TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Social Relationship Extraction from Natural Language Conversations

Project report 2

As part of the Research Master's program

Applied Research in Engineering Sciences

Author:	Markus Glas
Course of studies:	Applied Research in Engineering Sciences
Location:	Nuremberg, Germany

Abstract

Digital dialog systems often lack in the sense of a human-like and individualized interaction. Responses are often predefined and user utterances are treated following similar processes, regardless someones personality or social environment. Extracting and storing social relationships during human-computer conversations can help dialog systems to learn someones social network and therefore create more contextual-related and personalized responses.

In this work, an approach for extracting and storing social relationships from natural language conversations is being presented. After an introduction to the current state in information extraction and database systems for representing social networks, the implemented method is being described in detail. The method is split into two components using the grammatical dependency path and pattern-based rules, together with state-of-the-art NLP models for entity recognition and representing semantic word vectors. Relationships are stored into a database for permanently access through a dialog systems or similar, and meant to represent the social network of a user, learned throughout the conversation.

The performance of the new approach is being evaluated on a conversational dataset, built as a combination of crowdworker conversations and TV show subtitles. It turns out that the combination of grammatical and semantic features can handle misspellings or grammatical errors within conversations, and successfully extract the social relationships.

Contents

List of Figures III

1 Introduction 1
- 1.1 Research Questions . . . 1
- 1.2 Aim of thesis . . . 2
- 1.3 Thesis Outline . . . 2

2 Information Extraction 3
- 2.1 Definition . . . 3
- 2.2 Named Entity Recognition . . . 3
  - 2.2.1 Rule-based Entity Recognition . . . 5
  - 2.2.2 Machine Learning for Entity Recognition . . . 5
- 2.3 Relation Extraction . . . 6
  - 2.3.1 Definition of Relation Extraction . . . 6
  - 2.3.2 Approaches towards Relation Extraction . . . 7
- 2.4 Word Embeddings . . . 11
- 2.5 Dependency Parsing . . . 12
  - 2.5.1 Transition-based Parsing . . . 12
  - 2.5.2 Neural Network based Parsing . . . 14

3 Storing Social Relationships 16
- 3.1 RDF Triple Stores . . . 16
- 3.2 Graph Database Systems . . . 17
- 3.3 Advantages and Drawbacks . . . 18

4 Social Relation Extraction 20
- 4.1 Implementation Overview . . . 20
- 4.2 Natural Language Conversations . . . 21
- 4.3 Proposed Method . . . 21
  - 4.3.1 Notations . . . 21
  - 4.3.2 Algorithm Overview . . . 22
  - 4.3.3 Shortest Path Relation Extraction . . . 23
  - 4.3.4 Pattern-based Relation Extraction . . . 30

4.4 Experimental Evaluation 32
4.4.1 Datasets 32
4.4.2 Experimental Results 33

5 Conclusion **35**

A Appendix **37**
A.1 Social Relation Extraction 37
A.2 Shortest Path Relation Extraction 38
A.3 Pattern-based Relation Extraction 38

Bibliography **39**

List of Figures

2.1 Bi-LSTM sequence model for named entity recognition [JM18] 6
2.2 Approaches for Relation Extraction 7
2.3 Typical illustration of a dependency path using labeled arcs from head to dependents above the words [JM18]. 12
2.4 The basic transition based parsers consists of a stack, an input buffer and a set of relations containing an oracle for predicting the transition actions [JM18]. 13
2.5 The neural network architecture for dependency parsing presented by Chen and Manning [CM14] consists of an input layer, a hidden layer and a softmax layer. 15

3.1 Graphical representation of a RDF triple, with subject (S) and two objects (O) as nodes, and predicates (p) as edges. The object illustrated as rectangle refers to a label of S. 16
3.2 A labeled property graph (LPG) consists of labeled nodes and edges, each with a unique identifier and a set of key-value pairs, called properties. . . 17
3.3 A labeled property hypergraph consists of nodes and hyperedges, each capable of connecting multiple nodes. 17
3.4 In RDF it is not possible to add attributes to certain elements, hence attributes have to be defined as distinct elements. 18
3.5 In LPGs, certain properties, declared as key-value pairs, can be added to each element (node or edge). 19
3.6 In hypergraphs, the relation is represented as hyperedge (here *father-of*), connected to one or multiple nodes. 19

4.1 Overview of the the social relation extraction within a dialog system . . . 20
4.2 Natural Language Conversations, used within chatbot dialogs, are less formal and do often contain spelling or grammatical errors. 21
4.3 The two major parts of the algorithm 23
4.4 Sentences with POS-Tags and dependecy path 24
4.5 Sentences containing social relations with POS-Tags and dependecy path 25

4.6 Example of a dependency path within a sentence containing two *PER* entities. 27
4.7 Example of a dependency path within a sentence containing two entities. . 27
4.8 Word Vector Space of 100-dimensional word embeddings projected to a 2-dimensional space using Principal Component Analysis (PCA). The closest distance from vector *Peter-Tom* is the vector for representing the word *father*. 29
4.9 Graph representation of the Social Relation between *Peter* and *Tom* as given in the sentence *Peter is the father of Tom.* 29
4.10 Pattern based approach overview. 30
4.11 Noun chunks extracted as tree structures. 31

1 Introduction

Digital dialog systems, also known as chatbots, have gained popularity through their usage in smart phones and smart speaker in recent years. Nevertheless, conversations with chatbots often lack in terms of natural human-like conversations and are often based on question and answering dialogs. The ability to learn someone's social relationships during conversations can help to create personalized responses and lead to a more human-like and diverse conversation.

The field of relation extraction (RE) has become an important tasks to automatically discern relationships among detected entities within unstructured text. Within the field of information extraction (IE) several applications, most notably for optimizing web search engines, have pushed the development and research of new approaches during the last two decades. RE approaches have essentially focused on textual data in form of books or web documents. This work evaluates current approaches in RE and their usage within the domain of natural language conversations. It shows how current approaches in RE can be adapted to extract social relations from conversations and how to store these in a social network structure.

1.1 Research Questions

The field of relation extraction has been widely researched for large and interrelated texts, such as books, news articles or web site documents. A less considered area is the field of natural language conversations, used within chat message between people, or humans and chatbots. The ambiguous, informal and often erroneous characteristic of natural language conversations requires specific adaptions on common approaches in RE. Together with the possibility to spread coherent information across multiple messages, it is necessary to evaluate new techniques upon the well-known approaches for extracting relationships. This work addresses the following two research questions:

- How can social relationships, mentioned within human-to-human or human-to-computer conversations, be extracted, based on existing approaches in RE?

- How can extracted social relationships be efficiently stored, to enable permanent access to the social network?

1.2 Aim of thesis

The first research question is addressed by exploring existing techniques and algorithms for relation extraction. The strategy consists of evaluating well-established RE algorithms and their possible use within the new domain of natural language conversations. Following, a new approach for extracting social relationships is implemented and evaluated on a conversational dataset.

The second research question is addressed by comparing different systems for storing social relations. Therefore, graph database systems and triple stores are evaluated, together with their advantages and drawbacks.

Within this paper the following goals are being pursued: Evaluate current approaches in relation extraction, compare different database systems for storing social relationships and implement a novel approach for social relation extraction.

1.3 Thesis Outline

The remaining chapters of this work are organized as follows.

In chapter 2, an introduction to the key aspects of information extraction is given, followed by an explanation of different techniques for relation extraction, word vector representations and dependency parsing.

In chapter 3, graph database systems and RDF triple stores are surveyed, followed by an evaluation on the use within the context of social networks.

In chapter 4, a new approach for social relationship extraction is introduced, including an experimental evaluation on a conversational dataset.

In chapter 5, the work and its results are summarized.

2 Information Extraction

This chapter reviews related work on information extraction (IE) from textual data, together with its sub tasks of named entity recognition (NER) and relation extraction (RE). It gives an overview on the current approaches in IE, and exemplifies different techniques and methods. The chapter closes with an description of state-of-the-art word vector representations as word embeddings and principles in dependency parsing.

2.1 Definition

Information extraction (IE) describes the task of extracting unstructured information, embedded in texts, into structured data [JM18]. The overall goal of IE is to identify relevant information from documents and to put them in a structured format for further processing [GJJM05, JM18]. To achieve this objective, most IE tasks consist of multiple language processing steps, each of which can be accomplished by different techniques. The two basic steps *named entity recognition (NER)* and *relation extraction (RE)* are being discussed in the following. Furthermore, *word embeddings* for representing semantics in a vector space, and *dependency parsing* for analyzing the grammatical structure of sentences are being described within this chapter.

2.2 Named Entity Recognition

The task of named entity recognition (NER) tries to detect proper names or named entities in a text and to label them [JM18]. The denotation of a named entity is domain specific and might vary in certain approaches. Commonly used entities refer to the labels: persons, organizations, locations or dates. These labels are sometimes also referred to as *types* or *entity types*. Other more specific labels like brand names, products, genes or protein names might be relevant for other tasks. Table 2.1 shows some generic entity types, their tags, categories and example sentences.

Entity Type	Tag	Categories	Example sentence
People	PER	people, characters	**Turing** is a giant in computer science.
Organization	ORG	companies, sports teams	**BMW** is a German automobile manufacturer.
Location	LOC	regions, mountains, seas	The **Mt. Blanc** is the highest mountain of Europe.
Facility	FAC	bridges, buildings, airports	Consider the **Golden Gate Bridge**.
Geo-Political Entity	GPE	countries, cities, states	**Palo Alto** is raising the fees for parking

Table 2.1: List of generic named entities

NER is the first step towards extracting relations, but can also be useful in many other language processing tasks. Recognition is often difficult because of the ambiguity of segmentation and types. For instance, the proper noun *JFK* can refer to a person (*JFK* [PER]), the airport in New York (*JFK* [FAC]) or any other street, school or bridge named after *John F. Kennedy*, the 35th president of the United States of America.

Because of the ambiguity of natural language, it is necessary to take the context of words into account and classify sequences of words, rather than recognizing distinct words within a sentence. The standard algorithm for NER is designed as a so called *sequence labeling* task, where tags capture the boundaries as well as the type of a token. The specific NER tags are therefore augmented by a prefix, indicating multi-word entities.

Considering the first three words of the example *"Palo Alto is ..."* from the last row of table 2.1, a NER tagger using sequence labeling would recognize the entities as follows: *Palo* [B-GPE], *Alto* [I-GPE], *is* [O]. The prefixes of the first two tags indicate the *beginning* (B) and *inside* (I) of the multi-word entity *Palo Alto*. The *outside* tag *O* indicates an entity consisting of a single word, which is not classified as a named entity to one of the predefined classes. The process of labeling multi-word entities with theses prefixes is referred to as *IOB-Tagging*. The first token (word) of a multi-word entity is always labeled with a beginning (B) tag. All following tokens, which belong to the same entity, are marked with inside (I) tags. Entities that do not fall into one of the predefined types are labeled with an single O.

Recent applications are based on two standard families of algorithms for NER, which will be described in the following.

2.2.1 Rule-based Entity Recognition

Rule-based systems use combinations of hand-crafted rules, e.g. regular expressions, and predefined lists or dictionaries containing domain-specific information, for entity recognition. The hand-crafted rules can achieve extremely high precision, but often achieve only a low recall [JM18].

While in academic research, machine learning approaches are the norm, rule-based approaches are still being used in commercial systems [CLR13]. One reason for this distinction leads back to the different use within both areas. In commercial systems, it is often more important to identify entities of a specific domain within pre-processed texts, while in academia it is more important to identify general entities within standard labeled datasets. IBM's SystemT [CKL$^+$10, CDL$^+$18] is a text understanding system, which uses rules, defined in a declarative language (AQL) by the developer. These rules are later translated into an algebraic expression using regular expressions.

2.2.2 Machine Learning for Entity Recognition

Machine Learning algorithms have become the norm for NER in academic research by using recurrent neural networks (RNN) or maximum-entropy Markov models (MEMM).

Current state-of-the-art approaches in NER use a Long short-term memory (LSTM) variant of bidirectional recurrent neural networks, called bi-LSTM [LBS$^+$16, JM18, ABV18]. To consider the strong constraints of the neighbour tokens in named entity recognition, a conditional random field layer (CRF) is used on top of the bi-LSTM output.

Figure 2.1 shows a sketch of an algorithm for NER using bi-LSTM. As input to the bi-LSTM network, words have to be represented as vectors. State-of-the-art approaches uses *word embeddings* for word vector representations, trained over large collections of unlabeled data, like news articles or Wikipedia [ABV18] (details on *word embeddings* will be described in a separate section). The embedding representations of the sentence, in this example *"Mark Watney visits Mars"*, are then applied to the two LSTM layers, their output being concatenated and fed into the CRF layer. The CRF layer shows the final entity tags, in this example *B-PER*, *I-PER*, *O* and *B-LOC*.

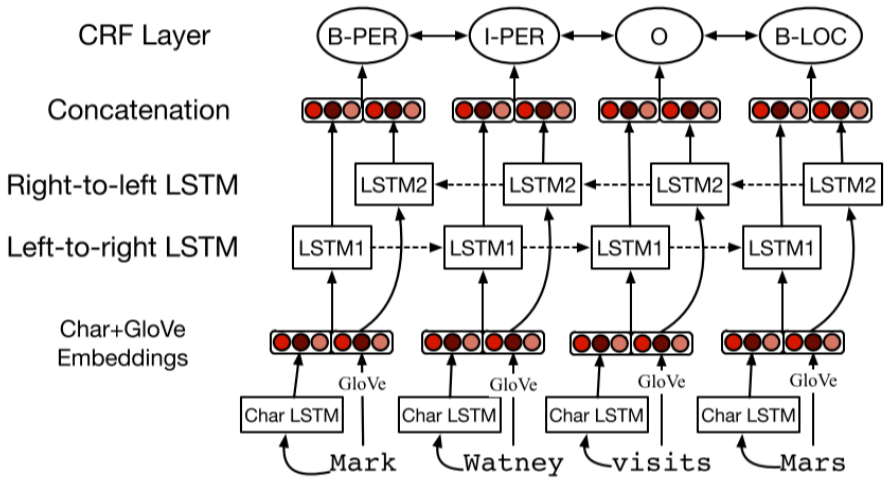

Figure 2.1: Bi-LSTM sequence model for named entity recognition [JM18]

2.3 Relation Extraction

Relation extraction (RE) describes the task of detecting and classifying semantic relations among named entities in texts [JM18]. These are commonly represented as binary relations between two entities such as *part-of*, *child-of*, *employed-at* or *located-in*. Relation extraction faces many challenges as relations are inherently ambiguous and the range of relations can vary from one domain to another. As other computational linguistic tasks, RE is also language-dependant.

To date different approaches for RE exist, which will be discussed in the following sections, after a general definition of RE.

2.3.1 Definition of Relation Extraction

Semantic relationships can be defined among multiple entities. Within this work, only binary relationships will be considered, as relationships among multiple entities can be considered as set of binary relationships [Bat16].

Extracted binary relations are denoted as a triple $< e_1, rel, e_2 >$, where e_1 and e_2 represent the extracted entities, and *rel* describes a specific relationship type that connects the two entities.

2.3.2 Approaches towards Relation Extraction

Approaches for semantic relation extraction can be roughly assigned to one of four categories shown in Figure 2.2 [Bat16, JM18]. Besides that, other approaches might exist, which will not be covered within this work.

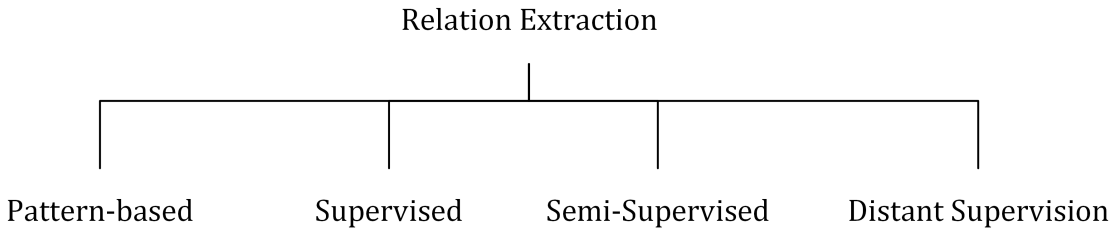

Figure 2.2: Approaches for Relation Extraction

Pattern-based Relation Extraction

Early and still common algorithms uses hand-written patterns for relation extraction, also known as pattern-based RE. First developed by Hearst [Hea92], these patterns try to infer the relationships between entities based on syntactic rules using regular expressions or similar. Consider the following sentence from Hearst's publication:

"The bow lute, such as Bambara ndang, is plucked and has individual curved neck for each string."

Even without knowing the meaning of *Bambara ndang*, a human can infer that it is a kind of *bow lute*. This is true, even if the reader has only few conception about a *bow lute*. The following lexico-syntactic pattern can indicate this relation:

$$NP_0 \; such \; as \; \{NP_1, \; NP_2 \; ..., (and|or)\} \; NP_n \tag{2.1}$$

implies the following

$$for \; all \; NP_i, 1 \leq i \leq n, \; hyponym(NP_i, \; NP_0) \tag{2.2}$$

allowing the inference

$$hyponym(\text{"}Bambara \; ndang\text{"}, \; \text{"}bow \; lute\text{"}). \tag{2.3}$$

Equation 2.1 and 2.2 indicate that a noun phrase (NP) followed by the words *such as*, and one or more noun phrases, connected by *and* or *or*, as a hyponym. Noun phrases are commonly composed of a noun (NN), with an optional preceding determiner (DET) and adjective (JJ). A regular expression for a noun phrase could be defined as follows: *<DT>?<JJ>*<NN>*.

Even if this method can reach a high precision on specific domains, it requires deep domain knowledge and lacks with respect to robustness and portability to other domains. Nevertheless it is still a commonly used approach in some enterprise systems [CLR13].

Supervised Relation Extraction

Supervised relation extraction, based on supervised machine learning, follows the method of hand-annotating a text with a fixed set of relations as training corpus, and uses them to train a classifier. In recent approaches, two classifiers are used to indicate if two entities are related to each other or not, before the final decision about the entitiy type is being made. This allows to speed up the final classification and allows the use of distinct features [JM18].

Commonly approaches for supervised relation extraction consist of the following steps:

1. Find pairs of named entities (mostly within one sentence).
2. Make binary decision whether the entities are related to each other or not (binary classifier).
3. Classify the type of the relation (multi-class classifier).

For each of the classifiers, standard classification techniques like Support Vector Machine (SVM) [MLG11], Logistic Regression, Neural Networks [SXZ15] or similar, can be used.

Supervised relation extraction can achieve high accuracy, if enough hand-labeled data is available. But labeling large training sets is extremely time consuming and error-prone. Besides that, supervised models are very domain-specific and can not generalize to different text genres, that were not part of the training set [JM18]. Most research has therefore focused on semi-supervised approaches in recent years.

Semi-supervised Relation Extraction

Semi-supervised or bootstrapping approaches for RE try to tackle the problem of hand-labelling large training sets. Starting with a set of known relationships, bootstrapping algorithms try to generate training data by iterating over large texts and extracting training sentences based on predefined seeds. The following example shows this approach for the task of extracting relations among cities and the countries they are located in (i.e. relation type: *located-in*):

1. Start with some example seeds, which are already known (*[Berlin, Germany], [Paris, France], [Nuremberg, Germany]* etc.).
2. Iterate over large texts and search for appearances of the entities pairs defined within the seeds.
3. If an entity pair is found within a sentence, extract the features (e.g. the words in between, before and after the entities, the entity types of the two nouns, and perhaps other features: "*Berlin is the capital of Germany...*").
4. Use the new patterns to search for additional tuples of the same entity types.

The first published system using this technique was *DIPRE*, developed by Brin [Bri98] specifically for the purpose of extracting books and their authors from web documents (*[author, book]* pairs). It uses a 7 tuple <*author, book, order, url, prefix, suffix, middle*> as features for every co-occurence of a seed example, where *order* represents a boolean if the author occures before the book, *prefix* and *suffix* the 10 words before and after the the matched entities and *middle* all words in between them. The tuples are grouped by the *middle* and *order* features and DIPRE verifies if new tuples are the same. The new learned pattern are used to search the corpus again and extract new relationships.

Agichtein & Gravano [AG00] have extended Brin's approach to a system called *Snowball*, which presents a more general approach for extracting further entity types from texts, than only books and authors. Similar to DIPRE, they are forming tuples of the entities (e1, e2) and the words before (BEF), in between (BET) and after (AFT) as context: <*BEF, e1, BET, e2, AFT*>. Each context tuple is then represented by a TF-IDF vector, and similar contexts are clustered by a single-pass clustering, using the cosine similarity between the vectors.

Batista et al. [BMS15] updated the *Snowball* system with state-of-the-art word vector representations using word embeddings. Together with a prior selection of words between the entities using hand-written patterns, they achieved a higher accuracy than previous systems.

The advantage of bootstrapping approaches lies in the omitted necessity of hand labelling large datasets to train a classifier. One downside of the examined bootstrapping approaches is the often limited usability to specific relation types, such as hypernyms (is-a) or meronyms (part-of) [GBM03, SJN05]. Another drawback of semi-supervised RE is the manual creation of seed examples, which requires domain specific knowledge.

Distant Supervision

Distant Supervision methods use knowledge bases, like *DBPedia*[1] or *Freebase*[2], containing semantic relations to automatically collect large amounts of seed examples. These seed examples are then used to extract training examples on large amounts of texts, similar to the bootstrapping approach.

Considering a system which tries to learn the relation type *place-of-birth*, instead of manually defining a small number of seed examples, *DBPedia* is used, containing over 100,000 examples for the relation type *place-of-birth*, like <*Albert Einstein, Ulm*>. Distantly supervised approaches then run a named entity tagger over large coherent texts, like Wikipedia Articles, and extract sentences containing both entities (e.g. *"Albert Einstein was born in Ulm..."*). The extracted sentences can then be used to train a supervised classifier, similar to the supervised approaches described above.

Mintz et al. [MBSJ09] use *Freebase* as source for their seeds, which is a former semantic database trained on Wikipedia info boxes and other sources. They use several syntactic and lexical features, like sequence of words between the entities and part-of-speech tags, to train a multi-class logistic classifier.

Aljamel et al. [AOA15] used a combination of *DBPedia* and *Freebase* for extracting their seed tuples. They also used a *JENA'S SPARQL* engine for extracting the mentioned relations within both databases, through a common interface. For the classification task, they compared three methods: Support Vector Machine (SVM), Perceptron Algorithm Uneven Margin (PAUN) and K-Nearest-Neighbour (KNN).

[1] https://wiki.dbpedia.org/
[2] https://developers.google.com/freebase/

The SVM outperformed in both Person-Location (Per-Loc), Person-Organisation (Per-Org) relations and had an equal accuracy to PAUN within Location-Organisation (Loc-Org) relations.

2.4 Word Embeddings

The representation of words in a continuous vector space is a standard approach in current Natural Language Processing (NLP) applications. Earlier techniques treat words as atomic units, each represented as an index in a vocabulary and similar words are represented close to each other within the vector space. However, the semantic similarity is not represented within the vector space and words having similar meanings, but different notations could spread over large distances.

To date different techniques have been proposed to create semantic word vectors, commonly known as *word embeddings*. The basic idea of these techniques lead back to the distributional hypothesis, first formulated by linguists Joos, Harris and Firth, which says that *"words that occur in similar contexts, tend to have similar meanings"*. Two of todays most popular systems *Word2Vec* [MCCD13] and *GloVe* [PSM14] are based on this hypothesis. Their models are trained on hundreds of millions of words, extracting target words and their corresponding *nearby words* within a predefined *window*. Both approaches use different techniques for calculating multi-dimensional word vector representations.

Word2Vec uses a feed-forward neural network with three layers for creating word vector representations. The model provides two different architectures: *Skip-gram* and *CBOW* (continuous bag-of-words). Both models are algorithmically similar, but different at training phase. The *Skip-gram* model learns word vector representations of the target word, that are good at predicting the nearby words. The *CBOW* model does the inverse, by learning word vector representations of the nearby words, that are good at predicting the target word.

GloVe is a count-based approach using a matrix factorization. First, a matrix X, containing *word-word* co-occurrence counts, is being created. Every entry X_{ij} within the matrix, denotes how often a word j occurs within the context of word i. After training over a large corpus, this matrix is factorized to yield a lower dimensional matrix, where every row yields a vector representation.

2.5 Dependency Parsing

Dependency Parsing describes a method to automatically analyze the grammatical structure of a sentence and to establish relationships between "head" words and their dependents [JM18]. Figure 2.3 shows a typical illustration of a sentence's dependency path, with relations denoted above the words using a labeled arc from heads to dependents.

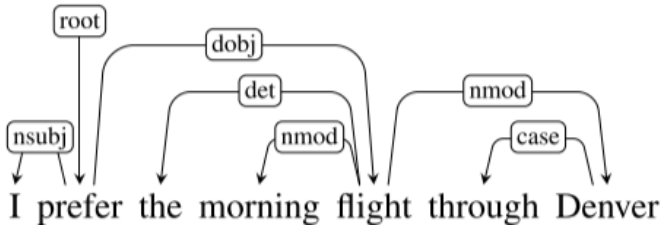

Figure 2.3: Typical illustration of a dependency path using labeled arcs from head to dependents above the words [JM18].

The root element, which is mostly the verb (predicate) within a sentence, is explicitly labeled. The resulting dependency path, sometimes also referred to as dependency graph, contains important information, which is usually hidden in complex phrasal structures. For example, the verb *prefer* from the sentence in Figure 2.3 is directly connected to it's argument *flight*, which is then connected to it's dependent noun *Denver*.

The dependency path can help to analyze structures, especially in languages with free word order, that offer a relatively flexible phrasal structure. With the provided information of the head-dependent arc, it is also possible to extract an approximation of the semantic relationships between predicates and their arguments.

2.5.1 Transition-based Parsing

Transition-based dependency parsing approaches are motivated by a stack-based *shift-reduce parsing*, originally developed for analyzing programming languages [JM18]. These parsers aim to predict a transition sequence from an initial configuration to a final configuration, which derives the target dependency tree. To generate the configurations, transition based parsers consists of an *input buffer* for the tokens, a *stack*, a set of *dependency relations* and an *oracle* (see Figure 2.4).

The basic principle applies a recursive process, until a final configuration is reached. The *oracle*, sometimes also called *guide*, takes the features extracted from the configuration and predicts, which transition (action) to take.

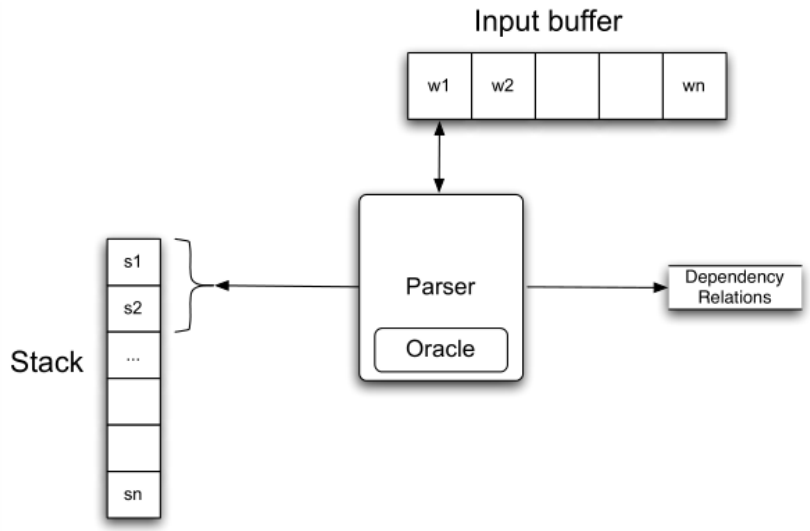

Figure 2.4: The basic transition based parsers consists of a stack, an input buffer and a set of relations containing an oracle for predicting the transition actions [JM18].

The standard approach for transition-based parsing, known as the *arc standard* [Niv03], consists of the following steps for producing new configurations:

1. Initialize the input buffer with words: one-by-one, left-to-right.
2. Initialize the stack with an generic *root* element.
3. Push the first word of the input buffer onto the stack.
4. Compare the first two elements on the stack.
5. Choose a transition action for these two words (left reduce, right reduce, no action), based on the oracles prediction.

The transition actions are of the types *left reduce*, *right reduce* or *no action*. These follow the graphical *arc representation* (left arc, right arc) represented in the dependency path, shown in Figure 2.3. The two transitions for *left reduce* and *right reduce* work as follows:

- Left-Arc: Add an arc from the top element on the stack to the second element $s_1 \rightarrow s_2$; remove the second element (s_2) from stack .
- Right-Arc: Add an arc from the second element on the stack to the first element $s_2 \rightarrow s_1$; remove the top element (s_1) from the stack;

The decision, which action to use in the current configuration, i.e. predicting a transition, is given by the *oracle*. In state-of-the-art parsing systems these oracles are represented by classifiers trained via supervised machine learning. The classifier learns, which action to taken given an appropriate configuration. As training data, representative treebank corpora like the *Penn Treebank* for English or the *TIGER Corpus* for German, containing dependency trees are being used.

The features, used for predicting a transition, are commonly the conjunction of one up to three elements from the stack using the words, their POS tags or arc labels. However, these concatenation of features is generally expensive and incomplete, as it does usually not cover all possible word combinations.

2.5.2 Neural Network based Parsing

Chen and Manning [CM14] presented a dependency parser, which uses a neural network classifier to make parsing decisions within a transition-based dependency parser. The aim is to overcome the incompleteness and expensive computation of features within transition-based systems, like the *arc-standard* mentioned above. A neural network is trained on the *Penn Treebank*, to predict the appropriate transition action.

The neural network parser consists of a shallow neural network with an input layer, one hidden layer using cube activation functions and a final softmax layer (see Figure 2.5). The words, their POS-Tags and arc-labels are being mapped to d-dimensional vectors.

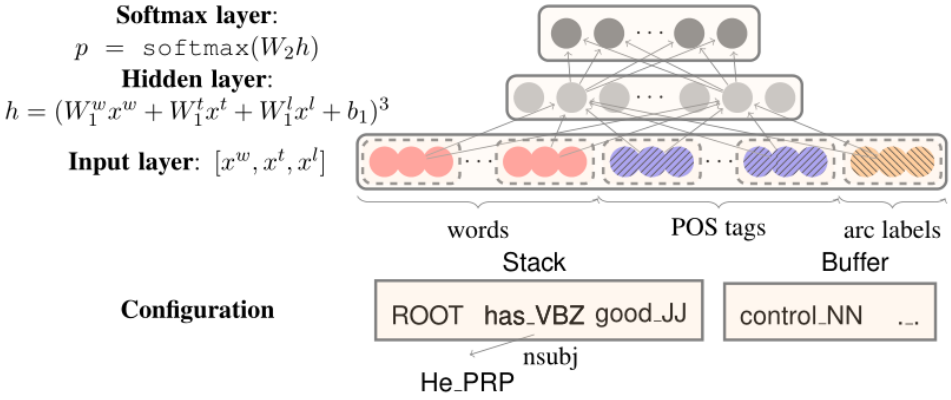

Figure 2.5: The neural network architecture for dependency parsing presented by Chen and Manning [CM14] consists of an input layer, a hidden layer and a softmax layer.

Instead of taking the discrete representations of POS tags and arc-labels, this approach uses word embeddings to cover regularities of the words. Since it is not necessary to calculate and concatenate the POS tags of all words, this is much faster than the standard approach.

3 Storing Social Relationships

This chapter covers approaches for permanently storing social relationships using semantic web and graph database technologies.

3.1 RDF Triple Stores

RDF (Resource Description Framework[1]) and the corresponding database system, called *RDF triple store*, has become one of the standard techniques in academia for structuring relationships. Originally developed for representing meta data of the web, it is nowadays used to a wide variety of semantic data. Large semantic databases, such as *DBPedia* or *Freebase*, already mentioned within in context of *Distant Supervision*, are using RDF semantic format to represent their data.

The basic RDF data model represents members as triples composed of the three elements *subject*, *predicate* and *object* [LS17]. The *subject* represents the resource and a node in the graph, the *object* another node or literal. The *predicate* illustrates the relationship, shown as edge, in between them (see Figure 3.1). An object can also be a label, used to describe further attributes of a subject. RDF stores are strongly index-based, so that each element has its own index.

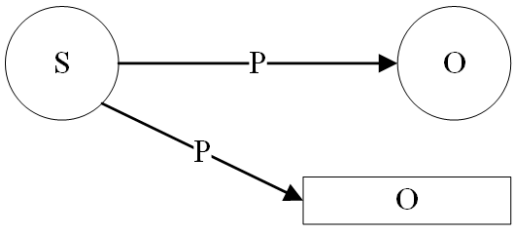

Figure 3.1: Graphical representation of a RDF triple, with subject (S) and two objects (O) as nodes, and predicates (p) as edges. The object illustrated as rectangle refers to a label of S.

[1] https://www.w3.org/RDF/

3.2 Graph Database Systems

A graph database is a storage system which uses graph structures for storing and representing data [RWE13]. The structures are represented as a combination of *nodes* and *edges*. A widely used model for graph databases is referred to as *labeled property graph (LPG)*. The *nodes* (*entities*) within this model can hold any number of *properties* (*attributes*) expressed as key-value pairs. Additionally, nodes and edges can be tagged with *labels* for representing a role or something similar. These *labels* are sometimes referred to as *types*, following the notation of object oriented programming.

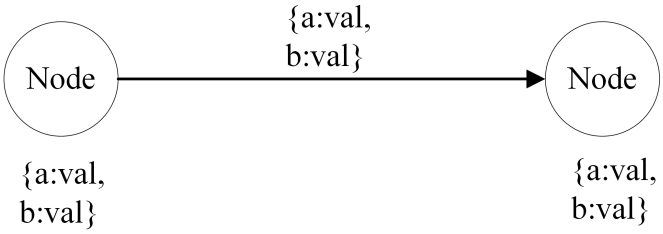

Figure 3.2: A labeled property graph (LPG) consists of labeled nodes and edges, each with a unique identifier and a set of key-value pairs, called properties.

Besides LPG, *hypergraph* models take a slightly different approach within graph database systems. A *hypergraph* presents a generalization of a graph, in which edges are substituted by *hyperedges* [Ior10]. Other than in LPG, where an edge is a connection between two nodes, hyperedges can connect an arbitrary set of nodes within the graph.

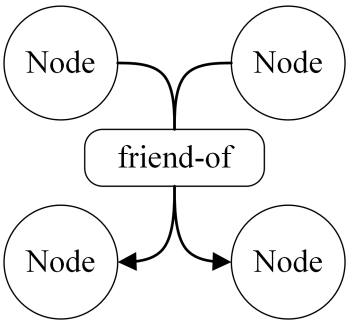

Figure 3.3: A labeled property hypergraph consists of nodes and hyperedges, each capable of connecting multiple nodes.

3.3 Advantages and Drawbacks

RDF has become a standard format for storing relations of various types, especially in academia, as it offers a well-defined and comparable format, officially adopted by the World Wide Web Consortium (W3C). Nevertheless, it has some drawbacks in storing social relationships compared to property graphs. Due to the unique indexing of RDF, it is not possible to have the same relationships between the same entities multiple times. Depending on how the social network is being constructed, there might be cases where multiple similar edges are necessary. Besides that, relationships (predicates) can not hold attributes in RDF, which means that attributes have to be stored by using distinct elements. This can hinder querying objects, as it results in multiple joined queries, which take longer to process. For instance, considering the storage of a social relation between the two real world persons *Peter* and *Tom*, *Person* has to be defined as a distinct object within the RDF triple store (see figure 3.4).

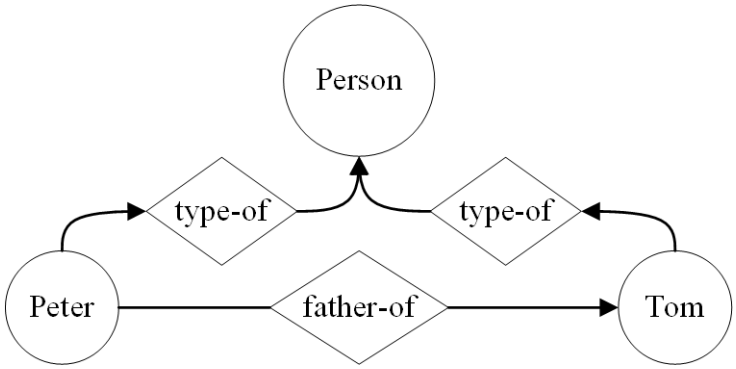

Figure 3.4: In RDF it is not possible to add attributes to certain elements, hence attributes have to be defined as distinct elements.

The LPG model facilitates denoting each element to a specific type, by adding the corresponding label. If the example from above is taken, it is possible to denote both nodes with a property *type* and the corresponding value *Person*. The edge can also be assigned with a property, which results in a compact representation and short queries.

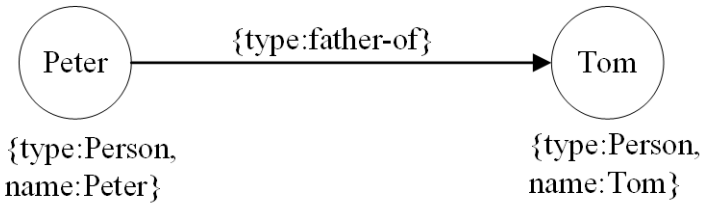

Figure 3.5: In LPGs, certain properties, declared as key-value pairs, can be added to each element (node or edge).

Hypergraphs can handle social relations in a similar way to LPG's. However, there is a need to provide relations as hyperedges, which are actually respresented as additional nodes within the graph. Even though this can be helpful in social networks, for instance having a *father* node connected to multiple child nodes via a hyperedge, this leads to larger graphs even in small networks.

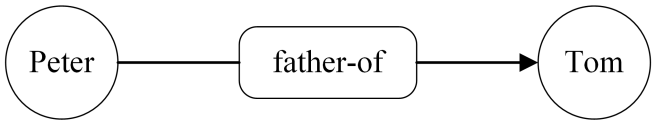

Figure 3.6: In hypergraphs, the relation is represented as hyperedge (here *father-of*), connected to one or multiple nodes.

4 Social Relation Extraction

This chapter describes an approach for social relation extraction. It starts by introducing the use of social relation extraction within dialog systems, and illustrates the characteristics of natural language conversations. The proposed approach is adapted for the use within conversational texts and uses two distinct components: a shortest path relation extraction based on semi-supervised bootstrapping approaches, and a pattern-based relation extraction using grammar rules.

4.1 Implementation Overview

The social relation extraction approach proposed within this work, is developed as part of a dialog system (chatbot) used for human-like social interaction. Figure 4.1 shows a schematic overview of the three main parts: *Dialog System*, *Relation Extraction* and *Social Network Graph*. The *Relation Extraction*, serves as an interface between the *Dialog System* and the *Social Network Graph*. The aim is to learn someone's social network through chatting with the dialog system, disregarding any other resources. Both parts, *Relation Extraction* and *Social Network Graphs*, are being exemplified within this chapter.

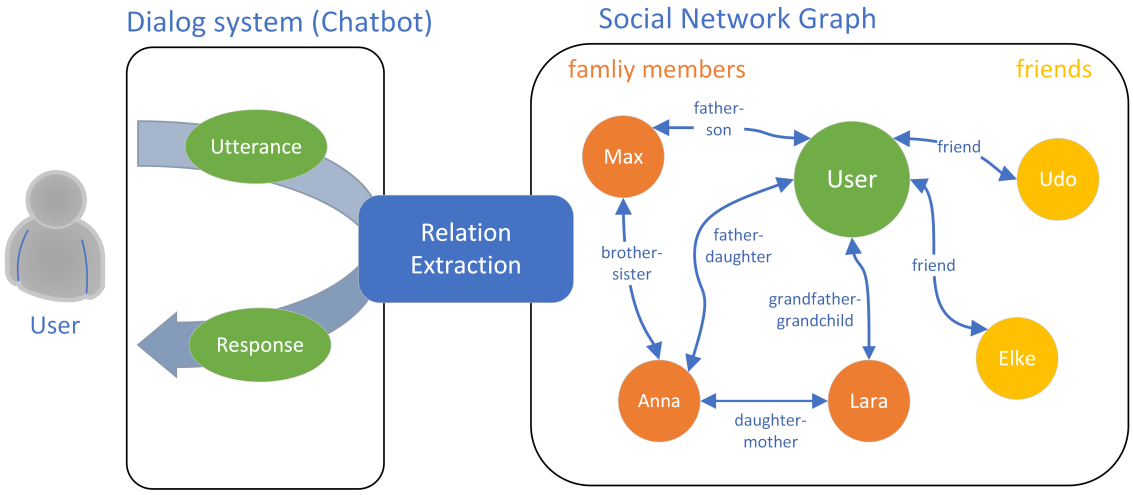

Figure 4.1: Overview of the the social relation extraction within a dialog system

4.2 Natural Language Conversations

Natural Language Conversations, as utilized within this work, are referred to as conversations between two or more persons, or a person and a chatbot (see Figure 4.2). These textual data differs from other unstructured text forms, such as news articles or books, as the used language is commonly informal and texts are short, less reviewed and therefore often erroneous. Missing punctuation or capitalization, misuse of grammar rules, and the ambiguity of natural language in general, are further characteristics, which makes analysis difficult.

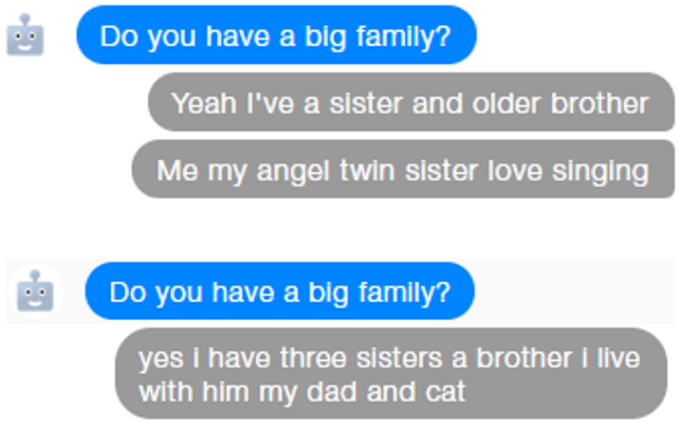

Figure 4.2: Natural Language Conversations, used within chatbot dialogs, are less formal and do often contain spelling or grammatical errors.

4.3 Proposed Method

The proposed method for social relation extraction uses two distinct components, depending on the recognized entity types within a sentence. Both components use different approaches, based on grammatical and semantic features.

4.3.1 Notations

Within this work, social relations are described as a finite number of relation types between two entities mentioned within a sentence. The entities are assigned to one of the following two types: *Person (PER)* or *User (USR)*.

Entity pairs must occur in one of the following orders within a sentence, to be successfully extracted: *Person-to-Person* (PER-PER) or *User-to-Person* (USR-PER). Table 4.1 shows an example of both combinations.

No.	Example	Relations	Entities
1	"**Peter** is the father of **Tom**."	<Peter, father-of, Tom>	PER-PER
2	"**My** daughter **Lisa** is moving to London next month."	<Lisa, daughter-of, USER>	USR-PER

Table 4.1: Social relationships are being described as a relation between two specific types of entities, which can be related to each other in two different ways shown as rows in the table.

The relations between two entities can be of one of the following types: *father-of*, *mother-of*, *son-of*, *daughter-of*, *brother-of*, *sister-of*, *grandfather-of*, *grandmother-of*, *grandson-of*, *granddaughter-of*, *husband-of*, *wife-of*, *uncle-of*, *aunt-of* or *friend-of*.

Relations are being denoted as triples $< e_1, rel, e_2 >$, and read from the left entity to right entity, e.g. the triple $< Peter, father-of, Tom >$ is being read as *"Peter is father of Tom"*.

The entity type *USR* describes the user interacting with the dialog system. If the user mentions him- or herself through a possessive pronoun, like *my* or *I*, it is denoted as *USR* entity. For instance, someone might utter the following sentence: *"My daughter Lisa is flying to London next week"*. The extracted relation triple in this case is expected to be $< USR, daughter, Lisa >$, where *USR* represents a generic entity referring to the user.

4.3.2 Algorithm Overview

The proposed method is split into two main components, using different approaches (see Figure 4.3). The shortest path approach is considered to extract relations, if at least two entities of the type *PER* or an entity pair of *USR* and *PER* are being recognized within a sentence. The Pattern-based extraction is used, if a single *PER* entity or a single *USR* is recognized within a sentence.

Pseudocodes of both approaches together with the general function for relation extraction can bee seen in appendix A.

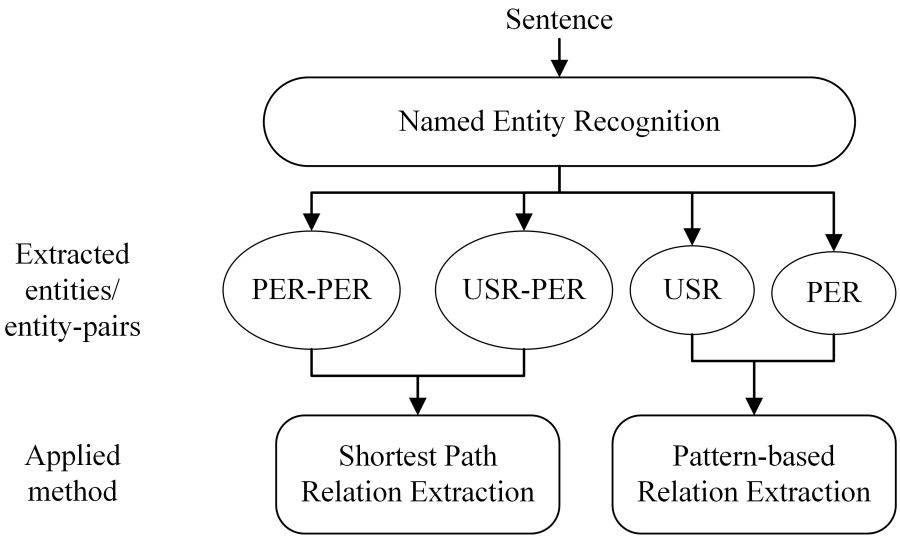

Figure 4.3: The two major parts of the algorithm

The decision of splitting the relation extraction into two distinct approaches, is based on the use within a dialog system. Sentences containing multiple entities can be extracted based on relation extraction algorithms, as described in chapter 2. Due to the possibility to ask the user a specific question through the dialog system, it is also helpful to extract sentences containing only one entity. The missing second entity can be identified within further utterances of the user.

4.3.3 Shortest Path Relation Extraction

The method for shortest path RE is based on the feature extraction process, used by Bunescu et al. [BM05] to train a multi-class classifier. Other than in [BM05], the proposed method does not train a classifier due to the lack of conversational training datasets. It will rather use a distance measurement and compare the extracted features with predefined vectors of known relationship types. Besides that, it uses word embeddings instead of combined features made of POS-tags, entity types and other syntactic features.

Shortest Path Hypothesis

As proposed by Bunescu et al. [BM05], the contribution of the dependency graph for establishing the relation between two entities, is almost exclusively concentrated in the shortest path between these in the undirected version of the dependency graph. If two entities are arguments of the same predicate (verb) and belong to the same predicate-argument structure, then the shortest path will pass through this predicate. Figure 4.4 shows two examples representing this case, where the two entities represent the subject (nsubj) and object (dobj) of each sentence, both related to the same predicate.

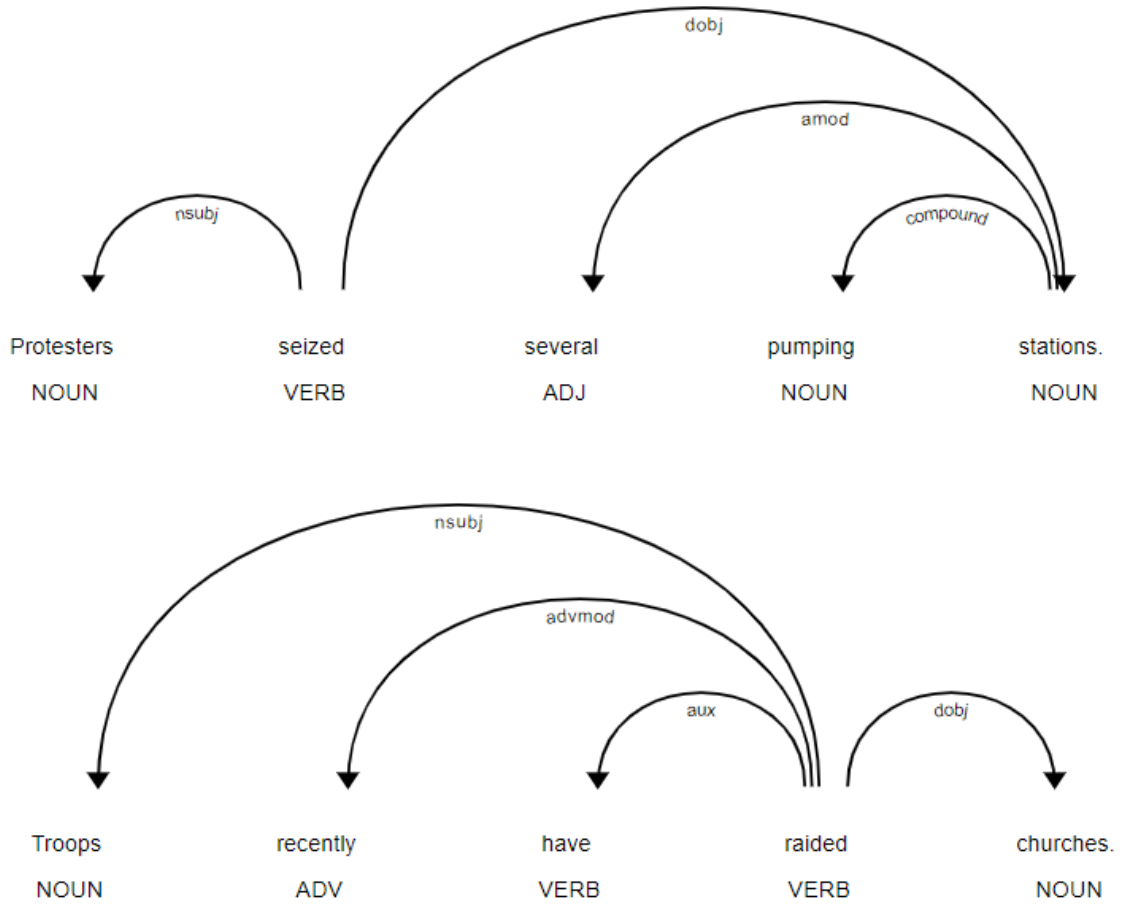

Figure 4.4: Sentences with POS-Tags and dependecy path

No.	Sentence	Shortest path in undirected dependecy graph
1	Protesters seized several pumping stations.	**protesters** ← *seized* → **stations**
2	Troops recently have raided churches.	**troops** ← *raided* → **churches**

Table 4.2: Shortest path representations of the sentence

In the first sentence of Figure 4.4, the shortest path between the two noun entities *protesters* and *stations* passes through the predicate (verb) *seized*. In the second sentence below, the two entities *troops* and *churches* are both related to the verb *raided*. In these basic examples, the predicate represents the only word on the shortest path, which might not be the case in more complex sentences. Nevertheless, even with more words on the shortest path, the predicate is always a part of it.

Based on the words on the shortest dependency path, annotation decisions can be made that show the relation between two entities. For instance, it is reasonable that if the person entity ('protesters') is doing some action ('seized') to a facility entity ('stations'), then the person entity is located at that facility [BM05].

Shortest Path Hypothesis for Social Relations

The shortest path hypothesis can be used to extract social relationships between two entities, as the shortest path commonly passes through the mentioned relation. This is also true, if the relationship is not mentioned between two entities or if other entities appear in between them. Figure 4.5 shows two examples.

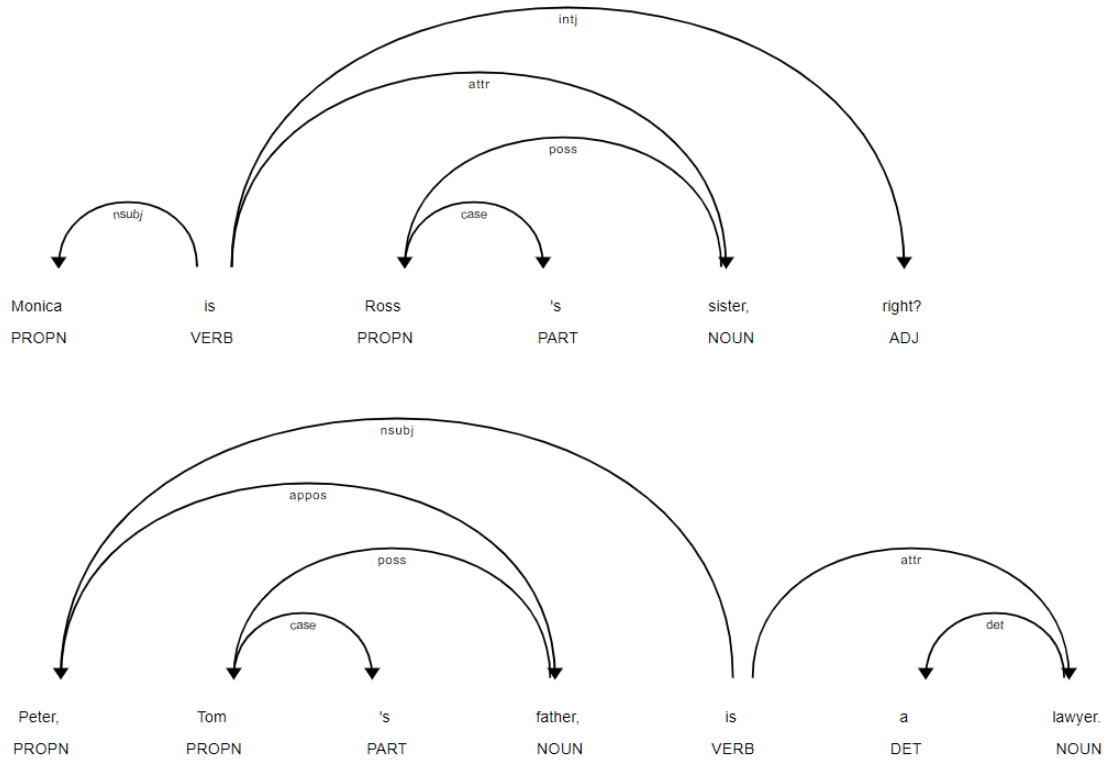

Figure 4.5: Sentences containing social relations with POS-Tags and dependecy path

No.	Sentence	Shortest path in undirected graph
1	Monica is Ross's sister, right?.	**Monica** $\leftarrow$ *is* $\rightarrow$ *sister* $\rightarrow$ **Ross**
2	Peter, Tom's father, is a lawyer.	**Peter** $\rightarrow$ *father* $\rightarrow$ **Tom**

Table 4.3: Shortest path representations of the sentence

The first sentence, shown in the figure 4.5, holds a social relation between the entities *Monica* and *Ross*. Even though the relation is not in between these entities, it is possible to extract the relation by following the shortest path in the undirected dependency graph.

The second example below, shows an inserted sub-clause, containing the social relation *father*. Even though both entities are directly following each other, it is possible to extract the social relation, mentioned afterwards, through the dependency path.

Method for extracting Social Relationships

The proposed method for extracting social relations, consists of eight steps which are shown in the following:

1. Recognize two or more entities within a sentence (PER or USR)
2. Extract the dependency path of the sentence
3. Build an undirected graph based on the dependency path
4. Search for the shortest path between entity pairs inside the undirected graph
5. Get word embedding representations for each word on the shortest path
6. Sum up word embeddings to a single vector
7. Measure cosine similarity between the shortest path vector and stored relationship vectors
8. Extract relationship type with the highest similarity as the final relation type

The first step consists of finding entities (*PER*, *USR*) within a sentence. *Person (PER)* entities are being recognized by using two pre-trained sequence tagger models, for English and German, provided by the open source library *Flair*[1]. Both models are trained over the *CoNLL-03* dataset and offer a 4-class named entity recognition. The architecture of the sequence labelling follows a BiLSTM-CRF approach, as described in section 2.2.2 [ABV18]. *USER (USR)* entities are recognized by using a predefined list.

[1] https://github.com/zalandoresearch/flair

For creating the dependency path (step 2), the pre-trained models *en_core_web_md* for English and *de_core_news_sm* for German, provided by the open source NLP library *spaCy*[2], are used. The English model is trained on the sources *CommonCraw*[3] and *OneNotes 5*[4], and consists of 20,000 unique word vectors and 685,000 keys. The German model is trained on the *TIGER*[5] corpus. Figure 4.6 shows the visualization of a dependency graph for the sentence *"Peter is the father of Tom"*, which will be used as an example throughout this section.

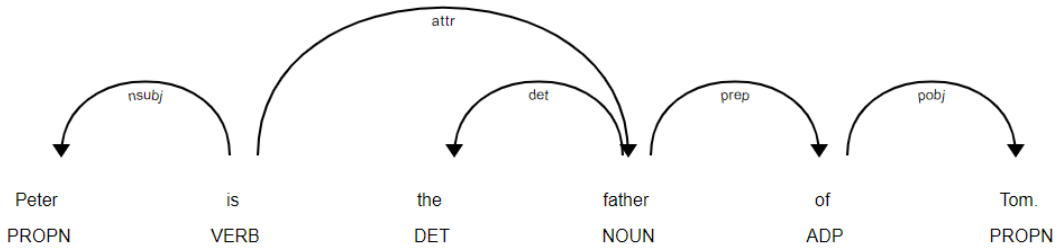

Figure 4.6: Example of a dependency path within a sentence containing two *PER* entities.

Within the third step of this approach, the dependency graph is transformed to an undirected graph, containing the words as nodes and the dependencies as edges (see Figure 4.7). As the entities are already known from the first step, the next step is to search for the shortest path between each entity pair using Dijkstra's algorithm. The entity pairs are constructed, following their appearance within a sentence from left to right, so that the leftmost entity is connected to each of following entities to the right. For instance, considering three entities e_1, e_2, e_3 within a sentence, the three entity pairs $< e_1 - e_2 >$, $< e_1 - e_3 >$ and $< e_2 - e_3 >$ are being constructed.

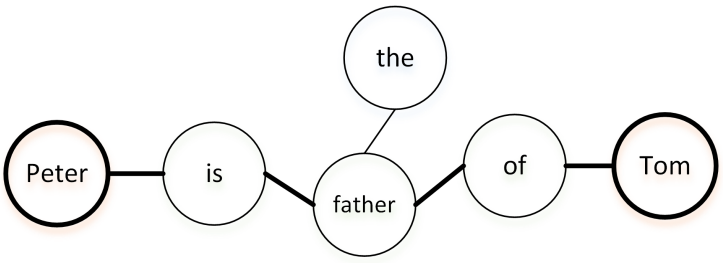

Figure 4.7: Example of a dependency path within a sentence containing two entities.

[2]https://spacy.io/
[3]http://commoncrawl.org/
[4]https://catalog.ldc.upenn.edu/LDC2013T19
[5]https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger.html

Following the sentence of the example introduced above, two entities being recognized, resulting in one entity pair $< Peter - Tom >$. Using the shortest path algorithm, the words *is*, *father* and *of* are extracted (step 4; see Table 4.4).

Sentence	**Shortest path in undirected graph**
Peter is the father of Tom.	**Peter** $\leftarrow is \rightarrow father \rightarrow of \rightarrow$ **Tom**

Table 4.4: Shortest path representations of the sentence

In the fifth step, for each word on the shortest path the corresponding word embedding representation is looked-up in a word embedding dictionary. The dictionary used here, contains vector representations of *GloVe*[6] word embeddings, trained over Wikipedia provided by the *FastText*[7] library. Each word is represented by a 100-dimensional word vector. Words that are not part of the dictionary, will be represented by a zero vector.

All word vectors of the shortest path are being summed up into a single 100-dimensional vector (step 6), further denoted as *shortest path vector* ($\vec{sp}$). This vector is being compared (step 7) to a set of stored *relation vectors* ($\vec{rel}$), each representing a relation type corresponding to one of types defined in 4.3.1. This results in a word vector for each of the following 15 words: *father*, *mother*, *sister*, *brother*, *son*, *daughter*, *husband*, *wife*, *grandson*, *granddaughter*, *grandmother*, *grandfather*, *uncle*, *aunt* and *friend*.

The closest *relation vector*, measured through the cosine similarity, indicates the relation type between the entities (step 8). The similarity is depicted by a single float value and needs to exceed a predefined distance threshold of 0.6 for successful RE. Similarities below this threshold are not extracted as social relationships.

The similarity is measured by using the cosine similarity, as denoted in Equation 4.1, where $\vec{sp}$ represents the shortest path vector and $\vec{rel}(n)$ a relation vector ($1 \leq n \leq 15$).

$$similarity = cos(\theta) = \frac{\vec{sp} \cdot \vec{rel}(n)}{||\vec{sp}|| \cdot ||\vec{rel}(n)||} \tag{4.1}$$

Figure 4.8 illustrates a two-dimensional plot of *relation vectors* (blue) and the shortest path vector (red) between the entities *Peter* and *Tom* from the example above. The closest vector in this case is the word vector for representing *father*.

[6] https://nlp.stanford.edu/projects/glove/
[7] https://fasttext.cc/docs/en/crawl-vectors.html

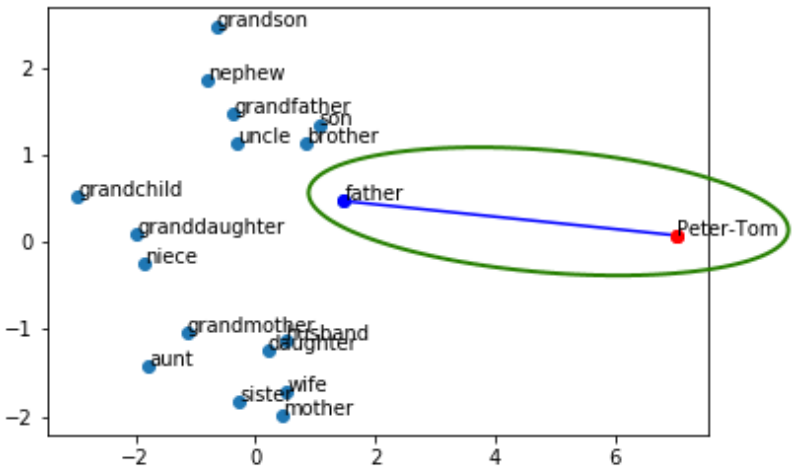

Figure 4.8: Word Vector Space of 100-dimensional word embeddings projected to a 2-dimensional space using Principal Component Analysis (PCA). The closest distance from vector *Peter-Tom* is the vector for representing the word *father*.

Storing extracted Relationships

Extracted relationships between PER-PER or USR-PER entities are stored in a *neo4j*[8] graph database using the LPG model. Entities are represented as labeled nodes, holding the attributes *name*, *age*, *gender*. The attribute *name* is filled, if a the corresponding entity is recognized as a person.

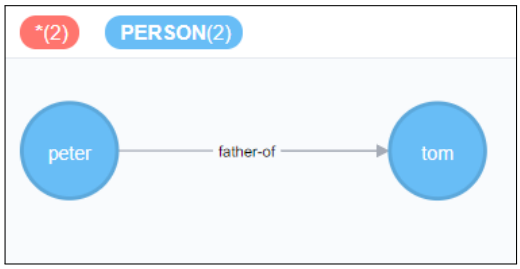

Figure 4.9: Graph representation of the Social Relation between *Peter* and *Tom* as given in the sentence *Peter is the father of Tom.*

[8] https://neo4j.com/

The other attributes are left blank during the initial creation and might be filled later during the progress of the conversation. The edge holds a single attribute *relation-type* as extracted during the above described process.

4.3.4 Pattern-based Relation Extraction

The second RE component uses a pattern-based approach, inspired by the basic principles of Hearst [Hea92] and using a modified version of the grammar introduced in the context of Fader's *ReVerb* system [FSE11]. The algorithm consists of four basic steps and extracts a relation if it meets syntactic and lexical constraints (see Figure 4.10). After sentence segmentation and entity recognition, achieved similar to the shortest path approach, the following steps are applied:

1. Tokenization of the sentence.
2. Part-of-Speech Tagging of each token.
3. Chunking: building noun phrases (NP) or personal pronoun phrases (PP).
4. Comparison of the words inside the chunks with known relations (similar to the shortest path approach).

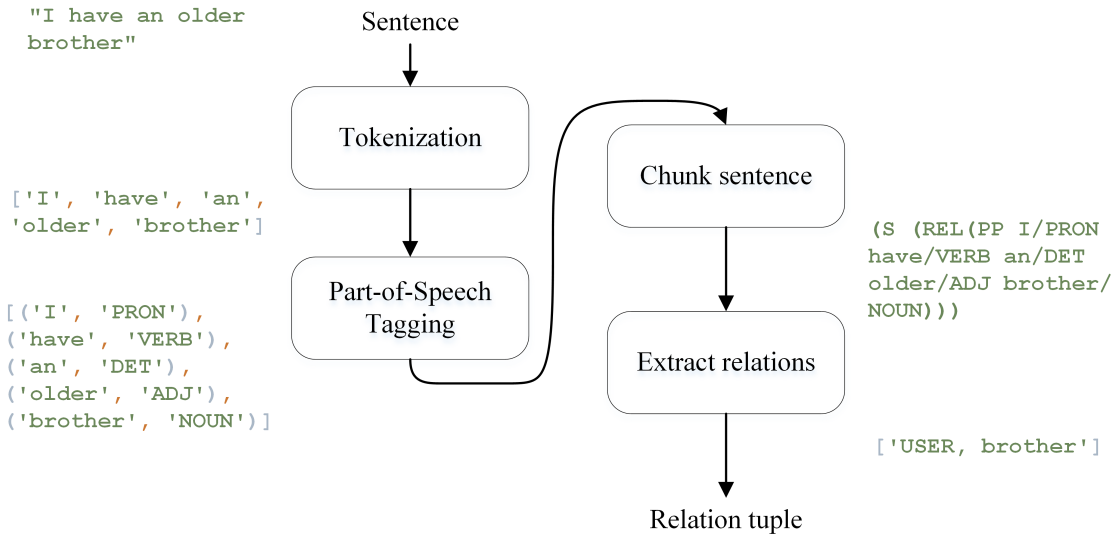

Figure 4.10: Pattern based approach overview.

After tokenization, each token is labeled with the corresponding Part-of-Speech (POS) tag, indicating the grammatical word category. For both, tokenization and POS tagging, the pre-trained *spaCy* model, similar to the one used for dependecy parsing in the previous section, is being used. The POS-tagged sentence is parsed using a *chunking* method, which labels multi-token, non-overlapping segments (chunks) of the sentence according to a pre-defined grammar [JM18]. Particularly common are noun phrases, verb phrases or adjective phrases.

The approach presented here uses a grammar as show in listing 4.1. It consists of two independent chunks denoted as *NP* for noun phrases and *PP* for (personal) pronoun phrases. A noun phrase refers to a group of words, starting with a determiner, followed by an optional adjective and ending with a noun, e.g. "My[DET] little[ADJ] Sister[NOUN]". A pronoun phrase refers to a chunk consisting of a pronoun, followed by a verb, an optional number, determiner or adjective, and an ending noun, e.g. "I[DET] have[VERB] an[DET] older[ADJ] brother[NOUN]". A chunk matching one of these two rules, is denoted as a possible relationship *REL*.

```
1  NP: {<DET><ADJ>?<NOUN>} #e.g. "My little sister"
2  PP: {<PRON><VERB><NUM>?<DET>?<ADJ>?<NOUN>} #e.g. "I have an older brother"
3  REL: {<NP>|<PP>}
```

Listing 4.1: Chunking grammar

The sentence is then projected to a tree structure for further processing. Subtrees labeled as *REL* indicate possible relationships and every word inside the subtree is decoded into its *GloVe* word embedding representation. Each word embedding vector is then compared to the stored relation type vectors, similar to step 7 of the shortest path approach. The relation type with the highest cosine similarity, i.e. the shortest distance inside the vector space, is being extracted as the final relationship.

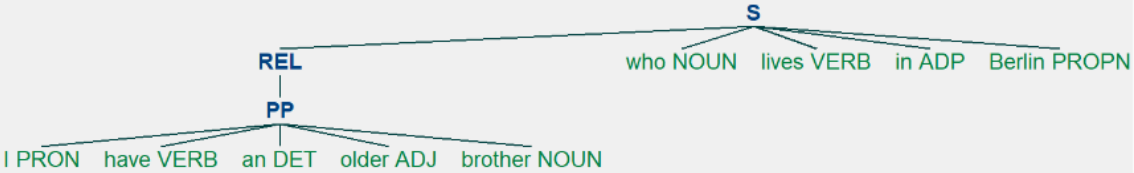

Figure 4.11: Noun chunks extracted as tree structures.

Due to the fact that the second entity is missing, the extracted relation is not instantly stored to the graph database, as in the shortest path approach. Instead, a specific question is returned back to the dialog system. If the user specifies the missing entity during the next conversation messages, the relation is stored to the *neo4j* database, otherwise it is discarded.

4.4 Experimental Evaluation

The experiments are conducted on a dataset, built as a combination from utterances of the *Persona-Chat*[9] dataset and the *Friends TV Corpus*[10]. From each dataset, 500 randomly chosen utterances are used.

4.4.1 Datasets

The *Persona-Chat* project [ZDU+18] was created by collecting 162,064 utterances between crowdworkers via *Amazon Mechanical Turk*. The crowdworkers were randomly paired and told to chat naturally and get to know each other during conversation (see Figure 4.5). This dataset is also known as a training set for the Conversational Intelligence Challenge 2 (ConvAI2).

Persona	**Utterance**
PERSON1	Hi
PERSON2	Hello! How are you today?
PERSON1	I'm good thank you, how are you
PERSON2	Great thanks! My children and I were just about to watch Game of Thrones.
PERSON1	Nice! How old are your children?
PERSON2	I have four that range in age from 10 to 21. You?

Table 4.5: For constructing Persona-Chat dataset [ZDU+18], two crowdworkers were randomly paired and told to chat to each other, like they are meeting the first time.

[9] https://github.com/facebookresearch/ParlAI/tree/master/parlai/tasks/convai2
[10] https://github.com/npow/friends-chatbot

The *Friends TV Corpus*, first introduced by Nio et al. [NSN+14], was constructed by using subtitles of the popular American *Friends*[11] sitcom. It contains subtitles from five seasons, with a total of 112 episodes. Each episode contains several scenes with several dialog turns. The dataset consists of 60,849 dialogs in total. Figure 4.6 shows some utterances from the dataset.

Actor	Utterance
Monica	There's nothing to tell! He's just some guy I work with!
Joey	C'mon, you're going out with the guy! There's gotta be something wrong with him!
Chandler	All right Joey, be nice. So does he have a hump? A hump and a hairpiece?
Phoebe	Wait, does he eat chalk?
Phoebe	Just, 'cause, I don't want her to go through what I went through with Carl- oh!

Table 4.6: The *Friends TV Corpus* [NSN+14] contains subtitles from the corresponding TV series.

4.4.2 Experimental Results

For evaluation, 1000 combined utterances of the two datasets, *Persona-Chat* corpus and the *Friends TV* corpus, are used. The combined dataset consists of 2313 sentences with 47 sentences containing social relations as defined in section 4.3.1. Within this category, seven sentences returned no relation and 4 returned a false or incomplete relation by the relation extraction system. This results in 11 false and 36 correct extracted sentences with an error rate of 0.31.

Sentences like *"my dad flies airplanes."* or *"That's what my mom said."* could be correctly extracted, even though the relation types *dad* and *mom* are not explicitly defined inside the relationship list. This is possible due to the use of semantic word vectors.

Table 4.7 shows some sentences, that have been erroneous extracted with different causes of error. The cause of a falsely extracted relation varies and is related to different components of the relation extraction system.

The named entity recognizer does not recognize every person as such or falsely identifies some other words as a person. Although the used *Flair* model outperforms previous models in NER tasks, with an F1-score of 93.18% for English and 88.27% for German, entity recognition errors are still feasible.

[11] https://en.wikipedia.org/wiki/Friends

Other errors occur due to the used wording of the relation type. In chat conversation it is common to use short notations or slang for some words, e.g. *gf* for *girlfriend* or *sisses* for *sisters*. Despite the semantic word vector representations used by the word embeddings model, short notations such as *gf* are very difficult to recognize as relation types.

No.	**Sentence**	**Extracted Relation**	**Cause of error**
1	*I exercise at home with my gf*	-	Short notation 'gf' for girlfriend not recognized as social relation
2	*my sister is madonna, crazy right ?*	<'sister-of', 'USER'>	Name 'madonna' not recognized as a person
3	*i actually have four sisses too !*	-	'sisses' is not being recognized as the relation type 'sisters'
4	*no , i want a son so we can watch the tigers my favorite sports team that would be fun .*	<'son-of', 'USER'>	The user utters, that he or her wishes to have a son but actual does not have one.
5	*Hey Pheebs, has Gary ever been shot at for real?*	<'hey_pheebs', 'KNOWS', 'gary'>	'Hey Pheebs' was falsely recognize as a person.
6	*my brother in law is president of paramount pictures what is your favorite show ?*	<'brother-of', 'USER'>	The relation 'brother-in-law' was recognized as 'brother', due to the missing hyphens.

Table 4.7: Sentences which are falsely extracted due to different causes of error.

5 Conclusion

In this work, an approach for extracting social relationships from natural language conversations, as used within chat(bot) messages, is presented. Starting with an overview of current and past approaches for information extraction from unstructured text, an introduction to different methods for named entity recognition, relation extraction, dependency parsing and word vector representations is given. It turns out that, despite recent advances in machine learning, pattern-based approaches for relation extraction are still used, especially to overcome the lack of available datasets in certain domains. In contrast, machine learning has become the norm for named entity recognition, dependency parsing and creating word vector representations, known as word embeddings, in almost any domain.

For storing social relationships, graph database systems seem to displace previous systems, as they are easy to understand and can be simply adapted to various programming languages due to growing availability of object-graph-mapping (OGM) libraries or well documented APIs. Despite the growing popularity of graph databases, RDF triple stores are still widely used and outclass graph databases in terms of standardized query constraints and unique model definitions.

The presented approach for social relation extraction is split into two distinct components, based on a shortest dependency path and a pattern-based method. Due to the diversity of chat messages and the lack of conversational training data, it is still necessary to manually define patterns and machine learning can only leverage certain specific tasks, such as creating word embeddings or recognizing named entities. The chosen approach has shown to be successful at extracting social relations from conversational data, even if the grammatical structure of a sentence is not correct or contains spelling errors. Additionally, different notations of relations can be extracted, due to the use of semantic word vector representations. Beyond that, the shortest path approach is able of extracting relations mentioned after two entities within a sentence.

In future work, the presented method can be expanded and optimized by using more features for classification or taking relations across sentence boundaries into account, using coreferences. The task of coreference resolution has been successfully tackled within certain approaches and might be suitable for adapting to the presented social relation extraction approach.

For more human-like conversations, it is necessary to take the context of the current messages into account. The assumption that the occurrence of a relation within a sentence also represents the real world relation, does not withstand hypothetical or ironic sentences. Besides that, the robustness of the proposed method could possibly be optimized by including more features during extraction and similarity measurement step. Clustering or classifying multiple feature vectors for each relation type might also help to improve the recognition rate.

A Appendix

A.1 Social Relation Extraction

The core function of the social relation extraction, used also as an entry point for the dialogue system, is shown in Algorithm 1. The decision for using one of the two methods, *ShortestPathRE* or *PatternBasedRE*, is based on the extracted entities, stored in the list *personEntities* (row 4).

Algorithm 1 Social Relation Extraction

```
1: function ExtractRelations(utterance)
2:     for all sentences in utterance do
3:         extractedRelations ← nil
4:         personEntities ← ExtractEntities(sentence)
5:
6:         if length of personEntities > 1 then
7:             extractedRelations ← ShortestPathRE(sentence, personEntities)
8:         else
9:             extractedRelations ← PatternBasedRE(sentence)
```

A.2 Shortest Path Relation Extraction

Algorithm 2 Shortest Path Relation Extraction

```
1:  function ShortestPathRE(sentence, entities)
2:      uGraph ← BuildUndirectedGraph(sentence)
3:      spDict ← SearchShortestPath(sentence, entities, uGraph)
4:      extractedRelations ← nil
5:
6:      for all entityPairs in spDict do
7:          shortestPath ← entityPair[value]
8:          wordEmbeddings ← GetWordEmbeddings(shortestPath)
9:          summarizedEmbeddings ← SummarizeEmbeddings(wordEmbeddings)
10:         mostlikelyRelation ← MeasureCosSimilarity(summarizedEmbeddings)
11:
12:         if length of mostlikelyRelation > 1 then
13:             relationTriple ← entityPair, mostlikelyRelation
14:         else
15:             relationTriple ← entityPair, 'KNOWS'
16:         extractedRelations ← relationTriple
```

A.3 Pattern-based Relation Extraction

Algorithm 3 Pattern-based Relation Extraction

```
1:  function PatternBasedRE(sentence)
2:      PosTaggedSentence ← PosTagSentence(sentence)
3:      chunkTree ← ChunkSentence(sentence)
4:      extractedRelations ← nil
5:
6:      for all subTrees in chunkTree do
7:          if label of subTree = REL then
8:              for all words in subTree do
9:                  wordEmbedding ← GetWordEmbeddings(word)
10:                 mostlikelyRelation ← MeasureCosineSimilarity(wordEmbedding)
11:
12:                 if length of mostlikelyRelation > 1 then
13:                     relationTriple ← entityPair, mostlikelyRelation
14:                 else
15:                     relationTriple ← entityPair, 'KNOWS'
16:                 extractedRelations ← relationTriple
```

Bibliography

[ABV18] AKBIK, Alan ; BLYTHE, Duncan ; VOLLGRAF, Roland: Contextual String Embeddings for Sequence Labeling. In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA : Association for Computational Linguistics, August 2018, 1638–1649

[AG00] AGICHTEIN, Eugene ; GRAVANO, Luis: Snowball: extracting relations from large plain-text collections. In: *Proceedings of the fifth ACM conference on Digital libraries - DL '00*. San Antonio, Texas, United States : ACM Press, 2000, 85–94

[AOA15] ALJAMEL, Abduladem ; OSMAN, Taha ; ACAMPORA, Giovanni: Domain-Specific Relation Extraction - Using Distant Supervision Machine Learning:. In: *Proceedings of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*. Lisbon, Portugal : SCITEPRESS - Science and and Technology Publications, 2015. – ISBN 978–989–758–158–8, 92–103

[Bat16] BATISTA, David S.: *Large-Scale Semantic Relationship Extraction for Information Discovery*, INSTITUTO SUPERIOR TÉCNICO, PhD Thesis, 2016

[BM05] BUNESCU, Razvan C. ; MOONEY, Raymond J.: A Shortest Path Dependency Kernel for Relation Extraction. In: *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Stroudsburg, PA, USA : Association for Computational Linguistics, 2005 (HLT '05), 724–731

[BMS15] BATISTA, David S. ; MARTINS, Bruno ; SILVA, Mário J.: Semi-Supervised Bootstrapping of Relationship Extractors with Distributional Semantics. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal : Association for Computational Linguistics, September 2015, 499–504

[Bri98] BRIN, Sergey: Extracting Patterns and Relations from the World Wide Web. In: *WebDB*, 1998, S. 172–183

[CDL+18] CHITICARIU, laura ; DANILEVSKY, Marina ; LI, Yunyao ; REISS, Frederick ; ZHU, Huaiyu: SystemT: Declarative Text Understanding for Enterprise. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*. New Orleans - Louisiana : Association for Computational Linguistics, Juni 2018, 76–83

[CKL+10] CHITICARIU, Laura ; KRISHNAMURTHY, Rajasekar ; LI, Yunyao ; RAGHAVAN, Sriram ; REISS, Frederick ; VAITHYANATHAN, Shivakumar: SystemT: An Algebraic Approach to Declarative Information Extraction. In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden : Association for Computational Linguistics, Juli 2010, 128–137

[CLR13] CHITICARIU, Laura ; LI, Yunyao ; REISS, Frederick R.: Rule-Based Information Extraction is Dead! Long Live Rule-Based Information Extraction Systems! In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA : Association for Computational Linguistics, Oktober 2013, 827–832

[CM14] CHEN, Danqi ; MANNING, Christopher: A Fast and Accurate Dependency Parser using Neural Networks. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar : Association for Computational Linguistics, 2014, 740–750

[FSE11] FADER, Anthony ; SODERLAND, Stephen ; ETZIONI, Oren: Identifying Relations for Open Information Extraction. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK. : Association for Computational Linguistics, Juli 2011, 1535–1545

[GBM03] GIRJU, Roxana ; BADULESCU, Adriana ; MOLDOVAN, Dan: Learning semantic constraints for the automatic discovery of part-whole relations. In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, Association for Computational Linguistics, 2003, S. 1–8

[GJJM05] GUODONG, Zhou ; JIAN, Su ; JIE, Zhang ; MIN, Zhang: Exploring various knowledge in relation extraction. In: *Proceedings of the 43rd Annual Meet-*

ing on Association for Computational Linguistics - ACL '05. Ann Arbor, Michigan : Association for Computational Linguistics, 2005, 427–434

[Hea92] HEARST, Marti A.: Automatic Acquisition of Hyponyms from Large Text Corpora. In: *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*, Association for Computational Linguistics, 1992

[Ior10] IORDANOV, Borislav: HyperGraphDB: A Generalized Graph Database. In: SHEN, Heng T. (Hrsg.) ; PEI, Jian (Hrsg.) ; OEZSU, M. T. (Hrsg.) ; ZOU, Lei (Hrsg.) ; LU, Jiaheng (Hrsg.) ; LING, Tok-Wang (Hrsg.) ; YU, Ge (Hrsg.) ; ZHUANG, Yi (Hrsg.) ; SHAO, Jie (Hrsg.): *Web-Age Information Management* Bd. 6185. Berlin, Heidelberg : Springer Berlin Heidelberg, 2010, S. 25–36

[JM18] JURAFSKY, Dan ; MARTIN, James H.: *Speech and Language Processing*. 3. 2018

[LBS+16] LAMPLE, Guillaume ; BALLESTEROS, Miguel ; SUBRAMANIAN, Sandeep ; KAWAKAMI, Kazuya ; DYER, Chris: Neural Architectures for Named Entity Recognition. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California : Association for Computational Linguistics, Juni 2016, 260–270

[LS17] LASSILA, Ora ; SWICK, Ralph: *Resource Description Framework (RDF) Model and Syntax Specification*. `https://www.w3.org/TR/PR-rdf-syntax/`. Version: 2017

[MBSJ09] MINTZ, Mike ; BILLS, Steven ; SNOW, Rion ; JURAFSKY, Dan: 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 - ACL-IJCNLP '09* Bd. 2. Suntec, Singapore : Association for Computational Linguistics, 2009, 1003

[MCCD13] MIKOLOV, Tomas ; CHEN, Kai ; CORRADO, Greg ; DEAN, Jeffrey: Efficient Estimation of Word Representations in Vector Space. In: *arXiv:1301.3781 [cs]* (2013), Januar. – arXiv: 1301.3781

[MLG11] MINARD, Anne-Lyse ; LIGOZAT, Anne-Laure ; GRAU, Brigitte: Multi-class SVM for Relation Extraction from Clinical Reports. In: *Proceedings of the*

International Conference Recent Advances in Natural Language Processing 2011. Hissar, Bulgaria : RANLP 2011 Organising Committee, September 2011, 604–609

[Niv03] NIVRE, Joakim: An efficient algorithm for projective dependency parsing. In: *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*, Citeseer, 2003

[NSN+14] NIO, Lasguido ; SAKTI, Sakriani ; NEUBIG, Graham ; TODA, Tomoki ; ADRIANI, Mirna ; NAKAMURA, Satoshi: Developing non-goal dialog system based on examples of drama television. In: *Natural Interaction with Robots, Knowbots and Smartphones*. Springer, 2014, S. 355–361

[PSM14] PENNINGTON, Jeffrey ; SOCHER, Richard ; MANNING, Christopher D.: GloVe: Global Vectors for Word Representation. In: *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, 1532–1543

[RWE13] ROBINSON, Ian ; WEBBER, Jim ; EIFREM, Emil: *Graph databases*. "O'Reilly Media, Inc.", 2013

[SJN05] SNOW, Rion ; JURAFSKY, Daniel ; NG, Andrew Y.: Learning syntactic patterns for automatic hypernym discovery. In: *Advances in neural information processing systems*, 2005, S. 1297–1304

[SXZ15] SANTOS, Cicero dos ; XIANG, Bing ; ZHOU, Bowen: Classifying Relations by Ranking with Convolutional Neural Networks. 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)*. Beijing, China : Association for Computational Linguistics, 2015, 626–634

[ZDU+18] ZHANG, Saizheng ; DINAN, Emily ; URBANEK, Jack ; SZLAM, Arthur ; KIELA, Douwe ; WESTON, Jason: Personalizing Dialogue Agents: I have a dog, do you have pets too? In: *arXiv:1801.07243 [cs]* (2018), Januar