its superiority over the coarse type system from CoNLL, with which the Stanford NER works with. However, while containing some very specific types, e.g. actor, architect, bridge, hotel, transit_system, etc., some other entities would be labeled with a very broad type, i.e. person, organization, location, product, art, event or building (apart from 30 other types not in these categories).

There are other NER datasets, e.g. based on Wikipedia (see [34]), but the above datasets are the most commonly used ones.

[2]For the rest of this document, we consider an *entity mention* as a reference to an entity, which itself is a node that can be added to the KG.

The TAC KBP EDL dataset[3] contains NEs of types *Facility*, *Geo-Political Entity*, *Location*, *Organization* and *Person* including nominal entity mentions, such as "president" or "forest reserve". Therefore this dataset contains valuable information for trained models in order to detect entities that are not mentioned by their name.

2.1.2 Entity Linking

The most commonly used dataset for EL is the CoNLL-AIDA dataset [40] with manual annotations of the aforementioned CoNLL 2003 NER dataset. Entities are linked to the YAGO2 database [39], which effectively contains links to Wikipedia articles.

Existing approaches, such as [78] or [29], are typically ignoring out-of-KB entities (entities without a match in the reference KB) for their evaluation and therefore are not able to distinguish between entities that are in-KB and out-of-KB. Furthermore, these approaches are not able to detect the mentions of entities themselves, instead they rely on an NER tool annotating a dataset beforehand, which is already done in case of the CoNLL-AIDA dataset.

These are the steps EL systems, such as the previously mentioned ones, follow after an NER tool extracts mentions of entities:

1. For each mention, a candidate set of entities from a KB is created. This can be done using an *alias dictionary*, which contains references, i.e. alternative names, to each entity in the KB, e.g. "NYC" and "New York" can both refer to the entity "New York City", among others.

2. A score for each candidate entity is computed and the mention is linked to the entity with the highest score. This score is typically computed by a model that takes several features into account (among others): The immediate context of the mention, other mentions of entities or their links (if available), Wikipedia articles of each candidate entity and the probability of the mention being linked to a candidate in Wikipedia. For

[3]Text Analysis Conference (TAC) Knowledge Base Population (KBP) Entity Discovery and Linking (EDL): https://catalog.ldc.upenn.edu/LDC2019T19

example, in order to link both entities in the sentence "Edmonton won against Calgary 6:1 yesterday.", the context has to be taken into account to notice that they should be linked to sports clubs rather than the cities.

However, other approaches, such as [48], are able to do NER and EL in one step, while still not being trained to detect out-of-KB entities as they are not designed for that. But of course a mention cannot be linked if its candidate set is empty. However, especially for first names, there will be multiple candidates with a high chance.

On the contrary, approaches trained on the TAC KBP EDL dataset, which also contains links to entities of an attached dataset, such as [96], are explicitly designed to detect out-of-KB entities, since this is part of the TAC KBP evaluation. Out-of-KB entities can presumably be found frequently in conversations, such as "uncle George". Although it is common to simply consider mentions as out-of-KB entities if the candidate set is empty, as done in the CMU submission for the KBP TAC 2017 Entity Discovery and Linking challenge [55]. In addition, conversations presumably do not contain many popular entities compared to news data, therefore these systems may not be able to help linking the majority of entities in our case.

2.1.3 Relation Extraction

Generally, RE is the task of detecting entities and how they are related to each other in text data. There are two different paradigms of RE used in the literature: Open Information Extraction (OpenIE) and an approach we refer to RE with a predefined set of relations.

Open Information Extraction

OpenIE systems, such as [27] and [75] based on hand-crafted grammatical patterns or more recently [5] with a classification-based approach, are able to extract simple subject-verb-object triples. They are open since there is no predefined set of relations, therefore it is unknown which exact relation is extracted. In addition, it is usually possible to express a relation in different ways

even with different words and OpenIE approaches would not be able to capture that since OpenIE relations consist of words directly taken from the processed piece of text. For example, the sentences "John is married to Sarah" and "John and his wife Sarah live in Edmonton" are both expressing the *spouse*-relation, but the sentence structure is very different and OpenIE systems would not extract the exact same relation between "John" and "Sarah".

RE with predefined relations

On the other hand, RE systems with predefined relations are able to match relations that can be expressed with different words. However, they are limited to a hand-crafted set of relations of a typically very limited size.

Previously, in order to extract datasets from large text corpora for RE Distant Supervision (DS) was used, a bootstrapping method that uses an existing KG and a corpus of text to label sentences with relations.

Due to the lack of large datasets with entities as well as their relations annotated, Mintz et al. [63] proposed to link entities in text corpora to their corresponding entries in a KG, e.g. Freebase [9], and whenever two entities that are linked in the KG appear in the same sentence, this sentence is considered as expressing this relationship in some ways. Although this can lead to obvious incorrect annotations, e.g. sentences that include Bill and Melinda Gates certainly do not always express that they are married. They created a dataset with 10,000 instances and 102 Freebase relations using Wikipedia, although entities are tagged with an NER and existing annotations are ignored. The reported human-evaluated precision of the extracted instances is 67.7%.

Riedel et al. [70] follow a similar approach, except that their assumption is slightly different. Instead of assuming every sentence expresses a certain relation, they hypothesized that given a set of sentences mentioning two specific entities, at least one of them expresses the relation of interest. They created an RE corpus based on the New York Times Annotated Corpus[4] with 430 Freebase relations.

However, even though using the DS assumption may lead to noisy RE

[4] `https://catalog.ldc.upenn.edu/LDC2008T19`

datasets, it can still help creating datasets for RE through speeding up the process of manual annotation. More details on this can be found in Chapters 4 and 7.

The most common dataset today is the TACRED dataset [97], which was annotated using Mechanical Turk crowd annotation of the TAC KBP challenge data with 41 relation types. It is split into a training, development and test set. However, even though it is based on human annotations, the quality of these is unknown since the authors did not provide any information on whether annotations were judged by several annotators. Moreover, the dataset contains obvious mistakes, as can be seen in the following part of a sentence from the development set:

> "Besides his wife, Mandelbrot is survived by (...)"

The word "his" is considered as the subject and the word "Mandelbrot" the object for the relation "per:spouse", although both words refer to the same entity.

State-of-the-art models trained on the NYT dataset typically lead to worse results than, for example, models trained on the newer TACRED dataset. Although this is not directly comparable due to different text data used for both datasets, the results measured with the F1 score are quite different (and still disappointingly low), e.g. 0.587 for [95] (NYT) versus 0.701 for [79] (TACRED).

Recently a dataset called DialogRE was created (see [93]) from dialogues from the Television show "Friends" with entity mentions, the speakers and their relations annotated by humans. Although unfortunately entities are only considered to be mentioned if a name appears, ignoring mentions such as "my brother", without a name. These relations are only considered if the name was mentioned somewhere else in the conversation, which the authors denoted as cross-sentence relations, which we believe is incorrect since the mention of "my brother" should already be considered as entity mention and relation to be extracted.

2.2 Other Approaches

Here we present approaches, which combine multiple relevant subtasks for KGP.

2.2.1 CoreNLP

CoreNLP [57] is a popular toolkit, which consists of many annotators for a variety of NLP tasks, from which a few can be used to build a KG:

- NER: A Conditional-Random-Field (CRF) based NER [28], being able to detect the classes *Person*, *Location*, *Organization*, *Miscellaneous* (and *Money*, *Percent*, *Date* and *Time*, if needed). The whole system can be re-trained, e.g. for a dataset with a different type set, but partially labelled datasets cannot be used here (see Chapter 6).

- EL: An EL system [80] solely based on a dictionary over Wikipedia articles, linking entity mentions based on the probability distribution of their surface form as anchor text of hyperlinks to articles in Wikipedia. Out-of-KB entities cannot be linked.

- OpenIE: A logistic regression classifier to detect relations in text without a pre-defined set of relations. The output of the system is a set of subject-relation-object triple. As previously mentioned, such a system can only extract relations directly mentioned in the sentence, typically as a verb.

- KBP: A rule-based as well as a classifier-based system [96] for detecting TAC KBP relations in text. A rigid set of relations is used, which may not be possible to extend, except that new rules can be added.

- Co-reference Resolution (CR): The CR system aims to detect co-references in text, mainly linking pronouns to their corresponding entity mention, if available. A deterministic [50], statistical [18] or neural CR system [19] can be applied. Such a CR system could also implicitly be part of EL.

In general, CoreNLP is a good starting point for many NLP tasks, but, if multiple annotators are combined in the pipeline, mistakes or other issues are carried over, typically leading to very few extractions for short pieces of text.

2.2.2 Hierarchical Multi-Task Learning model

The Hierarchical Multi-Task Learning model (HMTL) [74] combines the prediction for multiple NLP tasks into a single model (in contrast to multiple models/approaches used by CoreNLP), namely: NER, Entity Mention Detection (EMD)[5], CR and RE. A limited set of relations is used from the aforementioned ACE05 corpus.

HMTL encodes the input sentence using multiple types of word embeddings (GloVe [67], ElMo [68], CNN-extracted Char Features ([49] and [17])). These embeddings are later used for multiple encoders (Multi-layer BiLSTM) to make predictions for all four tasks.

Similar to CoreNLP, HMTL is not specifically developed or trained to do KGP, but aims to solve subtasks of KGP.

[5]Similar to NER, except that also entities, which do not have a name, can be detected. In this thesis, we refer to NER and EMD as Entity Recognition (ER), in general.

Chapter 3

Datasets derived from Wikipedia

Manually labelling datasets for NLP tasks is a time-consuming and potentially error-prone task. Text with entity annotations could be helpful in a variety of NLP tasks when used as a starting point. Therefore, in this chapter we present an approach aiming to create a dataset from Wikipedia with ideally all named entities being annotated, which can be of use for several related NLP tasks, important for KGP.

3.1 Introduction

Generally, in order to train well-performing models, for example, for NLP tasks, ideally large amounts of labeled data are required. However, high-quality manually annotated data is often limited, especially if languages other than English are of interest.

Wikipedia[1] is a valuable source of information and is currently available in 325 languages[2] with the English version containing 6,383,000+ articles[3]. It is free to download and is already exploited for various NLP tasks, including Entity Linking (EL) (using Wikipedia alias dictionaries for candidate selection), e.g. see [36], or Named Entity Recognition (NER) (using hyperlinks, among others, as entity annotations), e.g. see [34].

[1] `https://www.wikipedia.org/`
[2] As of Jan. 20, 2022: `https://meta.wikimedia.org/wiki/List_of_Wikipedias`
[3] As of Jan. 20, 2022.

Wikipedia, as-is, is already able to provide datasets in the following ways:

1. NER: If an NER-type to Wikipedia article mapping is available, e.g. from [34], existing annotations in Wikipedia can be tagged with their corresponding type and used for NER models to be trained on. However, Wikipedia is not exhaustively annotated, and training such models on partially annotated data is challenging and leads to worse results than if fully annotated datasets are used, e.g. see [44] or [59].

2. Co-reference Resolution (CR): Wikipedia sometimes contains annotations of the same entities within the same article, even though authors are discouraged from linking the same entity per article more than once. Still, a dataset with such co-references for some entities can be created and a model can be trained on this dataset, even though more annotations in Wikipedia would be beneficial.

3. EL: Wikipedia already has entities linked as hyperlinks, e.g. [36] created a dataset where they leveraged hyperlinks as links between entities and their articles in Wikipedia.

4. Relation Extraction (RE): Using Distant Supervision, as applied by [70], and a KG, such as DBpedia [8], an RE dataset can be created through considering a specific relation between a specific pair of entities, as it appears in the KG and sentences expressing this relation if both entities are present.

We present our work on Wikipedia EXhaustive Entity Annotations (WEXEA), which is able to add a large amount of training data for each of these tasks[4]. In addition, datasets extracted by WEXEA can be considered as a starting point for manual data annotation since it presumably needs minimal refinement by human annotators, compared to starting data annotations from scratch.

Therefore, our contributions are:

[4]WEXEA aims to exhaustively annotate all entities in Wikipedia. However, the difficulty of this task heavily depends on the definition of an entity and we cannot claim that *all* entities can be actually annotated. Therefore, information on the exact kind of entities, that are ideally exhaustively annotated by WEXEA, can be found in Section 3.4

- Devising a new approach to annotate ideally all entity mentions in Wikipedia to create an annotated text corpus that can be useful in downstream tasks, for the English, German, French and Spanish versions of Wikipedia.
- Highlighting the usefulness of our approach for the task of creating datasets for RE using Distant Supervision. Many more sentences, for more relations, can be extracted, following our approach, than using a basic Wikipedia-based corpus without additional annotations.
- Generally WEXEA can provide the following language resources for multiple languages (as long as an NER tool is available): Wikipedia with additional annotations, RE datasets using Distant Supervision (need to be refined manually, ideally), NER datasets (article-type-mapping needs to be available), EL datasets, CR datasets (especially for the article entity, since these are often mentioned), parsed Wikipedia articles, dictionaries for hyperlinks, aliases, redirects, categories and many more.

The remainder of this chapter is structured as follows: Section 3.2 provides information about similar existing approaches. In Section 3.3, Wikipedia and related statistics are presented. In Section 3.4 our method is described in detail with an evaluation in Section 3.5, and our conclusions are presented in Section 3.6.

3.2 Related Work

There are two main lines of previous work that are important to our method: (1) Datasets with annotations similar to ours (mostly for NER) and (2) semi-automatically extracted datasets for RE using DS. Both are described below.

3.2.1 Wikipedia Annotated Corpora for NER

In [64], entities in Wikipedia are classified into one of the CoNLL-2003 entity types, i.e. *Person*, *Organization*, *Location* or *Miscellaneous*, in order to create a large annotated corpus based on the English Wikipedia. Already linked

entities are classified and additional links are identified through the use of 3 different rules:

1. The title of an article and all redirects are used to find more links of this article.

2. If article A is of type *Person*, the first and last word are considered as alternative title for article A.

3. The text of all links linking to article A is used as alternative title as well.

Their work is based on a 2008 Wikipedia dump. Co-references such as *he* or *she* or others that can be used to refer to certain entities are not considered, if not already in Wikipedia (typically it is not the case).

Ghaddar et al. [34] created the WiNER corpus and follow a similar approach using a 2013 Wikipedia dump. Their annotation pipeline is similar to the one of [64] and consists of three steps as well (typing information is used from Freebase):

1. Each article reachable via a hyperlink, i.e. out-links, is considered for search. Co-references, i.e. alternative entity names such as *NYC* or *New York* for the article *New York City*, from [33] are included in addition to the article titles.

2. Article titles reachable via out-links of out-links are searched.

3. Co-references from articles found in step 2 are included as well.

Conflicts (certain words may refer to multiple entities) are resolved through linking such a mention to the closest already linked article before or after.

The resulting corpora of both systems are evaluated using common NER approaches and corpora, and showed a slightly better result than training an NER system on other typically smaller datasets. Even though both datasets are publicly available, classifying entities into one of the 4 CoNLL-2003 classes introduces errors. Moreover, the original annotations are removed and cannot

be obtained anymore. This removes valuable information, e.g. for creating distantly supervised RE corpora or training CR systems, since it is not clear which annotation refers to which Wikipedia article.

The same authors created the system WiFiNE [35], which builds up on WiNER and includes more co-references and fine-grained entity typing.

The closest to our approach is the work of Klang et al. [47]. It uses an EL system with a pruning strategy based on link counts in order to keep the number of candidates per mention low (after running a mention detection algorithm). It also uses PageRank [11] combined with a neural network classifier to link mentions to their Wikipedia articles. The authors did not publish the code, and the data is not downloadable, to work with it offline[5]. Therefore, it is not possible to compare against this approach. In addition, co-references are not resolved.

Another similar but smaller dataset is the *Linked WikiText-2* dataset from [54], which is publicly available[6]. It consists of only 720 articles (600 train, 60 dev, 60 test). It was created using a neural EL system [37] and a CR system [58] to generate additional annotations to the ones given by the editors of the articles. However, using automatic tools introduce additional errors. Conversely, Wikipedia can be considered as partially annotated data, and therefore it would certainly be beneficial to consider these annotations, as our work and others in the literature suggest [34], [64]. The main issue with using an Entity Linker in such an unrestricted way is that it tries to link all mentions of entities, regardless of whether they have an article in Wikipedia or not.

3.2.2 Distant Supervision

Another line of work, relevant to ours, is extracting datasets from large text corpora for RE using Distant Supervision (DS).

Riedel et al. [70] run a manual inspection on a DS dataset based on Wikipedia as well as the New York Times corpus comparing the number of

[5]The article provides links in order to inspect the datasets online, but they are not available anymore (as of Sep. 29, 2022).

[6]`https://rloganiv.github.io/linked-wikitext-2`

Language	Articles	Edits	Active Users
English	6,441,277	1,062,121,045	121,778
Cebuano	6,110,506	33,782,787	194
Swedish	2,723,659	49,986,114	2,318
German	2,655,326	217,914,814	19,203
French	2,390,851	189,594,878	18,738
Dutch	2,078,405	60,740,414	4,234
Russian	1,787,519	119,176,670	11,411
Spanish	1,746,575	140,623,695	13,997
Italian	1,736,786	124,879,290	8,445
Egyptian Arabic	1,538,022	6,320,459	210

Table 3.1: Statistics of the 10 largest Wikipedias.

violations of the DS assumption on three relations and found that it is a lot less often violated in Wikipedia ($\approx$ 31% vs. $\approx$ 13%). This indicates that Wikipedia can provide DS data for RE systems and high-quality annotations presumably lead to better extractions. As we show in Section 3.5, WEXEA is capable of extracting many more relevant sentences than only using Wikipedia without additional annotations.

3.3 Wikipedia

Wikipedia is a free encyclopedia that exists for 325 languages of varying content size. Table 3.1 shows statistics about the 10 largest versions (the Swedish and Cebuano Wikipedias were largely created by a bot[7])[8]. Although Wikipedia exists for 325 languages, only 18 of all Wikipedias contain more than 1,000,000 articles and 253 contain only less than 100,000 articles, see Table 3.2[9]. Nevertheless, Wikipedia is a Knowledge Base of impressive size with more than 6 million articles in the English version. This leads to a huge potential for NLP research, as shown by Ghaddar et al. [34] for NER.

Wikipedia contains many more pages than articles, such as redirect or disambiguation pages as seen in Table 3.3. In the following, we explain certain

[7] `https://en.wikipedia.org/wiki/Lsjbot`
[8] As of Jan. 20, 2022: `https://meta.wikimedia.org/wiki/List_of_Wikipedias`
[9] As of Jan. 20, 2022.

No. articles	Languages
1,000,000+	18
100,000+	53
10,000+	89
1,000+	124
100+	30
10+	0
1+	9
0	2

Table 3.2: Distribution of number of articles and language versions.

Page type	Number
Redirects	8,440,863
Disambiguations (other)	189,124
Disambiguations (geo)	38,507
Disambiguations (human)	59,988

Table 3.3: Number of important pages other than articles.

features of Wikipedia and the annotation scheme proposed for editors, which we deem important for a proper understanding of our approach:

- Redirect pages: Wikipedia contains many redirect pages, e.g. *NYC* or *The City of New York* referring to the article of *New York City*. Editors can create these pages through adding alternative names for an article or they are created automatically, e.g. in case the name of an article changes and therefore broken links can be avoided through creating a redirect page.

- Disambiguation pages: Wikipedia contains many disambiguation pages, which are similar to redirect pages, except they deal with mentions that are knowingly referring to several different articles. For example, the disambiguation page of *New York*[10] refers to a whole list of articles including the city, state and many sports clubs located in New York City or the state of New York.

[10] `https://en.wikipedia.org/wiki/New_York`

- Typically, entities are only linked once in an article when they are mentioned first. Subsequent mentions should not be linked anymore.[11] In addition, pages do not contain links to themselves, e.g. there is no link to the article of *Barack Obama* within itself, although he is mentioned in there many times.

- Links can consist of two parts: (1) The article name that the link refers to (mandatory), and (2) an alias for that article since it is not always convenient to include the linked article's full name. This could look like the following link (following the linking scheme of Wikipedia): *[[Barack Obama|Obama]]*, resulting in a hyperlink with anchor text *Obama*, linking to article *Barack Obama*.

- Links, in general, should help the reader to understand the article and therefore should only be added if helpful (over-linking is to be avoided). This also means that articles, most readers are familiar with, such as countries, locations, languages, etc., are typically not linked.

Wikipedia's linking scheme aims for the best readability, but in order to be useful for NLP tasks, more annotations can be beneficial and we show in the remainder of this chapter that it is possible to add more annotations automatically.

3.4 Method

In this section we present WEXEA, which annotates as many mentions of entities as possible in Wikipedia articles.

The reader may expect WEXEA to be actually able to annotate *all* entities that can be found in Wikipedia. However, the definition of an entity is difficult to make and depends on the application. We consider the tasks of Named Entity Recognition (NER) and Co-reference Resolution (CR) as references here. Therefore, Named Entities and their co-references are considered as the entities and mentions WEXEA aims to annotate in Wikipedia (or keep

[11] `https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Linking`

existing annotations). This means that annotations in Wikipedia, which are not related to these kind of annotations, are removed and do not appear in the resulting datasets. For Named Entities, we only keep entities and their articles that more often than not start with a capital letter (considering the variety of anchor texts of incoming links of an article; this is a similar approach as described in [34]). Details on co-references can be found in Section 3.4.4 below.

In order to illustrate the problem that our approach aims to solve, consider the following sentence from *Tony Hawk*'s Wikipedia article[12] (original annotations in blue):

> "Tony Hawk was born on May 12, 1968 in **San Diego, California** to Nancy and Frank Peter Rupert Hawk, and was raised in San Diego."

Apart from the entity *San Diego, California*, a few other non-annotated mentions of entities appear in this sentence: (1) *Tony Hawk*, the entity of the current article, (2) his parents *Nancy Hawk* and *Frank Peter Rupert Hawk* (both currently not in Wikipedia), as well as (3) another mention of *San Diego, California*. Therefore, if correctly annotated, this sentence includes 5 mentions of entities, although only a single one is already annotated.

In general, editors should avoid to link mentions if they refer to entities that were already linked before or if they refer to very popular entities, and linking them would not contribute to understanding the meaning of a sentence[13]. However, WEXEA aims to add all missing annotations in Wikipedia in order to create an annotated text corpus that can be more useful in downstream tasks, than solely relying on already existing links or using an NER and EL to find and link more mentions, which introduces unnecessary errors.

In order to achieve this task, it can be broken down into several subtasks, including (1) dictionary creation for linking mentions to articles, (2) initial

[12] `https://en.wikipedia.org/wiki/Tony_Hawk`
[13] `https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Linking`

annotations, (3) co-reference resolution and (4) candidate conflict resolution, which are described below.

3.4.1 Dictionary Creation

The first step of our approach is to create the following dictionaries that help with the initial Mention Detection (MD) and EL, considering the hyperlinks that are added by the editors of the article:

- Redirects dictionary: Wikipedia contains many redirect pages, e.g. *NYC*[14] or *The City of New York*[15] referring to the article of *New York City*. These redirects are useful alternative titles that, presumably, solely refer to a certain entity (otherwise it would not be a redirect page, it could be a disambiguation page instead if several candidate articles are possible). Redirect pages can be either created by an editor to provide alternative titles or they are created automatically in case the title of an article changes.[16]

- Alias dictionary: This is another dictionary containing alternative names for articles, created through collecting anchor texts of hyperlinks referring to an article, e.g. *U.S.* and *USA* are both included in the alias dictionary since they appear in other Wikipedia articles linking to the article of the *United States*. Overlaps with the redirects dictionary are possible, but typically the alias dictionary contains more ambiguous aliases, e.g. *New York* referring to the articles *New York City*, *New York (state)* and many more. We only keep aliases that start with a capital letter, since only these can refer to Named Entities, and we ignore alias-entity links that only appear once in Wikipedia, since these are often not meaningful and can introduce unnecessary errors.

- Disambiguation page dictionaries: Wikipedia contains many disambiguation pages, which are similar to redirect pages, except they deal with

[14] `https://en.wikipedia.org/w/index.php?title=NYC&redirect=no`

[15] `https://en.wikipedia.org/w/index.php?title=The_City_of_New_York&redirect=no`

[16] `https://en.wikipedia.org/wiki/Wikipedia:Redirect`

mentions that knowingly refer to several different articles, e.g. the disambiguation page *New York* refers to a whole list of articles including the city, state and many sports clubs located in New York City or the state of New York. Often, these disambiguation pages have a certain type attached, mainly *human*[17] for persons or *geo*[18] for geopolitical entities. In case the page contains several types of entities, such as *New York*, it typically does not fall under one of these two categories. However, if it does, at least we can infer that the mention type is *Person* or *Geopolitical Entity*, even if we cannot link it to a specific article.

- We are ignoring stub articles[19], which are articles that are very short and do not contain a great deal of information. Usually, these are articles that are just started, but none of the editors has taken a closer look into it, and therefore the expected quality could be lower than the more popular and longer articles.

- We compiled a list of persons from the Yago KG [56] in order to figure out whether an article refers to a person, in which case the first and last word of the name can be also considered as alternative name for that person, again, as done in [64].

- Often very popular entities, such as countries or certain events (e.g. *World War II*) are mentioned in text without being linked. In order to link these mentions with high confidence we keep a dictionary of the 10,000 most popular articles (regarding the number of incoming hyperlinks found in all of Wikipedia) that can be linked to mentions, without being linked in the text at all.

- Wikipedia contains many articles about given names. We collect all articles for this dictionary through looking for the categories *Given names*, *Masculine given names* or *Feminine given names*[20].

[17] `https://en.wikipedia.org/wiki/Template:Human_name_disambiguation`

[18] `https://en.wikipedia.org/wiki/Template:Place_name_disambiguation`

[19] `https://en.wikipedia.org/wiki/Wikipedia:Stub`

[20] Categories in Wikipedia can be used by editors to group articles: `https://en.wikipedia.org/wiki/Help:Category`

3.4.2 Direct Mention Annotations

We found that applying just 3 or 4 rules in order to annotate mentions of entities in Wikipedia articles, as done in [34] or [64], is not sufficient, therefore we apply a relatively extensive set of rules. Apart from keeping the links corresponding to articles mostly starting with capital letters, we are detecting new potential mentions of articles using the following rules (applied to text between already annotated mentions):

1. The first line of an article often contains mentions of the article entity in **bold**, representing alternative names and are therefore annotated with the article entity.

2. At any time throughout processing an article, we are keeping an alias dictionary of alternative names for each linked article up until this point. This includes all aliases in the alias dictionary, all redirects and first and last word of an article in case it is a person. Since each article is linked once by the author, when it was mentioned first, these alternative names can be searched throughout the text after an article link is seen. If a match is found, the found mention can be annotated with the matching article.

3. We search for acronyms using a regular expression, for example, strings such as "Aaaaaa Bbbbbb Cccccc (ABC)", linking the acronym to the matching string appearing before the brackets, which was linked to its article before.

4. For all entity mention spans that are not found so far, we apply the NER of the CoreNLP toolkit [58] in order to find and type more entity mentions. This includes dates and times found by the SUTime library [15].

5. In many cases, the corresponding article for a mention is not linked in the text before (or cannot even be found in Wikipedia) and therefore the following rules are applied in these cases:

- If the mention matches an alias of the current articles' main entity and does not exactly match any other entities, it is linked to it.
- If the article matches one of the 10,000 most popular entities in Wikipedia, the mention is linked to this article.
- If it matches a disambiguation page and one of the previously linked articles appears in this page, the mention is linked to this article. These mentions are solely found by the NER and therefore the entity type is kept too.
- If the mention matches an alias from the general alias dictionary, it is linked to the most frequently linked entity given the mention.

6. We also apply rules in case there are conflicts (more than one potential candidate for a mention using previous rules):
   - If all candidates correspond to persons (sometimes people with the same first or last names appear within the same article), the person that was linked with the current mention more often, is used as annotation.
   - If a mention matches an alias of the current articles' main entity and more entities in the current alias dictionary, these are discarded in case the corresponding articles do not match the mention exactly.
   - Otherwise, in some cases conflicts cannot be solved this way and EL has to be used (see Section 3.4.3).
7. If no entity can be found by these ways, the mention is annotated as *unknown entity* or, in case a disambiguation page matches, this page is used and it sometimes contains the information that the mention corresponds to a person or geopolitical entity.

3.4.3 Candidate Conflict Resolution

Even after applying the rules explained in Section 3.4.2, it is still possible to end up with multi-candidate mentions, which the following sentence (from

Wikipedia[21]) illustrates:

> "Leicestershire are in the second division of the County Championship and in Group C of the Pro40 one day league."

Leicestershire in this sentence refers to the *Leicestershire County Cricket Club* although an alternative candidate, which is exactly matching, would be *Leicestershire*[22], which was mentioned in a previous sentence within the same article.

In order to resolve these conflicts we use the Entity Linker from [37], as used for the *Linked Wikilinks-2* dataset as well. The authors made the code and the used models publicly available[23]. However, we only use the Linker for multi-candidate mentions and not to find and link all mentions (which leads to errors if a mention refers to an entity that is absent in Wikipedia since the system always finds a link) and we modified it in a way that it only considers our candidate set for linking, not the candidate set from its own alias dictionary, since this set would include many more articles that are presumably not relevant.

3.4.4 Co-reference Resolution

Co-references that are not Named Entities (do not start with a capital letter), such as *he* or *she* for humans, *the station* for *Gare du Nord*[24] or *the company* for *General Electric*[25], should be linked as well. Our approach to achieve this is explained below.

As mentioned before, CR systems work increasingly well, although their performance is still far behind the performance of, for example, NER tools, e.g. [1] or [51]. We experimented with the state-of-the-art system from [51], but there are two main issues with it: (1) For long articles it takes several seconds to process (AMD Ryzen 3700x, 64gb, Nvidia GeForce RTX 2070) and

[21] https://en.wikipedia.org/wiki/Leicestershire_County_Cricket_Club
[22] https://en.wikipedia.org/wiki/Leicestershire
[23] https://github.com/nitishgupta/neural-el
[24] https://en.wikipedia.org/wiki/Gare_du_Nord
[25] https://en.wikipedia.org/wiki/General_Electric

is therefore too slow to annotate approximately 3,000,000 articles within a reasonable amount of time. (2) The model was trained in a fully open way, i.e. it has to find all mentions of entities and create one cluster per distinct entity, which is a very hard task. Whereas in our setting, many mentions are already annotated and a system only has to figure out whether there are more mentions (non-named entities) of the annotated entities in the article. Therefore, we decided to use a simple rule-based system.

Our system for CR considers a small set of co-references, depending on the type of entity. In order to find the type of an entity, we use the Yago KG [56] in order to retrieve all types of each entity. Yago is a system that links Wikipedia articles to a set of types, which consists of Wikipedia categories, as well as words from WordNet [61]. In case an entity is not of type "person", all WordNet types are considered as co-references, with "the" as prefix, e.g. *the station* (Gare du Nord) or *the company* (General Electric). For persons, *he*, *she*, *her*, *him* and *his* are considered as co-references. Also sometimes the article name includes the type of an entity, which can be considered as a co-reference as well, e.g. the type of *Audrey (band)*[26] is *band*. This results in an initial dictionary with zero or more co-references per article in Wikipedia.

In order to find out which of the co-references in the initial dictionary is actually used, we simple searched and counted each co-reference for each article. For example, we found that *General Electric* has type *company* in Yago and *the company* appears 19 times in its article. If a co-reference appeared more than a user-defined threshold, it was accepted. For persons, we looked for *he* and *she*, and if one of them appeared more than a threshold, the one appearing more often was accepted. This includes *his*, *him* and *her* as well, depending on the previous decision. We found setting this threshold to 2 worked reasonably well. This resulted in a dictionary of at least one co-reference for 825,100 entities in Wikipedia.

Using this dictionary, we can add more annotations to our corpus using the following procedure for each article:

[26] `https://en.wikipedia.org/wiki/Audrey_(band)`

1. Processing an article sequentially and whenever an already annotated entity mention appears, all its co-references are added to the current co-reference dictionary.

2. The text in between two entities (or start/end of article) are tagged using this co-reference dictionary, making sure that only previously mentioned entities can be used.

At any time, there can only be one entity attached to a certain co-reference, which effectively results in using the article matching a co-reference that appeared most recently in the previous text. Mentions that do not have a Wikipedia article, but were still annotated are classified into male or female human or something else using the gender guesser package[27]. We classify a mention as male, if there are no female-classified words in the mention and vice-versa for female mentions. Otherwise, the mention is not considered for annotation.

3.4.5 Multi-Language

CoreNLP is the main entity detection tool (apart from mentions found through existing hyperlinks), which is available in the following languages: English, Arabic, Chinese, French, German and Spanish[28]. We tested WEXEA on English, French, German and Spanish, even though more languages can be introduced through training new models.

The SUTime library for detecting dates and times is available in English and Spanish. We extended the rule-set for German and French, which mainly involved translating weekdays and months as well as adjusting regular expressions for detecting dates. It can be adjusted for more languages of interest or left out.

The EL system used for mentions of entities with multiple candidates was trained on English Wikipedia text [37] and therefore cannot be used for other languages. In this case a greedy EL system is used, which links the candidate

[27] https://pypi.org/project/gender-guesser/

[28] `https://stanfordnlp.github.io/CoreNLP/human-languages.html`

with the highest prior probability, i.e. the one that appears the most often with the detected mention in Wikipedia. For example, if the entity "New York" is found and the set of candidates consists of the articles for "New York City" and the "New York Yankees", the former would be chosen since the hyperlink count from "New York" to the article "New York City" is higher than for the "New York Yankees".

In addition, the CR system used cannot directly be applied to other languages. Hence pronouns are translated from English to the currently used language, and English entity names are still used, other co-references from Yago are removed since they would have to be translated including the corresponding definite articles used. This reduces the set of entities with such co-references, although still some of them can be detected for languages other than English.

3.5 Evaluation

In this section we present statistics of the datasets we created with WEXEA for the English, German, French and Spanish versions of Wikipedia and a visualization of a paragraph of an article in English.

3.5.1 Dataset Creation

Wikipedia is large[29] and processing all articles can be time-consuming, depending on the language used. WEXEA has two main steps[30]:

1. Dictionary creation, article split (storing each article separately), resolving templates, removing non-content articles, e.g. redirects, lists, categories etc. ($\sim$ 5h).

2. Removing non-named entity annotations and adding annotations through the alias and redirect dictionaries as well as running the CoreNLP toolkit and finding acronyms ($\sim$ 2d).

[29] 6,383,000+ articles in the English Wikipedia as of January 14, 2022: `https://www.wikipedia.org/`

[30] Runtimes are based on dataset creation on a AMD Ryzen 3700X with 64G main memory and the English Wikipedia (version with the most articles).

Especially step (2) takes a significant amount of time due to CoreNLP annotating each article. However, once all necessary dictionaries are created in (1), articles can be prioritized in step (2), e.g. only processing the most popular articles, to speed up the process. Wikipedia for languages other than English usually contains significantly less articles and, therefore, run time would be faster.

3.5.2 Visualization

Figure 3.1 shows a paragraph from Queen Victoria's English Wikipedia article[31] with the original in Figure 3.1a and the corresponding one generated by WEXEA in Figure 3.1b. WEXEA annotated many more entities than the original paragraph contains. Multiple annotations of Queen Victoria can be found compared to no annotations in Wikipedia since this is her article, and editors are not supposed to add links to the same article.

[31] `https://en.wikipedia.org/wiki/Queen_Victoria`

Victoria's father was Prince Edward, Duke of Kent and Strathearn, the fourth son of the reigning King of the United Kingdom, George III. Until 1817, Edward's niece, Princess Charlotte of Wales, was the only legitimate grandchild of George III. Her death in 1817 precipitated a succession crisis that brought pressure on the Duke of Kent and his unmarried brothers to marry and have children. In 1818 he married Princess Victoria of Saxe-Coburg-Saalfeld, a widowed German princess with two children—Carl (1804–1856) and Feodora (1807–1872)—by her first marriage to the Prince of Leiningen. Her brother Leopold was Princess Charlotte's widower. The Duke and Duchess of Kent's only child, Victoria, was born at 4:15 a.m. on 24 May 1819 at Kensington Palace in London.

(a) Original paragraph from Wikipedia. Multiple entities are not annotated.

Victoria's father was Prince Edward, Duke of Kent and Strathearn, the fourth son of the reigning King of the United Kingdom, George III. Until 1817, Edward's niece, Princess Charlotte of Wales, was the only legitimate grandchild of George III. Her death in 1817 precipitated a succession crisis that brought pressure on the Duke of Kent and his unmarried brothers to marry and have children. In 1818 he married Princess Victoria of Saxe-Coburg-Saalfeld, a widowed German princess with two children—Carl (1804–1856) and Feodora (1807–1872)—by her first marriage to the Prince of Leiningen. Her brother Leopold was Princess Charlotte's widower. The Duke and Duchess of Kent's only child, Victoria, was born at 4:15 a.m. on 24 May 1819 at Kensington Palace in London.

(b) Corresponding paragraph from the file generated by WEXEA with additional entity annotations. In order to simplify, only the anchor text for hyperlinks is shown, the linked entity is left out. Hyperlinks (blue) refer to an entity in Wikipedia, for all other annotations (green) no entity in Wikipedia can be found (mostly dates, times, numbers as well as other named entities). The original annotation of *succession crisis* (third sentence) was removed since it does not correspond to a named entity.

Figure 3.1: Same paragraph from Queen Victoria's English Wikipedia article as well as her corresponding article generated by WEXEA. Hyperlinks/Annotations in blue, linked articles are omitted due to readability.

3.5.3 WEXEA Statistics

Table 3.4 provides basic statistics of WEXEA when run on the English[32], German[33], French[34] and Spanish[35] Wikipedia versions. The number of rel-

[32]Wikipedia dump enwiki-20210220
[33]Wikipedia dump dewiki-20220101
[34]Wikipedia dump frwiki-20220101
[35]Wikipedia dump eswiki-20220101

evant articles[36] and sentences as well as the number of original annotations in Wikipedia and all annotations in WEXEA ("Entities total") are shown in total, averaged per sentence and article.

Language	English		German	
Type	Wikipedia	WEXEA	Wikipedia	WEXEA
Articles	2,676,086		1,929,698	
Sentences	148,866,723		83,426,382	
Entities total	62,026,078	320,142,453	45,383,570	149,776,874
- avg per sentence	0.42	2.15	0.54	1.80
- avg per article	23.18	119,63	23.52	77.62
Language	French		Spanish	
Type	Wikipedia	WEXEA	Wikipedia	WEXEA
Articles	1,568,460		1,137,844	
Sentences	64,214,837		43,794,620	
Entities total	26,113,542	97,482,667	17,059,545	74,824,888
- avg per sentence	0.41	1.52	0.39	1.71
- avg per article	16.65	62.15	14.99	65.76

Table 3.4: Annotation statistics for WEXEA compared to Wikipedia, for English, German, French and Spanish.

WEXEA extracts a number of potentially useful dictionaries and articles of a specific type, with statistics presented in Table 3.5:

- Article: Number of articles (in order to put numbers into perspective).
- Categories (assigned): Wikipedia contains a category hierarchy[37] to group together entities of a similar subject. Each article can have multiple categories attached, hence more category assignments than articles can be found.
- Disambiguation pages: These pages often contain lists of articles, which can be referred to by the name of the disambiguation page[38].
- Lists: List articles group articles of the same type[39].

[36]Not all article entities are considered as named entities, therefore the shown number of articles is lower than the one found on `https://www.wikipedia.org/`.

[37]`https://en.wikipedia.org/wiki/Help:Category`

[38]Disambiguation page of "New York": `https://en.wikipedia.org/wiki/New_York`

[39]List of sovereign states: `https://en.wikipedia.org/wiki/List_of_sovereign_states`

- Aliases: Each hyperlink in Wikipedia contains an anchor text (i.e. an alias) and the entity it refers to (not necessarily the same). Therefore this dictionary contains aliases and linked articles, important for EL in order to create a list of candidates per entity found in text.

- Links: Some EL systems, e.g. see [36], consider the links of a candidate to other entities in a document in order to link an entity to a candidate. This dictionary contains all links extracted through hyperlinks from an article to the linked entity.

- Redirects: This dictionary provides alternative names for many Wikipedia articles, e.g. *New York state* refers to the article with name *New York (state)*. Whenever an article in Wikipedia is renamed, a redirect page is created (if it is not ambiguous). Editors can also add redirects manually.

- Given names and surnames: Specific templates and categories mark articles about a *given name* or *surname*. These articles are stored in these dictionaries and can be used to identify entities of type person.

	English	German	French	Spanish
Articles	2,676,086	1,929,698	1,568,460	1,137,844
Categories (assigned)	21,067,025	11,702,075	9,509,203	5,641,208
Disambiguation pages	303,508	311,151	106,963	58,845
Lists	111,103	83.664	27,229	55,992
Aliases	4,620,268	3,019,351	2,315,007	1,818,029
Links	104,820,886	62,790,995	47,263,448	33,434,730
Redirects	4,811,019	1,444,249	1,089,731	1,434,897
Given names	10,482	7,103	1,327	976
Surnames	35,600	6,318	2,379	142

Table 3.5: Further annotation statistics for WEXEA and all languages.

Table 3.6 breaks up the annotation types of WEXEA:

- Annotations: Main annotations from Wikipedia.

- Article Entity: The article's entity, mentioned many times, but typically not annotated.

- Popular entities: Matching one of the 10,000 most popular articles in Wikipedia (after sorting by each article's number of hyperlinks), which are often not annotated due to the assumption the reader knows which article is mentioned.
- Single candidate: Annotations with a single candidate, typically referring to articles previously linked in the same article.
- Multi candidate: Multiple candidates are available, solved by the EL system.
- Co-references: Pronouns (found by the CoreNLP NER) and co-references for many entities based on their type.
- Acronyms: Acronyms and corresponding entities, which were not already annotated.
- NER: All other entities found and typed by the NER (including numerical entities only available in English).

	English	German	French	Spanish
Annotations	62,026,078	45,383,570	26,113,542	17,059,545
Article Entity	19,873,610	11,215,066	6,507,156	4,695,562
Popular entities	14,507,645	7,667,027	4,110,588	3,502,393
Single candidate	42,833,403	27,162,472	10,228,398	8,786,032
Multi candidate	2,861,934	894,593	316,171	223,584
Co-references	36,300,537	2,604	1,092	1,140
Acronyms	745,470	207,876	84,113	117,951
NER	140,993,776	57,243,665	43,808,292	32,023,229
Total	320,142,453	149,776,874	91,169,353	66,409,439

Table 3.6: Breakup of mention types found by WEXEA for all languages.

Since all annotations are marked accordingly, unwanted annotations, e.g. numerical entity types, can be filtered out.

3.5.4 Entity Annotations

Table 3.7 shows a small evaluation of a subset of the main entity annotations from WEXEA (apart from NER annotations) for 20 randomly selected

Annotation type	Accuracy	Number
Article entities	0.97	98
Popular entities	0.96	56
Candidate entities (single)	0.91	44
Candidate entities (multi)	0.67	3
Co-reference entities	0.69	67

Table 3.7: Accuracy and number of annotations for four different annotation types and 20 randomly selected articles from the English WEXEA corpus.

Wikipedia articles. Overall, the annotation quality for article entities, popular entities and single candidate entities is excellent. Multi-candidate entities do not appear often, a large number of articles needs to be evaluated to achieve a higher confidence level. The CR system of WEXEA is sometimes mislead, carrying an incorrect entity through a number of sentences.

3.5.5 Baseline Comparison

We compare the output of WEXEA to WiNER [34], which was described in Section 3.2. Unfortunately, the source code of WiNER is not available and furthermore, even though the resulting dataset can be downloaded, it is based on a 2013 Wikipedia dump with NER annotations only, i.e. actual entity annotations cannot be reconstructed anymore.

In order to fairly compare the rule-set of WEXEA and WiNER, we re-implemented WiNER's rules, although we had to make a few assumptions due to missing resources and missing explanations in [34]:

- We used an intermediate dataset from WEXEA's pipeline, which contains raw Wikipedia articles without irrelevant Wikipedia markup, such as templates and infoboxes, only existing annotations of Named Entities can be found here. Therefore, we ensure that both approaches start from the same source of text data.

- WiNER relies on an alias dictionary of articles of interest from [33]. As this dictionary is out-dated (again, based on a 2013 Wikipedia dump), we re-use WEXEA's alias dictionary as it is independently collected from the rule-set both approaches use for further annotations.

- In addition, WiNER claims to only annotate Named Entities, i.e. proper nouns. Therefore, we cleaned the alias dictionary so that co-references (such as pronouns and other non-proper noun aliases) are removed as they cannot be detected by WiNER, i.e. only aliases starting with a capital letter are kept.

- When aliases of article names are searched for in the article at hand, they are sorted in reverse order by length and exact matches considering word boundaries are kept. It is unclear how WiNER originally handled this, but we found that longer aliases lead to better annotations as this avoids recognizing the shorter nested entities, which both approaches are not necessarily aiming to recognize. For example, the longer mention of *New York Yankees* should be linked as a whole without considering just the first shorter part *New York* for annotation.

- We only consider the English dataset in this evaluation since WiNER was developed for English text only.

In this comparison we ensure that both approaches start from the same text source and alias dictionary. Differences are solely based on the rule-set used for further annotations (original ones matching the assumptions regarding proper nouns are kept), the core of both approaches.

We randomly chose 100 articles annotated by WiNER and WEXEA and kept the abstract of each article for evaluation. Wikipedia articles can be long and we aimed to evaluate a wide range of articles as one approach may work better on certain kinds of articles or topics than the other. This resulted in 100 abstracts with 462 additional annotations from WiNER and 312 from WEXEA. We considered these rules in order to consider annotations as correct/incorrect or missing:

- We only considered the union of annotations of both approaches as potential annotations, i.e. articles that were not mentioned by both approaches were not considered. The reason for this is that it is not always clear for the annotator whether the actual entity can be found

in Wikipedia, which is often not the case. In this case, WiNER and WEXEA typically annotate a nested entity, which may be correct or incorrect and is counted as such. Therefore, if both approaches missed an annotation, we ignore it.

- If one approach annotates a certain mention of an entity correctly, but the other one missed the annotation or used an incorrect entity, we counted a missing or an incorrect annotation, respectively, for the latter approach, and a correct one for the former.

Table 3.8 shows the result of the manual evaluation. WiNER has a notably lower precision than WEXEA, i.e. instead of missing certain annotations, WiNER prefers to add an annotation, even if it is incorrect, hence the higher recall. We found many annotations of commonly used words, such as *The* for the article *The New York Times* or *It* for the article *Italy*. WiNER was developed based on a 2013 Wikipedia dump. Today's Wikipedia consists of over 50% more articles[40] and presumably many more aliases per article as articles grew as well. This seems to result in more incorrect annotations than originally anticipated by the authors, considering WiNER's rule set. In addition, WiNER was developed to provide NER tags, instead of Wikipedia articles, as annotations. A later step converts article annotations to NER tags. Therefore, for WiNER there is no distinction between articles with the same aliases, such as *New York (state)* and *New York City*, as they are both locations.

WEXEA, on the other hand, achieved an F1-score of 0.82, compared to 0.65 for WiNER. Even though WEXEA missed more annotations than WiNER, the added annotations are better, leading to a much higher precision.

Approach	P	R	F1
WiNER	0.54	**0.81**	0.65
WEXEA	**0.91**	0.75	**0.82**

Table 3.8: Results on WEXEA vs. WiNER, based on the abstracts of 100 articles.

[40] `https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia`

The source code of WiNER (using WEXEA dictionaries) and the set of articles considered for evaluation, is publicly available[41].

3.5.6 Dataset Creation

WEXEA corpora can be used for multiple NLP tasks, such as NER, EL, CR and RE, and here we describe how datasets can be extracted or how these corpora can be used without modification. Since WEXEA does not provide gold-standard annotations, these datasets can be used as a starting point to pre-train models or manually annotate data, especially if high-quality training data is very limited or unavailable (specifically for languages other than English).

Named Entity Recognition

Models for NER are typically trained for token classification, e.g. see [49], i.e. every token in the training set needs to be properly annotated. Wikipedia cannot be used due to many missing annotations and partially annotated datasets for NER can significantly reduce model performance, e.g. see [44] or [59]. WEXEA can fill these annotation gaps and therefore the resulting corpora can be used for NER, if an article-type mapping can be provided. DBpedia, which contains type-relationships for each entity[42], or the system NECKAr [32] for typing Wikidata entities can be used here.

Entity Linking

Similar to the dataset created by [36], large datasets for EL can be extracted from WEXEA corpora without modification. Annotations linking entities to an article in Wikipedia can be used to train such systems. Furthermore, EL systems rely on alias dictionaries in order to create a set of candidates for each entity found in text. These alias dictionaries should be updated frequently since the set of articles/entities of interest increases constantly and WEXEA can be used to provide these dictionaries.

[41] `https://github.com/mjstrobl/WEXEA`

[42] For example, see Barack Obama's DBpedia article: `https://dbpedia.org/page/Barack_Obama`

Co-reference Resolution

CR systems aim to detect clusters of co-references for each mentioned entity, e.g. see [51]. Therefore, datasets for CR need to contain spans of text corresponding to entity annotations and a unique identifier for each entity mentioned to link co-references to a unique entity. WEXEA does contain additional entity annotations and unique identifiers for many annotations (except NER tagged annotations without articles attached). For example, the article entity is in average mentioned over seven times per article in the English WEXEA corpora (see Table 3.5 and 3.6), which corresponds to one entity cluster, which can be used for CR model training.

Relation Extraction

Many relations and corresponding entity pairs can be extracted from DBpedia[43]. These pairs of entities can be found in Wikipedia and WEXEA corpora, and datasets for RE based on Distant Supervision can then be created. Table 3.9 shows the amount of data (sentences), which can be extracted from these corpora, i.e. sentences that contain annotations of entities, which are linked with a certain relation in DBpedia. This is done for all relations for which at least 100 sentences each can be extracted[44]. Using WEXEA, many more relations, sentences and unique pairs of entities per relation can be found, when compared to Wikipedia. When a large number of sentences can be extracted for a low number of unique pairs, an RE model may memorize relations for certain pairs rather than recognizing relations in text. WEXEA is able to extract a more significant number of unique pairs than Wikipedia contains.

In the following, we show the sentence distribution of each dataset, one

[43]It is important to know that DBpedia is based on Wikipedia, i.e. DBpedia mainly contains information from Wikipedia infoboxes. Therefore, each entity in DBpedia contains a direct link to its Wikipedia article name, which can be used to link entities in Wikipedia/WEXEA articles to their corresponding DBpedia entity. Since relations in DBpedia have a subject and an object, pairs of entities, which are linked in DBpedia, are of interest here.

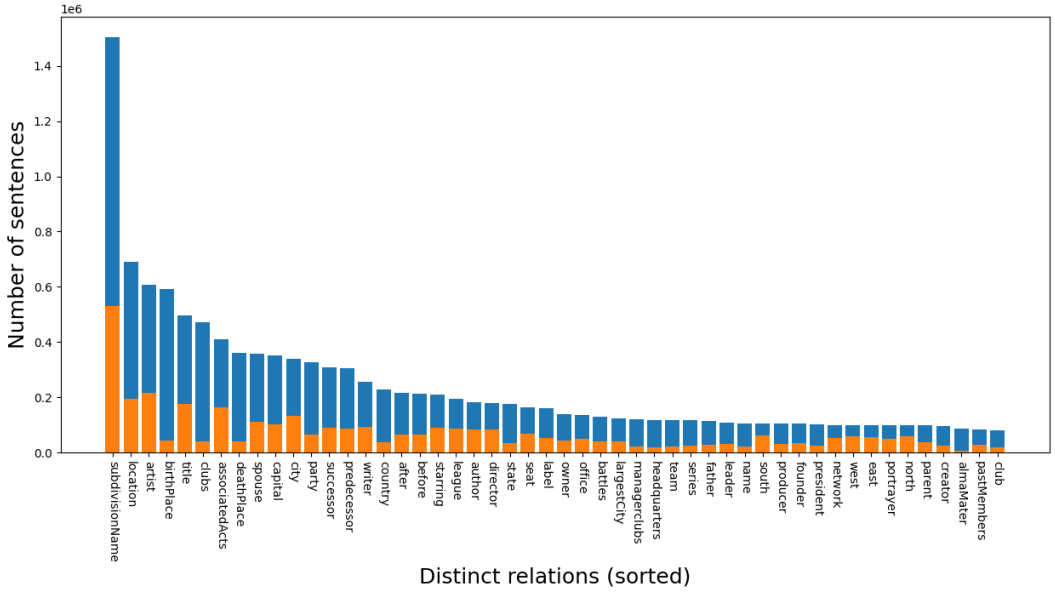
[44]This was mainly done to avoid obscure relations in DBpedia. In addition, datasets for RE created with Distant Supervision are typically more noisy than manually annotated datasets as described in Chapter 7. Therefore, small datasets with less than 100 sentences per relation do not seem to be useful.

Language	English		German	
Type	Wikipedia	WEXEA	Wikipedia	WEXEA
Relations	1,458	2,322	616	928
Sentences (total)	6,3sec23,758	20,970,686	1,702,887	5,399,588
Unique pairs (avg)	1,017	1,540	709	1,935
Sentences (avg)	4,337	9,031	2,764	5,819
Language	French		Spanish	
Type	Wikipedia	WEXEA	Wikipedia	WEXEA
Relations	746	1,048	640	893
Sentences (total)	3,392,807	6,577,523	1,862,071	4,689,909
Unique pairs (avg)	971	1,289	651	1112
Sentences (avg)	4,548	6,276	2,909	5,252

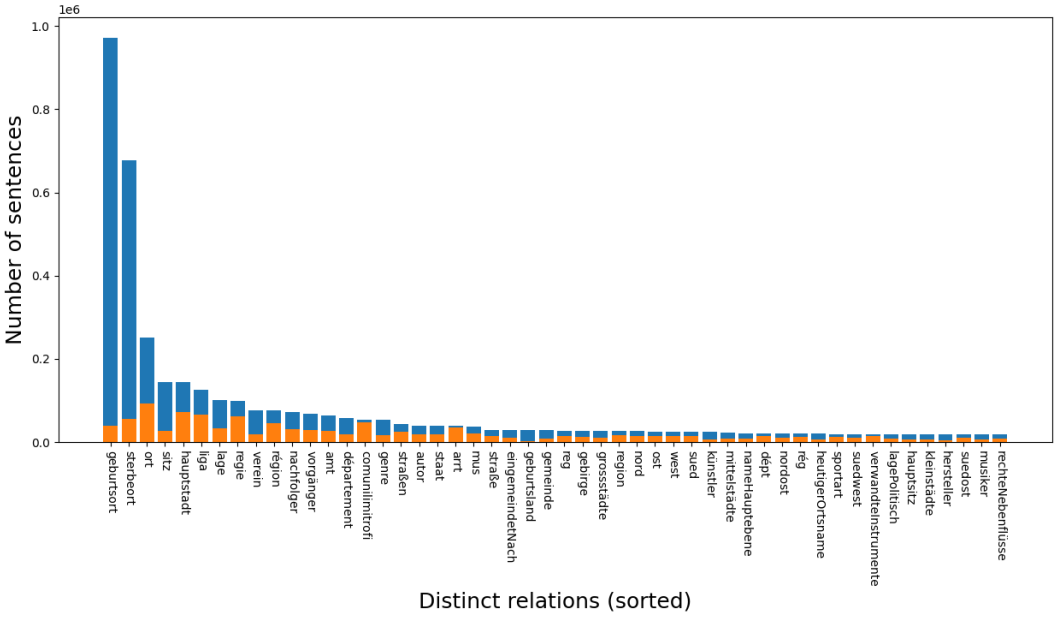
Table 3.9: Statistics for datasets based on Distant Supervision created using Wikipedia and WEXEA corpora with DBpedia: Number of relations for which at least 100 sentences can be extracted, extracted sentences total, unique pairs of entities averaged per relation and average number of sentences per relation.

for each relation, extracted from both corpora for all four languages. More specifically, if a sentence contains a pair of entities, which are linked with a certain relation in DBpedia, this sentence can be extracted and added to the Distant Supervision dataset for this relation.

Figure 3.2 shows the distribution of sentences for the top 50 relations (based on the number of sentences extracted from WEXEA) for all four Wikipedias (orange) and WEXEA (blue). Again, many more sentences can be extracted from WEXEA for each of these relations.

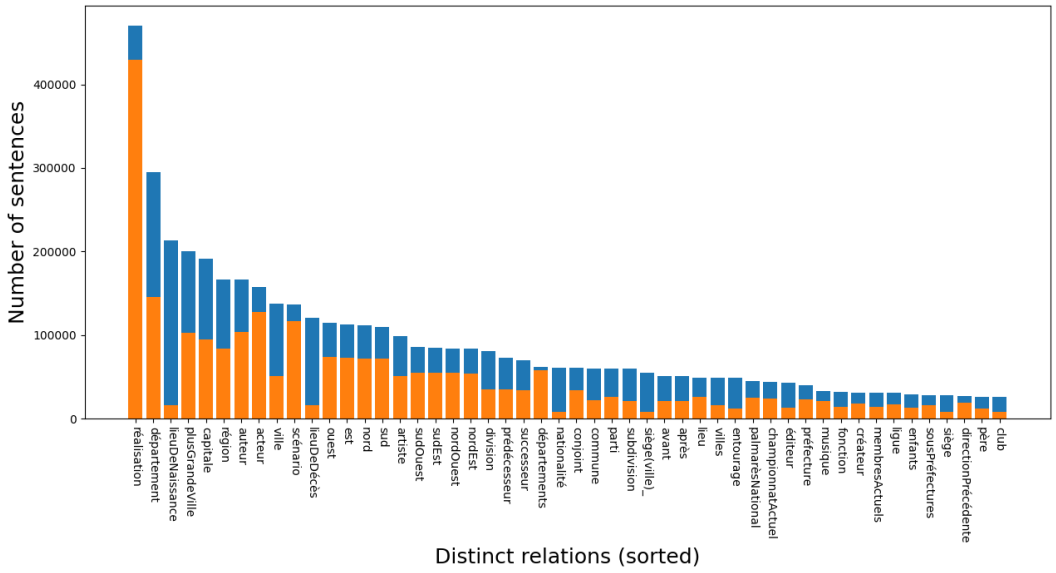


(a) English Wikipedia and WEXEA.

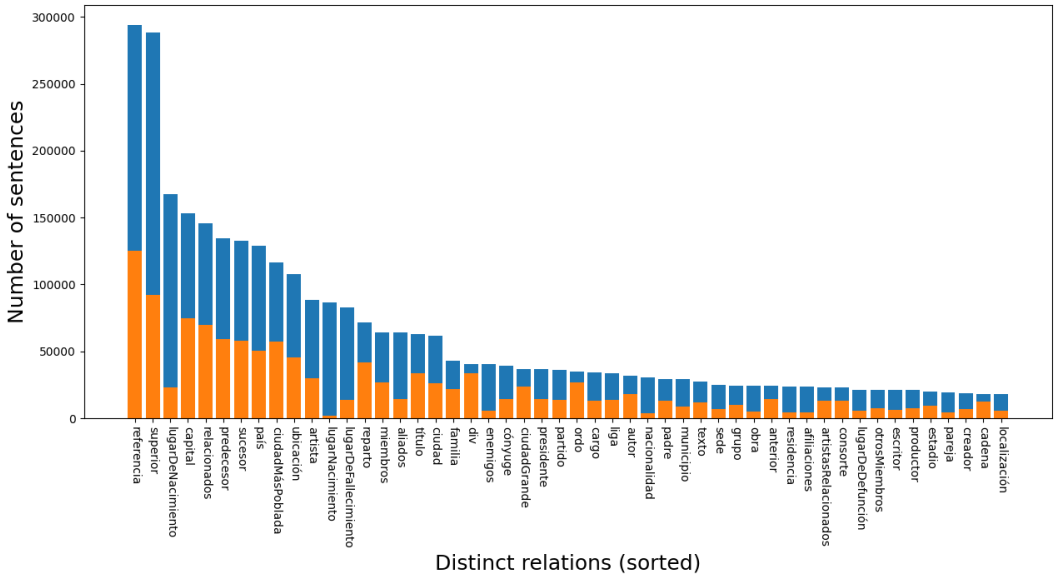


(b) German Wikipedia and WEXEA.

Figure 3.2: Number of sentences per relation (top 50) for Wikipedia (orange) and WEXEA (blue; in addition to Wikipedia).



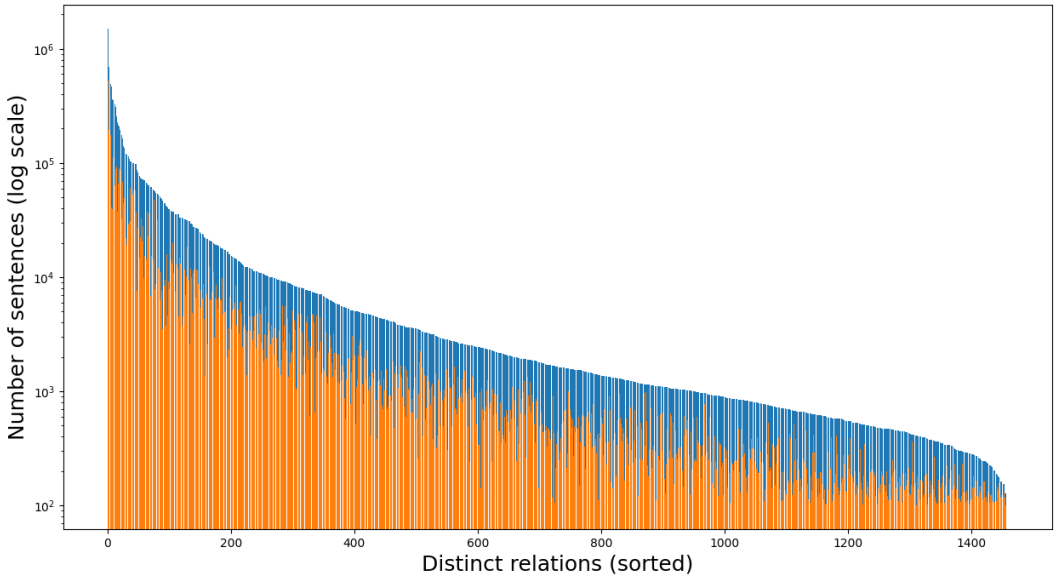
(c) French Wikipedia and WEXEA.



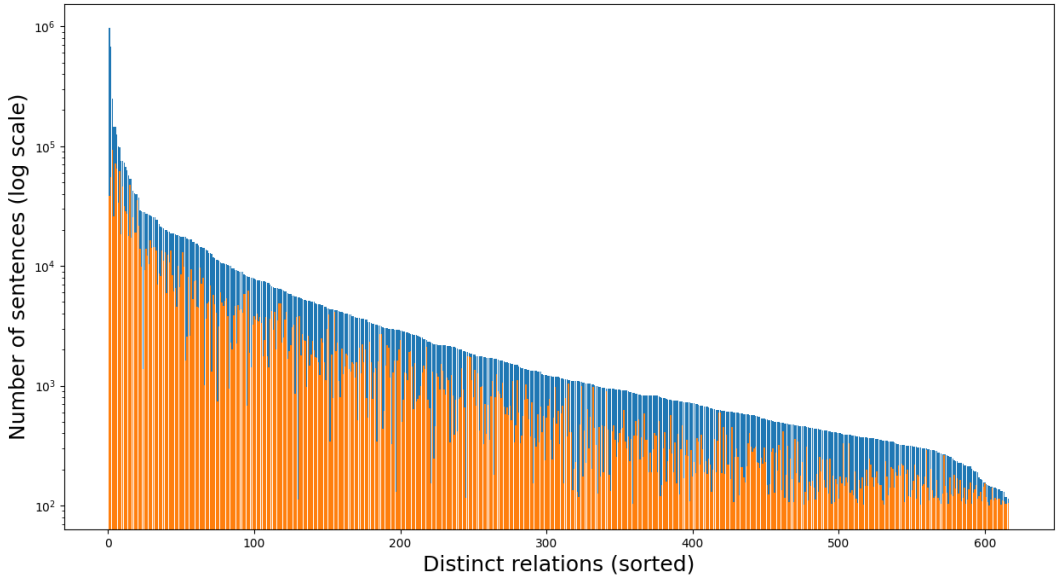
(d) Spanish Wikipedia and WEXEA.

Figure 3.2: Number of sentences per relation (top 50) for Wikipedia (orange) and WEXEA (blue; in addition to Wikipedia) (cont.).

Figure 3.3 similarly shows the distribution for all relations and number of sentences (note the log scale on the y-axis in this case).

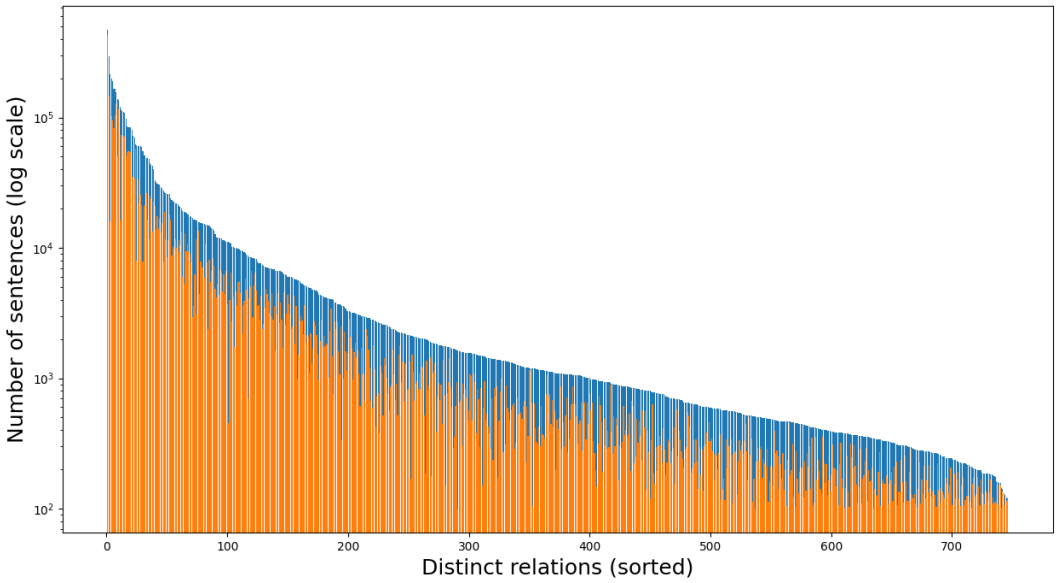


(a) English Wikipedia and WEXEA.

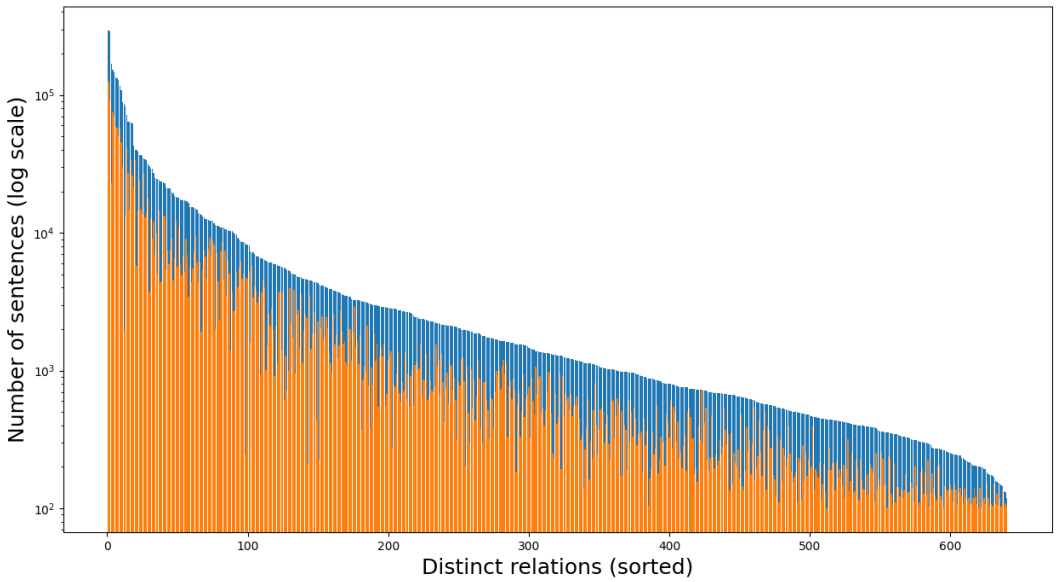


(b) German Wikipedia and WEXEA.

Figure 3.3: Number of sentences per relation (log scale) for Wikipedia (orange) and WEXEA (blue; in addition to Wikipedia).



(c) French Wikipedia and WEXEA.



(d) Spanish Wikipedia and WEXEA.

Figure 3.3: Number of sentences per relation (log scale) for Wikipedia (orange) and WEXEA (blue; in addition to Wikipedia) (cont.).

3.6 Conclusion

We created WEXEA, in order to annotate additional mentions of named entities in Wikipedia, resulting in a large corpora and many more annotations

than the original Wikipedia dump contains. The code is publicly available[45] (including the implementation of WiNER [34] and the set of articles used for evaluation in Section 3.5.5) and can be applied to the latest Wikipedia dump, which is free to download, by everyone. Furthermore, we showed how this can be useful for Relation Extraction datasets based on such a corpus, DBpedia and Distant Supervision. Again, many more sentences can be extracted for more relations than using a Wikipedia-based corpus without additional annotations.

So far we were only concerned with creating a corpus using the English, German, French and Spanish versions of Wikipedia. However, Wikipedia is available for 325 languages and although the number of articles per language varies significantly, we believe that our approach can be used for other versions as well in order to create similar corpora, especially since we are only using minimal language-dependent resources.

Chapter 4, 6 and 7 show how datasets created through WEXEA can be used to train models and potentially refined.

[45] `https://github.com/mjstrobl/WEXEA`

Chapter 4

Flexible Relation Extraction Data Annotation

In order to train accurate models for many NLP tasks, such as Named Entity Recognition (NER), Entity Linking (EL), Co-reference Resolution (CR) or Relation Extraction (RE), sufficient and properly labeled data is required. Adequately labeled data is difficult to obtain and annotating such data is a tricky undertaking and, typically, either efficiency or effectiveness has to be sacrificed. Therefore, this chapter presents our tool for Flexible Relation Extraction Data Annotation (FREDA) to manually annotate data as quickly and accurately as possible, mainly shown for the task of RE, as it is a more complex task to annotate data for, at least compared to ER and CR.

4.1 Introduction

The main goal for the task of RE is to recognize relations, which are expressed between two entities mentioned in the same sentence or document. At present, this is usually achieved by a model based on neural networks, which is trained on, ideally, large amounts of labeled data. However, there are very few publicly available labeled datasets. When available, these are usually limited to specific relations, and often lack the relations that one is interested in. Therefore, depending on the application, the creation of new datasets may be necessary.

For other tasks, such as NER, similar issues can arise. The most commonly used dataset for NER is the CoNLL 2003 dataset [72] from newswire

text data, annotated for the classes *Person*, *Location*, *Organization*, *Miscellaneous*[1]. However, other datasets may use a different set of entity classes, e.g. the ACE 2005 dataset[2] [41] for the types *Person*, *Organization*, *Location*, *Facility*, *Weapon*, *Vehicle* and *Geo-Political Entity*. In addition, there may be a mismatch between the type of text used for training versus testing, e.g. newswire and social media text. This could lead to decreasing model performance, as can be seen in [34].

Even though manual data creation may be time-consuming, the use of crowd-sourcing may lower the time overall needed to produce large amounts of required data. The TAC Relation Extraction Dataset (TACRED) [97] was created using Mechanical Turk crowd annotation, although recently Alt et al. [3] gave some insights on TACRED, suggesting that more than 50% of the challenging examples, i.e. trained models make mistakes on, may need to be relabeled. On the other hand, carefully manually annotated datasets for RE with multiple annotators (which can lower the chance of incorrect annotations ending up in the dataset), such as KnowledgeNet [60], are extremely time-consuming to create and therefore often not feasible.

In this chapter, we propose a tool that makes it possible to create high-quality datasets for various NLP tasks with a moderate amount of time and effort. An evaluation is provided in Section 4.4 comparing annotation times with our tool and a baseline.

4.2 Related Work

There is a variety of web-based annotation tools available, open-source tools, such as BRAT [81], as well as proprietary ones, such as Prodigy[3], which can be used for RE data annotation, among a variety of other NLP tasks.

We used BRAT as open-source example of such tools and Figure 4.1 shows its annotation interface. A span of text can be selected with a computer

[1]The *Miscellaneous* class contains names which do not fit one of the other classes, without further specification.

[2]`https://catalog.ldc.upenn.edu/LDC2006T06`

[3]`https://prodi.gy/`

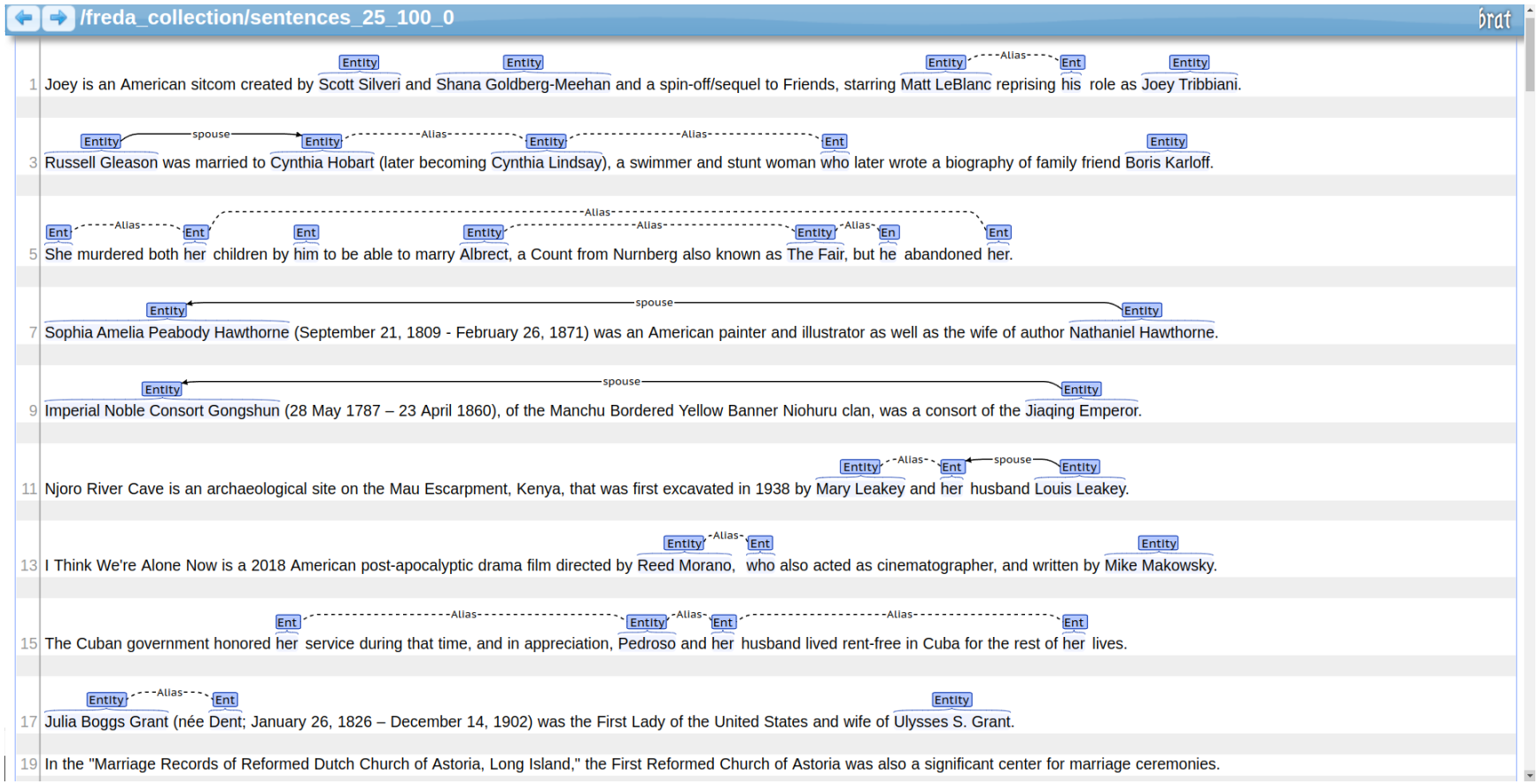


Figure 4.1: Annotation interface of BRAT, including annotations for the *spouse*-relation.

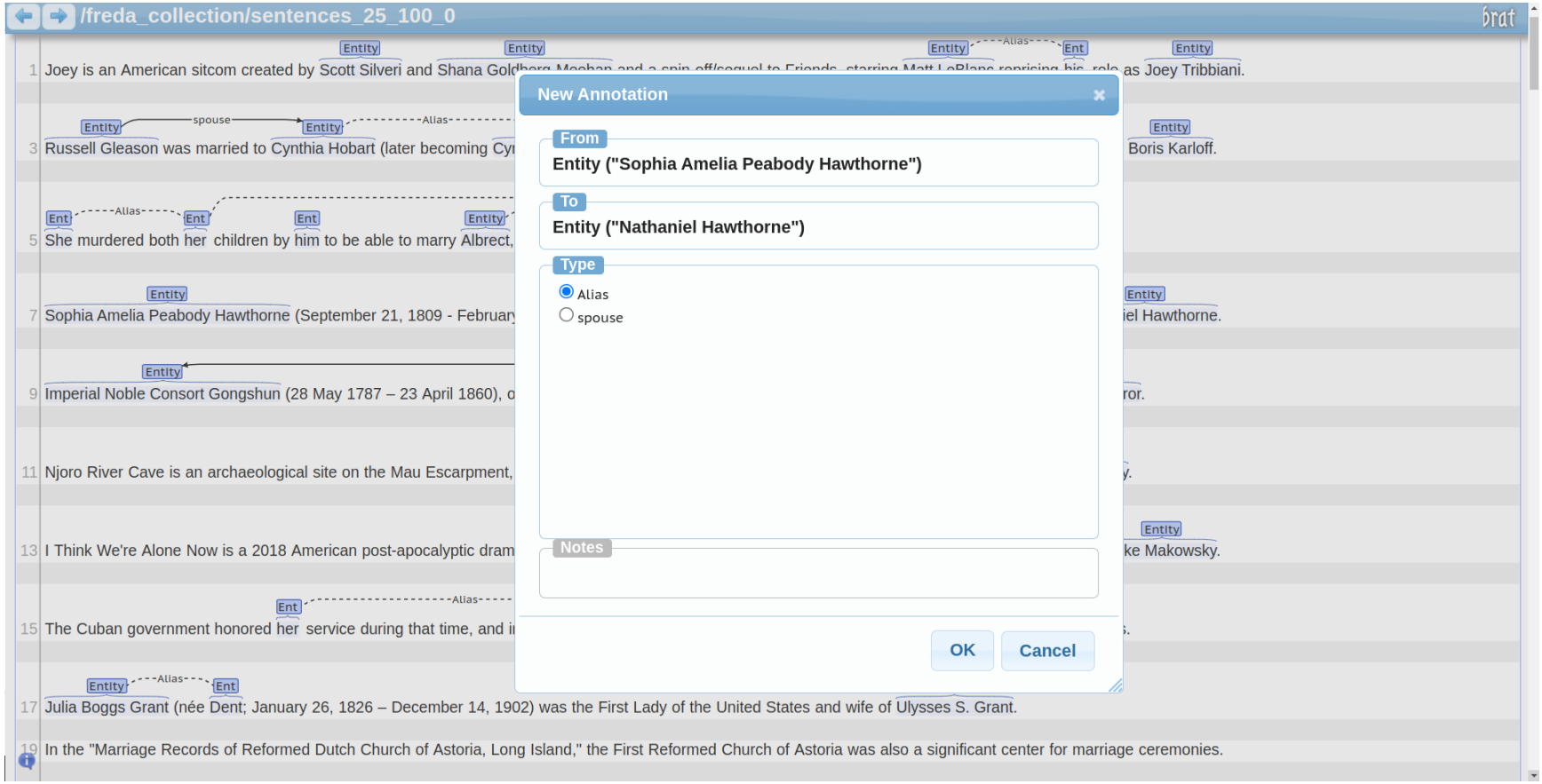


Figure 4.2: Popup for relation annotations with BRAT, similar for entity type selection.

mouse, triggering a popup which is used to select a type of entity (mainly for NER annotations) and adding the entity annotation to the interface. In addition, entities can be connected through dragging an arrow from one entity to another to add a directed relation, again, through triggering a popup (see Figure 4.2) where the relation can be selected. In order to add co-references, this functionality can also be used to add these as *aliases*. While it is possible for a second or third annotator to double-check previous annotations, there may be a bias (the subsequent annotator can either see no or all previous annotations). Therefore, without additional data processing, an independent or partially independent annotation scheme for multiple annotators cannot be realized with BRAT. Apart from such annotations, BRAT maintains a collection of datasets and their annotations and multiple users can be added.

However, data annotation with BRAT turns out to be a tedious and time-consuming task, similar to what was reported in [60], where annotations of a single sentence took 3.9 minutes in average[4].

4.3 Flexible Relation Extraction Data Annotation

In this section we describe the general architecture of our annotation tool for Flexible Relation Extraction Data Annotation (FREDA).

4.3.1 General Architecture

Figure 4.3 shows the architecture of FREDA, involving the following subsystems:

1. WEXEA as data source: Pre-annotated data, even if not perfect, can reduce the workload per annotated sentence significantly. Therefore, our approach utilizes data from WEXEA, which contains large amounts of

[4]This includes two or three annotators (third only in case the previous two disagree), although an EL step was included, which accounted for 28% of the time.

text from multiple languages for a wide variety of topics and entities and can be used for various annotation tasks.

2. Pre-processing: Depending on the task at hand, the data needs to be preprocessed. This involves a filtering step, e.g. for sentences to be annotated for a specific relation for RE. In addition, sentences need to be prepared for display.

3. The database contains all data subject to annotation and keeps track of the annotation progress.

4. The server serves data to the user and, therefore, communicates with the database and the user application. Also it ensures that multiple annotators can work on the same dataset and a third annotator is asked whenever the previous two disagreed. The measurement of "disagreement" has to be implemented for each task individually.

5. The tablet application used for the actual data annotation task.

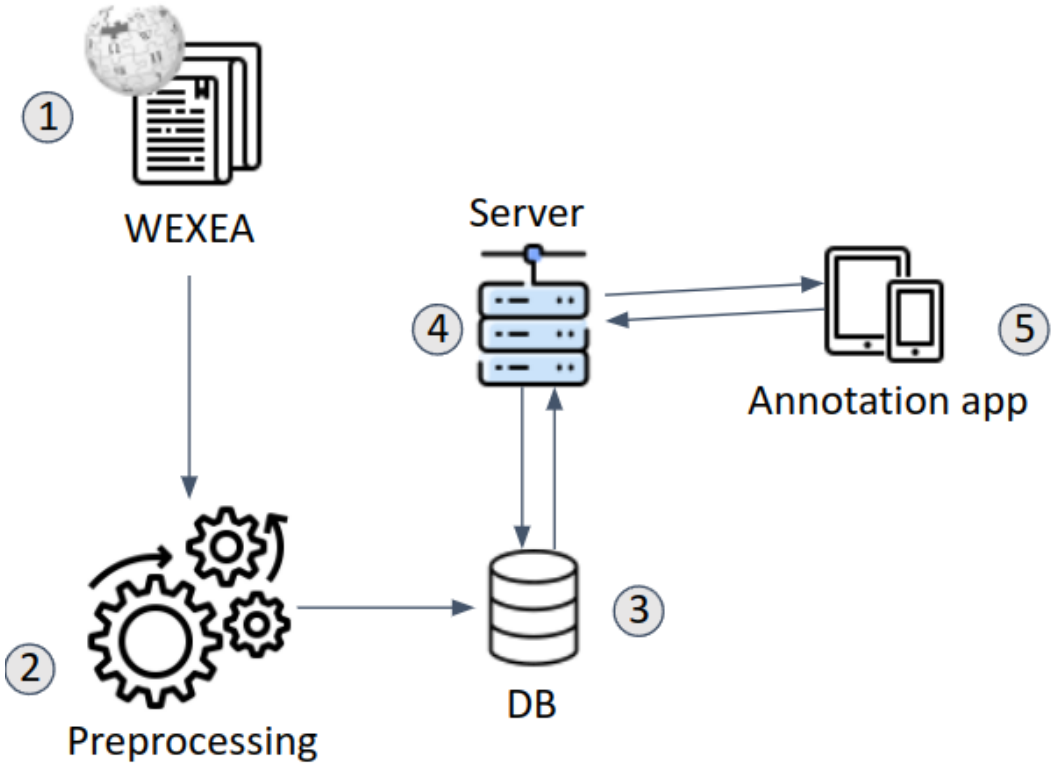


Figure 4.3: General architecture.

4.3.2 WEXEA as Data Source

A major part of various NLP data annotation tasks is selecting entities and their co-references. Therefore, in the ideal case, a corpus should be used with entities already labeled. For example, a corpus for NER, such as the CONLL 2003 dataset [72], could be used. However, manually labeled corpora are usually limited in size and variety of topics and types of text, therefore it is unlikely that they contain enough data that could be "interesting" or even relevant for a specific task. To avoid this issue of potentially running out of text data, we are using sentences from WEXEA (see Chapter 3), which are semi-automatically pre-annotated and WEXEA contains millions of processed Wikipedia articles for four languages: English, German, French and Spanish. Since WEXEA is sentence-based, only a small amount of data is displayed to the user at once since the goal is to transform the tedious tasks of annotating large amounts of data into a better flowing task, which can be paused and picked up again without much effort.

4.3.3 Adding Data from WEXEA

If all available sentences are considered for annotation for each relation, the percentage of relevant sentences[5] is expected to be very low. This is an issue since the number of relevant sentences for each relation is presumably very imbalanced for a corpus without pre-selection, as it is the case for TACRED, for example. The by far most common relation in this dataset is *per:title* with 3,862 examples, whereas 37 out of 41 relations have less than 1,000 examples with 4 relations having even less than 100. This lead to the development of a more challenging and balanced RE dataset and models trained on TACRED can be evaluated on it [71]. Therefore, sentence filtering ensures that each relation has enough relevant examples to make sure a model can be trained on all relations properly. We are using a similar approach as used by [60]:

- Keywords: We define a set of keywords relevant to each relation, which

[5]It is somewhat subjective what a relevant sentence is for a specific relation. It could be a sentence which contains entities of both entity types participating in the relation or a sentence being somehow related to the relation topic-wise.

are used to filter sentences. This is done for each relation separately. We chose relevant words for each relation including WordNet synonyms [61]. The keywords used by [60] were not reported. We also used keywords related to entities for specific relations, e.g. *company* or *university*.

- Distant Supervision (DS): Since it is difficult to define an exhaustive set of keywords, a DS approach is used to find sentences not matching these keywords, but still containing entities, which are known to be related to each other with a relation of interest. DS is typically used to add positive examples before training. In our case, we use DS to add candidate sentences for annotation, i.e. the actual DS annotations are not presented to the user, it is only used to filter sentences for a specific relation DS-datasets are available for. We are using DBpedia [6] as KB and extracted all entity pairs for each relation, which can be found in DBpedia. Elsahar et al. [26] pointed out that Wikipedia itself often does not contain entity links, even though a specific entity is mentioned in a sentence through a pronoun or other co-references. This is due to the aforementioned Wikipedia guidelines, which ask authors not to link entities multiple times and not to link an entity the article is about in the same article. However, WEXEA is supposed to fix this and contains links to these entities. Depending on the number of distinct entity-pairs, which can be found in DBpedia, this should lead to a higher number of extracted sentences compared to using a raw Wikipedia dump, as described in chapter 3.

DS for a specific relation can only be used if this relation is part of the KB (DBpedia in our case), which is not the case for all relations we annotated sentences for. But if it was the case, up to 50% of the relevant sentences are extracted this way.

4.3.4 Data Annotation

The application for data annotation contains a dataset overview for each task (corresponding to a single relation for RE) from which a user can select, shown

in Figure 4.4. The overall progress is shown for each available dataset. The columns for *Annotator 1/2/3* show the number of sentences annotated by the user of this device as Annotator 1 (first one to see a sentence), 2 (second one to double-check whether first annotator was correct) or 3 (in case of disagreement; see below for an explanation of how this is measured in this case). In addition, the last three columns show the overall progress on each dataset with *Once* counting sentences with one or more annotators, *Twice* counting sentences with at least two annotators and *Full* counting sentences with two (no disagreement) or three annotators (disagreement resolved by third annotator).

Figure 4.5 provides an overview of the annotation interface. The objective of the tool is to facilitate the annotation task and reduce the cognitive load for its users. This would lead to collecting annotations of decent quantities and high quality in a relatively short amount of time.

Given a particular relation, the tool initially provides a sentence with entity annotations highlighted in different colours in the *Sentence View*. These initial entity annotations are extracted and loaded by leveraging the WEXEA dataset. The *Entity View* contains distinct entity buttons which are unique labels representing each entity annotated in the *Sentence View*. Thus, if multiple different mentions or co-references to the same entity occur in a sentence (*Sentence View*), they would all be represented by one entity button in the *Entity View*. The colour of that entity button and all corresponding mentions in the *Sentence View* would be the same. This would help reduce ambiguity and facilitate the decision making.

Another practical advantage of the tool consists of reducing the indication of the subject (SUBJ) and the object (OBJ) entities in the relation to just a simple press of the corresponding entity buttons in the *Entity View*. In other words, it is not necessary to look through every single mention of an entity and indicate the role. Annotating only the representative entity button in the *Entity View* is sufficient.

The *Word View* contains buttons representing every single token in the sentence at hand. By using this view, conducting different editing operations

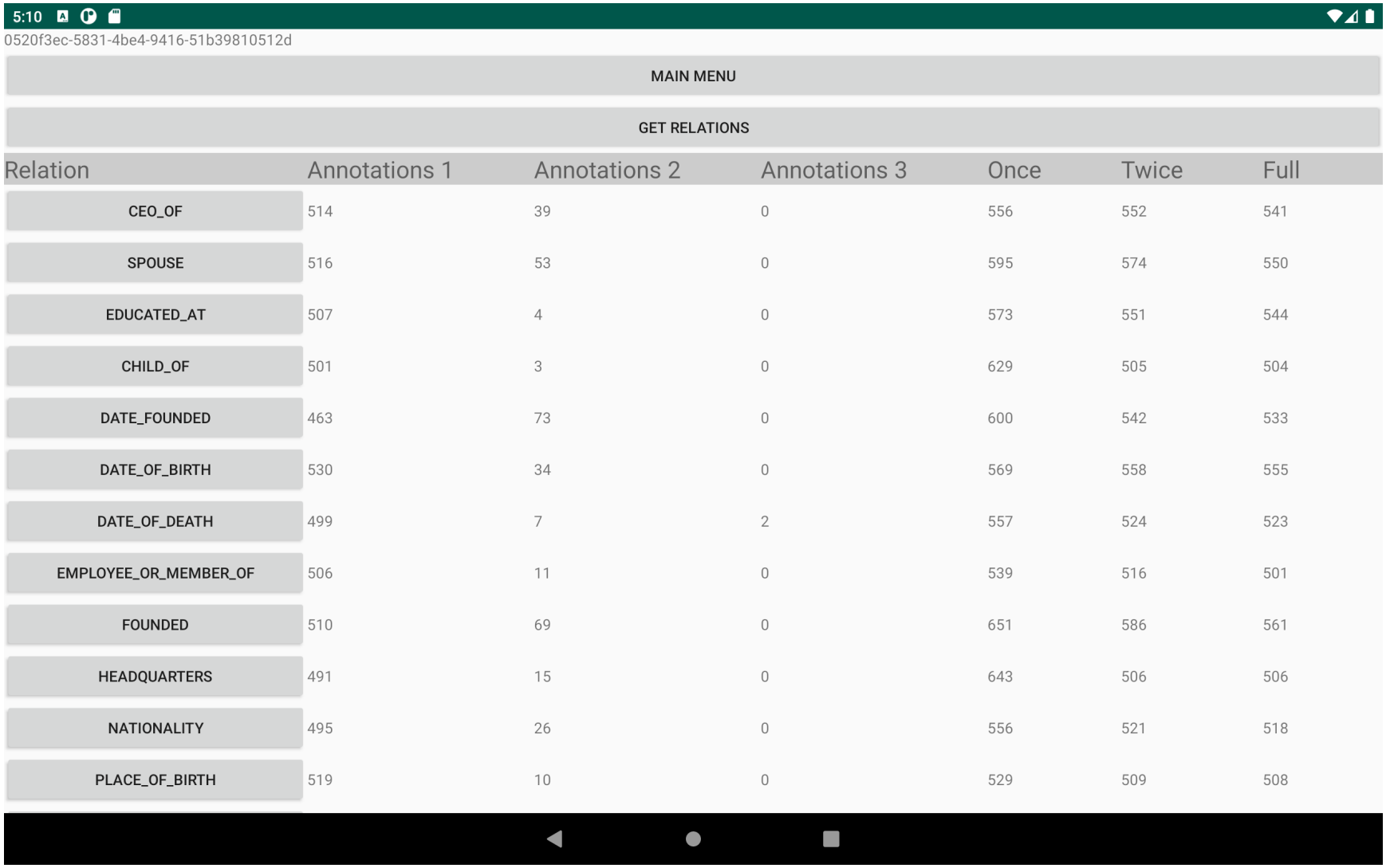


Figure 4.4: Overview of available datasets for RE data annotation.

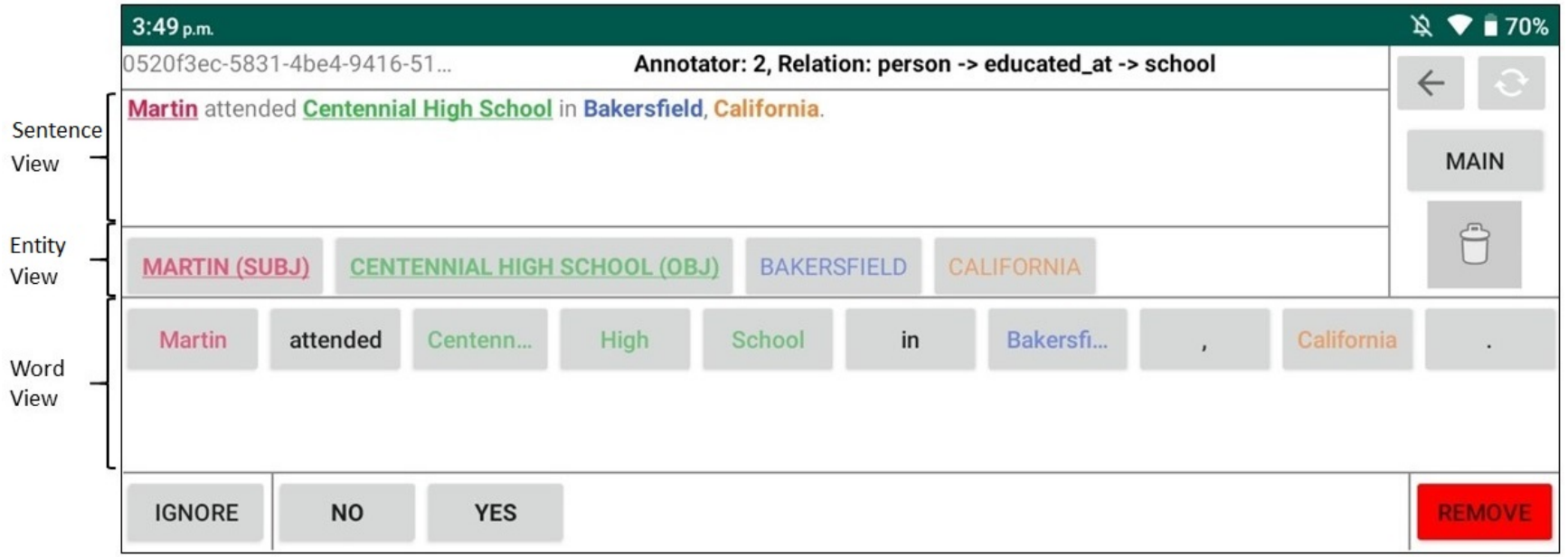


Figure 4.5: Interface for RE data annotation. The relation considered here is *educated at* with *Martin* as subject and *Centennial High School* as object. The locations *Bakersfield* and *California* are not participating in the relation, but can be used for creating negative examples.

becomes straightforward. For instance, new entities can be easily created through dragging and dropping one of the word buttons in the *Word View* up into the *Entity View*. Similarly, entities can be removed or fixed (e.g. adding a missing word) by just dragging and dropping. Web-based tools, e.g. BRAT[81], require the user to select spans of text using a computer mouse to create entities, requiring very precise (i.e. slow) moves.

In most sentences, editing operations using FREDA require a very minimal amount of time. Once done with editing, a user indicates whether the relation holds or not, i.e. a simple binary decision is made at the end. An annotator can also remove the sentence from the database (e.g. sentence is broken in some ways) or ignore it to use it in the context of other relations (e.g., the sentence looks good, but not relevant for the current relation, but could be for other relations).

Note that since a sentence may be relevant for multiple relations and potentially also for other tasks such as co-reference resolution, NER, or similar tasks, we do not instruct annotators to remove already highlighted entities that are irrelevant to the relation at hand, but still potentially useful for other relations. Removing them would indeed take additional time and is not necessary. Actually, our models, which are trained on these datasets, are able to accurately distinguish between subjects, objects, and the rest of the entities in a sentence, because all entities are considered for training.

For such a complex task it is expected that a single annotator is not able to be accurate and consistent over hundreds of sentences. Alt et al. [3] show a detailed analysis of all the mistakes that are possible, especially for crowdsourced annotation tasks, where presumably time matters more than accuracy. Therefore, similar to Mesquita et al. [60], our system relies on at least two annotators per sentence with a third annotator to break ties if the previous two disagree in their decision. The second annotator gets to see the entity annotations from the first annotator (and the third from the second). Note that only the entity annotations are carried over in this way. Thus, the final decision as well as indicating the subject and object entities is still every annotator's independent decision. But carrying over entity annotations saves time

for subsequent annotators (also deleted sentences are not shown to annotators thereafter). Therefore, the time spent for subsequent annotators is likely to be lower than for the first. More on the datasets we created and models trained on them for the task of RE in chapter 7.

4.3.5 More Annotation Tasks

The interface for data annotation is very flexible and we adapted it to a few more common NLP tasks, which is described thereafter.

Named Entity Recognition

NER data annotation can be mainly seen as a subtask of data annotation for RE, although entity types have to be included. Unlike relations for RE, Named Entities are very common for text data from all sorts of sources, e.g Wikipedia, newswire or social media text. Therefore, there is no obvious need to filter sentences for NER data annotation. Although under certain circumstances, e.g. an (expected) imbalance of types, sentences can be filtered beforehand.

Figure 4.6 shows the annotation interface for the task of NER. The interface is similar to the RE annotation interface with some adjustments: (1) Once the sentence is perfectly annotated, the user has to press the *Done*-Button (apart from *Remove* and *ignore*), decision making as done for RE is not necessary here. (2) The main change is the *Type View* right below the *Entity View*, which shows all types the current dataset can be annotated for (each dataset can include its own type-set). In order to add a type to an entity, the corresponding entity button can be dragged and dropped on the appropriate type button.

Co-references for each entity can be included as well, similar to the RE task, even though this may not always be desirable as not all mentions of an entity can be considered as Named Entities.

Entity Linking

Data annotation for EL largely follows the scheme of the previous two tasks with finding entities and co-references, although, in addition, the corresponding KB-entity can be selected from a set of candidate entities, if applicable.

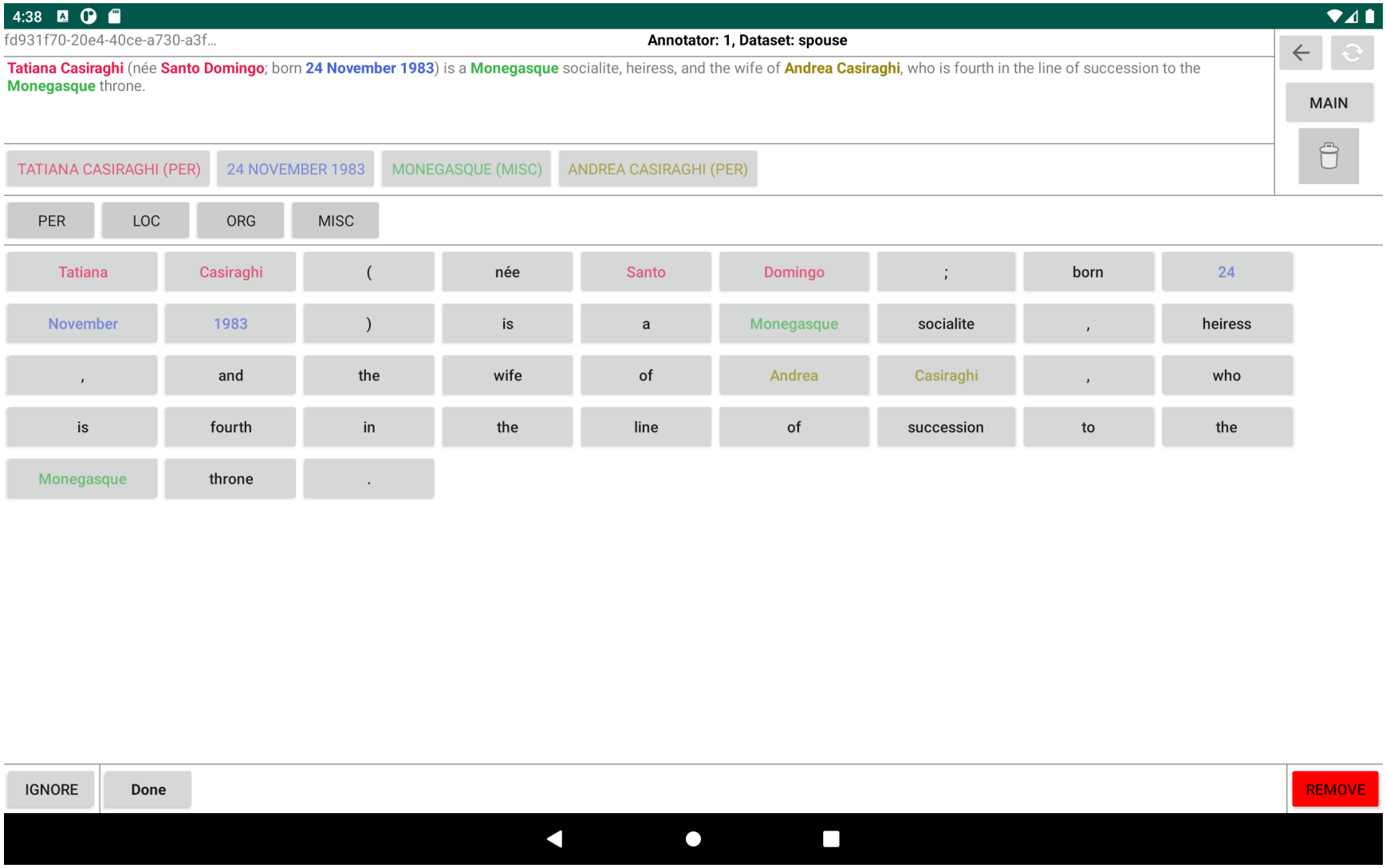


Figure 4.6: Data annotation for the task of NER.

As previously mentioned for the NER task, sentence filtering is in general not necessary for this tasks since entities appear frequently for all kinds of text types and topics. However, there may be an over-representation of very popular entities, such as countries and cities, which could lead to the necessity to filter based on the frequency of all entities (WEXEA contains the actual Wikipedia entity annotations).

The annotation scheme for EL is mainly the same as for RE, except that, again, the decision buttons are replaced by a single *Done*-Button and a dialog is opened whenever an entity button is pressed, for which at least one candidate can be found in Wikipedia, as shown in Figure 4.7. If none of the suggested entities is correct, *No Entity* can be selected. Each entity in the candidate dialog has the first 100 characters of its abstract in Wikipedia attached, more text is usually not necessary and would be overwhelming, and therefore time-consuming, for the user.

Co-reference Resolution

CR annotation is included in all previously mentioned tasks. However, in order to simplify the task for users who are solely interested in CR, it is added to the framework.

Similar to previous tasks, sentence filtering may not be needed. Since the *Sentence View* contains a relatively large amount of space, it is possible to display more than one sentence to capture inter-sentence co-references, which can be included in this step.

Using the CR interface for CR annotation can speed up the task since no type or candidate or subject/object selection has to be done. Using a simple *Done*-button finishes the current sentence(s) and moves to the next example.

4.4 Evaluation

Data annotation for the aforementioned tasks can be prohibitively time-consuming. Therefore, approaches for quick dataset construction are essential in order to be able to easily extend existing datasets with more entity types, relations or

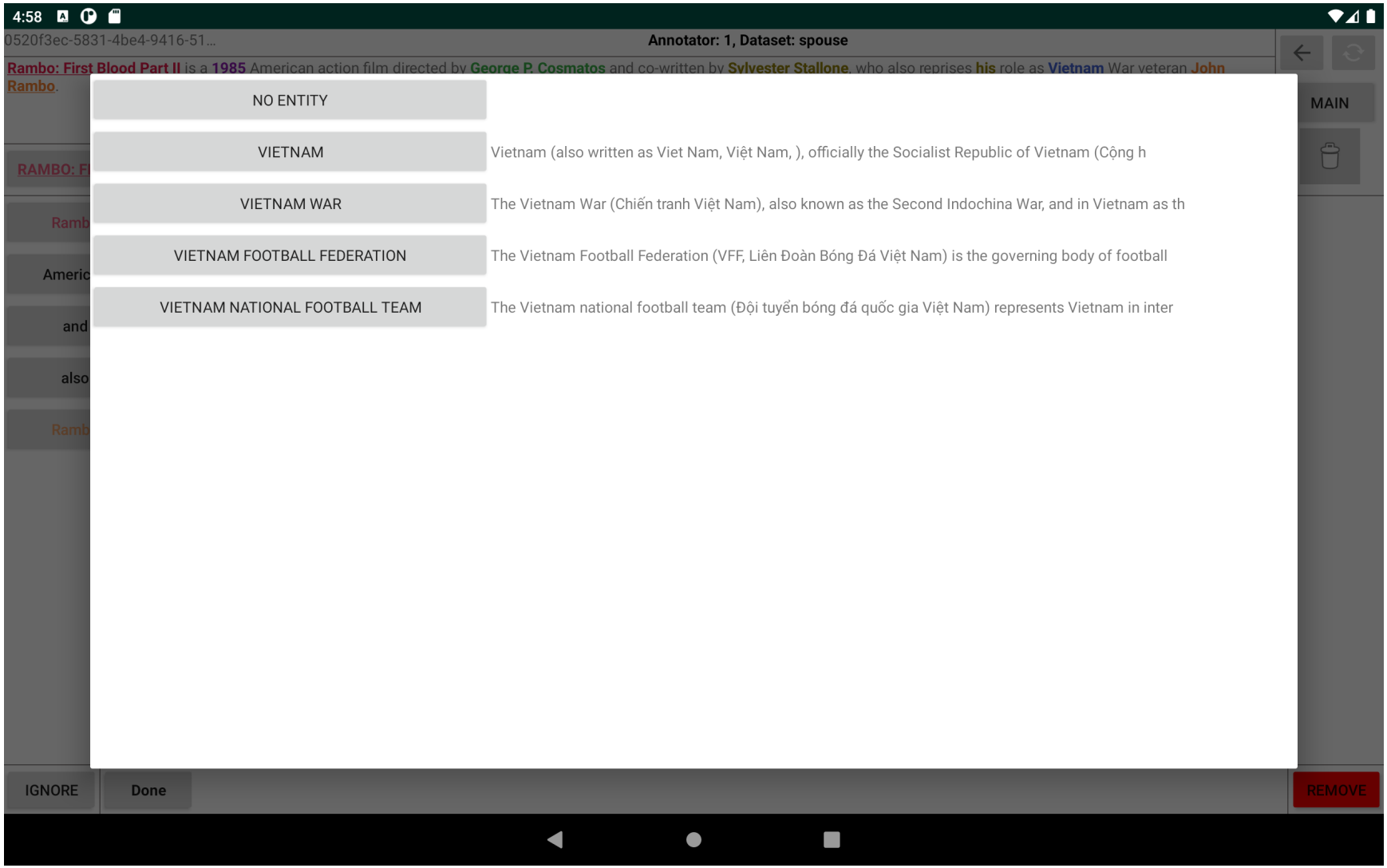


Figure 4.7: Candidate selection for EL data annotation. The entity *Vietnam* was selected in the text view and all related entities in Wikipedia are are shown, ranked in descending order by the number of hyperlinks of the anchor text *Vietnam* to those articles.

co-references or create entirely new datasets.

4.4.1 Baseline

We compared **FREDA** to the open-source tool **BRAT** [81]. BRAT is a web-based tool for data annotation for a variety of NLP tasks[6]:

- NER: Entities of different types can be annotated through selecting spans of text (types are given to the tool in advance, we use the commonly used types *Person*, *Location*, *Organization* and *Miscellaneous*). After selecting an entity, a window appears in which the type can be selected. FREDA represents each entity as its own entity button, which can be typed. There is no need to find co-references for this task and since WEXEA is able to provide annotations, such as dates, which are not compatible with this type set, we remove all annotations for both approaches, i.e. the annotator starts from scratch.

- CR: After annotating entities and their co-references, an *alias*-relation is used to connect clusters of co-referent entities, i.e. can be added through dragging an arrow from one entity to the other. FREDA represents each entity and their co-references as its own entity button.

- RE: Starting with the task CR. Afterwards, entities can be connected with specific relations given to the tool (we used the *spouse*-relation for comparison; if an entity has multiple co-references, a single one of them is sufficient to connect with a relation). FREDA asks the user whether the relation of interest (*spouse* in this case) is expressed between two selected entities (subject and object) in the sentence at hand.

.

4.4.2 Comparison

A single annotator labelled 100 sentences from the *spouse*-relation-dataset (see Section 4.3.3 for an explanation how this dataset was created). A new dataset

[6]Only NER, CR and RE are considered here since BRAT is not designed to be used for EL data annotation.

of unseen sentences was used each time. In order to keep the workload as consistent as possible, each sentence consists of 25 words.

Table 4.1 shows the result of comparing BRAT and FREDA for the tasks NER, CR and RE. FREDA out-performs BRAT consistently, even though the difference for NER can be considered as minor. For this task pre-annotated entities from WEXEA were removed as they often do not correspond to Named Entities. The more words an entity consists of, the easier it is to add such an annotation with BRAT, since long text can be simply selected with a mouse, whereas the user of FREDA has to drag and drop every single word of an entity to its entity-button. However, the rest of the annotation with FREDA, such as selecting entity types, seems to counterbalance this disadvantage. For CR and RE in this evaluation, the user of BRAT needs 62% and 44%, respectively, more time to annotate these datasets, compared to FREDA.

	BRAT	**FREDA**
NER	22.6s	**21.9s**
CR	14.9s	**9.2s**
RE	14.8s	**10.3s**

Table 4.1: Number of seconds (average per sentence) to annotated 100 sentences per task for both approaches.

Criticism may be raised due to the fact that the annotator may have been biased towards one or the other approach and adjusting annotation speed correspondingly. Therefore, we made videos of the annotations for this evaluation publicly available[7].

4.5 Conclusion

We presented our tool for Flexible Relation Extraction Data Annotation (FREDA) for a variety of NLP tasks. The interface is simple to use and can also be easily extended or adjusted, depending on the data annotation task. The evaluation shows its competitiveness when compared to the open-source tool BRAT. We

[7]Links can be found here: `https://github.com/mjstrobl/FREDA`

hope that releasing FREDA to the public[8] will encourage the community to quickly create more annotated data for NLP tasks.

For Future Work, it may be possible to include more tasks, such as Word Sense Disambiguation or the creation of evaluation data for Dialogue Systems.

[8] https://github.com/mjstrobl/FREDA

Chapter 5

A Knowledge Graph Population System for Conversations

In this chapter we provide a more detailed description of our system for Knowledge Graph Population (KGP) from conversations.

5.1 Introduction

Building a KG from text is often done using a pipeline-based approach, such as the one proposed by [96]. In our case, the pipeline consists of three parts: Entity Recognition (ER), Entity Linking (EL) and Relation Extraction (RE):

1. ER: Typically a Named Entity Recognition system (NER) is used (e.g. see [72] or [49]), which only detects Named Entities (NEs). However, sometimes noun phrases in general may correspond to entities of interest, e.g. *my sister*, which refers to an NE without a specific mention of the name. Therefore a similar system has to be used that is able to detect all kinds of entity mentions or at least it has to be flexible enough to be extended.

2. EL: EL systems are popular in Natural Language Processing (NLP) recently, e.g. see [78] or [48]. Typically these systems link an already detected entity (through the ER system) to an entity in a KB (e.g. Wikipedia). However, although entities mentioned in news text, i.e. the kind of data these systems are usually trained on, can mostly be found in KBs, this is typically not the case in a conversation. In case one

wants to talk about *uncle George*, current EL systems would link him to an existing entity, although he does not exist in the KB. Therefore, the challenge is to decide whether an entity already exists in the KG or refers to a new entity, which can be added.

3. RE: We argue that the kind of relations we typically find in conversations are very different from what can be found in RE datasets, e.g. TACRED [97]. Therefore, we need to define a new set of relations that have to be detected.

In order to illustrate the task of KGP, consider the following sentence, which could be part of a conversation:

> "John's sister is married to Christopher and she lives in Edmonton."

The output of each pipeline part that can be extracted from this sentence is shown in Figure 5.1. The relationships *spouse* and *sibling* require two entities as subject and object, whereas the *name*-relation points to a Literal (a string in this case). Although *John's sister* was not explicitly mentioned by a name, we can still create a node for her and link her to *Christopher* with the *spouse*-relation.

Previous RE systems for dialogues do no explicitly deal with cases like this where entities are referenced through a relationship, e.g. *John's sister*, without mentioning a name. This results in a low recall due to missing either the relation between *John* and his sister or the relation between his sister and *Christopher*, e.g. see [93]. Therefore, these nouns phrases should be detected and added to the KG. In [93], John's sister would be extracted as an entity, if she was referenced somewhere else in the conversation, which the authors consider as cross-sentence relation, even though relationships are still expressed in a single sentence or utterance.

While ER, EL and RE are popular research topics in the field of NLP, the combination of all subtasks aiming to create a KG is often neglected and in

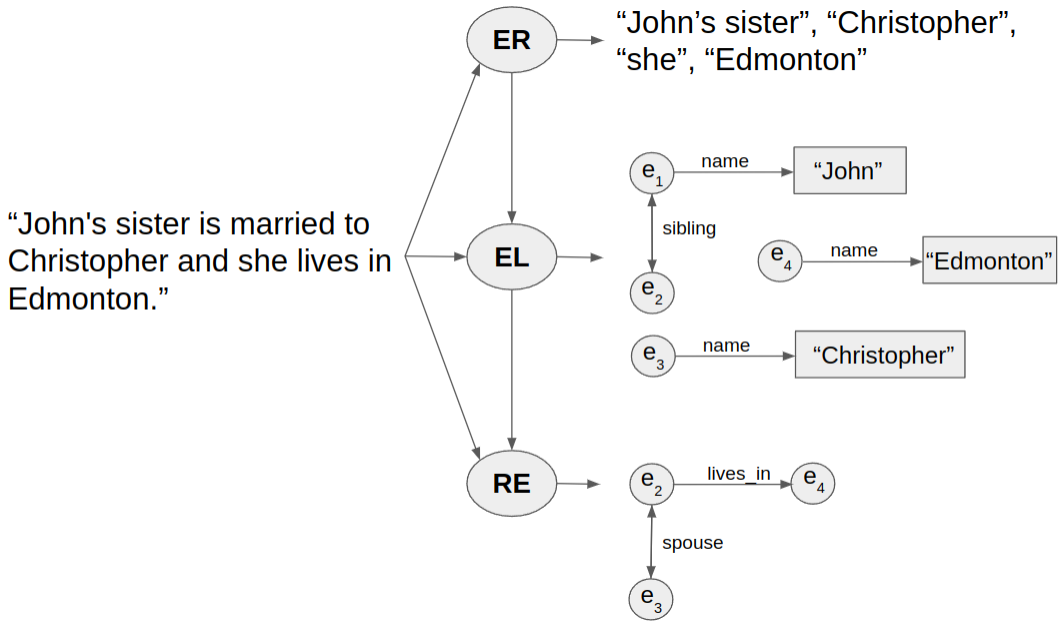


Figure 5.1: Output of each pipeline part from the example sentence. e_1, e_2 and e_3 correspond to nodes in the graph, arrows denote relations between two nodes or a node (subject) and a Literal (object). The EL system further adds (not shown here) all entities to the KG through merging or keeping nodes. Although not shown here as well, it is also possible to add the gender, as it is encoded in the schema.org ontology[1].

order to test or demonstrate a system, a framework with graphical user interface is necessary, which is typically not available. Therefore, in this chapter, we describe a system with a graphical user-interface showing a conversation with a chatbot and the resulting KG. Basic rule-based components for ER, RE and EL are included, which can be further extended (see Chapter 6 and 7).

The remainder of this chapter is structured as follows. Section 5.2 provides information about Related Work. The proposed system is described in Section 5.3 with a discussion of the approach used in Section 5.4. The chapter is concluded in Section 5.5.

[1] `https://schema.org/gender`

5.2 Related Work

There are a variety of works related to ours, although they are typically missing at least one key component.

The CoreNLP toolkit [58] is a popular tool for multiple common NLP tasks and loosely implements the pipeline described in the Stanford 2016 TAC KBP submission [96]. While the user-interface of CoreNLP[2] consists of visualizations of all its subtasks, including RE, no KG is created, only entities and their relations in the current sentence are analyzed and shown, which is not suitable for KGP since it is not keeping track of extractions and does not link entities to their corresponding node in the KG. Furthermore, extending CoreNLP with more relations to detect (RE) or new entity types (NER) is not straightforward.

Sanh et al. [74] proposed a multi-task learning approach, aiming to detect entities and relations jointly. A user interface is provided[3], which shows the output of the system for each provided sentence independently, similarly to CoreNLP. Again, no KG is maintained.

There is also work on dialogues available, e.g. [94], which focuses on the extraction of relations from dialogues and provides a human-annotated dataset (*DialogRE*) for Information Extraction from dialogues. However, even though baseline models are provided, a user interface is missing and it was only tested on RE, ignoring ER and EL.

5.3 KGP from Conversations

We are considering a pipelined approach with the following individual sub-modules:

1. Entity Recognition (ER): The ER module recognizes entity mentions.

2. Entity Linking (EL): Entities found by the ER sub-module can potentially be linked to existing nodes in the KG or a new node is created in

[2] `https://corenlp.run/`
[3] `https://huggingface.co/hmtl/`

case no match was found.

3. Relation Extraction (RE): A sentence can contain several entities and the RE module can detect relations expressed between entities, if not already recognized by the ER module[4]. Existing datasets (and therefore resulting models trained on them), such as TACRED, may not always contain appropriate relations for conversations. Therefore our system includes a rule-based RE system, which can be easily extended with models trained on RE dataset, and tested.

5.3.1 Entity Recognition

Commonly used ER approaches do not consider noun phrases, such as *my brother* as entities. However, especially in conversations these noun phrases containing expressions of relations are very common since mentioning only names of entities is too ambiguous. We propose an approach including rule-based noun phrase detection as well as NER with the CoreNLP toolkit. In addition, since the CoreNLP NER may not be flexible enough regarding the type of models used, a custom NER in order to recognize other types of entities is added[5], while still being able to use the rule-based entity detection of CoreNLP with Tokensregex [16]. Tokensregex is a framework for recognizing and typing entities based on regular expressions, e.g. entities such as *my sister* can be detected this way, if the relation *sister* or *sibling* is part of the pre-defined set of relations to be recognized.

Entities can consist of three parts:

- Named Entities (NE): Names referring to unique entities by name, e.g. *John*, *Sarah* or *Christopher*, are detected by the NER.

- Personal Pronouns (PRP): Same function in text as NEs, recognized by CoreNLP's Part of Speech tagger.

[4]Some entities itself can contain information about relationships as described in the remainder of this section.

[5]This custom NER can be easily used to completely replace the model-based part of the CoreNLP NER, while still using the regular-expression-based NER.

- Entity Mentions (EM): Noun phrases; typically roles of a specific entity (with or without a name mentioned in text), from the perspective of another entity, e.g. *John's sister* may not be the same as *Sarah's sister*. Therefore, we consider EMs only if they were mentioned as a possessive property of an NE or PRP. These can be recognized by Tokensregex using the pre-defined set of relations.

Although EMs can already contain relations, we do not consider these for the RE module since they are considered to be either subject or object of a sentence, whereas the RE module aims to find relations expressed in the sentence between the subject and object, often with multiple words in between.

A custom NER can be used as well. We implemented a regex-based entity recognizer for the type *Food*, considering a dictionary compiled from Wikipedia's food-related categories. It can be used within CoreNLP.

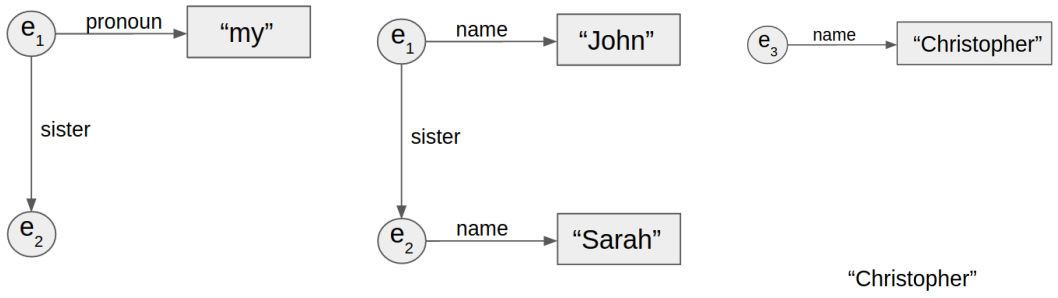
5.3.2 Entity Linking

Once entities are detected, Entity Graphs (EG) can be created by the EL system, which in turn can be translated to RDF[6] expressions (can be directly added to the KG). Since these entities usually follow a very specific scheme, we propose a rule-based approach to break them down into single entities and their links. Figure 5.2 shows examples of EGs from entities commonly found in conversations.

The second goal of EL for KGP is to link each entity in the EGs to its corresponding node in the KG or a new node is created, in case there is no suitable match. Especially if a lot of personal information is shared in a conversation, conventional EL systems, such as [78] or [48], are not suitable since they cannot link out-of-KB-entities, i.e. entities that cannot be found in the reference Knowledge Base (typically the set of articles in Wikipedia). We propose a partially rule-based system that finds candidates for entities the following way:

- NEs: Perfect match with entity names in the KG, or new node.

[6] `https://www.w3.org/RDF/`



(a) *my sister*: Two nodes, the relation *sister* in between and the pronoun attached to the first node.

(b) *John's sister Sarah*: Two nodes, both with names attached and the *sister* relation in between.

(c) *Christopher*: A single name without any other relations or pronouns is translated into a node and the name attached as RDF Literal.

Figure 5.2: Examples of detected entities and corresponding EGs.

- PRP: CoreNLP's CR system is used for linking these pronouns to their corresponding entity in text.
- EMs: Considering NEs and PRPs are already linked, relations can be linked through perfect matches as well, e.g. if *John's sister* is already stored in the KG, she can be retrieved and linked if mentioned again, even though a name is not present.

In case a model-based EL system, which is capable of EL for all entities (in- as well as out-of-KB), it can be added in order to replace the rule-based system.

5.3.3 Relation Extraction

This module basically extends the ER module through adding binary relationships as edges expressed in text between a subject and an object, which are already detected and linked. We added a rules-based matcher detecting the most common expressions of each relation. Neural-network-based RE approaches are often used today and can be added as well. This follows a similar scheme as already done by Stanford's TAC-KBP submission [96].

Table 5.1 shows the currently detectable relations and their type signatures. For each relation, a number of synonyms exist in order to be able to

detect them through the ER system. In addition, each relation has a set of Tokensregex as well as Semgrex [14][7] rules attached, detected by the RE system.

Relation	**Types**
sibling	Person ↔ Person
child_of	Person →Person
relative	Person →Person
spouse	Person ↔ Person
lives_in	Person →Location
born_in	Person →Location
date_of_birth	Person →Date
family_doctor	Person →Person
likes	Person →Food

Table 5.1: Detectable relations and corresponding type signatures.

5.3.4 User Interface

Figure 5.3 shows the user interface for KGP from conversations. On the left, chat messages are shown, on the right the KG as graph[8] and as printed subject-property-object triples is shown. In order to restart from a previous checkpoint, the graph can be stored to a file and reloaded later. Chat messages from the user are show in *green* and responses from sub-modules are shown in *red*.

5.4 Discussion

The proposed KGP approach is able to extract entities and relations from text and add corresponding facts to a KG. However, such facts are never corrected (belief revision) and there is no interaction with the user, which, for example, could resolve ambiguity. This is discussed hereafter, including the reason for a the lack of a scientific comparison with existing approaches as baselines.

[7]Similar to Tokensregex, except that rules can be written for dependency parses.
[8]Visualizations are re-created dynamically using D3: `https://d3js.org/`

5.4.1 Belief Revision

The task of Belief Revision, e.g. see [31], aims to revise a KB in case it would otherwise contain inconsistencies. In our case, such inconsistencies could arise if facts are added to the KG, which contradict existing facts. For example, the *place_of_residence* relation typically exists only once per person, but it can be updated over time. Other relations, such as *parent*, do not change over time, but facts involving this relation could be erroneously added if the corresponding utterance is incorrect.

This could be partially resolved through considering the latest added fact as correct and earlier facts as invalid. Depending on the relation, this does not mean older facts are incorrect, they may still have been correct in the past. A reasoning system or conversational agent using the resulting KG needs to be aware of this issue.

5.4.2 Interactivity

The proposed systems is adding facts to a KG. These facts could be ambiguous or miss interesting information, such as names, or even be incorrect, if the system made mistakes. An interactive system, which is able to ask the user to provide certain pieces of information or ask for missing facts or simply clarify previous extractions, could use the KG to improve completeness and quality of the extracted entities and relations.

In [20] as well as the STEP project (Schema Tuple Expression Processor) [62], the goal was to provide an NLI (Natural Language Interface) that allows the user to use natural language to query a database. This was done in an interactive fashion in order to be able to ask the user for clarifications through paraphrasing the questions or asking for specific fields that can be found in the database.

Even though it is beyond the scope of this work, a Response Generation system could consider these ideas and potentially do Belief Revision through asking the casual user for clarifications in order to avoid inconsistencies in the KG.

5.4.3 Missing Baseline Comparison

The main reason for developing this system is the lack of tools to test KGP approaches as a whole. CoreNLP, for example, is capable of all subtasks of KGP, but is not able to continuously add facts the the KG, specifically as it is not able to connect entity mentions over time, add models for RE, or answer questions based on a KG. Therefore, a scientific comparison is not possible. However, in order to attempt to clarify its usefulness for KGP, examples are added to the following chapters, which should also help visualizing and, therefore, understanding those approaches.

5.4.4 Example

The resulting KG from the piece of a conversation in Figure 1.2a using the described system can be seen in Figure 5.4. The entity of type *Food* ("Pancakes")[9] as well as the relations *child_of* cannot be detected yet as rules matching these specific cases are missing.

[9] Although, the relation *likes* can actually be detected. It is considered as missing here as the object of this relation is not detected yet.

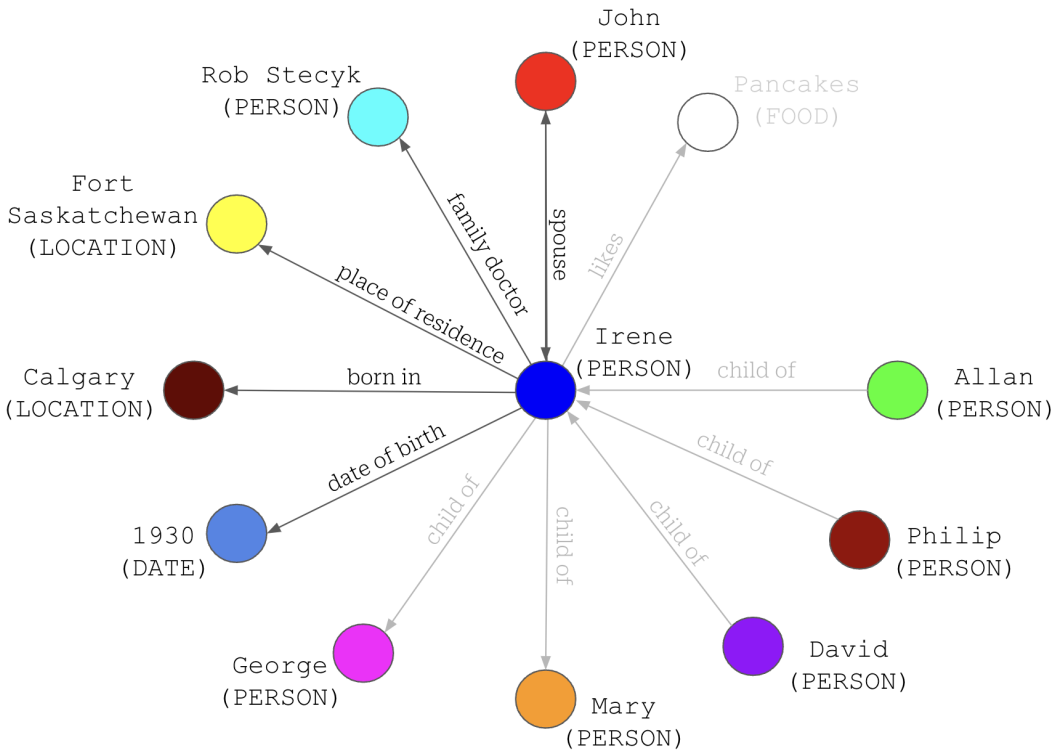


Figure 5.4: KG extracted from the conversation in Figure 1.2a by the proposed system. Light grey relations and entities cannot be detected yet.

5.5 Conclusion

We proposed a system for Knowledge Graph Population from Conversations with baseline approaches for each sub-module for Entity Recognition, Entity Linking and Relation Extraction. It can be easily extended with neural-network-based models and is suitable for testing and demonstrating Information Extraction capabilities of newly developed or existing approaches for conversations. The codebase is publicly available[10].

For Future Work, we are planning to add a variety of state-of-the-art models for each submodule as well as a Response Generation module taking the Knowledge Graph into account, automatically replying to chat messages considering the current Knowledge Graph.

[10] https://github.com/mjstrobl/KGP_from_Conversations

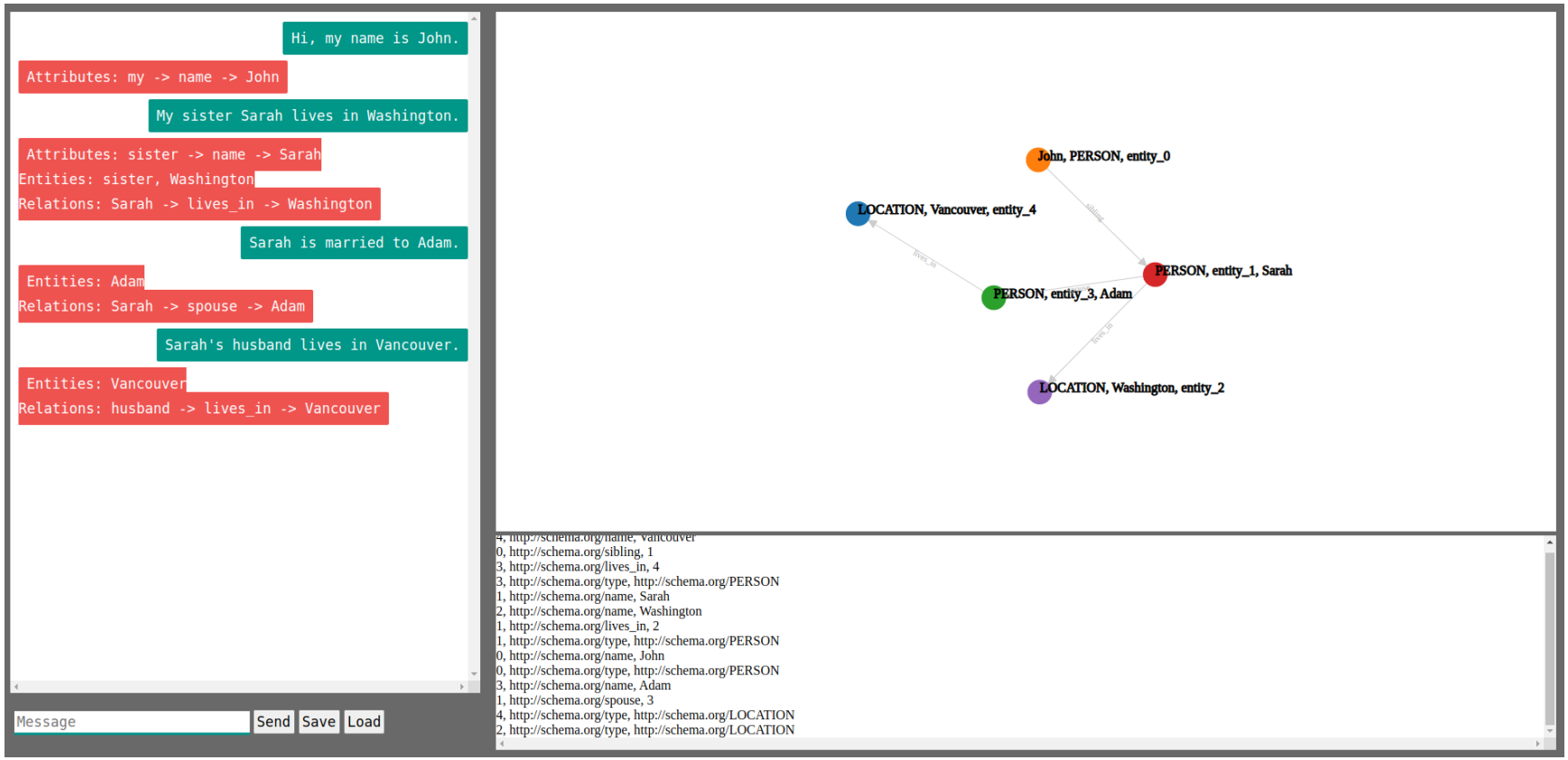


Figure 5.3: Knowledge Graph Population user interface. Messages can be sent and graphs can be stored and loaded.

Chapter 6

Entity Recognition

We focus on KGP from conversations for elderly people, including the topics Nutrition and Health. However, common NER models are not able to recognize related entity types, such as *Food* and *Drugs*, due to missing training data. Even if datasets with annotations for new types would be available, it is not straightforward to include them in the training procedure. All datasets used for training need to be exhaustively annotated with all entity types, the model is supposed to recognize. This chapter describes an approach on how to create partially annotated datasets for new entity types, semi-automatically, and investigates training procedures for such datasets.

6.1 Introduction

Named Entity Recognition (NER) is one of the most popular tasks in NLP. The goal for NER is to classify each token (usually corresponding to a word) in a sequence of tokens according to a scheme and a set of classes. The BIO scheme (Beginning, Inside, Outside) is typically used and each entity label starts with *B-<class>* (first word) and continues with *I-<class>* (all subsequent words). This way multiple entities in a row can be annotated/detected, even if there is no separator. The set of classes a model can recognize is dependent on the dataset it is trained on, e.g. *Person*, *Organization*, *Location* and *Miscellaneous* for the popular CoNLL 2003 NER dataset [72]. Typically a sequence tagging model (e.g. [49] or [21]) is used to achieve this task, which takes a sentence as a sequence of tokens as input and outputs a sequence of classes all at once.

In order to train such a model, high-quality manually annotated data is necessary with each entity in the dataset assigned its correct class. Datasets with partial annotations, mainly without all entities being annotated, are noisy and lead to worse model performance. This is due to the fact that entities in the dataset, which are not annotated as such, are automatically considered as belonging to the *Outside* class. Therefore, the model is trained not to recognize these, even though it should, leading to a model which may be tempted to ignore a certain number of entities when used for making predictions.

However, partially annotated data is often easier to come by, e.g. through using hyperlinks from Wikipedia[1] as entities. This is especially useful if classes other than the ones seen in common NER datasets are of interest. Furthermore, intuitively, why should it be necessary to annotate every single entity in a dataset to make a model learn this task? Humans are perfectly able to recognize mentions of an entity consistently after having it "classified" once. Consider the following sentence with partial links for animals from Wikipedia[2]:

> "Stauffer's animal crackers include bear, bison, camel, **cow**, **cat**, **donkey**, elephant, hippopotamus, horse, lion, **mountain goat**, rhinoceros, and tiger."

This sentence originally contains four links to animals, while the other nine animals are not linked. Wikipedia often contains partially annotated sentences due to a Wikipedia policy discouraging editors either from linking the same article more than once or linking popular articles all together since they would not provide any new information to the reader. However, if the goal is to train a model, which is capable of recognizing the class *Animal*, being able to use such data without manual annotations would significantly simplify the task.

Therefore, the work presented in this chapter aims to describe how it is possible to train commonly used models for NER on partially annotated datasets for new classes, without a significant manual effort for data annotation.

These are our main contributions:

[1] `https://www.wikipedia.org/`
[2] `https://en.wikipedia.org/wiki/Animal_cracker`

- Describing a procedure on how we can create partially annotated datasets for new classes derived from Wikipedia categories semi-automatically.
- Providing and comparing training strategies for NER models on partially annotated datasets.
- Releasing two manually annotated dataset of 500 sentences each for the classes *Food* and *Drugs* in order to test how generalizable our data extraction techniques are.

The remainder of this chapter is outlined as follows: Section 6.2 shows some related work on how the problem of training models on partially annotated datasets has been approached before. We propose our method for data extraction from Wikipedia and model training strategies in Section 6.3. The experimental evaluation can be found in Section 6.4 with a conclusion in Section 6.5.

6.2 Related Work

Jie et al. [44] proposed an approach to train a BiLSTM-CRF model on partially annotated datasets. Their iterative approach tried to find the most likely labelling sequence that is compatible with the existing partial annotation sequence, i.e. the model is supposed to learn to assign the highest weight to the most likely (ideally correct) labelling sequence.

The CoNLL 2003 dataset for English was used for the evaluation (in addition to the CoNLL 2002 dataset for Spanish NER [86]). 50% of the labelled entities were removed for testing their model, effectively lowering the recall when trained on this dataset. The best model achieved a 1.4% F1-score reduction on CoNLL 2003 (compared to the same model architecture trained on the complete dataset without any entities removed). Only fully annotated (yet artificially perturbed) datasets were considered for the evaluation. While it would be technically possible to use partially annotated datasets with non-entity annotations[3] through ruling out some potential labelling sequences, this

[3]These are annotations indicating that this is knowingly not an entity, which is possible

was not tested.

A different approach was proposed by Mayhew et al. [59] for training BiLSTM-CRF models on existing datasets for a variety of languages, e.g. the popular CoNLL 2003 dataset for English NER [72]. They used an iterative approach in order to learn a weight of 1.0 or 0.0 for each token, depending on whether it should be included for training, i.e. considered in the loss function, or not. Whenever a span of tokens representing an entity is considered as non-entity, the weight should be close to 0.0, and in case of a proper entity annotation, the weight should be 1.0. The dataset was artificially perturbed to reduce precision as well as recall, i.e. to lower recall some entity annotations were removed and to lower precision some random spans of tokens were annotated as entities. Therefore, instead of trying to label the training sequence correctly, they tried to figure out which tokens are of class *Outside* with high confidence (weight = 1.0) and which ones are probably entities (weight = 0.0) that the model should not use for training.

Their best model still suffered from an F1-score reduction of 5.7% and in the same experiments the models from [44] had an 8% reduction. Note that the dataset also contained random spans of tokens added as entities, which was not tested by [44]. Although it is not known which mistakes can be attributed to lowering recall or precision in the training dataset. The obvious drawback of their approach is that false negatives (entities in the dataset without annotations) are not supposed to be considered for training, effectively reducing the set of entities the model can be trained on. Only the weights for each token are adjusted, but not the labels. Furthermore, assuming a partially annotated dataset contains entities of some class as well as non-entity annotations of this class, i.e. entities that are known not to belong to this class (but it may not be known which class they actually belong to), their model cannot take advantage of this kind of information, it simply tries not to use these annotations for training.

In addition, both aforementioned model architectures seem to be outdated

for partially annotated datasets derived from Wikipedia, which is described in more detail in the next section.

for today's standards as models based on the Transformer architecture [88], specifically the pre-trained BERT model [21], achieve a significantly higher F1-score than BiLSTM-CRF models when trained on unperturbed datasets, e.g. see the results on CoNLL 2003 from [21] compared to the popular LSTM-based approach from [49], which was specifically developed for NER. Also, both models are not tested on new datasets with new entity classes.

6.3 Method

This section proposes our approach to create partially annotated NER datasets from Wikipedia for new classes and strategies to train models on these datasets without sacrificing prediction performance on entities from existing classes.

6.3.1 Data creation

When creating datasets for new classes for NER there are two problems to solve:

1. Where can we get text data from? Entities of the class of interest maybe less abundant than common classes, such as *Person*, *Location* or *Organization*. If simply random sentences, e.g. from the web, are included, the fraction of useful token spans maybe quite low[4].

2. How can we annotate relevant token spans? Manual data annotation is a time-consuming task, even if done partially, which should be avoided. In addition, if a token span looks like a relevant entity, does it make sense to annotate it as well as non-entity for the class of interest? This mainly depends on whether a model can take advantage of that, i.e. whether the model can be told not to predict this class. The approach proposed by [59] would not be able to.

Wikipedia as a whole can be considered as a partially annotated dataset. Hyperlinks in articles correspond to entities and the hierarchical category sys-

[4]Mainly those token spans are useful that could potentially be part of the current class of interest, but are not always, e.g. "Tomato" could refer to the class Food or it could be a musician: `https://en.wikipedia.org/wiki/Tomato_(musician)`

tem can be considered as a class hierarchy, which can be used to classify entities. Each article can have several categories attached by editors, although not all of them refer to a class in an NER-sense. For example, the article "Salt"[5] has the following categories attached: "Edible Salt", "Food additives", "Sodium minerals" and "Objects believed to protect from evil". In case the class of interest is "Food", we can consider articles in the first two categories as relevant, whereas the latter two categories do not necessarily refer to food-related articles. Therefore, we only need to know which categories are relevant for this class in order to extract a set of articles and sentences they are linked to create a partially annotated dataset.

Algorithm 1 outlines a simple procedure to extract categories and articles from the Wikipedia category hierarchy using Breadth-first-search. We start at a base-category, e.g. "Category:Food and Drink"[6] for the class *Food*. At each iteration of the loop in line 3 a category and 10 articles in this category (if available) are presented to the user and they have to decide whether to keep the category (and potentially the articles as well[7]). If it is kept, all sub-categories (if available) are added to the queue. Ultimately, all categories the user wants to keep including all articles in these categories (if not explicitly excluded) are considered for the class of interest and text from Wikipedia can be extracted.

These categories were added by the editors of Wikipedia and are often redundant, therefore it may be necessary to restart the procedure to avoid adding too many categories. However, in our experience it seems to be possible to finish it within 1 to 2 hours, at least for our test classes.

Since the training corpus for the new class C should be somewhat difficult for a model to be trained on, it is also necessary to consider articles, which share aliases with articles in C (many entity mentions can refer to entities of different types). This can be done with an alias dictionary derived from Wikipedia hyperlinks. We used the parser from WEXEA including its alias

[5] `https://en.wikipedia.org/wiki/Salt`

[6] `https://en.wikipedia.org/wiki/Category:Food_and_drink`

[7] Sometimes intermediate categories, the user wants to keep, contain uninteresting articles in which case these articles can be ignored.

Algorithm 1 Extract articles and sub-categories

Input Wikipedia type hierarchy.
Output Partial hierarchy corresponding to entity class of interest.

```
1: procedure BFS(cat_start)
2:     queue = [cat_start]
3:     while queue not empty do
4:         cat_current = queue.pop(0)
5:         print cat_current and 10 articles
6:         input_user = input()
7:         if input_user =' y' or input_user n =' s' then
8:             for cat_sub in cat_current do
9:                 queue.append(cat_sub)
10:                if input_user =' y' then
11:                    Keep all articles in cat_current
```

dictionary in order to extract sentences from Wikipedia, which contain hyperlinks of articles in C (annotated as entities of class C) or hyperlinks of other articles that share aliases with those in C (annotated as non-entities). This will result in a partially annotated corpus of entities and non-entities of class C, i.e. a set of sentences can be extracted from Wikipedia that contain relevant entities identified through their hyperlinks.

In addition, the alias dictionary is used to annotate potential entities with an unknown type since, as we pointed out, not all entities are annotated in Wikipedia. Depending on the model, these entities would be excluded from training since the type is unknown. This is applied to all datasets used for training, e.g. CoNLL 2003 may contain entities of type *Food*, which can be found and potentially excluded this way. And since an NER model should be trained on this kind of dataset as well as other datasets, such as CoNLL 2003, the CoreNLP NER [58] is used to find all other entities of type *Person*, *Location*, *Organization* and *Miscellaneous*. These additional entities are also considered as non-entities, although since they are not gold-standard entities, we do not use their types otherwise for training, i.e. if a *Location* was found this way, the model is only trained not to recognize it as one of the new classes (if applicable), but it is not trained to recognize it as *Location*.

6.3.2 Model training strategies

When partially annotated data is introduced, the main goals for model training are:

1. Classification accuracy for entities of existing classes should not suffer from the introduction of new data: We still rely on the CoNLL 2003 dataset for NER with classes *Person*, *Location*, *Organization* and *Miscellaneous*. Entities of these classes will very likely appear in the new datasets, but it is unknown where they are mentioned. A sequence tagger used for NER considers all tokens outside of the spans of entities as token of class *Outside*. Therefore, these unlabeled mentions could mislead the classifier if trained on, leading to a decreased classification accuracy for existing classes.

2. Similarly, a new entity class is introduced through partially annotated data and predictions by a model trained on such data should still be of high quality: A common multi-label single-class classifier, as used for NER in [21] or [49], needs a label for each token. But in some cases such a label can either not be provided at all or it contains only partial information about the class, i.e. that it is not part of a specific class, but the class membership is unknown otherwise (a non-entity for a certain class). If in doubt, the *Outside* class could be used here, leading to the aforementioned problem, or the tokens within the span of these entities should be excluded from training. But since this should be valuable information for the classifier, a model, that can take advantage of these mentions, is desirable.

Figure 6.1 shows examples of sentences, which could occur in a partially annotated dataset for NER.

In the following, we are proposing multiple strategies for NER with partially annotated data with an evaluation in Section 6.4 showing how well they meet these goals. The BIO scheme was used in all models to encode entity spans.

	Tomatoes	have	been	designated	the	state	vegetable	of	New	Jersey	.
Gold	B-FOOD	O	O	O	O	O	O	O	B-LOC	I-LOC	O
Partial	B-FOOD	O	O	O	O	O	O	O	O	O	O

(a) A partially annotated dataset for the class *Food* may not contain annotations for other classes, such as *Location*. When trained on such data, a model may not be able to recognize, for example, entities of type *Location* appropriately.

	Tomato	is	an	American	musician	.
Gold	B-PER	O	O	B-MISC	O	O
Partial	NO-FOOD	O	O	O	O	O

(b) Partially annotated datasets derived from Wikipedia can contain information about certain entities and a class they are not part of, e.g. not *Food* in this case.

Figure 6.1: Examples of sentences and their annotation, which can occur in a partially annotated dataset for NER and new classes, extracted from Wikipedia.

Ordinary sequence tagger

The model described here is used as a starting point for all subsequent models. We are using the NER model proposed in [21], which was published with the ubiquitous BERT model and shown in Figure 6.2. An output layer with a softmax activation is used for token classification. The Categorical Cross-Entropy is used as loss function:

$$J_1(\Theta) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{M} y_i^{(n)} \log \hat{y}_i^{(n)} \tag{6.1}$$

with N as the number of tokens, M the number of classes, $y_i^{(n)} = 1$ if token n belongs to class i, 0 otherwise, and $\hat{y}_i^{(n)}$ the predicted probability for token n and class i. It is straightforward that this loss function is not capable of taking advantage of non-entities as well as entities of unknown type in the training dataset. Therefore, these entities are considered to belong to the *Outside* class for this model, which is denoted as *Softmax*-model.

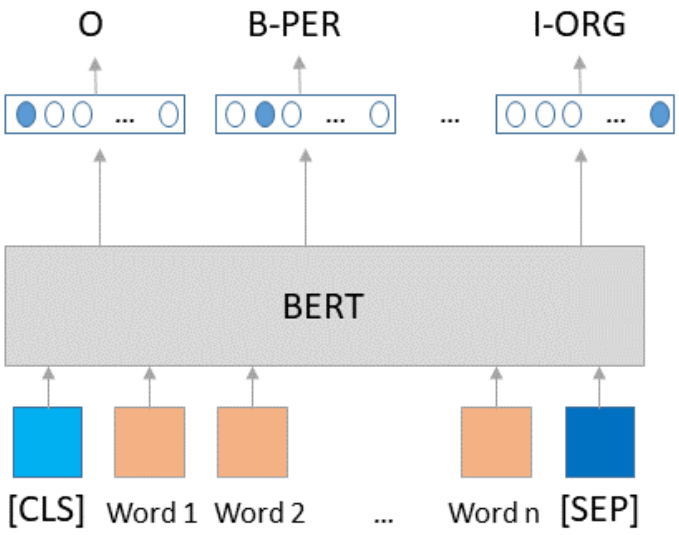


Figure 6.2: NER with BERT. Each token is embedded with the BERT model, which is used as input for the final classification layer.

Sequence tagger ignoring entities

If a span of tokens was annotated as non-entity or an entity of unknown type, it is probably detrimental for the model to simply classify it as *Outside*. Therefore, through adjusting the loss function to ignore these tokens when training, we do not harm the model. This can be done with the following loss:

$$J_2(\Theta) = -\frac{1}{N}\sum_{n=1}^{N} w^{(n)} \sum_{i=1}^{M} y_i^{(n)} \log \hat{y}_i^{(n)} \tag{6.2}$$

A weight w is added for each token with $w = 0$ for non-entity tokens and tokens of unknown type, the model should not be trained on, and $w = 1$ for all other tokens. This model is denoted as *Softmax (weighted)*.

Sequence tagger trained on non-entities

So far the problem of taking advantage of the fact that some spans of tokens are known to be entities not belonging to the new class C, but unknown which is the true class, has not been solved yet. In order to do so, our model can be slightly adapted through using a multi-class multi-label approach with a sigmoid activation for each class in the output layer. The following loss

function based on binary cross-entropy can be used here:

$$J_3(\Theta) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{M} w_i^{(n)}(y_i^{(n)} \log \hat{y}_i^{(n)} + (1 - y_i^{(n)}) \log(1 - \hat{y}_i^{(n)})) \tag{6.3}$$

Now the model is able to specifically learn that an entity is not part of a particular class i, i.e. $w_i^{(n)} = 1$ and $y_i^{(n)} = 0$ for a token $t^{(n)}$ known not to belong to class i. This model is denoted as *Sigmoid (weighted)* and is able to take advantage of all available information in the partially annotated datasets.

While this would be out-of-scope for this chapter, such a model can also be used for assigning multiple labels to a single entity, in case of overlapping entity classes, or detecting nested entities[8].

6.4 Evaluation

In this Section, we provide an evaluation of all three training strategies. This includes details about the datasets we created from Wikipedia semi-automatically.

6.4.1 Datasets derived from Wikipedia

We created two datasets, for the classes *Food* and *Drugs*, referring to the Wikipedia categories "Food and Drink"[9] and "Drugs"[10], respectively. Table 6.1 shows statistics about these datasets. *Positive Entities*[11] refer to the number of entities of the corresponding new class, detected through matching the set of class-related articles and hyperlinks in Wikipedia. *Non-Entities*[12] denote entities that are linked to other articles in Wikipedia. *Excluded Entities*[13] are entities that could potentially refer to an entity of the type of interest, e.g. through matching an alias of a corresponding article, but a hyperlink is missing. *Entities* correspond to the number of entities, i.e. distinct

[8] At least if the nested entity is of a different type than the entity containing it, otherwise the annotation could be ambiguous.

[9] `https://en.wikipedia.org/wiki/Category:Food_and_drink`

[10] `https://en.wikipedia.org/wiki/Category:Drugs`

[11] All three models can be trained on these.

[12] Only *Sigmoid (weighted)* can properly use these, excluded from training for *Softmax (weighted)*, considered as *Outside* for the *Softmax* model.

[13] Excluded from training for the *weighted* models, *Outside* for the *Softmax* model.

Wikipedia articles, for each type and *Sentence* to the number of sentences in each dataset.

Entity Type	Pos. Entities	Non-Entities	Excl. Mentions	Entities	Sentences
Food	246,292	139,825	293,926	17,164	283,635
Drugs	93,439	16,772	65,350	27,863	82,498

Table 6.1: Statistics for datasets derived from Wikipedia for the types Food and Drugs.

It is worth mentioning that it seems like Wikipedia knows many more drugs than food items (see column *Entities*), but still, the food-related dataset contains many more sentences (see column *Sentences*). However, this is not necessarily an issue for model training, as we show in the remainder of this section.

Some of the sentences were left out for manual annotation, resulting in datasets with 280,000 and 80,000 sentences for *Food* and *Drugs*, respectively. 500 of each of these sets of sentences were manually annotated by an annotator familiar with the task. These datasets are referred to as *Food* and *Drugs gold* in the following.

6.4.2 Model parameters

We trained all models with the following settings:

- Batch size: 32
- Optimizer: Adam [46] with $lr = 5e^{-5}$ and $\varepsilon = 1e^{-8}$ without weight decay.
- 1-10 epochs, model with best F1-score on the dev dataset was selected.
- Train-dev-test split: 80%-10%-10% with corresponding Wikipedia and CoNLL datasets merged and shuffled.[14]

[14]Since the CoNLL datasets could potentially contain mentions of *Food* and *Drugs*, the alias dictionary for each dataset was used to exclude those from training for the *weighted* models.

The CoNLL 2003 NER dataset is split into three parts: Train, Test A (dev) and Test B (test). While Train and Test A are similar, i.e. they were extracted from news data from the same years, Test B was extracted from news articles from different years resulting in a more challenging dataset. We simply added our train, dev and test data to the appropriate CoNLL datasets.

6.4.3 Baselines

We compare our approaches to the following baselines:

- *Mayhew et al.* [59]: BiLSTM-CRF model architecture for partially annotated datasets (similar to Jie et al. [44], although performing better), focusing on learning weights for each token. High weights indicate a high confidence that such tokens can be learned from using the known annotation, low weights indicate that such tokens should be ignored in the loss of the model.

- *Dictionary*: We collected a dictionary of Wikipedia articles for each dataset (see Table 6.1 for the sizes), which can be used to detect entities of these classes through exact string matching.

Due to long training times for *Mayhew et al.*, we had to reduce the amount of words from the new datasets used for the evaluation to 1,000,000 tokens from each dataset (in addition to CoNLL 2003 data). The resulting datasets represent ≈11% and ≈40% of the available words for the *Food* and *Drugs* datasets, respectively. Results for our models trained on the full datasets can be found in [85].

6.4.4 CoNLL + Wikipedia

The overall task for such a model is to learn to fully annotate a dataset with all available entity types, i.e. *Person*, *Location*, *Organization* and *Miscellaneous* from CoNLL and *Food* or *Drugs* from our new datasets[15]. However, there are

[15]There are two separate evaluations for CoNLL and *Food* or *Drugs* as the latter types are based on partially annotated datasets and we found it more interesting to distinguish here as there could be difference between those two datasets, which may be missed if combined.

no fully annotated datasets, fulfilling these requirements, available. Therefore, the evaluation of trained models is done separately, for CoNLL as well as for the partially annotated and manually annotated datasets for *Food* and *Drugs*, even though we trained two models for CoNLL types plus *Food* or *Drugs*.

Table 6.2 shows results (*Precision*, *Recall* and *F1-score*) of the trained models when tested on all available datasets. The results for CoNLL test A/B can be seen as a sanity check whether the newly added data is too noisy and the output layer and loss function may or may not be able to compensate for this. In order to compare, *Softmax (no food)* and *Mayhew et al. (no food)* denote models, which were trained on CoNLL only, without any new Wikipedia-based datasets and entity classes[16].

Dataset	CoNLL test A			CoNLL test B					
Model	P	R	F1	P	R	F1			
Softmax (no food)	0.95	0.96	**0.95**	0.90	0.92	**0.91**			
Mayhew et al. (no food)	–	–	–	–	–	0.90			
Softmax	0.94	0.92	0.93	0.90	0.89	0.90			
Softmax (weighted)	0.95	0.94	**0.95**	0.91	0.92	**0.91**			
Sigmoid (weighted)	0.94	0.94	0.94	0.89	0.90	0.90			
Mayhew et al.	0.89	0.92	0.91	0.83	0.89	0.86			
Dataset	Food test A			Food test B			Food gold		
Model	P	R	F1	P	R	F1	P	R	F1
Softmax	0.69	0.72	0.70	0.68	0.73	0.70	0.71	0.42	0.53
Softmax (weighted)	0.85	0.88	**0.86**	0.84	0.88	**0.86**	0.53	0.67	0.59
Sigmoid (weighted)	0.84	0.85	0.84	0.82	0.84	0.83	0.66	0.62	**0.64**
Mayhew et al.	0.25	0.85	0.39	0.25	0.86	0.39	0.44	0.52	0.48
Dictionary	–	–	–	–	–	–	0.28	0.51	0.36

Table 6.2: Results for CoNLL 2003 mixed with the Wikipedia dataset for *Food*. Best F1-score for each dataset in bold.

All three approaches (*Softmax*, *Softmax (weighted)* and *Sigmoid (weighted)*) are able to produce reasonable results for CoNLL. Our experience was that the Wikipedia dataset for *Food* did not contain a lot of entities of class *Person*, *Location*, *Organization* or *Miscellaneous*. Therefore, it is not surprising that our Wikipedia-based *Food* dataset did not add too much noise harming the ability of the model to still recognize the original entity classes. Whereas

[16]In [59], only the F1-score for the CoNLL test B dataset was provided.

Mayhew et al. was not able to keep up with the original result without new data (0.90 vs. 0.86 F1-score). This approach assumes a certain amount of entities of the original types in the whole dataset used for training (CoNLL 2003 and *Food* data), inevitably leading to misclassifications, hence the lower precision.

Results for *Food test A/B* show the ability of the model to adapt to presumably noisy data added. The *weighted* approaches clearly outperform the *Softmax* approach. This shows the necessity of at least excluding mentions that are known to be ambiguous from training. The *Sigmoid (weighted)* approach does not seem to add any benefit in this setting. *Mayhew et al.* performs very poorly on both datasets. While it is able to recognize many *Food* entities, leading to a high recall, it recognizes many entities as *Food* incorrectly, i.e. leading to a low precision.

The results on the *Food gold* dataset are slightly more diverse. *Sigmoid (weighted)* approach returned the best results with almost balanced *Precision* and *Recall*. Both, *Softmax* and *Softmax(weighted)* result in a much larger gap. The dictionary-based baseline approach performs poorly. *Mayhew et al.* is able to outperform this baseline, while still achieving a significantly worse classification result than the weighted approaches.

Softmax on the one hand produces a high *Precision* and low *Recall*, i.e. it is capable of recognizing very few entities relatively consistently. It is possible that these entities are also often linked in Wikipedia, while others are not or at least not very often, explaining why the results on *Food test A/B* do not show this phenomenon. In addition, it seems like this model pays more attention to the surface form of the mention, mainly recognizing entities it has seen before and not necessarily considering the context.

Softmax (weighted) on the other hand produces a high *Recall* and low *Precision*. This indicates that this model learned to recognize *Food* entities, but since it was not trained to not recognize certain mentions that could be *Food* items as well, but are for sure not (either they are linked to *non-Food* Wikipedia articles or they are tagged by the CoreNLP NER), it never learned to appropriately label ambiguous spans of text. This is not an issue for *Food*

test A/B, presumably since (as previously mentioned), certain *Food* entities are more consistently linked through hyperlinks in Wikipedia than others, while these other entities are annotated in the *Food gold* dataset.

Table 6.3 shows results for all models when trained on CoNLL 2003 and the partially annotated dataset for the class *Drugs*. Overall, the results look similar to Table 6.2 with similar conclusions. Test results on *CoNLL test A/B* are very close to the results of the model without new data added, achieving the goal of not harming the prediction quality for those classes. *Mayhew et al.* performs slightly better than previously, when *Food* data is added.

For *Drugs test A/B*, again, the *weighted* approaches adapt a lot better to the new class than the *Softmax* approach. While still performing poorly, *Mayhew et al.* is able to narrow the gap to these approaches slightly.

However, the distinction between *Softmax (weighted)* and *Sigmoid (weighted)* is less clear for the dataset *Drugs gold*. The latter approach has, again, a more balanced *Precision* and *Recall*, while the final *F1-score* is the same. We assume that drug names in general are less ambiguous and therefore less non-entities are found (in fact only ≈10% of all found entities from Wikipedia in the *Drugs*-dataset are non-entities, compared to ≈20% for the *Food*-dataset). In addition, we noticed that, while manually annotating sentences for the dataset *Drugs gold*, this dataset is less noisy than the *Food* dataset. The *Softmax* model is even able to outperform the other approaches at least on the *Drugs gold* dataset, while still under-performing on *Drugs test A/B*. Again, *Mayhew et al.* is able to narrow the gap and achieving only a slightly worse result, while having a large gap between precision and recall.

6.4.5 Example

The proposed system is able to detect entities of the types *Food* and *Drugs*. In addition to what has been extracted by the system described in Chapter 5, we can now detect such entities and extract "Pancakes" from the conversation in Figure 1.2a, which can be seen in Figure 6.3.

Dataset	CoNLL test A			CoNLL test B		
Model	P	R	F1	P	R	F1
Softmax (no drugs)	0.95	0.96	**0.95**	0.90	0.92	**0.91**
Mayhew et al.	–	–	–	–	–	0.90
Softmax	0.95	0.94	**0.95**	0.90	0.91	0.90
Softmax (weighted)	0.95	0.95	**0.95**	0.91	0.92	**0.91**
Sigmoid (weighted)	0.94	0.94	0.94	0.89	0.91	0.90
Mayhew et al.	0.92	0.91	0.92	0.86	0.89	0.87

Dataset	Drugs test A			Drugs test B			Drugs gold		
Model	P	R	F1	P	R	F1	P	R	F1
Softmax	0.84	0.88	0.86	0.85	0.88	0.87	0.73	0.63	**0.68**
Softmax (weighted)	0.93	0.95	**0.94**	0.93	0.96	**0.94**	0.59	0.80	**0.68**
Sigmoid (weighted)	0.93	0.94	0.93	0.93	0.93	0.93	0.62	0.74	0.67
Mayhew et al.	0.42	0.94	0.58	0.41	0.94	0.57	0.50	0.88	0.64
Dictionary	–	–	–	–	–	–	0.28	0.44	0.34

Table 6.3: Results for CoNLL 2003 mixed with the Wikipedia dataset for *Drugs*. Best F1-score for each dataset in bold.

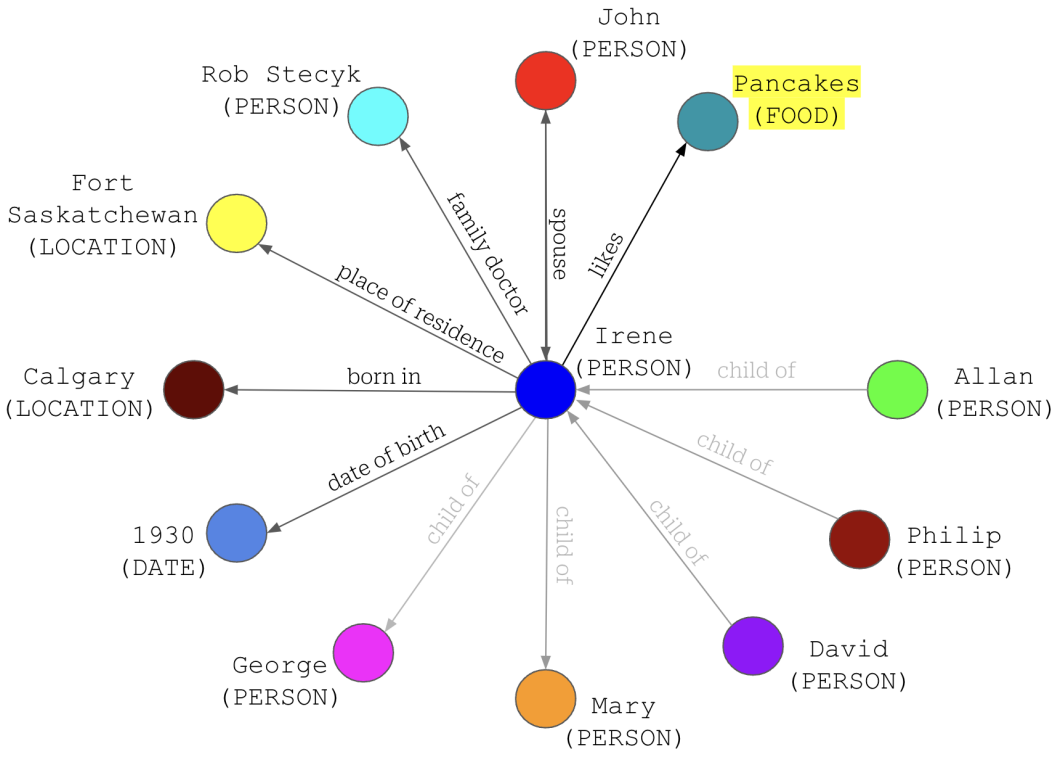


Figure 6.3: KG extracted from the conversation in Figure 1.2a by the proposed systems. The entity "Pancakes" is newly extracted. Light grey relations cannot be detected yet.

6.5 Conclusion

We proposed an approach to extract partially annotated datasets for Named Entity Recognition semi-automatically from Wikipedia. In addition, three model architectures, based on the commonly used BERT model, differing only in the activation function of the output layer as well as the loss function, were compared. A state-of-the-art model [59] was used as a baseline in addition to a simple dictionary-based approach. Two of the tested models, introducing simple changes to the base model, show promising results when trained on partially annotated Wikipedia-based data and tested on similar data as well as on a small amount of manually annotated data for the classes *Food* and *Drugs*. Performance on the CoNLL 2003 NER dataset is not harmed significantly through adding data for the tested new entity classes. The two baselines can be outperformed by a very large margin. Datasets and code are publicly available[17].

For future work, it would be worth investigating how to narrow the gap between test results on semi-automatically and manually annotated data. This could include a classification approach for Wikipedia articles in order to rule out articles, which do not fit a certain category very well.

[17] `https://github.com/mjstrobl/NER_for_partially_annotated_data`

Chapter 7

Relation Extraction

In this chapter we provide more details on how to create datasets for the task of Relation Extraction (RE) with our Flexible Relation Extraction Data Annotation (FREDA) framework.

7.1 Introduction

The task of Relation Extraction (RE) is one of the major parts of Knowledge Base Population (KBP) [43], i.e. the augmentation of an existing Knowledge Base (KB). The main goal is to recognize relations, which are expressed between two entities mentioned in the same sentence or document. At present, this is usually achieved by a model based on neural networks, which is trained on, ideally, large amounts of labeled data. However, there are very few publicly available labeled datasets. When available, these are usually limited to specific relations, and often lack the relations that one is interested in.

To illustrate the difficulty of the annotation task for RE, consider the following example[1]:

> "**Melinda** began dating **Microsoft** CEO **Bill Gates** in 1987, after meeting **him** at a trade fair in **New York**."

Four entities are mentioned (**Melinda**, **Microsoft**, **Bill Gates** with his co-reference **him**, and finally **New York**). If the task is detecting the *ceo_of*-relation, the fact is explicit in the sentence. However, if the task is detecting

[1]From Melinda Gates' Wikipedia article `https://en.wikipedia.org/wiki/Melinda_Gates`

the *spouse*-relation, one may indicate such relationship between Melinda and Bill Gates. While this might be true, it is only based on the annotator's knowledge and such annotation could improperly mislead a classifier and train it to perhaps associate "dating" with the *spouse*-relation.

Another possible annotation error can be illustrated by this example:

> "**Bill Gates** has received an honorary Doctorate from **Cambridge University**."

An annotator confused about the meaning of **alma mater** could erroneously tag the *alma_mater*-relation between Gates and the university while Bill Gates never attended Cambridge. Although not always done (see TACRED [97]), this problem could be mitigated by using multiple annotators, as suggested by [3].

This leads to the question, how is it possible to create RE datasets, which:

(1) are of high enough quality and validated; **(2)** are large enough to train effective models; **(3)** can be relatively easily extended with more relations; **(4)** can be quick to annotate and construct;

We believe these four conditions are essential for the creation of useful RE datasets.

Previous work encountered some difficulties to fulfill these conditions. [3] suggest that generating high quality and validated data (condition (1)) may not be met through crowd-sourcing or at least not if there are no measures in place that ensure data quality. Therefore, ideally, multiple annotators per example are necessary to ensure consistency. In addition, [71] suggested that all entities in a sentence need to be annotated in order to train models that generalize properly, which is not always done.

Creating large enough datasets (condition (2)) is not a trivial task and the required size of labeled datasets is usually unknown in advance. Indeed, labeling entities, co-references, and their relations to each other, even in a single sentence, can be complicated, depending on the relation in question, as well as time-consuming [60]. Moreover, existing datasets often contain a predetermined rigid set of common annotated relations like *located_in*, *founded* and

spouse. Therefore, an easily extendable and flexible framework (condition (3)) is practical to gather data for new specific or uncommon relations. Enabling a quick construction (condition (4)) is not about the haste in annotation, which in turn could lead to errors, but more about the ease of annotation to avoid a repetitious tedious task. This ease of annotation is conducive to the collection of larger datasets (condition (2)).

We provide a dataset with 10,022 sentences annotated for 19 relations (15 used by [60] and 4 new relations) with at least two annotators per sentence (third annotator for tie-breaking in case the first two disagree). Our FREDA framework was used for data annotation, as described in Chapter 4. Models trained on these sentences and their labels show a significant performance gain in F1 scores than previously reported results on common RE datasets, demonstrating that it is possible to obtain significantly better results when the annotations are of high quality.

The remainder of this chapter is structured as follows: Section 7.2 presents related work on RE datasets and their construction. Section 7.3 delineates the model architecture used for training models for RE on data created with our FREDA framework. Section 7.4 details the evaluation procedure with our conclusions in Section 7.6.

7.2 Related Work

The TAC Relation Extraction Dataset (TACRED) [97] is a large dataset which used Mechanical Turk crowd annotation with 41 relations and 106,264 examples. However, each example was only annotated by a single annotator and, as pointed out by [3], there is a large number of labeling errors misleading trained models. Although TACRED is still a popular dataset and widely used, e.g. by [45] or [4], presumably since previous semi-automatic labelling approaches, such as text annotations with Distant Supervision (see [70]), i.e. aligning sentences with facts from KBs only through matching entities, are even more error-prone. It seems to be important to double check annotations with more annotators, even though less text can be annotated that way in a

specific time-frame.

In addition, [71] investigated the heuristics that a model trained on TACRED may learn to score high on the test set without solving the underlying problem: (1) Only 17.2% of the sentences in TACRED have more than a single pair of entities annotated. Therefore, in most sentences a model will only encounter a single pair of entities, for which it is asked to predict a relation. Instead of predicting a relation for this pair, it may rather learn to predict whether a sentence expresses a certain relation, ignoring the potential subject and object. This would lead to a high recall, but may also result in many false positives, leading to a low precision when tested. (2) A classifier may predict a relation solely based on whether the types of entities in that relation are present in a sentence, especially for relations, which have a unique entity-type-pair, such as *per:religion* with *Person* as subject and *Religion* as object. The authors did a manual investigation of this relation with models trained on TACRED, often leading to false positives, if entities of type *Person* and *Religion* were present in sentences from Wikipedia, but the relation actually not expressed. One of their conclusions was that all entities in a sentence need to be annotated to lower the impact of these problems on the trained model, i.e. a model may generalize better to unseen data in this case.

A similar approach to Distant Supervision is T-REx [26]. In this case, the text-KB-fact-alignment is done more carefully using keyword matching, resulting in a significantly higher annotation quality. However, these approaches suffer from a restriction problem: only relations found in existing KBs can be considered. Therefore, if new relations or relations with a very low coverage in these KBs are of interest, such an approach cannot be used.

KnowledgeNet [60] is a project aiming to manually annotate 100,000 facts for 100 properties, although at the time of publication 13,425 facts for 15 relations were available. Their annotation pipeline consists of the following steps:

1. Fetch sentences: Using T-REx to find sentences that could describe facts

from Wikidata[2] and, in addition, sentences that contain certain keywords, chosen for each property, and therefore potentially expressing a fact of interest.

2. Mention Detection: Annotators are asked to highlight entity names.

3. Fact Classification: Pairs of mentions are classified as positive or negative examples for a relation.

4. Entity Linking: Linking mentions to their corresponding Wikidata entity.

Each sentence is labeled by at least two annotators to ensure high data quality. The authors report an average of 3.9 minutes to annotate a single sentence by up to 3 annotators. In our approach, Entity Linking is not explicitly done. However, this step is only responsible for 28% of the time spent according to their study, and therefore a significant amount of time (about 2.8 minutes) is still needed to annotate a sentence.

In addition, it is important to note that there has been work conducted on few- or even zero-shot learning for RE in order to avoid the necessity of creating new datasets for relations, which do not appear in existing datasets. Han et al. [38] described a framework for few-shot RE and published the FewRel 1.0 dataset. It consists of 700 examples from Wikipedia for each of the 100 relations considered and was annotated by crowd workers. In contrast to our problem, the task for a model trained on FewRel 1.0 is to match an example, a sentence with two entities annotated, to a reference example from the set of examples of the true relation. Effectively, the goal is to rank relations and their reference examples. The relation corresponding to the best fitting example is selected. This allows the model to be trained on an arbitrary number of sentences (or even zero) and, hopefully, still select the correct relation through ranking it the highest. Based on FewRel 1.0, the FewRel 2.0 dataset [30] aims to fix two issues: (1) Adapt models to select relations from new domains and (2) avoid selecting a relation, if none of the available ones fit.

[2] `https://www.wikidata.org/wiki/Wikidata:Main_Page`

Even though the models used in [38] and [30] show promising results, Brody et al. [12] revealed similar issues with the FewRel framework to the ones found for TACRED by Rosenman et al. [71]. In particular, they found that models trained on FewRel 1.0 seem to heavily rely on entity types and adding training data for relations with similar entity types may mitigate this issue. Furthermore, the evaluation metrics used in FewRel ignores the fact that models may perform much better on some relations than others (in fact, they found a large gap between best and worst), potentially due to the aforementioned entity type issue. This, again, leads to the question how to annotate more data for more relations quickly and accurately.

7.3 Model Architecture

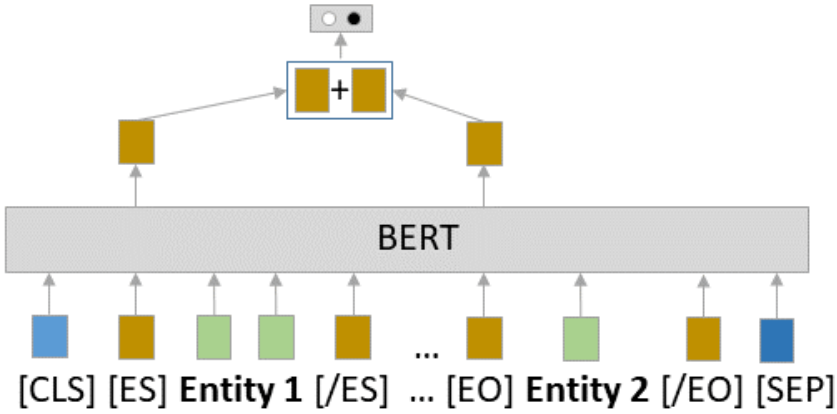


Figure 7.1: Binary classification model used for Relation Extraction from a sentence with annotated subject and object, similar to the one proposed by [79]. Both entities are encapsulated in special tokens and the embeddings of the begin tokens of both are used for classification.

We adapted the best model architecture from [79], which is based on the BERT Transformer model [21] and showed good results on TACRED. Although instead of a multi-class classifier, as commonly used for models trained on TACRED, we altered the model to do binary classification. It is depicted in Figure 7.1. Apart from BERT's special tokens, two new tokens are introduced for the subject (entity start token *[ES]* and entity end token *[/ES]*) and for the object (entity start token *[EO]* and entity end token *[/EO]*), which are

referred to as entity markers. All new tokens are fine-tuned during model training. The BERT embeddings of the start tokens of both entities concatenated are used as input of a classification layer with a sigmoid activation function, which makes a binary decision whether the relation of interest is expressed between the marked entities. Since sentences can express multiple relations, even between the same entity-pair, independent classification decisions have to be made and one model per relation can be trained. Therefore, the input of the model for a given relation is a sentence with one entity mention marked as subject (with *[ES]* and *[/ES]*) and one entity mention marked as object (with *[EO]* and *[/EO]*).

This model architecture aims to solely recognize and decide whether or not the context of two entities suggests that the relation, the model was trained for, is expressed. There are no additional tasks such as NER or CR involved, which have to be learned and may influence the quality of the predictions. Therefore, this architecture should be suitable for finding out what performance is possible for the task of RE if accurately annotated and large enough datasets are available for training.

7.4 Evaluation

In this Section, we provide a detailed evaluation of how models trained on datasets created with FREDA perform, how much data is required to reach a certain performance level for selected relations and how fast these annotations can be acquired.

7.4.1 Model Training and Dataset Statistics

We are using the previously mentioned model architecture outlined in Figure 7.1 with the cased large model of BERT. Learning rate of $5 * 10^{-6}$ (linear decay), Adam optimizer [46], batch size 32, 1-10 epochs (varies per relation; determined using 5-fold cross-validation). Test sets contain 10% of the whole dataset, one per relation.

Table 7.1 shows statistics of our datasets for each relation and all together.

7 annotators annotated in total 10,022 sentences with >500 examples for each relation. The first 15 relations can be found in the KnowledgeNet dataset as well. In addition, we annotated 4 more common relations, which are also part of the schema.org ontology for persons[3]. From all sentences with a positive response (a sentence is deemed to express the relation at hand), a number of *positive facts* can be extracted with a positive label, which can be used for model training. Often it is possible to extract multiple such facts for a single sentence since subjects and objects can be mentioned several times in the sentence and we may have different subjects or objects in the same sentence. All other potential facts, which can be extracted from a sentence, are considered as *Negative Facts*.

For instance, consider the following sentence:

> "**Princess Alberta** was the fourth daughter of **Queen Victoria** and **Prince Albert**."

Two positive facts can be extracted for the *child_of* relation:

- **Princess Alberta** *child_of* **Queen Victoria**
- **Princess Alberta** *child_of* **Prince Albert**

Facts with a negative label can be easily created by considering all other pairs of entities, which do not express the relation at hand. Therefore, four negative facts can be extracted from the previous sentence for the same relation[4]:

- **Queen Victoria** *child_of* **Princess Alberta**
- **Prince Albert** *child_of* **Princess Alberta**
- **Prince Albert** *child_of* **Queen Victoria**
- **Queen Victoria** *child_of* **Prince Albert**

[3] `https://schema.org/Person`

[4] It is possible that these negative facts still express another relation, e.g. *parent* or *spouse*. But since the corresponding model is trained to do binary classification, they are considered as negative facts in this case.

Since there are typically more negative facts than positive ones, each training example is weighted in the loss function (binary cross-entropy loss) accordingly, in order to account for the class imbalance.

We calculated the inter-annotator agreement for the first and second annotator with Cohen's Kappa. Two annotators agree when they both consider that the relation in question is expressed (or inexistent) in the sentence at hand. Overall, the results can be considered as excellent with $\kappa = 0.85$ for all relations together. Although it ranges between 0.48 (*place_of_residence* and 0.96 (*date_of_birth*). Some relations require more discussion between annotators are therefore more difficult and time-consuming to annotate for humans, leading to more disagreement.

Relation	**Sentences**	**Positive facts**	**Negative facts**	**Inter-annotator kappa**
date_of_birth	555	683	11,447	0.96
date_of_death	523	721	13,739	0.89
place_of_residence	510	624	12,256	0.48
place_of_birth	508	541	14,915	0.94
nationality	518	595	12,319	0.93
employee_or_member_of	501	604	12,114	0.73
educated_at	544	500	11,892	0.92
political_affiliation	504	638	16,218	0.95
child_of	504	522	12,312	0.93
spouse	550	1,524	14,672	0.88
date_founded	533	265	10,303	0.93
headquarters	506	1,022	9,070	0.69
subsidiary_of	544	320	10,664	0.64
founded	561	317	12,665	0.89
ceo_of	541	604	7,374	0.84
award	521	366	8,494	0.74
alma_mater	528	278	12,850	0.65
place_of_death	517	366	18,898	0.93
sibling	554	670	10,476	0.90
Total	10,022	11,160	232,678	0.85

Table 7.1: Data statistics (Total and per relation): Number of sentences, number of positive and negative facts extracted from these and inter-annotator kappa (between the first two annotators).

7.4.2 Baselines

We conducted multiple experiments to gain insights on the quality of the datasets created with FREDA for the task of RE[5]:

- KnowledgeNet [60]: While the annotation framework is not publicly available, a portion of the datasets annotated by the authors using their tool are available to download and use here. Data annotation with the KnowledgeNet interface presumably leads to high-quality dataset for RE, even though very time-consuming. Therefore, this dataset can be used to evaluate models, which were trained on dataset annotated with FREDA (see Section 7.4.3).

- Challenge RE [71]: A manually annotated and balanced (relation-wise) dataset for RE closely related to TACRED (hence a similar set of relations), which we used for evaluating models trained on datasets annotated with FREDA and a state-of-the-art model trained on TACRED, i.e. the KnowBert+W+W model [69] (see Section 7.4.4).

- BRAT [81]: A tool to manually annotate sentences with entities, co-references, and relations. We conducted a comparison of FREDA and BRAT, measured the time to annotate sentences for both approaches for the *spouse*-relation and compared models trained on the resulting datasets.

7.4.3 Test Results

We trained models on all datasets created with FREDA and tested them on the corresponding test sets as well as on unseen data provided by KnowledgeNet, which can be considered as high-quality.

The KnowledgeNet training data can be downloaded from their repository[6], which can be used for testing models trained on our datasets. Since this dataset does not contain exhaustive entity annotations (only entities participating in

[5]The main goal was to evaluate the quality of the resulting datasets, not the comparison of approaches using or trained on these datasets.

[6]`https://github.com/diffbot/knowledge-net`

a specific relation are annotated), negative examples for testing can only be generated from sentences expressing a relation. These are presumably the more challenging sentences since the model needs to figure out which entity is subject, which is object and which entities are neither. Also, entity-pairs for negative examples can be extracted through considering mentions of the same entity, which cannot be related to each other. For all relations, this results in 10,895 positive and 46,347 negative examples, compared to 11,160 positive and 232,678 negative examples for our datasets.

We are reporting Precision, Recall and F1 score on the FREDA and KnowledgeNet test sets in Table 7.2, broken down for each relation as well as **Interim** results for relations which can be found in the datasets from both approaches, and the **Total** which includes the four additional relations we annotated with FREDA.

The **Interim** F1 score of 0.86 on the FREDA test sets is relatively high overall. Even though it is not possible to directly compare this result against results of state-of-the-art models trained on the commonly used TACRED dataset, these latter usually achieve a significantly lower F1 score on TACRED[7]. Models trained on these datasets created with FREDA also show a similarly high F1 score on the KnowledgeNet dataset with 0.87. The latter dataset was created by different annotators and presumably similar, but still, in detail, different approaches for sentence filtering.

It is often not reported, but we can gain some insights on how such models perform on different relations. While relations, which do not leave a lot of room for interpretation, such as *date_of_birth*, *place_of_birth*, *child_of*, *spouse* or *sibling*, show a very high F1 score, others, such as *subsidiary_of* or *place_of_residence*, show significantly worse results. Cohen's Kappa for inter-annotator agreement for these two relations is quite low with 0.64 and 0.48, respectively, compared to the average of 0.85. Therefore, the datasets corresponding to these two relations can be considered as more challenging to train on, and thus more sentences may need to be annotated.

[7] `https://paperswithcode.com/sota/relation-extraction-on-tacred`

	Datasets					
	FREDA (test)			KnowledgeNet		
Relation	**P**	**R**	**F1**	**P**	**R**	**F1**
date_of_birth (PER →DATE)	0.96	0.97	0.96	0.90	1.00	0.94
date_of_death (PER →DATE)	0.93	0.96	0.94	0.93	0.92	0.93
place_of_residence (PER →LOC)	0.71	0.76	0.73	0.86	0.74	0.79
place_of_birth (PER →LOC)	0.85	1.00	0.92	0.95	0.81	0.87
nationality (PER →LOC)	0.84	0.95	0.89	0.92	0.92	0.92
employee_or_member_of (PER →ORG)	0.68	0.91	0.78	0.95	0.82	0.88
educated_at (PER →ORG)	0.87	0.94	0.90	0.98	0.90	0.94
political_affiliation (PER →ORG)	0.96	1.00	0.98	0.90	0.90	0.90
child_of (PER →PER)	0.75	0.83	0.79	0.91	0.89	0.90
spouse (PER ↔ PER)	0.93	0.91	0.92	0.95	0.89	0.92
date_founded (PER →DATE)	0.83	0.95	0.89	0.94	0.88	0.91
headquarters (ORG →LOC)	0.80	0.84	0.82	0.94	0.86	0.90
subsidiary_of (ORG →ORG)	0.51	0.71	0.59	0.87	0.74	0.80
founded (PER →ORG)	0.72	0.94	0.82	0.49	0.82	0.61
ceo_of (PER →ORG)	0.81	0.89	0.85	0.94	0.91	0.93
Interim	0.83	0.90	0.86	0.88	0.86	0.87
award (PER →AWARD)	0.78	0.83	0.80	–	–	–
alma_mater (PER →ORG)	0.70	0.62	0.65	–	–	–
place_of_death (PER →LOC)	0.79	0.90	0.84	–	–	–
sibling (PER ↔ PER)	0.77	0.79	0.78	–	–	–
Total	0.82	0.89	0.85	–	–	–

Table 7.2: Test set results of the models trained on the FREDA training sets for each relation and both approaches. The last 4 relations are not part of KnowledgeNet's dataset, therefore the results are missing. **Interim** corresponds to the overall results for all 15 relations in both datasets. **Total** includes results on all relations in FREDA's test set.

7.4.4 Challenge RE dataset

Rosenman et al. [71] created a more challenging dataset based on 30 out of 41 TACRED relations, called Challenge RE (CRE). It is relatively balanced, i.e. the number of positive examples is similar to the number of negative examples, whereas TACRED is highly imbalanced. Furthermore, each annotated sentence contains at least two entity pairs that are compatible with the relation the sentence is annotated for, aiming to reveal models that learned to classify sentences rather than classifying entity pairs in a sentence, which would lead to high recall but low precision. Therefore, this dataset is considered to be

	Models					
	FREDA			**KnowBert-W+W**		
Relation	**P**	**R**	**F1**	**P**	**R**	**F1**
date_of_birth	0.96	0.93	**0.95**	0.67	0.99	0.80
date_of_death	0.74	0.78	**0.76**	0.61	0.74	0.67
educated_at	0.85	0.72	0.78	0.68	0.93	**0.79**
sibling	0.76	0.87	**0.81**	0.53	0.89	0.67
spouse	0.84	0.87	**0.85**	0.56	0.86	0.68
founded	0.86	0.53	0.66	0.82	0.76	**0.79**
date_founded	0.86	0.60	0.71	0.60	0.89	**0.72**
Total	0.83	0.76	**0.79**	0.63	0.87	0.73

Table 7.3: Challenge RE dataset test results. The previously trained models from FREDA were used as well as the KnowBERT-W+W model, trained on TACRED and showing state-of-the-art performance on the TACRED test set. The best F1 scores per relation and overall are in bold.

more challenging than the TACRED test set. CRE was specifically created as a challenging dataset in order to test the generalization capabilities of models, for example, trained on TACRED.

We identified 7 relations in CRE that are fully compatible with 7 of FREDA's relations[8], therefore the previously trained models on FREDA datasets can be used to be tested on the CRE dataset for these relations. In order to compare, we chose the KnowBert-W+W model from [69], a knowledge-enhanced version of BERT through the integration of WordNet [61] and a subset of Wikipedia. It also uses entity markers for relation prediction and is trained on TACRED and shows state-of-the-art results on the TACRED test set.

Table 7.3 shows the results on the CRE dataset for both approaches. Overall, the F1 scores show that models based on FREDA often perform significantly better than KnowBERT-W+W, resulting in a higher total average. Another observation is that KnowBERT-W+W shows a very high recall compared to precision, which is expected due to the nature of TACRED and the resulting lack of generalization when tested on a more challenging dataset, such as CRE. FREDA's models, on the other hand, show a more balanced

[8]CRE and TACRED also contain partially overlapping relations with ours, such as *per:children*, *per:parents* (inverse of the previous), *per:city_of_birth* or *per:country_of_birth*. However, these relations can be seen as subsets of some of our relations and can therefore not be used for testing.

precision and recall, indicating that they pay more attention to which entity is subject and which is object, i.e. our datasets may lead to better generalization properties for models when trained on them, compared to TACRED. This also indicates that our sentence pre-selection step with keywords and Distant Supervision, which is important to end up with balanced datasets, is relatively general and does not necessarily only pre-select easy sentences, while still leading to a high F1 score when trained and tested on (see Table 7.2).

7.4.5 How many sentences do we need per relation?

TACRED contains a variety of relations with a high variance in the number of examples per relation (3,862 for *per:title* and only 33 for *org:dissolved*). However, typically only the overall performance of a model trained and tested on TACRED is reported in the literature, i.e. it is unknown how well these models perform on each relation and how many examples or sentences per relation are needed to reach a certain performance level.

We want to shed some light into the question of how many sentences have to be annotated per relation and how is it possible to find out whether more annotated sentences may be beneficial. Figure 7.2 shows the model performance on the FREDA test sets (same as used for the experiments reported in Table 7.2) for five different relations (*date_of_birth*, *spouse*, *educated_at*, *place_of_residence* and *subsidiary_of*), when trained on 100, 200, 300, 400 or all available sentences we annotated using FREDA and shuffled before sampling. These relations were selected since the models trained on the corresponding datasets show different performance levels (Table 7.2) as well as the inter-annotator agreement varies widely (Table 7.1).

The model corresponding to the least controversial relation among annotators (*date_of_birth*), i.e. the one with the highest Kappa, already shows stable performance after being trained on only 100 sentences. Models for the *spouse* and *educated_at* relations need slightly more sentences, but barely improve after being trained on more than 300. Whereas for the *place_of_residence* and *subsidiary_of* relations, even close to 500 sentences[9] seem to be insufficient to

[9]As previously mentioned, we annotated at least 500 sentences per relation. However,

possibly get to a similarly high performance than the models for the other relations.

The Pearson Correlation Coefficient between model performance for each relation and the inter-annotator agreement is 0.75, i.e. both values are highly correlated. It can be concluded that if annotators often do not agree on annotations for certain relations, models have more difficulties to predict these relations as well, indicating that more data is needed. Whereas, if annotators barely ever disagree a relatively small amount of data is necessary.

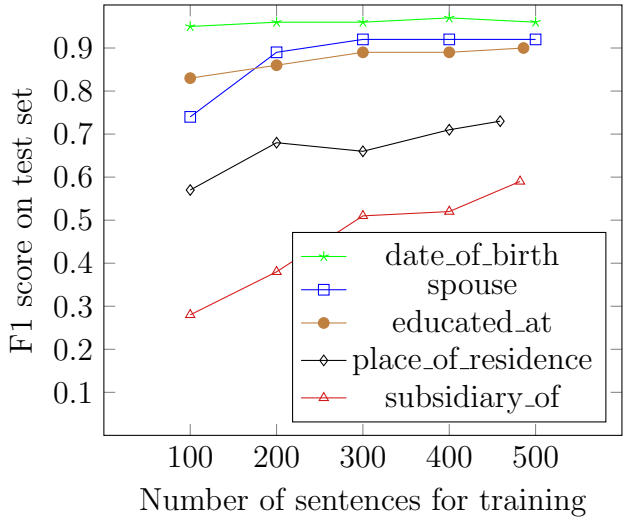


Figure 7.2: F1-score on FREDA (test) for different training set sizes and a selection of relations.

7.4.6 Annotation Speed

Data annotation for RE can be prohibitively time-consuming. Therefore, approaches for quick dataset construction are essential in order to be able to easily extend existing datasets with more relations or create entirely new datasets. We asked two annotators, familiar with the task, to annotate sentences for the *spouse*-relation using the following approaches:

- **BRAT** [81]: BRAT is an open-source web-based tool for data annotation. Entities of different types can be annotated through selecting spans

10% of these sentences are kept aside as test sets, i.e. the rest of the training sets may contain slightly less than 500 sentences.

of text. Relations are annotated through connecting these entities.

- **FREDA (plain)**: In order to solely and fairly compare FREDA's annotation interface with BRAT, WEXEA entity annotations were removed for this approach.

- **FREDA**: Full framework, including annotations from WEXEA.

Sentences were randomly selected from WEXEA. In order to keep the workload as similar as possible for each annotator and approach, all sentences contain exactly 25 words and a new unseen set of 100 sentences was used every time. In addition, the annotators were asked to only select entities of relevant types (*Person* in this case), which reduces the workload and the *spouse*-relation is not supposed to be applied to other types. However, WEXEA itself contains entity annotations of other types in which case annotators were not asked to remove entities for the approach **FREDA**.

	BRAT		**FREDA (plain)**		**FREDA**	
	sec.	**F1**	**sec.**	**F1**	**sec.**	**F1**
Annotator A	23.3	0.53	17.7	0.55	**9.2**	**0.69**
Annotator B	33.1	0.48	25.3	0.43	**12.4**	**0.56**

Table 7.4: Average annotation speed in seconds per sentence for each annotator, lower is better. A model was trained for each dataset and the F1 score on the CRE dataset for the *spouse*-relation as test set is reported. Best results per annotator are in bold.

Table 7.4 shows the average annotation speeds in seconds per sentence for both annotators and all three approaches. In general, annotation speeds vary significantly for each annotator since multiple steps are required and the speeds of each of them depend on individual abilities. The resulting datasets were used to train models, which were tested on the CRE dataset for the *spouse*-relation. Annotations with **FREDA** show the best F1 score for both annotators. Note that entities of other types are annotated as well using **FREDA**, while still keeping the average time to annotate a sentence low. This resulted in more examples for training and therefore better models when

tested on CRE. Results for datasets from the other approaches are similar, suggesting annotation quality is similar[10].

For both annotators, the FREDA interface, represented through the approach **FREDA (plain)**, lead to a 24% increase in annotation speed compared to **BRAT** and another 48% to 51% increase for pre-annotated sentences from WEXEA, i.e. for the approach **FREDA** compared to **FREDA (plain)**, even though more entity types were annotated with **FREDA**. This should give an order of magnitude for the time required to annotated a sentence for a relation and how FREDA can help reduce the workload for annotators through its easy-to-use interface as well as using pre-annotated sentences.

7.4.7 Example

The trained models can now be used to detect relations in text, independent of any rules, which has been described in Chapter 5. In addition to what has been extracted by the systems described in Chapter 5 and 6, we can now detect all *child_of*-relations from the conversation in Figure 1.2a, which can be seen in Figure 7.3.

[10]All F1 scores in Table 7.4 are lower than reported in Table 7.3 since significantly less data was used for training.

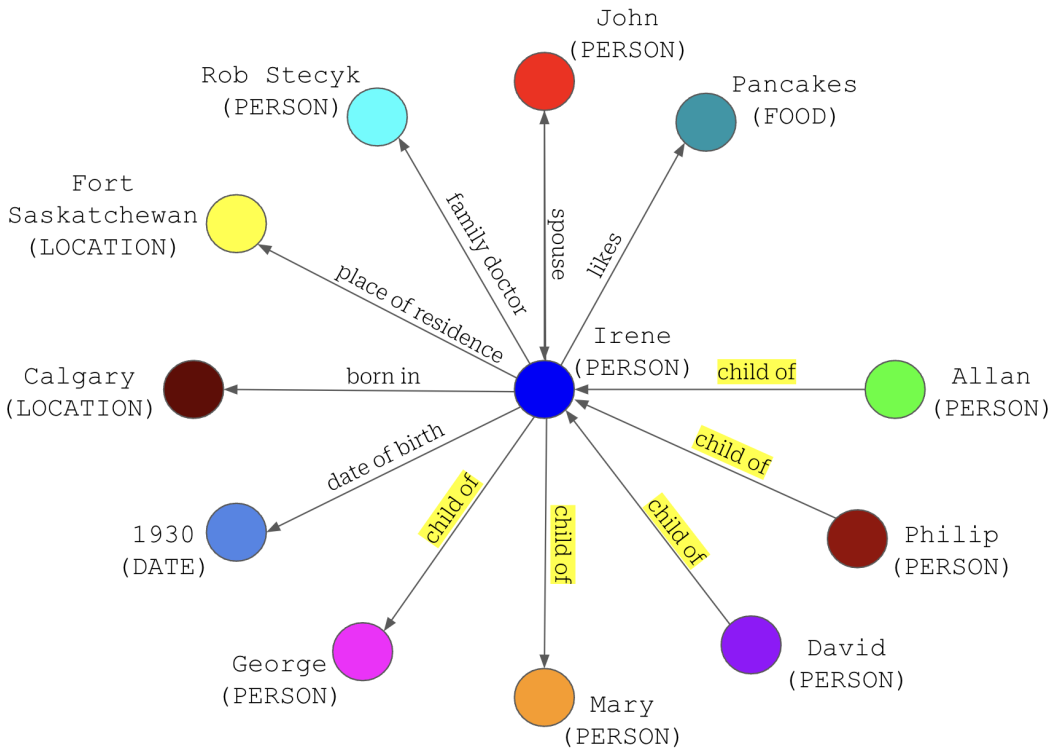


Figure 7.3: KG extracted from the conversation in Figure 1.2a by the proposed systems. All *child_of*-relations are newly added. Now the extractions are complete.

7.5 Acknowledgements

We would like to thank the annotators of the datasets described in this chapter for their hard work towards creating these datasets.

7.6 Conclusion

Previous works on data annotation indicated either that large amounts of data are necessary in order to achieve moderate model performance (see TACRED [97]) or data annotation, if done carefully, is extremely time-consuming (see KnowledgeNet [60] or BRAT [81]). We showed that it is possible to create high-quality datasets for RE for a variety of relations, with a moderate amount of time and effort, using freely available text data from Wikipedia. The resulting models trained on these datasets showed state-of-the-art results for RE and are robust when tested on datasets from different annotators than they were

trained on.

Annotated datasets are made publicly available for future research[11].

[11] https://github.com/mjstrobl/FREDA

Chapter 8

Knowledge Graph Question Answering

So far, we were mainly concerned with KGP and its subtasks, i.e. how to construct a KG. In order to access knowledge in our KG, a KG Question Answering (KGQA) system is suitable to retrieve such data. Figure 8.1 shows the workflow for answering a question from the user. The KGP system extracts and links the entity *Adam* and passes the question to the KGQA system[1]. The goal of the KGQA system is to extract a relation for the detected entity as subject or object and constructs a SPARQL query, selecting the missing entity, which can be used, in combination with the KG, to answer the question.

Therefore, in this chapter, we propose a method for KGQA in order to answer natural language questions from our KG.

8.1 Introduction

A Knowledge Graph (KG) provides a structured representation of textual knowledge. Facts in a KG are commonly stored using the Resource Description Framework (RDF)[2] as subject-relation-object triples. There are existing KGs, such as Wikidata [91] (maintained by humans) and DBpedia [52] (mainly derived from Wikipedia infoboxes), as well as approaches in order to automatically create a KG, denoted as Knowledge Graph Population (KGP), e.g. see

[1] In order to distinguish questions and statements, the system assumes questions to end with a question mark.

[2] `https://www.w3.org/RDF/`

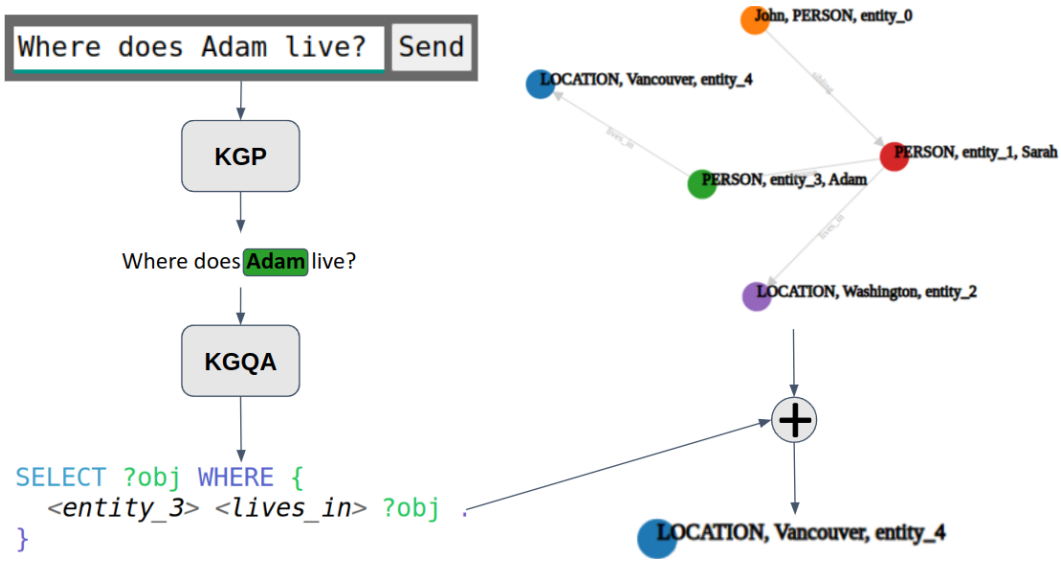


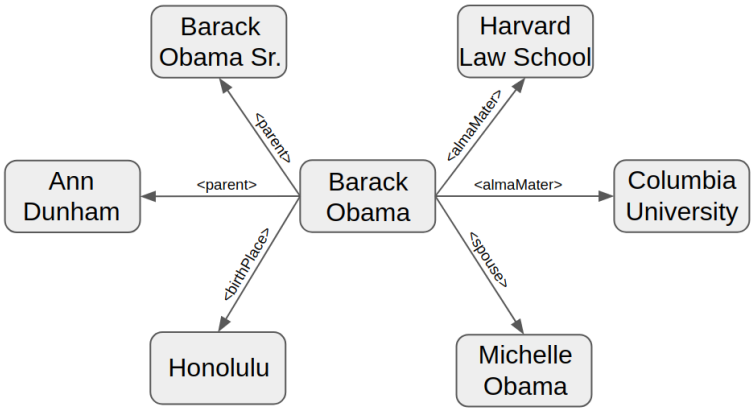
Figure 8.1: Workflow for answering questions using the Knowledge Graph extracted by our KGP system.

submissions to the TAC Cold Start KB/SF challenge[3], such as [13].

One advantage of knowledge representation in a KG using RDF over text is to be able to use SQL-like query languages, such as SPARQL[4]. Using SPARQL, one can retrieve information stored in the KG, i.e. finding answers for SPARQL-encoded questions, denoted as Knowledge Graph Question Answering (KGQA), e.g. see [90] or [22]. Figure 8.2 shows a sub-KG of DBpedia for the entity *Barack Obama*[5], including its RDF representation and a SPARQL query to answer the question: *Who is Barack Obama's spouse?*

Being able to query a KG with SPARQL provides a structured way to access knowledge from the KG, but may not be intuitive for users without domain knowledge. Therefore, approaches and datasets exist in order to translate natural language questions into SPARQL queries (or train models to do so), e.g. see [10]. However, these approaches only work on a specific KG they were developed/trained for. Specifically, the set of relations in one KG may not match the set of relations of another. While the overlap may be large for existing KGs, such Wikidata and DBpedia, a KG created from scratch

[3] https://tac.nist.gov/2017/KBP/ColdStart/index.html
[4] https://www.w3.org/TR/rdf-sparql-query/
[5] https://dbpedia.org/page/Barack_Obama




```
dbr:Barack_Obama  dbo:almaMater dbr:Harvard_Law_School .
dbr:Barack_Obama  dbo:almaMater dbr:Columbia_University .
dbr:Barack_Obama  dbo:spouse  dbr:Michelle_Obama .
dbr:Barack_Obama  dbo:birthPlace  dbr:Honolulu .
dbr:Barack_Obama  dbo:parent  dbr:Ann_Durham .
dbr:Barack_Obama  dbo:parent  dbr:Barack_Obama_Sr. .

SELECT ?obj WHERE {
  dbr:Barack_Obama dbo:spouse ?obj .
}
```

Figure 8.2: Sub-KG of DBpedia for the entity *Barack Obama*, including RDF representation and SPARQL query. DBpedia namespace prefixes are omitted.

through a KGP system, such as ours, may be domain-specific and may contain an entirely different set of relations. This really reduces the effectiveness of such approaches on newly created KGs.

Creating new manually labeled datasets for KGQA is a tedious task. Annotators have to come up either with a question to start with, for which a SPARQL query has to be written as well, or, in order to save time, they have to write questions for existing SPARQL queries, an approach followed by [24], which may or may not be meaningful. However, KGP approaches typically take advantage of existing Relation Extraction (RE) datasets, such as the TAC Relation Extraction Dataset (TACRED) [97] or our FREDA-derived datasets (see Chapter 7), to train models that are able to recognize relations in text. Now the question arises, whether it is possible to use the exact same datasets to recognize relations in questions as well, potentially with a minimum or no dedicated KGQA data, i.e. few-shot or zero-shot learning, respectively. Since such a KGQA system only needs to be aware of the relations that can be extracted by the KGP system, other datasets, even if containing questions for

more relations, are not needed. In this chapter, we investigate the possibility to use these existing RE datasets for KGQA.

Our contributions are the following:

- Since creating new KGQA datasets for a specific KG and its relations, populated by a KGP system, may not be feasible, we propose an approach to use existing RE datasets to extract entity-relation pairs, which can be combined to SPARQL queries, used for KGQA.
- A suitable model architecture as well as an evaluation based on an existing RE dataset, showing that such a model can effectively detect entity-relation pairs in natural language questions, even if no dedicated KGQA data (zero-shot learning) or small amounts of such data is included in the training procedure (few-shot learning).

The remainder of this chapter is structured as follows: Section 8.2 provides information about related work on KGQA. In Section 8.3 we show how to train models on KGP data and prepare them for KGQA with an evaluation in Section 8.4. The chapter is concluded in Section 8.5.

8.2 Related Work

There is a number of datasets available related to the problem of KGQA, which are presented in this section.

8.2.1 Simple Questions

The *SimpleQuestions* dataset [10] consists of 108,442 questions paired with a fact (subject-relation-object triple) from the Freebase KG [9]. A simple question is considered to contain either subject or object and a relation as supporting fact. For this dataset, objects are always considered to be the answer, i.e. subject and a mention of the relation can be found in each sample question. Table 8.1 shows four examples from *SimpleQuestions*. The corresponding SPARQL query would use the template *SELECT ?object WHERE*

{ <*subject*> <*relation*> *?object* . }. Therefore, if subject and relation can be retrieved from a simple question, such a SPARQL query is trivial to construct.

Where was **Guy Pnini** born?
What is the nationality of **Chris Gwynn**?
Who was the founder of **Honeywell**?
What's the name of a child of **Frank Gibney**?

Table 8.1: Examples from *SimpleQuestions* for the relations *born_in*, *nationality*, *founded* and *parent*. The subject is highlighted in bold.

The dataset was introduced due to a lack of large-scale datasets for Question Answering. The collection of the dataset involved two phases: (1) Selecting facts from Freebase with defined relation types, i.e. not containing the word *freebase*, and removing subject-relation pairs with more than 10 distinct objects, which could lead to uninformative questions. (2) Selected facts were presented to human annotators to write questions in natural language, which are as diverse as possible.

Each sample in the dataset consists of a question and a supporting fact from Freebase. Unfortunately, the subject is not annotated in the question itself. The dataset contains questions for 1,629 relations, although only 133 relations are represented more than 100 times and 563 relations are only mentioned once. A variety of commonly used relations was not considered at all: *spouse*, *date of birth/death*, *educated at*, *ceo of*, *sibling* and many more. In addition, Freebase is outdated as it is not developed anymore. It was merged into Wikidata [65].

8.2.2 Large Complex Question Answering Dataset

The Large-Scale Complex Question Answering Dataset (LC-QuAD) in its second revision (2.0 [24]; see [87] for the first one) is a dataset with 30,000 simple as well as complex questions (up from 5,000), their corresponding SPARQL queries and paraphrases. Queries for the Wikidata as well as DBpedia KG are attached[6]. Figure 8.2 shows four examples of questions involving more than a single fact from the KG.

[6]Wikidata was added to DBpedia and even though the relation scheme is slightly different for DBpedia, SPARQL queries can be automatically rewritten using known templates for

What is a cause of death that begins with the letter "p" and can be found on a CT scan?
Former champion Francisco Alarcon gave what award to Art Spiegelaman?
Located in the Central District, what is the county seat whose twin cities include Feodosiya?
What kind of career does Grigori Kozintsev have in the screenwriting field?

Table 8.2: Examples from *LC-QuAD 2.0* involving more than a single fact from the KG.

The annotation process starts with *vital* Wikipedia articles[7], i.e. Wikipedia's most important articles from a variety of topics. Then, question templates for the corresponding Wikidata/DBpedia entities are created manually considering multiple types of questions, simple as well as complex ones, e.g. involving multiple facts or temporal information for a relation. Entities and SPARQL templates are used to create SPARQL queries, which are presented to Amazon Mechanical Turk annotators, who are asked to write questions and paraphrases of those questions in a later step.

The LC-QuAD 2.0 dataset covers 22,792 entities and 3,627 relations. 2,374 of these relations are mentioned just once and only 76 are mentioned more than 100 times. Due to the fact that it relies on Wikidata/DBpedia KGs, similar issues with relation coverage exist as in the aforementioned datasets.

8.3 Method

In this section, we present our approach on how to construct SPARQL queries from entity-relation pairs and a model architecture to detect these pairs in natural language questions.

8.3.1 SPARQL Query Construction

Approaches aiming to translate natural language questions to SPARQL queries for a given KG, such as [90] or [22], consider specific SPARQL templates and a Slot Filling approach is used to fill the missing variables, i.e. find relations and

each relation.

[7] `https://en.wikipedia.org/wiki/Wikipedia:Vital_articles/Level/5`

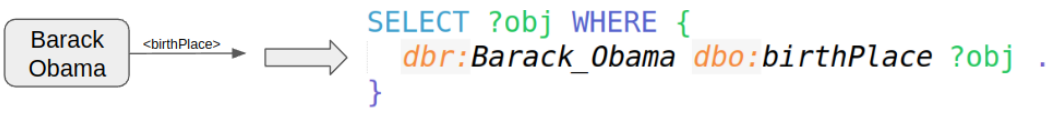


Figure 8.3: Relation Extraction and query translation for the simple question "Where was Barack Obama born?". DBpedia namespace prefixes are omitted.

link entities in the questions to their corresponding identifiers in the KG. In our case, we do not aim for linking entities to a KG. Instead, we assume the linking step has already occurred since this is part of the KGP pipeline. Existing datasets for KGQA, whether simple or complex questions are considered, may also include relations the KGP system is not able to extract. In order to still be able to test whether datasets for RE can be helpful for model training, we aim to predict parts of SPARQL queries, which contain relations that can be extracted by our KGP system. This way, even more complex queries can be written as no templates are used.

Figure 8.3 shows an example of a SPARQL query, which can potentially appear in a dataset, and the corresponding subject-relation tuple, which can be extracted by the KGQA system and used for translating the question into SPARQL. In advance, the entity *Barack Obama* is detected by the KGP system and linked to its node in the KG. In order to detect a relation, we consider each relation in the dataset, that can be extracted, as a unique class (including a *no-relation* class) and one entity in text as subject or object. If classified correctly, both, the KG identifier for the entity *Barack Obama* as well as the detected relation, can be added to the query. For more complex queries, involving multiple facts, a system like this would be able to add more subject-relation-object triples[8] to a SPARQL query. Finally, all missing subjects/objects are variables to be selected, e.g. Barack Obama's birthplace in Figure 8.3.

This approach is able to produce *SELECT* as well as *ASK*[9] SPARQL queries, the latter ones in case no subject or object is missing in the query triples. However, some simplifications have to be made: Queries containing

[8]Either subject or object would be considered as a variable, if not detected in a later step. All entities are considered as subject as well as object in order to predict relations and only positive, i.e. relations other than the *no-relation* class, are added to the query.

[9]ASK queries aim to answer yes/no-questions, e.g if two people are married.

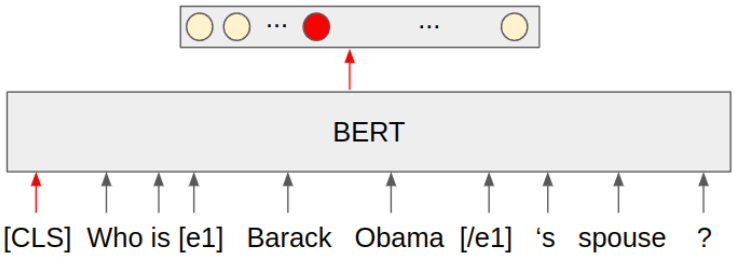


Figure 8.4: Model for extracting a relation for a marked entity as subject or object (see entity markers). Therefore, this models extracts an entity as subject or object for a detected relation (or no extraction, if no relation is detected). This entity-relation double can be plugged into a SPARQL query.

LIMIT, COUNT[10] and FILTER[11] are not producible this way, as well as Blank Nodes[12] cannot be considered. Although, it is still possible to test how RE datasets can help this way, which we show in Section 8.4.

8.3.2 Model Architecture

The problem at hand can be considered as a sentence classification problem. The model we considered is depicted in Figure 8.4, similar to what was considered in [79] for RE.

The subject or object of the question is marked with entity markers, *[e1]/[/e1]* and *[e2]/[/e2]*, respectively, to let the model know where the entity is, mainly relevant for complex questions, which could potentially contain more than one entity. But also the question may not always ask for the object of a relation, in which case the object itself would appear. The transformer-based [89] BERT model [21] is used to embed the question and a softmax-layer is used for relation classification considering the BERT's CLS-token as question embedding.

[10]LIMIT and COUNT are similar to the corresponding operators in SQL queries.
[11]FILTERs can be used to filter literals, i.e. strings, dates or numbers.
[12]Blank Nodes can be used to represent n-ary relations, e.g. if a marriage date is attached to a *spouse*-relation, see `https://www.w3.org/TR/rdf11-concepts/#section-blank-nodes`

8.4 Evaluation

In this section, we present the results when the aforementioned model architecture is trained on an RE dataset with various amounts of KGQA data, i.e. questions from two existing KGQA datasets, SimpleQuestions and LC-QuAD 2.0, their annotated subjects/objects and attached relations.

8.4.1 Baselines

Similar to the evaluation in Chapter 7, we focus on the comparison of datasets or, more specifically, the augmentation of existing datasets for KGQA using RE datasets annotated with our FREDA tool.

We use portions of various sizes of existing KGQA datasets, namely SimpleQuestions [10] and LC-QuAD 2.0 [24], to train models as baselines (see the architecture in Section 8.3.2) and test them on the corresponding test sets. These datasets are time-consuming to annotate for with the additional difficulty of how to achieve a large variety of questions from the same annotators. Therefore, we compare the baseline models and models additionally trained on data from existing non-KGQA sources, namely RE datasets manually annotated using our FREDA framework (see Chapter 7). Such datasets are simpler to annotate for and since they are already used to detect relations added to the KG, they already contain data for all relations of interest.

We use various amounts of questions from SimpleQuestions and LC-QuAD 2.0 in order to test how many questions per relation are really necessary. Assuming these datasets do not contain certain relations of interest, one needs to manually annotate questions for KGQA. The less manually annotated questions are needed to train models achieving good results, the less time is needed to produce these.

Therefore, these existing KGQA datasets are considered as baselines and adding RE datasets annotated with FREDA are used for augmenting these baseline dataset in order to train models that are better at predicting relations for KGQA.

8.4.2 Dataset Preparation

Even though TACRED is a commonly used dataset for RE, we consider our FREDA datasets, which we denote as *FREDA*, as multiple issues with TACRED were pointed out by [3] and [71], and FREDA datasets are used by our KGP system.

In addition, we consider the datasets SimpleQuestions, for simple questions with just a single fact involved, and the LC-QuAD 2.0 dataset for more complex questions with potentially multiple facts involved. However, both datasets have to be preprocessed in order to feed them into the model, which is described thereafter.

SimpleQuestions

Each sample in the SimpleQuestions dataset consists of a fact from Freebase (subject-relation-object) and a question in natural language asking for the object. Therefore, subject and relation are mentioned in the question. This already fits our models input of a question with a marked entity. However, the exact position of the entity is unknown and only a Freebase ID is attached. We used a Freebase-Wikidata mapping from Wikidata and an alias dictionary for Wikipedia entities, as provided by [84], in order to find alternative names of all such entities, which could appear in a question. This way we were able to annotate the actual entities in these questions. Four relations in SimpleQuestions are matching FREDA relations, namely *place_of_birth*, *nationality*, *child_of* and *founded*. All but *nationality* also appear as inverse relation in SimpleQuestions, which can be used as well.

In addition, a number of samples can be linked to the *no-relation* class for questions, which cannot be answered. The construction of a fully specified SPARQL query is trivial for SimpleQuestions. From the training set, we were able to extract 4,995 questions with annotated entities and relations attached as well as one no-relation example for each of these questions through labelling the detected entity opposite, i.e. subjects for a specific relation are labelled as objects and *no-relation*, and vice versa. From the SimpleQuestions test set,

we extracted 2,728 questions in total and 5,570 questions from FREDA for the aforementioned 4 relations.

LC-QuAD 2.0

This dataset mostly contains complex questions involving more than one fact from the KG. SPARQL queries as well as DBpedia queries are attached to each sample. We cannot claim to produce a full SPARQL query for each of them since some questions mention relations, which cannot be detected by our KGP system. In order to still take advantage of these samples, we only aim to predict those facts mentioned in the query, which contain a relation that can be predicted as well as at least one entity, either subject or object. Even though full SPARQL queries may not be written this way, it is still possible to test whether an RE dataset for KGP can help answering questions.

Again, entities in the question (multiple are possible) are not annotated, they have to be retrieved through string matching (Wikidata IDs leading to entity names are attached). This was not possible for all question, but we extracted 10,107 questions and paraphrased questions for 12 relations, which can be found in FREDA as well: *place_of_residence*, *place_of_birth*, *place_of_death*, *nationality*, *educated_at*, *child_of*, *spouse*, *headquarters*, *subsidiary_of*, *founded*, *ceo_of*, *sibling* and *award*. 7,067 *no-relation* questions are included through adding the opposite entity annotation, where applicable[13]. From the LC-QuAD 2.0 test set, we were able to extract 4,290 questions in total for all 12 relations and 21,550 questions from FREDA for the same relation set, used for training.

8.4.3 Model training

We use the model architecture depicted in Figure 8.4 and the BERT large cased model to embed questions. The hyperparameters were set as follows: Learning rate of $5 * 10^{-5}$ (linear decay), Adam optimizer [46], batch size 8, 1-5 epochs (varies per relation; determined using 5-fold cross-validation). The

[13]The relations *spouse* and *sibling* are bidirectional, therefore the opposite annotation is added as annotation with the same relation.

test sets of both KGQA datasets, SimpleQuestions and LC-QuAD 2.0, were used for testing models.

8.4.4 Test results

Table 8.3 shows test results for models trained on FREDA plus various amounts of dedicated questions from the preprocessed KGQA datasets SimpleQuestions and LC-QuAD 2.0. This was compared to training solely on these various amounts of questions from KGQA datasets. The corresponding preprocessed test sets were used to report Precision, Recall and F1.

In both cases, all extracted sentences from FREDA were used for model training and 0, 10, 25, 50, 100 (per relation) or all questions were added from KGQA datasets (4,995 questions from SimpleQuestions and 10,107 from LC-QuAD 2.0). The point of this experiment is to see whether any dedicated KGQA data is needed at all and how model performance is affected if a minimal amount is added. Ideally, no KGQA data should be necessary to add, since for some relations, depending on the underlying KGP system, there may be no annotated data available. However, it may be possible to annotate/write a few questions per relation relatively quickly and it would be interesting to know how model performance can be improved through adding a certain number of annotated questions per relation.

SimpleQuestions

Even if no questions from the SimpleQuestions training set were added to FREDA data, an F1 of 0.74 clearly indicates that an RE dataset can be used to detect an entity-relation pair from a natural language question relatively well. Adding questions to the training data, even just 10 per relation, increases model performance by a large margin, but it seems that adding more questions than 10 per relation, may not result in large improvements anymore. Training on questions only (no FREDA datasets; see *SimpleQuestions* column in Table 8.3), model performance is always significantly lower than when FREDA data is added or used solely. Except, FREDA data does not help anymore if all questions from SimpleQuestions are used.

	SimpleQuestions			+ FREDA		
Questions	**P**	**R**	**F1**	**P**	**R**	**F1**
0	–	–	–	0.93	0.62	0.74
10	0.39	0.77	0.52	0.95	1.0	**0.97**
25	0.32	0.85	0.46	0.95	1.0	**0.97**
50	0.50	0.97	0.66	0.95	1.0	**0.97**
100	0.52	0.95	0.67	0.94	1.0	**0.97**
All	0.98	1.0	**0.99**	0.98	1.0	**0.99**
	LC-QuaD 2.0			**+ FREDA**		
Questions	**P**	**R**	**F1**	**P**	**R**	**F1**
0	–	–	–	0.86	0.86	0.86
10	0.25	1.0	0.39	0.82	0.95	**0.88**
25	0.47	1.0	0.64	0.83	0.96	**0.89**
50	0.54	1.0	0.70	0.82	0.97	**0.89**
100	0.56	0.96	0.70	0.83	0.98	**0.90**
All	0.93	0.96	0.94	0.94	0.97	**0.95**

Table 8.3: Test results for models trained on FREDA including no or various amounts of questions from SimpleQuestions and LC-QuAD 2.0 as well as results for models trained without FREDA data. Precision, Recall and F1 are reported for models tested on the corresponding KGQA datasets test sets.

LC-QuAD 2.0

For LC-QuAD 2.0, results are very similar (see *LC-QuAD 2.0* and the following column in Table 8.3). When trained on questions data only, satisfying model performance can only be achieved through training on all questions. Whereas training solely on FREDA data already achieves and F1 of 0.86 with minor improvements when questions from LC-QuAD 2.0 are added. The more complex questions from LC-QuAD 2.0 are longer than in SimpleQuestions and are, presumably, syntactically more similar to sentences found in FREDA, therefore adding questions data does not result in improvements as large as in the SimpleQuestions case. In this case, FREDA slightly improves test results when all questions are used for training.

For both KGQA datasets, these results indicate that it is indeed possible to use RE datasets, used for creating a KG, to train a model that can predict entity-relation pairs in order to construct SPARQL queries.

8.4.5 Examples

Consider the conversation and resulting KG from Figure 1.2, which has been created by the systems described in the Chapters 5, 6 and 7. Using the approach proposed in this chapter, the model trained with the aforementioned *LC-QuAD 2.0+FREDA* (all) dataset and the Entity Recognition capabilities of the approaches described in Chapters 5 and 6, we can answer the questions in Table 8.4 based on the given KG.

The described KGQA system uses KGP (see Chapter 5) in order to detect entities, for which relations are predicted. The entity in the questions is then added as subject or object (both are tested[14]) for the model and a relation (or no relation) is predicted and converted into a SPARQL query (see Figure 8.1 for the actual workflow). This query was then applied to the KG at hand in order to retrieve the answers.

Question	**KGQA**	**Answer**
Who are Irene's children?	? child_of Irene	Allan, Philip, David
Who are Irene's parents?	Irene child_of ?	George, Mary
Who is Irene married to?	Irene spouse ?	John
Where was Irene born?	Irene place_of_birth ?	Calgary
Where does Irene live?	Irene place_of_residence ?	Fort Saskatchewan

Table 8.4: Questions, corresponding KGQA extractions (that lead to SPARQL queries), and answers from the KG extracted from the conversation in Figure 1.2a.

8.5 Conclusion

In this chapter, we showed how our FREDA datasets, not dedicated to predict relations in questions, can be used to write SPARQL queries, which can be applied to an RDF-based KG. While it is possible to use existing KGQA datasets, e.g. SimpleQuestions and LC-QuAD 2.0, to train models to do so on large KGs, such as Wikidata and DBpedia, it may not be possible to find such datasets for other KGs. Specifically, if a KGP system to create a KG from scratch is used, the set of relations found in existing datasets may not match

[14]This is irrelevant for the *spouse*-relation as it is bidirectional; the model predicted the relation for *Irene* as subject as well as object, both leading to the correct result.

the ones extracted by the system. Therefore, either potentially large amounts of questions have to be written and labelled with a corresponding SPARQL query, which may not be feasible, or the proposed techniques can be used to take advantage of RE datasets, which are already used by the KGP system to extract relations to add to the KG. We showed examples of questions which are converted into SPARQL queries and answered for the KG created in previous chapters.

The SPARQL queries, which can be created through the entity-relations pairs predicted by our models, are still limited as the proposed approach is not able to add blank nodes, filters, counts or limits. For future work, we would like to adjust the model to be able to predict these missing SPARQL functionalities.

Chapter 9

Conclusion

This chapter aims to summarize the key findings and main contributions of this thesis regarding the research questions, as described in Chapter 1, followed by a discussion and opportunities for future work.

9.1 Key Findings

We presented a general KGP system for conversations. It follows a pipelined approach with the subtasks NER, EL and RE. A KGQA system is added in order to be able to translate natural language questions into SPARQL queries, which can be applied to the resulting KG. The focus of the subtasks NER, with its classes, and RE, with its set of relations, lays on the interest of elderly people, mainly in the areas of Family/Friends, Health, and Nutrition. In the following, we present the key findings of this thesis.

We described the necessity of approaches that are able to provide meaningful text data, which can be manually annotated, and reduce the cognitive load for the annotator as much as possible. Our tool FREDA and, specifically, the datasets we manually annotated for the task of RE provide a possibility to create manually annotated datasets for the NLP tasks of interest in a less time-consuming way than before. The models trained on the resulting RE datasets for 19 relations with at least 500 fully annotated sentences per relation show good performance on the corresponding test sets as well as on other datasets, namely KnowledgeNet and CRE.

NER models typically only detect the classes of entities that can be found

in the datasets they were trained on. Whenever a new class should be added, a completely new dataset needs to be acquired with exhaustive annotations of all entities and all classes, since the models commonly used for NER would not necessarily work well on, for example, partially annotated datasets. However, these partially annotated datasets are much easier to come by and we showed an approach to derive these kind of datasets for classes of interest from the Wikipedia category hierarchy with a limited amount of time and effort. In addition, we showed three approaches to train models on existing NER datasets, namely CoNLL 2003, combined with datasets derived from Wikipedia for the classes *Food* and *Drugs*. In order to test our approach, we manually annotated 500 sentences per class.

Due to the lack of complete systems for demonstration and testing purposes for KGP, we developed a web-based system, which is capable of all KGP subtasks and keeping track of the state of the conversation in a KG. Each module can be replaced or extended, e.g. through adding rules for NER or RE. It is suitable for accepting utterances by the user and display the current state of the KG.

One way of retrieving information from the KG, e.g. for validation purposes, is to ask questions in natural language and use the KG to retrieve an answer, which is done using SPARQL queries. There are existing datasets for translating such questions into SPARQL queries for the Wikidata, DBpedia or Freebase KGs, which are not suitable for our KG due to the different set of relations stored. In order to avoid the manual creation of a new datasets matching our KG, we developed an approach, which is capable of re-using the aforementioned RE datasets, to help translate natural language questions into SPARQL queries.

9.2 Main Contributions

These are the main contributions of this thesis, including the corresponding codebases, which are publicly available:

- KGP from Conversations[1]: A proposal of a complete KGP system, including NER, EL and RE.

- Wikipedia EXhaustive Entity Annotations (WEXEA)[2]: WEXEA is an approach aiming to add additional annotations to Wikipedia dumps for the languages English, German, French, and Spanish. We showed how datasets for the tasks NER, EL, CR and RE (using Distant Supervision) can be extracted.

- Flexible Relation Extraction Data Annotation (FREDA)[3]: This tool allows the users to create manually annotated datasets for a variety NLP tasks, quickly and accurately. Specifically, for the task of RE, we showed that text data from WEXEA can help significantly speeding up manual data annotations.

- NER for Partially Annotated Datasets[4]: We described a procedure on how to create partially annotated datasets for new classes derived from Wikipedia categories semi-automatically. Training strategies for NER models on such datasets were compared and we released two manually annotated datasets of 500 sentences each for the classes *Food* and *Drugs* in order to test how generalizable our data extraction techniques are.

- Knowledge Graph Question Answering (KGQA)[5]: This is an approach, which is capable of taking advantage of ordinary RE datasets, used for KGP, in order to translate natural language questions into SPARQL queries, which can be applied to a KG. A suitable model architecture to achieve this task as well as an evaluation based on an existing RE dataset, showing the effectiveness, is included.

[1] `https://github.com/mjstrobl/KGP_from_Conversations`

[2] Codebase as well as multilingual datasets: `https://github.com/mjstrobl/WEXEA`

[3] Codebases and datasets for FREDA are available at `https://github.com/mjstrobl/FREDA`.

[4] Codebase and datasets: `https://github.com/mjstrobl/NER_for_partially_annotated_data`

[5] `https://github.com/mjstrobl/KGQA_using_KG_Construction_Data`

9.3 Future Work

For future work, we identified a number of potential improvements to the aforementioned contributions as well as the usefulness of a KGP system for downstream NLP tasks:

- Even though WEXEA can be applied to the English, German, French, and Spanish Wikipedia, many other languages are currently not supported. This is due to the fact that certain Wikipedia keywords are language-specific, i.e. they have to be translated, as well as the CoreNLP NER used by WEXEA does not support all of these languages. Therefore, the barrier of language-dependent keywords has to be removed and a more universal NER model, which can be trained on more languages should be used instead of CoreNLP.

- Similarly, through adding more partially annotated datasets for the NER subtask, more entity classes can be detected. In addition, there is an obvious gap between model performance on these partially annotated datasets and the corresponding manually annotated ones. This gap has to be further investigated and narrowed, if possible.

- We are using a fixed set of relations of our KGP system, therefore more relations with rules and models trained on datasets created using our FREDA framework would be beneficial.

- Our KGQA system is currently not able to process certain SPARQL queries, such as COUNT, LIMIT or queries including FILTERs. It needs to be adapted to be able to return general SPARQL queries including these.

- People with dementia, which especially elderly people are affected by, may often mention similar or the same facts. If such facts can be detected by the KGP system, it may be able to contribute detecting signs of dementia.

- As mentioned in Chapter 1, a Response Generation system could take advantage of the extracted KG. Typically sequence-to-sequence models are used for such systems, which read previous utterances and generate a new one. However, the older an utterance is, important to generate the next one, the less likely it is to be taken into account. Therefore, a structured KG could provide this information continuously and help producing more interesting responses.

9.4 Limitations

Our approach for KGP has a number of known limitations, which are addressed in this section.

9.4.1 Triple-based Storage of Facts

The triple-based, yet commonly used approach, of storing facts in a KG based on RDF has the limitation that only a single object per relation can be extracted. This can be illustrated by the following example:

> "John lives in Vancouver since Jun 8, 2022."

From this sentence our KGP approach is able to extract the relation *place_of_residence* with *John* as subject and *Vancouver* as object. However, the date as second object is ignored in this case. This issue is caused by the fact that our RE approach is limited to extracting only two arguments per relation, i.e. it can be considered as binary RE, whereas an n-ary RE approach would be able to additionally cover cases where the relation has less or more than two arguments. In addition, RDF itself is triple-based, even though so-called *Blank Nodes* could be used to represent relations with various numbers of objects, i.e. multiple triples can be added for a single relation, which is shown by Figure 9.1.

There is existing work on n-ary RE, e.g. shown in [66] for a medical dataset based on Distant Supervision. The models our system uses, see Figure 7.1, could be adjusted to accept more entity markers as well. However, the lack of

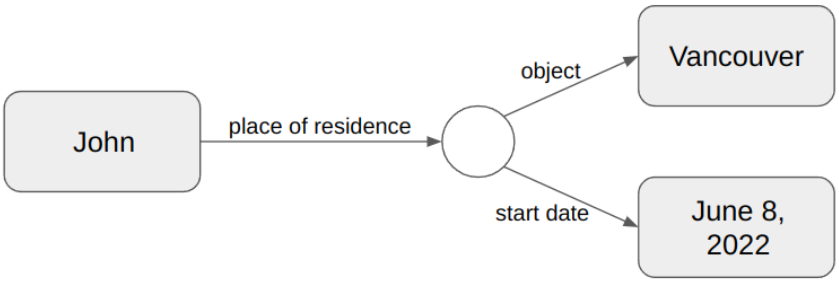


Figure 9.1: Corresponding RDF representation with a Blank Node.

high-quality datasets for RE remains. This is even more challenging to address in case of n-ary relations.

9.4.2 Maintenance of Existing Facts

As mentioned in Section 5.4, existing facts in the KG are not revised by the proposed approaches. This could lead to inconsistencies if the entirety of extracted facts, i.e. the current KG, is considered as the truth at any given moment. There are two ways to handle this: (1) Replace facts in case they are not compatible with existing facts. (2) Adding a timestamp for each relation, which can be realized through Blank Nodes.

In the latter case, a system using the resulting KG needs to be able to understand that certain facts can be seen as updates of older facts, either because relations of a person, e.g. *place_of_residence*, can change over time or the KG may contain incorrect facts that were corrected later on. A Response Generation or Dialogue System, which is able to actively asking the user, i.e. adding interactivity to the system, in case a contradiction is detected, can resolve these facts and help revising the KG.

9.4.3 Downstream Tasks

In the context of this work, the ultimate goal is a chatbot capable of chitchatting. We provided approaches, which lead to the extraction of facts from utterances, or text data in general, in order to build a KG. This KG can be considered as a long-term memory, which can, for example, be used for downstream tasks, such as a Dialogue System. Typically, these systems follow a

sequence-to-sequence modelling approach, reading the history of the dialogue and producing new utterances, e.g. as described in [76]. Such a KG can provide important information of related entities to such a system without forgetting as there is no guarantee that the model's encoder[6] captures this information from the history of utterances. It is beyond the scope of this work to show the usefulness of such a KG for downstream tasks other than KGQA, e.g. Dialogue Systems.

[6]The encoder of a sequence-to-sequence model encodes the history of utterances, which is fed into the decoder, producing a new utterance.

References

[1] A. Akbik, T. Bergmann, and R. Vollgraf, "Pooled contextualized embeddings for named entity recognition," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 724–728.

[2] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 1638–1649.

[3] C. Alt, A. Gabryszak, and L. Hennig, "TACRED revisited: A thorough evaluation of the TACRED relation extraction task," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 1558–1569. DOI: 10.18653/v1/2020.acl-main.142. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.142.

[4] C. Alt, M. Hübner, and L. Hennig, "Improving relation extraction by pre-trained language representations," in *Automated Knowledge Base Construction (AKBC)*, 2018.

[5] G. Angeli, M. J. J. Premkumar, and C. D. Manning, "Leveraging linguistic structure for open domain information extraction," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 344–354.

[6] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, ser. ISWC'07/ASWC'07, Busan, Korea: Springer-Verlag, 2007, pp. 722–735, ISBN: 3540762973.

[7] ——, "Dbpedia: A nucleus for a web of open data," in *The semantic web*, Springer, 2007, pp. 722–735.

[8] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann, "Dbpedia-a crystallization point for the web of data," *Web Semantics: science, services and agents on the world wide web*, vol. 7, no. 3, pp. 154–165, 2009.

[9] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, AcM, 2008, pp. 1247–1250.

[10] A. Bordes, N. Usunier, S. Chopra, and J. Weston, "Large-scale simple question answering with memory networks," *arXiv preprint arXiv:1506.02075*, 2015.

[11] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer networks and ISDN systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[12] S. Brody, S. Wu, and A. Benton, "Towards realistic few-shot relation extraction," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 5338–5345.

[13] A. T. Chaganty, A. Paranjape, J. Bolton, M. Lamm, J. Lei, A. See, K. Clark, Y. Zhang, P. Qi, and C. D. Manning, "Stanford at tac kbp 2017: Building a trilingual relational knowledge graph," in *TAC*, 2017.

[14] N. Chambers, D. Cer, T. Grenager, D. Hall, C. Kiddon, B. MacCartney, M.-C. De Marneffe, D. Ramage, E. Yeh, and C. D. Manning, "Learning alignments and leveraging natural logic," in *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, Association for Computational Linguistics, 2007, pp. 165–170.

[15] A. X. Chang and C. D. Manning, "Sutime: A library for recognizing and normalizing time expressions.," in *Lrec*, vol. 2012, 2012, pp. 3735–3740.

[16] ——, "TokensRegex: Defining cascaded regular expressions over tokens," Department of Computer Science, Stanford University, Tech. Rep. CSTR 2014-02, 2014.

[17] J. P. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," *Transactions of the association for computational linguistics*, vol. 4, pp. 357–370, 2016.

[18] K. Clark and C. D. Manning, "Entity-centric coreference resolution with model stacking," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 1405–1415.

[19] ——, "Deep reinforcement learning for mention-ranking coreference models," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 2256–2262.

[20] E. F. Codd, "Seven steps to rendezvous with the casual user," in *IFIP Working Conference Data Base Management*, 1974.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[22] D. Diomedi and A. Hogan, *Question answering over knowledge graphs with neural machine translation and entity linking*, 2021. DOI: 10.48550/ARXIV.2107.02865.

[23] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. M. Strassel, and R. M. Weischedel, "The automatic content extraction (ace) program-tasks, data, and evaluation.," in *Lrec*, Lisbon, vol. 2, 2004, p. 1.

[24] M. Dubey, D. Banerjee, A. Abdelkawi, and J. Lehmann, "Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia," in *Proceedings of the 18th International Semantic Web Conference (ISWC)*, Springer, 2019.

[25] N. Dziri, E. Kamalloo, K. Mathewson, and O. R. Zaiane, "Augmenting neural response generation with context-aware topical attention," in *Proceedings of the First Workshop on NLP for Conversational AI*, 2019, pp. 18–31.

[26] H. Elsahar, P. Vougiouklis, A. Remaci, C. Gravier, J. Hare, F. Laforest, and E. Simperl, "T-rex: A large scale alignment of natural language with knowledge base triples," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[27] A. Fader, S. Soderland, and O. Etzioni, "Identifying relations for open information extraction," in *Proceedings of the conference on empirical methods in natural language processing*, Association for Computational Linguistics, 2011, pp. 1535–1545.

[28] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the 43rd annual meeting on association for computational linguistics*, Association for Computational Linguistics, 2005, pp. 363–370.

[29] O.-E. Ganea, M. Ganea, A. Lucchi, C. Eickhoff, and T. Hofmann, "Probabilistic bag-of-hyperlinks model for entity linking," in *Proceedings of the 25th International Conference on World Wide Web*, 2016, pp. 927–938.

[30] T. Gao, X. Han, H. Zhu, Z. Liu, P. Li, M. Sun, and J. Zhou, "Fewrel 2.0: Towards more challenging few-shot relation classification," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 6250–6255.

[31] P. Gärdenfors, *Belief revision*, 29. Cambridge University Press, 2003.

[32] J. Geiß, A. Spitz, and M. Gertz, "Neckar: A named entity classifier for wikidata," in *International Conference of the German Society for Computational Linguistics and Language Technology*, Springer, 2017, pp. 115–129.

[33] A. Ghaddar and P. Langlais, "Coreference in wikipedia: Main concept resolution," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016, pp. 229–238.

[34] ——, "Winer: A wikipedia annotated corpus for named entity recognition," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2017, pp. 413–422.

[35] ——, "Transforming wikipedia into a large-scale fine-grained entity type corpus," in *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*, 2018.

[36] Z. Guo and D. Barbosa, "Robust named entity disambiguation with random walks," *Semantic Web*, vol. 9, no. 4, pp. 459–479, 2018.

[37] N. Gupta, S. Singh, and D. Roth, "Entity linking via joint encoding of types, descriptions, and context," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 2681–2690.

[38] X. Han, H. Zhu, P. Yu, Z. Wang, Y. Yao, Z. Liu, and M. Sun, "Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 4803–4809.

[39] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum, "Yago2: A spatially and temporally enhanced knowledge base from wikipedia," *Artificial Intelligence*, vol. 194, pp. 28–61, 2013.

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

[41] S. Huang, S. Strassel, A. Mitchell, and Z. Song, "Shared resources for multilingual information extraction and challenges in named entity annotation," in *Proceedings of the IJCNLP-04 Workshop on Named Entity Recognition for NLP Applications*, 2004, pp. 112–119.

[42] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[43] H. Ji, R. Grishman, and H. T. Dang, "Overview of the tac2011 knowledge base population track," in *Third text analysis conference*, 2010.

[44] Z. Jie, P. Xie, W. Lu, R. Ding, and L. Li, "Better modeling of incomplete annotations for named entity recognition," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 729–734.

[45] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "Spanbert: Improving pre-training by representing and predicting spans," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 64–77, 2020.

[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[47] M. Klang and P. Nugues, "Linking, searching, and visualizing entities in wikipedia," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[48] N. Kolitsas, O.-E. Ganea, and T. Hofmann, "End-to-end neural entity linking," in *Proceedings of the 22nd Conference on Computational Natural Language Learning*, 2018, pp. 519–529.

[49] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," *arXiv preprint arXiv:1603.01360*, 2016.

[50] H. Lee, A. Chang, Y. Peirsman, N. Chambers, M. Surdeanu, and D. Jurafsky, "Deterministic coreference resolution based on entity-centric, precision-ranked rules," *Computational linguistics*, vol. 39, no. 4, pp. 885–916, 2013.

[51] K. Lee, L. He, and L. Zettlemoyer, "Higher-order coreference resolution with coarse-to-fine inference," *arXiv preprint arXiv:1804.05392*, 2018.

[52] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, *et al.*, "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic web*, vol. 6, no. 2, pp. 167–195, 2015.

[53] X. Ling and D. S. Weld, "Fine-grained entity recognition," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012, pp. 94–100.

[54] R. Logan, N. F. Liu, M. E. Peters, M. Gardner, and S. Singh, "Barack's wife hillary: Using knowledge graphs for fact-aware language modeling," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 5962–5971.

[55] X. Ma, N. Fauceglia, Y.-c. Lin, and E. Hovy, "Cmu system for entity discovery and linking at tac-kbp 2017," in *TAC*, 2017.

[56] F. Mahdisoltani, J. Biega, and F. M. Suchanek, "Yago3: A knowledge base from multilingual wikipedias," in *Conference on Innovative Data Systems Research*, 2015.

[57] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 2014, pp. 55–60.

[58] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, pp. 55–60. [Online]. Available: `http://www.aclweb.org/anthology/P/P14/P14-5010`.

[59] S. Mayhew, S. Chaturvedi, C.-T. Tsai, and D. Roth, "Named entity recognition with partially annotated training data," in *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, 2019, pp. 645–655.

[60] F. Mesquita, M. Cannaviccio, J. Schmidek, P. Mirza, and D. Barbosa, "Knowledgenet: A benchmark dataset for knowledge base population," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 749–758.

[61] G. A. Miller, "Wordnet: A lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[62] M. J. Minock, "A step towards realizing codd's vision of rendezvous with the casual user," in *VLDB*, 2007.

[63] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervision for relation extraction without labeled data," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, Association for Computational Linguistics, 2009, pp. 1003–1011.

[64] J. Nothman, J. R. Curran, and T. Murphy, "Transforming wikipedia into named entity training data," in *Proceedings of the Australasian Language Technology Association Workshop 2008*, 2008, pp. 124–132.

[65] T. Pellissier Tanon, D. Vrandečić, S. Schaffert, T. Steiner, and L. Pintscher, "From freebase to wikidata: The great migration," in *Proceedings of the 25th international conference on world wide web*, 2016, pp. 1419–1428.

[66] N. Peng, H. Poon, C. Quirk, K. Toutanova, and W.-t. Yih, "Cross-sentence n-ary relation extraction with graph lstms," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 101–115, 2017.

[67] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[68] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of NAACL-HLT*, 2018, pp. 2227–2237.

[69] M. E. Peters, M. Neumann, R. Logan, R. Schwartz, V. Joshi, S. Singh, and N. A. Smith, "Knowledge enhanced contextual word representations," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 43–54.

[70] S. Riedel, L. Yao, and A. McCallum, "Modeling relations and their mentions without labeled text," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2010, pp. 148–163.

[71] S. Rosenman, A. Jacovi, and Y. Goldberg, "Exposing shallow heuristics of relation extraction models with challenge data," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 3702–3710.

[72] E. F. Sang and F. De Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," *arXiv preprint cs/0306050*, 2003.

[73] V. Sanh, T. Wolf, and S. Ruder, "A hierarchical multi-task approach for learning embeddings from semantic tasks," *arXiv preprint arXiv:1811.06031*, 2018.

[74] ——, "A hierarchical multi-task approach for learning embeddings from semantic tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6949–6956.

[75] M. Schmitz, R. Bart, S. Soderland, O. Etzioni, *et al.*, "Open language learning for information extraction," in *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, Association for Computational Linguistics, 2012, pp. 523–534.

[76] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau, "Building end-to-end dialogue systems using generative hierarchical neural network models," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[77] Y. Shao, S. Gouws, D. Britz, A. Goldie, B. Strope, and R. Kurzweil, "Generating high-quality and informative conversation responses with sequence-to-sequence models," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 2210–2219.

[78] A. Sil, G. Kundu, R. Florian, and W. Hamza, "Neural cross-lingual entity linking," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[79] L. B. Soares, N. FitzGerald, J. Ling, and T. Kwiatkowski, "Matching the blanks: Distributional similarity for relation learning," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 2895–2905.

[80] V. I. Spitkovsky and A. X. Chang, "A cross-lingual dictionary for english wikipedia concepts," 2012.

[81] P. Stenetorp, S. Pyysalo, G. Topić, T. Ohta, S. Ananiadou, and J. Tsujii, "Brat: A web-based tool for nlp-assisted text annotation," in *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, 2012, pp. 102–107.

[82] M. Strobl, A. Trabelsi, and O. Zaiane, "Enhanced entity annotations for multilingual corpora," in *Proceedings of the Language Resources and Evaluation Conference*, Marseille, France: European Language Resources Association, Jun. 2022, pp. 3732–3740. [Online]. Available: `https://aclanthology.org/2022.lrec-1.398`.

[83] ——, "Freda: Flexible relation extraction data annotation," *arXiv preprint arXiv:2204.07150*, 2022.

[84] M. Strobl, A. Trabelsi, and O. R. Zaiane, "Wexea: Wikipedia exhaustive entity annotation," in *Proceedings of The 12th Language Resources and Evaluation Conference*, 2020, pp. 1951–1958.

[85] M. Strobl, A. Trabelsi, and O. Zaïane, "Named entity recognition for partially annotated datasets," in *Natural Language Processing and Information Systems*, P. Rosso, V. Basile, R. Martínez, E. Métais, and F. Meziane, Eds., Cham: Springer International Publishing, 2022, pp. 299–306, ISBN: 978-3-031-08473-7.

[86] E. F. Tjong Kim Sang, "Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition," in *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*, 2002. [Online]. Available: `https://www.aclweb.org/anthology/W02-2024`.

[87] P. Trivedi, G. Maheshwari, M. Dubey, and J. Lehmann, "Lc-quad: A corpus for complex question answering over knowledge graphs," in *International Semantic Web Conference*, Springer, 2017, pp. 210–218.

[88] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: `https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

[89] ——, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[90] D. Vollmers, R. Jalota, D. Moussallem, H. Topiwala, A.-C. Ngonga Ngomo, and R. Usbeck, "Knowledge graph question answering using graph-pattern isomorphism," in *Further with Knowledge Graphs*, IOS Press, 2021, pp. 103–117.

[91] D. Vrandečić, "Wikidata: A new platform for collaborative data collection," in *Proceedings of the 21st international conference on world wide web*, 2012, pp. 1063–1064.

[92] T. Winograd, "Understanding natural language," *Cognitive psychology*, vol. 3, no. 1, pp. 1–191, 1972.

[93] D. Yu, K. Sun, C. Cardie, and D. Yu, "Dialogue-based relation extraction," *arXiv preprint arXiv:2004.08056*, 2020.

[94] ——, "Dialogue-based relation extraction," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 4927–4940. DOI: 10.18653/v1/2020.acl-main.444. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.444.

[95] X. Zeng, D. Zeng, S. He, K. Liu, and J. Zhao, "Extracting relational facts by an end-to-end neural model with copy mechanism," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 506–514.

[96] Y. Zhang, A. T. Chaganty, A. Paranjape, D. Chen, J. Bolton, P. Qi, and C. D. Manning, "Stanford at TAC KBP 2016: Sealing pipeline leaks and understanding chinese.," in *TAC*, 2016.

[97] Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning, "Position-aware attention and supervised data improve slot filling," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 35–45.