FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

State-of-the-Art Approaches for German Language Chat-Bot Development

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Nathaniel Boisgard, BSc
Registration Number 0025323

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.Ing. Dr. Hannes Werthner
Assistance: Mag. Dr. Julia Neidhardt

Vienna, 27th August, 2018

Nathaniel Boisgard

Hannes Werthner

Erklärung zur Verfassung der Arbeit

Nathaniel Boisgard, BSc
Hammerschmidtgasse 5/1/1, 1190 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 27. August 2018

Nathaniel Boisgard

Danksagung

Ich bedanke mich bei meinen Betreuern Mag. Dr. Julia Neidhardt und Prof. Dipl.-Ing. Dr. Hannes Werthner und den Kollegen von der e-commerce Forschungsgruppe für das laufende und immer hilfreiche Feedback und die Unterstützung bei der Fertigstellung dieser Arbeit. Vor allem aber bedanke ich mich bei meiner Familie, die mir durch ihre aufopfernde Unterstützung die Erstellung dieser Arbeit überhaupt erst möglich machte. Vielen Dank.

Acknowledgements

I give my thanks to my advisors Mag. Dr. Julia Neidhardt and Prof. Dipl.-Ing. Dr. Hannes Werthner, as well as my colleagues from the e-commerce research group for the continuing and always helpful feedback and their support towards the creation of this thesis. I am especially grateful for my family, who with their enormous efforts and support were the main reason that I could finish this thesis. Thank you.

Kurzfassung

Chat-Bots sind eines der meistbeachteten Themen der letzten Jahre. Der Hauptfokus der wissenschaftlichen Publikationen, sowie der existierenden Implementationen liegt jedoch auf Chat-Bots in Englischer Sprache. Es ist daher nicht klar, ob die beschriebenen und verwendeten Methoden sowie Werkzeuge auch in anderen Sprachen in der gleichen Art und Weise eingesetzt werden können. Diese Arbeit setzt sich zum Ziel, den aktuellen Stand der Technik im Bereich der Chat-Bot Entwicklung darzulegen, und die Anwendbarkeit der meistverbreiteten und aktuell angewendeten Methoden und Werkzeuge in Deutscher Sprache anhand einer Fallstudie zu evaluieren.

Abstract

Chat-bots have seen increased public interest over the last few years. The main focus of scientific publications, as well as that of existing implementations, however, has been almost solely on English language applications. It is therefore unclear, if the methods and tools mentioned or applied are applicable in languages other than English, and if so, to what degree. The goal of this thesis is to describe the current state-of-the-art in chat-bot development, and to evaluate the applicability of the most widely used state-of-the-art methods and tools in German language with the help of a custom case study.

Contents

CHAPTER 1

Introduction

1.1 Motivation and Problem Statement

Chat-bots have recently gained increased public interest. This can be explained by a multitude of reasons. Increased diffusion of technology into our everyday lives - like the now ubiquitous smartphones or navigation systems in cars - have lead to a renewed interest in natural language based interfaces [Song et al., 2017, Xu et al., 2017, Hearst, 2011, Bang et al., 2015, Braslavski et al., 2017]. The inherent restrictions that come with those small or integrated devices, such as small or no screens and limited input possibilities, showed that common practices of graphic-based user interfaces are not always viable, and that the intuitive and simple nature of speech- or text-only interfaces provide a good answer to that challenge [Fadhil and Villafiorita, 2017, Følstad and Brandtzæg, 2017, Radlinski and Craswell, 2017, Graf et al., 2015, Cimiano et al., 2007]. An alternative approach to speech- and text-based interfaces is presented in [Neidhardt et al., 2015], where the authors use a picture-based interaction to bridge gaps in the user's domain-literacy. Another reason is the steady growth of the world wide web, which demands more efficient and natural interfaces to effectively search and filter the enormous amount of data available to us [Cimiano et al., 2007, Kolomiyets and Moens, 2011], and gives a large number of people access to natural language based systems [Webber and Webb, 2010]. The latter part is especially true when we look at the popularity of instant messaging systems, which have recently surpassed social networks in both size of the user base and usage rates [Intelligence, 2016]. These applications serve as platforms for text-based natural language based systems and contribute a lot to the increased interest in chat-bot technology [Avula, 2017, Graf et al., 2015, Hill et al., 2015]. Steady advancements in natural language processing and information retrieval, especially driven by conferences like "Text REtrieval Conference" (TREC) [Webber and Webb, 2010], combined with access to scalable cloud computing technology have also added to the popularity [Kincaid and Pollock, 2017, Webber and Webb, 2010]. Figure 1.1 visualizes

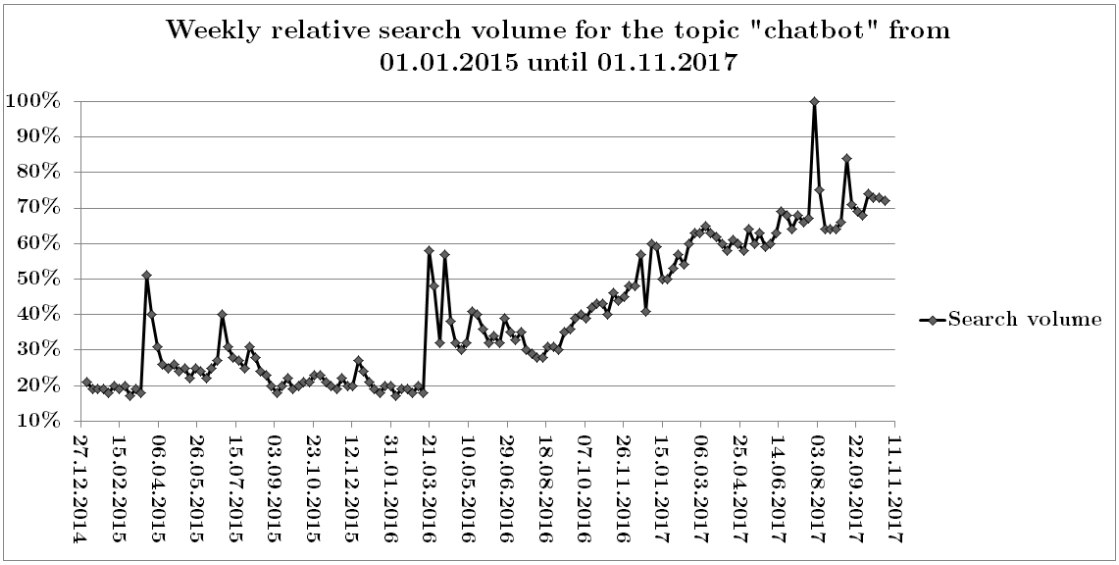


Figure 1.1: The global Google search volume for the topic of "chatbot", relative to the peak interest within the chosen time frame from 01.01.2015 until 01.11.2017. A significant increase in interest can be seen in March of 2016 and a constant growth from that time onwards. Source: `https://trends.google.com/trends/explore?date=2015-01-01%202017-11-01&q=%2Fm%2F01305y`

the significance of the rise in public interest. Google Trends[1] offers data on the relative popularity of a search term or topic within a given time-frame.

This increase in academic and public interest is also reflected, and contributed to by, major releases from big IT companies, like IBM, Google, Apple and Microsoft [Moore et al., 2017, Ameixa et al., 2014]. The most famous examples are IBM's Watson [Ferrucci et al., 2010], a question answering system which has famously participated in and won a game of Jeopardy, WolframAlpha[2], a large-scale expert system with free-form natural language interface, and, more recently, the launch of a new generation of intelligent personal assistants. Some famous examples are Apple's Siri[3], Microsoft's Cortana[4] and Amazon's Alexa[5] system, all of them capable of interpreting natural language utterances in both text and speech. Interest from developers has also been fueled by the releases of easy to use platforms and APIs [Zamora, 2017], to either integrate an existing chat-bot application into an instant messenger ecosystem, like Slack[6] or Facebook Messenger[7], or

[1] `https://trends.google.com/trends/`
[2] `http://www.wolframalpha.com`
[3] `https://www.apple.com/ios/siri/`
[4] `http://microsoft.com/cortana`
[5] `https://developer.amazon.com/alexa`
[6] `https://slack.com`
[7] `https://www.messenger.com`

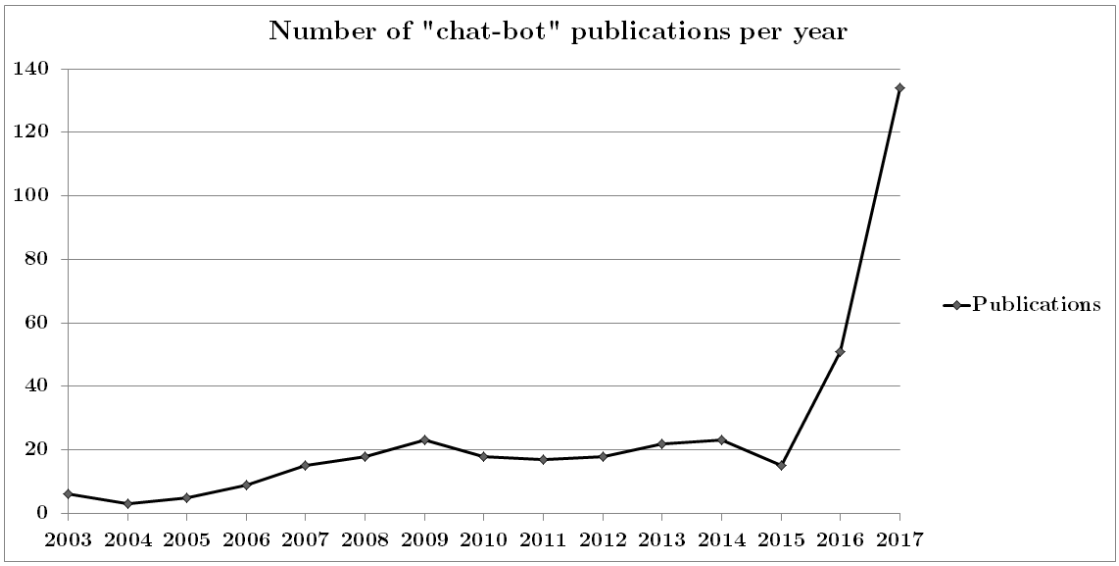


Figure 1.2: The number of publications containing the terms “chatbot”, “chatterbot”, “chat-bot” or “chat bot” released in the years 2000 - 2017 (as of November of 2017). The search was conducted using scopus.com. A significant increase in volume can be seen in the years 2016 and 2017. Source: `http://www.scopus.com`

to create, host and integrate chat-bot applications, e.g. using Microsoft’s Bot Framework[8] or IBM’s Watson Assistant (formerly known as Watson Conversation Services)[9]. Due to the large increase in literature output under that topic, it is hard to get an overview over the current state-of-the-art in the field of chat-bots. As can be seen in Figure 1.2, the number of publications has risen significantly after several announcements in 2016, with the release of Microsoft’s first Twitter bot Tay [Dewey, 2016], based on their Bot Framework, as well as allowing access to Facebook Messenger via a public API [Rosenberg, 2016].

However, most of these publications concentrate on using English as the language of choice, as it is the language of choice in scientific literature [van Weijen, 2012], and the market for English speaking customers is obviously much larger compared to German speaking customers. There is, however, a market for German speaking chat-bots, and to the best of my knowledge there is no literature on the efficiency and practicality of state-of-the-art methods, tools and services used in chat-bot development using German as the language of choice.

[8] `https://docs.microsoft.com/en-us/azure/bot-service/?view=azure-bot-service-3.0`

[9] `https://www.ibm.com/watson/services/conversation/`

1.2 Aim of the Work

The aim of this work is to give an overview of the current state-of-the-art in chat-bot development, and to evaluate the most commonly used methods and tools regarding their efficiency and practicality in both English and German language. Efficiency and practicality in this context means comparable features like availability of tools, documentation and available models for the latter, and performance measures like recall or precision for the former. The goal is to find out whether or not the current state-of-the-art methods and tools deliver a competitive German language performance, and if there are methods or tools which should be avoided or preferred.

1.3 Research Questions

This leads to the following research questions:

RQ1: *Is there a common definition of the term chat-bot, and if so, what is it?*

RQ2: *What is the current state-of-the-art regarding language-dependent methods, tools and services involved in creating chat-bots?*

RQ3: *Is there a measurable difference in the performances of current methods, tools and services when using German language, as compared to English?*

RQ3.1: *If so, are there methods, tools or services for which these differences are smaller or non-existent?*

1.4 Methodology

First, an introduction to the domain and a definition of the used terminology is given. To provide a better overview of the terminology, a literature review is conducted to identify the most commonly used terms and definitions. Then, a taxonomy is derived, defined and visualized using the terms and definitions found in the literature review as a basis.

To find the current state-of-the-art, a thorough literature review is conducted and the methods and tools mentioned in the literature are presented, explained and filtered based on a quantitative evaluation.

Further evaluation of the filtered number of methods and tools is done by implementing a case study as a testing framework. This case study is situated in the sports domain, more precisely Football. The supported languages are English and German. To evaluate the performance and practicality of the implemented methods and tools in both English and German, an evaluation and training dataset is manually created which contains (i) utterances in German language, (ii) the same utterances translated into English language, and (iii) the intent, following a domain-specific taxonomy. This dataset is used to train

a custom implementation called “GermaNLU", using only the most popular methods and tools identified in the aforementioned literature review. This system is then used for a quantitative evaluation to identify the best combinations of methods and tools in both English and German language. The results of this evaluation are then analyzed and discussed.

These performances of the best methods and feature combinations within the “GermaNLU" system are then compared to the performances state-of-the-art publicly available commercial services to provide a point of reference for the performance achieved. These systems are trained using the same training data as the “GermaNLU" system, and also evaluated using the same evaluation data.

The results of this comparison are then analyzed and discussed.

1.5 Structure of the Work

First, we provide an introduction into the terminology used in the field of chat-bots. This is followed by a description of the anatomy of a chat-bot system, presenting the current state-of-the-art. Next, the findings of the literature review regarding the state-of-the-art methods and tools in the field of chat-bot development are presented, followed by a detailed description of the case study implementation, the evaluation results and a closing summary and discussion of the findings. A detailed review of the literature is provided in the appendix, covering existing surveys, as well as detailed descriptions of the analyzed systems. Lastly, examples of training data used to train the state-of-the-art publicly available commercial services used in the final evaluation are also provided in the appendix.

CHAPTER 2

Terminology

As already established, the topic of chat-bots has seen a surge of interest in recent months. What became apparent during the initial research for this thesis was that the meaning of the term is not clearly defined. Definitions given in media and academic publications vary, from being specifically non-task oriented gadgets for entertainment purposes only, to more generic definitions as natural language interfaces to some form of service. But this blurry definition makes it hard to communicate the scope and applicability of a chat-bot. But not only the definition of the term chat-bot is unclear (and sometimes even contradictory to the popular usage of the term), there is also a multitude of different, often synonymously used terms, especially in academic literature. While in popular media systems are most commonly just labeled chat-bots, *"virtual private assistants"*, *"question answering systems"*, *"intelligent personal assistants"* or *"conversational recommender systems"* are just a few of the terms currently used in recent publications. The systems that are described vary heavily in functionalities and scope. In the following we present an overview of current terms and their definitions used in academic literature, and present a novel taxonomy to map these terms and definitions onto a shared and clear terminology. This taxonomy is based on functional attributes which define and distinguish the terms used. Common features are deduced from the literature and assigned to the different terms. Features in this context are all capabilities, system behaviors and attributes attributed to one or more chat-bot systems in the the cited literature.

2.1 Related Work

Very little literature explicitly approach the topic of chat-bot or conversational system classification. [Jurafsky and Martin, 2017a] provide a basic classification of conversational systems into *"non-task oriented"* and *"task oriented systems"*, chat-bots being members of the non-task oriented class, and systems like Apple's Siri being associated to the task-oriented class. It is also stated there, that the actual usage of the term chat-

bot in both media and industry does not always adhere to that classification. [Rubin et al., 2010] aim to classify conversational systems according to their purpose and the environment that they are used in. Initially focused on conversational systems in the context of a library, they define four distinct classes for conversational systems outside of a library environment: (i) educational, i.e. for training and instructional purposes, (ii) informational, i.e. for information seeking or promotions, (iii) assistive, i.e. supporting individuals with disabilities and (iv) socially interactive, i.e. for entertainment or social companionship. [Mazur et al., 2012] do not attempt to introduce a new taxonomy or classification, but argues that the existing classification of conversational systems into task-oriented and non-task oriented (as defined in [Jurafsky and Martin, 2017a]) is not sufficient to describe modern chat-bots, as they are showing traits of both classes. [Klopfenstein et al., 2017] argue in a similar fashion, that the current definition of chat-bots being non-task oriented is not sufficient to describe such systems, as it creates expectations in the users it might not be able to fulfill. The literature found focuses primarily on the shortcomings of the current classification of conversational systems, or suggests a classification with regard to the purpose of a conversational system. To my best knowledge, there is no publication dealing with the unclear landscape of terms used in both academia and media based on a system's features. Therefore, a more clear definition is needed to find out which features are relevant and central in the field of chat-bot development.

2.2 Definitions in the Literature

The following sections summarize the definitions used in scientific literature for terms used in the topic of chat-bots and conversational systems. The terms in use have been identified during the initial phase of literature review as being closely related to each other, and the definitions given for those terms are included to provide a better understanding of the current state-of-the-art in the field of conversational systems.

2.2.1 Conversational Systems

Conversational system, also known as dialog systems, are computer programs which communicate with users using natural language [Jurafsky and Martin, 2017a]. They fall into two distinct classes: *task-oriented conversational systems* and *non-task-oriented conversational systems*.

Task-oriented conversational systems: These are systems *"designed for a particular task and set up to have short conversations"* [Jurafsky and Martin, 2017a], with a *"deep strategic purpose of conversation and directing it to achieve a certain goal"* [Mazur et al., 2012]. These systems attempt to *"get the conversation back on track using its rule base and knowledge obtained during the conversation"* [Mazur et al., 2012]. Examples for such systems are virtual private assistants and conversational recommender systems.

Non-task-oriented conversational systems: These are systems *"set up to mimic the*

unstructured conversational or 'chats' characteristic of human-human interaction, rather than focused on a particular task like booking plane flights" [Jurafsky and Martin, 2017a] and *"mostly for entertainment purposes"* [Mazur et al., 2012]. [Jurafsky and Martin, 2017a] argue that chat-bots belong to this group.

2.2.2 Chat-Bot Systems

Already the term chat-bot itself is not uniquely defined. Amongst the terms used synonymously are *"chatterbot"*, *"chat-bot"* and *"chat bot"*. To get a complete picture of the definitions currently in use, all of these terms were included in the search for literature. The search was conducted using the scientific literature search engine *"scopus"*[1]. An analysis of the literature found showed that besides the previously introduced variations, also a number of different terms are explicitly said to also be synonymous for the term chat-bot (or it's common derivations). The terms and the publications that introduce them as synonymous to chat-bot are listed in Table 2.1.

Alternative Term	**References**
Commercial conversational agent	[Radlinski and Craswell, 2017]
(Intelligent) Virtual assistant	[Kincaid and Pollock, 2017, Dale, 2016]
Digital assistant	[Dale, 2016]
Conversational Interface	[Dale, 2016, Al-Zubaide and Issa, 2011]
Human-computer dialog systems	[Hirzel et al., 2017]
Dialogue system	[Madhu et al., 2017]
Virtual agent	[Hirzel et al., 2017, Madhu et al., 2017]
Chat-oriented dialogue system	[Bang et al., 2015]
Chatting system	[Bang et al., 2015]
Machine conversation system	[Madhu et al., 2017]
Talkbot	[Juang et al., 2015]
Artificial conversation system	[Juang et al., 2015]

Table 2.1: A list of the terms proposed as synonymous to the term "chat-bot" in the literature analyzed.

Some of the terms can be seen as in lieu with the classification used in [Jurafsky and Martin, 2017a], chat-bots being merely chit-chat systems for entertainment purposes (*"chatting system"*, *"talkbot"*, *"chat-oriented dialogue system"*). Some are very unspecific and leave a lot of room for interpretation, e.g. *"virtual agent"*, while others indicate synonymity to the term conversational system itself (*"artificial conversation system"*, *"human-computer dialog system"*) or indicate a task orientation (*"intelligent virtual assistant"*, *"digital assistant"*). The latter two kinds of terms are in direct conflict with the definition used in [Jurafsky and Martin, 2017a]. It can be argued that we are either

[1] `https://www.scopus.com/`

seeing a concept drift of the concept of *"chat-bots"*, or that the academic community is not fully aware of the existing classifications. The chat-bot definitions proposed in the literature also supports the findings from the analysis of the synonyms given. While all sources agree that chat-bots are systems conducting natural language conversations, they do not agree upon the question whether or not a chat-bot is task-oriented or not. Table 2.2 lists the defining features extracted from the literature and aggregates them into 2 groups, supporting non-task orientation and supporting task-orientation.

Non-task Oriented		**Task Oriented**	
Feature	**References**	**Feature**	**References**
Emulate human conversation only	[Ferrara et al., 2016, Song et al., 2017, Bang et al., 2015, Jurafsky and Martin, 2017a, Madhu et al., 2017, Hill et al., 2015, Juang et al., 2015]	Fulfill specific task	[Radlinski and Craswell, 2017, Xu et al., 2017, Kincaid and Pollock, 2017, Fadhil and Villafiorita, 2017, Dale, 2016, Vtyurina et al., 2017, Madhu et al., 2017, Juang et al., 2015]
For entertainment only	[Jurafsky and Martin, 2017a, Papaioannou and Lemon, 2017]	Spread fraudulent content	[Gianvecchio et al., 2011]
Conduct smalltalk	[Hill et al., 2015, Bang et al., 2015, Jurafsky and Martin, 2017a, Wei et al., 2017]	Give assistance and guidance	[Kincaid and Pollock, 2017, Dale, 2016, Juang et al., 2015]
		Give suggestions	[Radlinski and Craswell, 2017, Juang et al., 2015]
		Answer questions	[Radlinski and Craswell, 2017, Xu et al., 2017, Kincaid and Pollock, 2017, Setiaji and Wibowo, 2017, Juang et al., 2015]

Table 2.2: A list of features taken from definitions of the terms chat-bot and chatterbot grouped into features connected to the two types of conversational systems defined in [Jurafsky and Martin, 2017a]. Features in this context are capabilities, behaviors or attributes attributed to one or more chat-bot systems in the cited literature. Chat-bots are assumed to be a non-task oriented system, but a lot of features contradict this assumption.

As can be seen in Table 2.2, some sources give chat-bots features that are in line with

the definitions of both task orientated and non-task oriented behavior, which further supports the issues brought up in [Mazur et al., 2012] and [Klopfenstein et al., 2017] and supports the need for an updated taxonomy.

Related terms and expanded literature analysis: the analysis of the definitions gathered for the term chat-bot has shown that there is an enormous amount of synonymous terms, as well as features that can usually be found in different systems. Therefore the literature search is expanded by common names for systems which are described by said features. Table 2.3 shows a mapping of the expanded set of search terms to the features from Table 2.1.

Search Terms	**Feature**
Conversational recommender system	Give suggestions
Conversational search system, interactive information retrieval	Give guidance, answer questions
Virtual private assistant, intelligent private assistant	Give assistance, answer questions
Question answering system	Answer questions

Table 2.3: A set of additional search terms deduced from the features attributed to chat-bots. The names are either directly taken from the list of synonymous terms in Table 2.1, or identified via an online search for commonly used terms for systems showing those features

In the following the describing features for the above search terms are extracted from the definitions found in the search results. The search was conducted using scopus. Multiple search terms in the same field indicate the use of an "OR" operator in the conducted search, e.g. `"conversational search system" OR "interactive information retrieval"`.

Conversational recommender system: these are systems inspired by conversations of customers with sales persons, with the goal to guide the user through a potentially big product space, and to ultimately present the user with one or more suggestions according to the preferences the system could identify. The preferences of the user are deduced via a multi-step, mixed-initiative natural language conversation, either directly by asking or by inference. Usually these system offers the possibility to make a transaction, e.g. to buy a product. Table 2.4 lists the identified features from the literature and the references.

Feature	References
Inspired by conversation with sales person	[Baizal et al., 2016b, Baizal et al., 2016a]
Mixed initiative (i.e. user and system can both inquire information)	[Baizal et al., 2016b, Genc and O'Sullivan, 2017, Aha et al., 2001, McCarthy et al., 2004, McGinty and Smyth, 2003, De Carolis et al., 2017, Reilly and Reilly, 2004, Mcginty and Smyth, 2003, Bridge, 2002, Allen et al., 1999, Mahmood and Ricci, 2007]
Multi-step dialogue	[Baizal et al., 2016b, Baizal et al., 2016a, Genc and O'Sullivan, 2017, Aha et al., 2001, McCarthy et al., 2004, McGinty and Smyth, 2003, De Carolis et al., 2017, Reilly and Reilly, 2004, Mcginty and Smyth, 2003, Mahmood and Ricci, 2007]
Use conversation to elicit information about the user	[Baizal et al., 2016b, Baizal et al., 2016a, Aha et al., 2001, De Carolis et al., 2017, Mcginty and Smyth, 2003, Mahmood and Ricci, 2007]
Interactive decision process to meet users requirements	[Baizal et al., 2016b, Baizal et al., 2016a, Genc and O'Sullivan, 2017, Aha et al., 2001, De Carolis et al., 2017]
Integrate user feedback	[Baizal et al., 2016b, Baizal et al., 2016a, Genc and O'Sullivan, 2017, Aha et al., 2001, McCarthy et al., 2004, McGinty and Smyth, 2003, Mcginty and Smyth, 2003, Mahmood and Ricci, 2007]
Progressively adapt to the user	[Baizal et al., 2016b, Baizal et al., 2016a, Genc and O'Sullivan, 2017, Aha et al., 2001, McCarthy et al., 2004, McGinty and Smyth, 2003, Mcginty and Smyth, 2003]
Guides the user through the product space	[Baizal et al., 2016b, Genc and O'Sullivan, 2017, McCarthy et al., 2004, McGinty and Smyth, 2003, Reilly and Reilly, 2004, Mahmood and Ricci, 2007]
Navigation by asking	[Shimazu, 2002]
Navigation by proposing	[Shimazu, 2002]

Table 2.4: A list of the features of conversational recommender systems, as found in current literature.

Conversational search system: these are systems that use multi-step, mixed-initiative natural language conversation to improve search quality. They assist the potentially ill-literate user in formulating her/his information need. They differ from normal search engines in providing concise results, if possible a single fact, rather than a list of relevant documents. Table 2.5 lists the identified features from the literature and the references.

Feature	References
Model the user's information need	[Radlinski and Craswell, 2017, Vtyurina et al., 2017, Bickmore et al., 2016, Shiga et al., 2017]
Mixed-initiative	[Radlinski and Craswell, 2017, Vtyurina et al., 2017, Joho et al., 2017, Bickmore et al., 2016]
Use conversation to improve search quality	[Vtyurina et al., 2017]
Multi-step dialogue	[Radlinski and Craswell, 2017, Vtyurina et al., 2017, Joho et al., 2017, Bickmore et al., 2016]
Give concise answers	[Radlinski and Craswell, 2017, Trippas et al., 2017, Trippas et al., 2015]
Assist non-literate users which are unaware of domain vocabulary	[Radlinski and Craswell, 2017, Bickmore et al., 2016]
Use fully formulated sentences as input rather than sets of keywords	[Shiga et al., 2017]

Table 2.5: A list of the features of conversational search systems, as found in current literature

Intelligent personal assistant: also known as virtual private assistants, virtual assistants or digital assistants, these systems provide assistance for various different tasks. Sometimes anthropomorphised, they usually offer search, command & control of applications and devices, question answering and/or recommender system functionalities. They are capable of autonomous and proactive behavior and continuously keep track and assess the user's intent. Most often they offer a speech interface. Famous examples are Apple's Siri and Microsoft's Cortana. Table 2.6 lists the identified features from the literature and the references.

Feature	References
Provide assistance for various different tasks	[Vtyurina et al., 2017, Fonte et al., 2016, Weeratunga et al., 2016, **?**, Cowan et al., 2017, Kumar and Joshi, 2017, Porcheron et al., 2017, Yuksel et al., 2017, Gunaratna et al., 2017, Tintarev et al., 2016, Graus et al., 2016, Chung et al., 2017, Hauswald et al., 2016]
Anthropomorphised to some degree	[Porcheron et al., 2017]
Provide complex, context-rich search	[Vtyurina et al., 2017, Kiseleva et al., 2016, Weeratunga et al., 2016]
Control a device	[Weeratunga et al., 2016, Porcheron et al., 2017, Cowan et al., 2017]
Control services or applications	[Porcheron et al., 2017, Weeratunga et al., 2016, Cowan et al., 2017, Kumar and Joshi, 2017, Graus et al., 2016, Chung et al., 2017, Hauswald et al., 2016]
Usually with speech interface	[Kiseleva et al., 2016, Porcheron et al., 2017, Cowan et al., 2017, Chung et al., 2017, Hauswald et al., 2016]
Control connected devices (IoT)	[Porcheron et al., 2017, Chung et al., 2017]
Answer questions	[Porcheron et al., 2017, Fonte et al., 2016, Cowan et al., 2017, Kumar and Joshi, 2017, Sun et al., 2017, Gunaratna et al., 2017, Graus et al., 2016, Chung et al., 2017, Hauswald et al., 2016]
Make recommendations	[Fonte et al., 2016, Cowan et al., 2017, Kumar and Joshi, 2017, Sun et al., 2017, Porcheron et al., 2017, Gunaratna et al., 2017, Tintarev et al., 2016, Chung et al., 2017, Hauswald et al., 2016]
Act reactively and proactively	[Sun et al., 2017, Yuksel et al., 2017, Graus et al., 2016]
Continuously track and assess the users intent	[Sun et al., 2017, Yuksel et al., 2017]
Show autonomous behavior	[Sun et al., 2017, Yuksel et al., 2017, Chung et al., 2017]

Table 2.6: A list of the features of intelligent personal assistants, as found in current literature.

Question answering system: these are systems giving concise answers to questions formulated in natural language. They use conversation to elicit information on the user. Table 2.7 lists the identified features from the literature and the references.

Feature	References
Answer questions formulated in natural language	[Li et al., 2017, Bouziane et al., 2015, Jijkoun and de Rijke, 2007, Razzaghnoori et al., 2018, Jurafsky and Martin, 2017c, Xie et al., 2017]
Give concise answers	[Bouziane et al., 2015, Jijkoun and de Rijke, 2007, Razzaghnoori et al., 2018, Xie et al., 2017]
Mixed initiative	[Bodoff and Raban, 2016, Li et al., 2016]
Use conversation to elicit information about the user	[Bodoff and Raban, 2016, Li et al., 2016]

Table 2.7: A list of the features of question answering systems, as found in current literature.

2.3 Updated Taxonomy

For this taxonomy, we propose to use already well established terms, extended by the relationships they have to each other. The analysis so far could identify four distinct systems that can be considered chat-bots:

- Non-task oriented conversational system
- Intelligent personal assistants
- Conversational recommender systems
- Question answering systems

Non-task oriented conversational systems is one of the currently used definitions for the term chat-bot, but as we could establish in the previous sections, this is clearly not sufficient. As new term, that is both well-known and descriptive of the chit-chat and generally light nature of such systems, we propose the use of *"chatterbot"* as synonym for "non-task-oriented conversational systems". *"Chatterbot"* is therefore no longer a synonym of chat-bot, but rather a distinct class of systems with focus on non-goal-oriented conversation. What is currently regarded as "task-oriented conversational systems" should be considered synonymous to the term "chat-bot" and is extended by three subgroups: (i) *"intelligent personal assistants"*, (ii) *"conversational recommendation systems"* and (iii) *"question answering systems"*. This updated taxonomy and concept shift more efficiently reflects the common use of the terminology, while maintaining some compatibility to the existing, often insufficient, definitions by using the term *"chatterbot"* as a stand-in for the previous definition of chat-bots. Figure 2.1 shows a visualization of this new taxonomy.

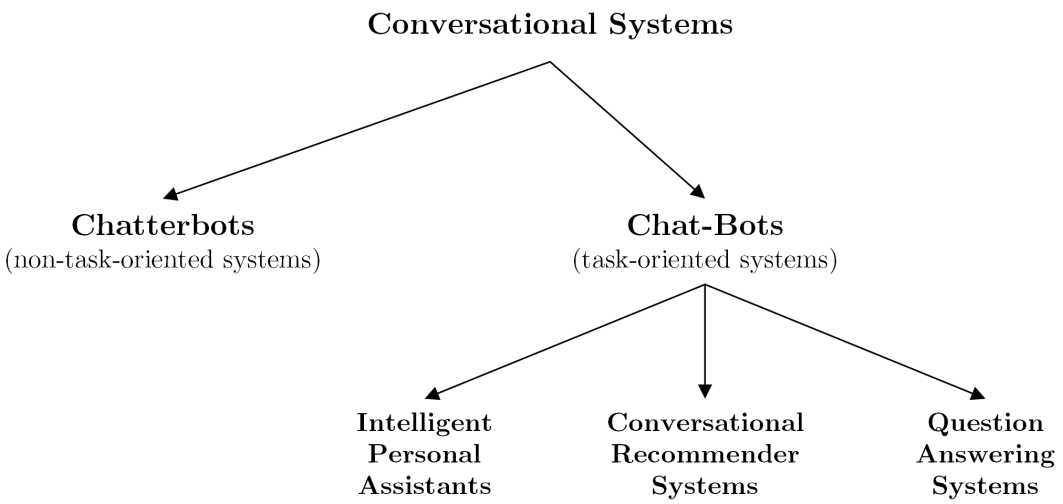


Figure 2.1: A visualization of the proposed taxonomy. It shows the relations between the different terms and their relative position within the taxonomy.

2.4 Summary

It became clear during the research phase that there is no definitive terminology in the field of chat-bots and conversational systems in general. While some classification attempts exist, they do not seem to be thoroughly applied. Especially the term *"chat-bot"* is often used in contradiction to the most common definition of the term, defining chat-bots as conversational systems which do not fulfill a specific task. It is unclear if this definition is unknown to those authors who use the term in a different manner, or that we are witnessing a general change in the way that chat-bots are generally defined and perceived. With regard to the research question 1 (*"Is there a common definition of the term chat-bot, and if so, what is it?"*), the answer is twofold: for one, yes, an explicit definition of the term *"chat-bot"* exists, however, neither scientific literature nor commercial publications are following it, and a large number is even directly contradicting it. Therefore, we come to the second part of the answer: while there is no explicitly stated, official definition for the term *"chat-bot"*, one can be deduced by analyzing the way that chat-bot systems are described in the literature. In doing so, we present a novel taxonomy, with *"chat-bot"* being defined as *"task-oriented conversational system"*, which is, at the most basic level, a software agent capable of interactions using natural language, which is used to fulfill a specific task. Also, several subtypes could be identified during the literature analysis, namely *"intelligent personal assistants"* (e.g. Apple's Siri), *"conversational recommender systems"* (i.e. conversational systems with internal user modeling) and *"question answering systems"* (i.e. conversational systems capable of answering questions formulated in natural language). Combining the updated definition with these subtypes, we can build a novel taxonomy which can support future literature research, as it is more in lieu with the common usage of the terms. In the next chapter,

we will have a look at proposed and existing architectures of chat-bots, and try to deduce a common denominator, which will define what to concentrate on in the search for state-of-the-art methods and tools.

CHAPTER 3

Anatomy of a Chat-Bot

As laid out in the previous sections, neither the term chat-bot itself nor the scope of such systems are clearly defined in the literature. Therefore there is also no ground truth as to how a high level architecture of a chat-bot system should look like. However, a thorough analysis of architectures presented in recent literature shows several shared concepts which can be used to create a unified, high level architecture of a general chat-bot system. In the following this basic architecture is presented. The following section will show the findings of the literature analysis. Section 3.2 gives an overview of the shared architecture (see Figure 3.2), the modules and their interactions. As it would go beyond the scope of this thesis, speech functionalities are omitted, all inputs and outputs are assumed to be text-based, either coming directly from the user or produced by some form of speech-to-text technology.

3.1 Related Work

As the terminology in the domain of chat-bots is still unclear, it is non-trivial to find fitting literature presenting generic architectures. Some publications use the term "conversational system/agent", others use "chat-bots", while others use "question answering system" or "digital assistants". To give an overview of a basic architecture shared amongst all these systems, all of these terms are considered in the literature search. Some publications present a generic high-level architecture, [Sansonnet et al., 2006] for example present a very basic architecture of a embodied digital assistant. [Babar et al., 2017], [Amit et al., 2017] and [Braun et al., 2017] introduce an approach towards a generic chat-bot architecture and give an overview of the possible actions that are happening in each module. [Kolomiyets and Moens, 2011], [Frank et al., 2007], [Damiano et al., 2017] and [Jurafsky and Martin, 2017c] show high level architectures for question answering systems, [Bang et al., 2015] and [Shimazu, 2002] for conversational recommender systems. [Waltinger et al., 2011], [Al-Zubaide and Issa, 2011], [Schwarzer et al., 2016], [Hoque and

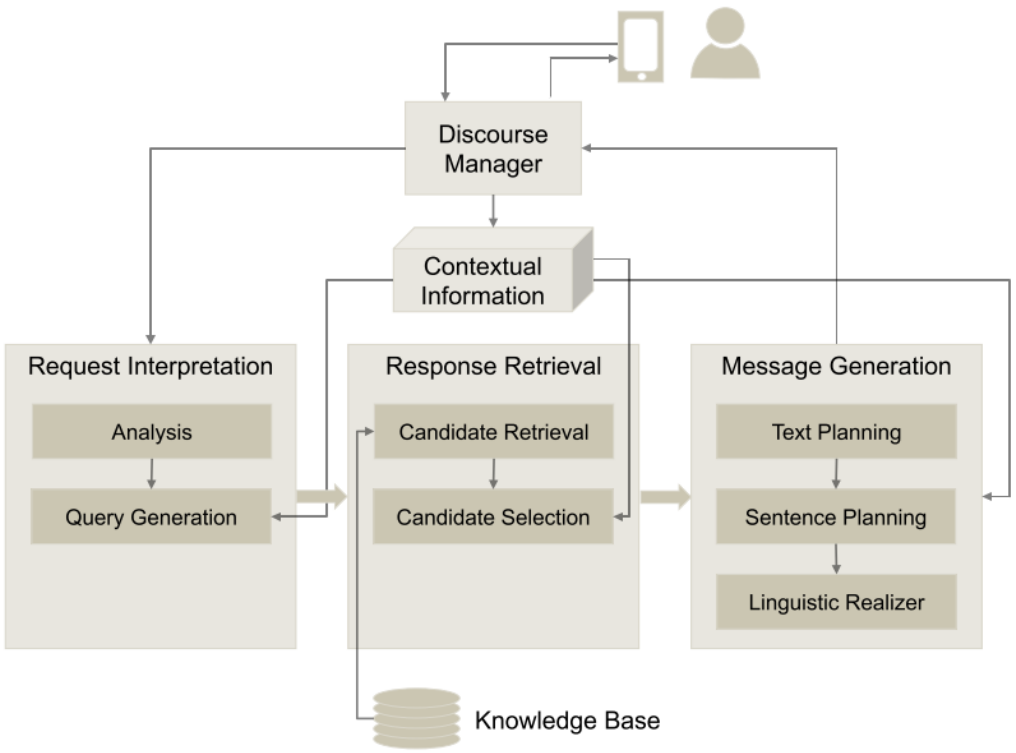


Figure 3.1: Generic chat-bot architecture proposed by [Braun et al., 2017]. It shows in high detail the inner functionalities of each module and their interactions.

Quaresma, 2015], [Frank et al., 2007], [Xie et al., 2017], [Freitas et al., 2011], [Cabrio et al., 2012], [Belyaev et al., 2017], [Damiano et al., 2017], [Kwok et al., 2001], [Lopez et al., 2012], [Lopez et al., 2007] and [Chandurkar and Bansal, 2017] present more specific examples of chat-bot system architectures of implemented systems. [Konstantinova and Orasan, 2013], [Jaya Kumar et al., 2017] and [Hirzel et al., 2017] present very basic architectures of systems using automated speech recognition and speech synthesis components to convert spoken utterances to textual input and to convert textual answers from the chat-bot system to speech, showing where these components are situated within the architectures and how they interact with the other components. An example of a generic chat-bot architecture can be seen in Figure 3.1

3.2 Architecture, Modules and Interactions

Combining the architectures from the literature, some shared concepts emerge. In the following these concepts are presented and merged into modules and explained. Figure 3.2 shows a graphical depiction of the proposed architecture. For example, all

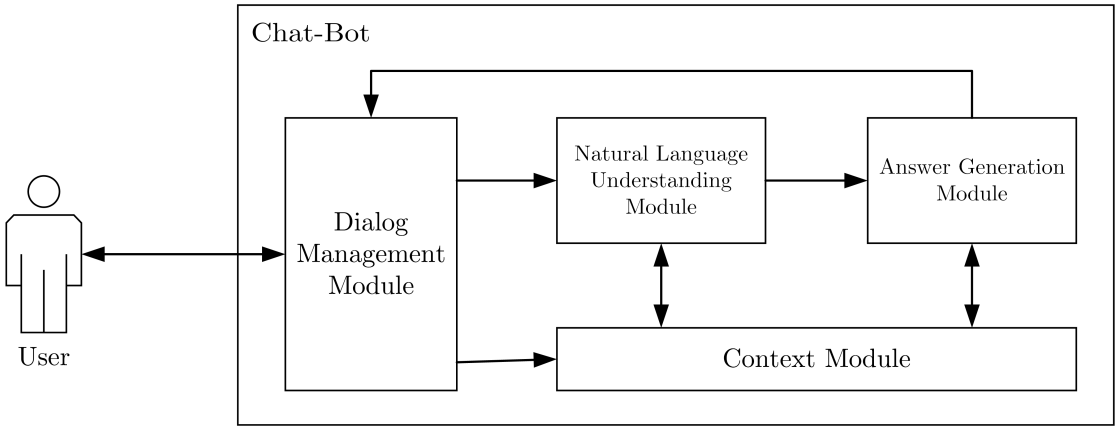


Figure 3.2: Basic unified architecture of a chat-bot system. It is derived from the architectures presented in the cited works. Four distinct modules could be identified, as well as their interactions.

architectures mentioned above contained one or more components to manage the textual inputs generated by a user, and the outputs generated by the system. Besides that, all systems rely on some form of mechanism to interpret these inputs via natural language understanding. How the systems generate the aforementioned outputs varies greatly, but all the systems employ one method or another to generate a textual output depending on the interpretation of the user's inputs. Figure 3.1 for example depicts in more detail that response candidates are retrieved from a knowledge base, a detail which is omitted in this combined architecture. The data sources used are part of the answer generation module, and further details about the nature of this data source is considered out of scope in this context. Some of the more sophisticated systems like conversational recommender systems inherently rely on some form of context that is used in addition to the user's inputs to generate a fitting answer. Also intelligent personal assistants sometimes rely on sensor inputs, data from external services or stored previous conversations to provide better answers. This can be interpreted as a separate "context module", which, depending on the use-case, can range in complexity from a very minimalistic conversation storage, to a highly sophisticated user model.

3.2.1 User

The user provides the chat-bot system with textual input, and receives textual output generated by the system. The input can either be natural language utterances or interactions with structured message types provided by an intermediary service, e.g. lists in Facebook Messenger (see Figure 3.3).

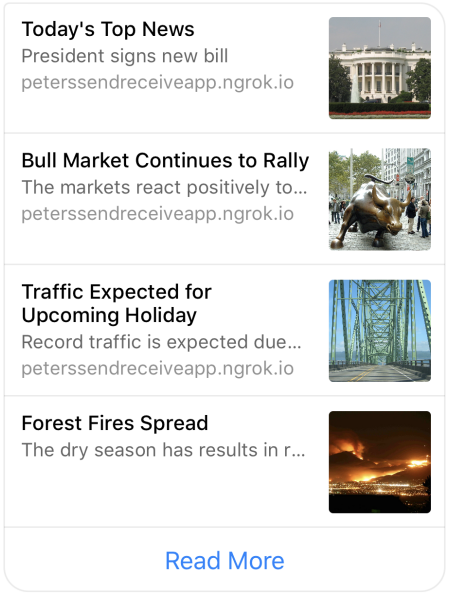


Figure 3.3: Example of a structured list message type provided by Facebook Messenger. The user can select one item, and the selection is communicated to the chat-bot system. Source: List-Template - Messenger Platform (`https://developers.facebook.com/docs/messenger-platform/send-messages/template/list`).

3.2.2 Dialog Management Module

This module is responsible for input/output handling. It manages the utterances received from the user and how to present the answers from the system to the user. In case of structured answers, e.g. user choices from a list of elements, it can update the state of the conversation. It is also responsible to provide the possibility for context switching, i.e. a user changing the focus of a conversation.

3.2.3 Natural Language Understanding Module

This module contains the natural language processing (NLP) functionalities necessary to extract the meaning of a natural language utterance. This includes (i) pre-processing, i.e. steps to clean the input and prepare it for further processing, using e.g. automated spelling correction, (ii) feature extraction, i.e. annotating the utterance with extracted structural and semantic information, e.g. part-of-speech (POS) tags and (iii) intent classification, i.e. to classify the core point of an utterance, e.g. according to a predefined taxonomy. All theses steps can be context-sensitive, i.e. depending on the state of the conversation and the data available about the user, the results can differ. An example would be the disambiguation of named entities depending on previous utterances, e.g. when using relative terms like “before” or “after”.

3.2.4 Answer Generation Module

In this module the annotated input utterance and the user's intent are used to query for potentially fitting answers. These could be facts, a list of recommendations, responses from connected services or retrieved or generated chit-chat utterances. The generation of potential answers can be dependent on the context of a conversation, e.g. generating only recommendations for restaurants in the near vicinity of the user, according to meta-data available from the context.

3.2.5 Context Module

This module functions as the state manager and memory of the systems. It can hold meta-data available about the user, e.g. the user's location or language, the current state of the conversation, e.g. the intent and previous utterances in a multi-step order process, or even a complex user model modeling the preferences of a user to provide fitting recommendations. This module interacts with both the natural language understanding module and the answer generation module, and can vary in complexity.

3.3 Summary

Combining the architectures described in the literature, we could deduce a common, basic architecture which consists of four modules: the dialog management module, which is responsible for input/output handling, the natural language understanding module, which interprets the natural language inputs, the answer generation module, which produces output depending on the interpretation of the input and the state of the application, and the context module, which manages the state of the chat-bot application. It becomes clear, that only the natural language understanding module is consistently dependent on the language the application is used with. While it can be argued that to some extent this also holds true for the answer generation module, which might need to interpret natural language texts to provide a fitting answer, the methods and tools applied there are the same as are used in the natural language understanding module. Many publications even explicitly suggest reusing the same module which is used for interpretation of the natural language input for handling natural language sources in such cases. Therefore, the following analysis of the current state-of-the-art methods and tools will be limited to those involved in natural language understanding. The following chapter will present the findings of this literature analysis.

CHAPTER 4

Methods and Tools: State-of-the-Art

In this section we will present the current state-of-the-art methods and tools used in developing chat-bots which are affected by the language used in the interaction. As previously introduced, the general chat-bot architecture consists of several distinct modules, and most of these modules are not affected by the language used. Relating to the unified architecture shown in Figure 3.2, the only module actually affected by the language spoken by the user is the natural language understanding module (see Section 3.2.3). In the following we will present the findings of a thorough literature review, concentrating on question answering systems. First we will present the reasoning behind choosing question answering systems as the main focus of the literature review. Then we will give a summary of existing surveys in the field of question answering systems, followed by a detailed listing of all analyzed question answering systems, and the natural language understanding methods the authors applied. Since for most of the systems no source code is available, a qualitative analysis of the methods used is considered out of scope in the context of this thesis. The underlying assumption is, that if a method was applied to the final approach presented by the authors, it contributes positively to the overall performance of the system. Finally, we will present the most popular methods and tools used in the analyzed systems in more detail. To measure the popularity of used methods and tools, they are grouped into 8 different groups, following existing classifications where applicable. The popularity score of a method in this context is measured by the number of applications relative to the total number of systems, i.e. if a method group was applied by 9 out of 10 systems, the popularity measure would be 9/10. No further analysis of popularity is made within a method group. The focus of this chapter lies on the natural language understanding methods and tools used in chat-bot development, as they form the very center of every conversational system: to identify the intention behind a natural language utterance and to extract all the necessary information to provide the user with

a meaningful answer. It can be argued, that question answering systems (QAS) are the most basic chat-bot systems. Therefore they were chosen as the main research target, as it has the least "overhead" compared to intelligent personal assistants (IPAs) and conversational recommender systems (CRS). As a reminder, IPAs offer a multitude of functionalities and their most defining trait is some degree of autonomous behavior, as is the interaction with external services or devices. These functionalities lie beyond the scope of this research, as does the focus on conversational capabilities, i.e. the quality of the conversation. CRS are also defined by functionalities beyond natural language understanding (NLU), which is, as is the case with IPAs, a prerequisite. CRS require some form of user modeling, to give the system some idea of a users preferences, as well as some form of projection of those preferences into a complex product space, for which the user requests recommendations. QAS also offer the most commonly used and most concisely defined terminology, which leads to a large quantity of available literature, which also helps in the research effort. These reasons combined offer an explanation as to why the state-of-the-art research focuses on QAS-specific literature. As most of the literature available is about English language systems, this chapter mainly describes the state-of-the-art in English language. The following sections are organized as follows: Section 4.1 is an introduction to the field of natural language understanding (NLU), its definition and relation and/or difference to natural language processing, Section 4.2 will present existing NLU surveys and their relation to this work. Section 4.4 will give an overview of existing QAS surveys, Section 4.5 will give an overview of existing systems, Section 4.6 will focus on existing question answering systems in languages other than English, Section 4.7 on current trends and challenges in QAS development. Lastly, Section 4.8 will give an overview of the methods and tools used for the interpretation of natural language utterances.

4.1 What is Natural Language Understanding?

To properly react to a natural language utterance, a computer system needs to be able to gain a certain understanding of the utterance it was provided with. The process of gaining this understanding is called *"natural language understanding"* (NLU). To extract an understanding from a natural language utterance, one or more natural language processing techniques are applied, e.g. tokenization, lemmatization, part-of-speech tagging, named entity recognition and tree parsing. These techniques provide an NLU system with morphological, syntactic and semantic feature information to infer the meaning of the natural language expression in a structured and repeatable way. Therefore the process of natural language understanding can be split into two steps: (i) extraction of features from a natural language text, and (ii) classification of the natural language text based on the extracted features.

4.2 Existing Natural Language Understanding Surveys

To the best of my knowledge, only one recent survey on natural language understanding methods and tools exists. [Braun et al., 2017] compare the performances of several popular NLU services, namely *"Microsoft LUIS"*, *"IBM Watson Conversation"*, *"API.ai"* (now Google Dialogflow) and the open-source alternative *"rasa"*. They compare the performances when classifying intents and named entities using different English language datasets: (i) Chatbot Corpus, (ii) StackExchange - Ask Ubuntu Corpus and (iii) StackExchange - Web Apps Corpus. The StackExchange corpora were created using the StackExchange Data Explorer. The authors come to the conclusion that *Microsoft LUIS* offers the best performance for all of the corpora used, and that the open-source alternative *rasa* offers competitive performance, outperforming both *Watson Conversation* and *API.ai* in an overall comparison. As this evaluation was only done in English language, it does not provide any insights into potential performance differences when using other languages. It does, however, present an overview of current and popular NLU services.

4.3 Literature Sources for Question Answering Systems

First, we conducted a search for existing survey in the field of question answering systems. we used the search engines Scopus[1] and Google Scholar[2] . The search was conducted between August of 2017 and January of 2018. The search terms used were:

```
("question" AND "answering" AND "system") AND "survey"
("question" AND "answering" AND "systems") AND "survey"
("question" AND "answering" AND "system") AND "overview"
("question" AND "answering" AND "systems") AND "overview"
question answering system survey
question answering systems survey
```

The literature connected to the question answering systems analyzed in the surveys was taken from the corresponding bibliographies. To further the scope, another search was conducted, concentrated on literature on recent question answering systems. Recent in this context means a publication year of 2016 and onwards. The search was conducted between August of 2017 and February of 2018. Again, the search engines Scopus and Google Scholar were used. The search terms were:

```
"question" AND "answering" AND "system"
question answering system
```

[1] http://www.scopus.com
[2] http://scholar.google.com

The search for surveys on the field of question answering systems returned more than 250 results, of which 190 were released in 2010 or later. Of those candidates, 11 surveys were taken into closer consideration, based upon an analysis of the titles and abstracts. Closer inspection lead to a final number of 6 surveys used for this chapter. Most of the 190 initial search results were either out of scope (e.g. about visual question answering, which does not focus on textual inputs), about specific methods or techniques used in question answering (e.g. neural networks), or about a specific subset of question answering systems, like question answering systems in Arabic language. The explicit search for question answering systems provided more than 9000 results, of which 1500 were published in 2015 or later. Only the most recent publications with at least 10 citations were considered for further consideration, which again was based on titles and abstracts. At the time of the initial search this included 50 publications. Again, most of the search results could be considered out of scope in the context of this thesis, as they concentrated on very specific sub-topics like training datasets, or methods used in the field of answer generation, while the focus of this search was on methods and tools used in natural language understanding applied to textual inputs. Out of these 50 candidates, 29 publications were chosen for further consideration. After closer inspection, 16 of those candidates were included in this chapter.

4.4 Question Answering System Surveys

In this section we present a short overview of the findings of the identified QAS surveys. [Liu et al., 2016] give an overview of the different approaches in QAS development, especially noting that rule-based approaches are not flexible enough to compete with more modern, machine learning based approaches. [Höffner and Lehmann, 2017] note that the language specific tasks in QAS development lie in combining natural language processing methods with methods from information retrieval, and that the lack of mature methods and tools is still a problem. [Bouziane et al., 2015] argue, that the type of data source a QAS uses defines the way their natural language understanding pipeline works, structured data sources demanding an approach more akin to transforming the input into a statement in a structured query language, and unstructured data sources demanding more sophisticated focus on information extraction methods. [Diefenbach et al., 2017] give a very thorough overview of the state-of-the-art approaches used in question answering over linked data (QALD). They identify the part of the QAS process that is affected by the language to be the question analysis phase, where a syntactic analysis of the input utterance is executed. Semantics are resolved in a language-independent manner in the following steps, highly intertwined with the data sources used. [Kolomiyets and Moens, 2011] present an overview of QAS using information extraction techniques, therefore focused on unstructured, textual data sources. They present main approaches used, such as “bag-of-words", and define the types of features that can be extracted from a natural language text, e.g. morpho-syntactic features like part-of-speech tags. [Mishra and Jain, 2016] present a classification scheme for QAS, depending - amongst other criteria - on the data source (structured/unstructured) and the domain (open/closed), the question

types (e.g. asking for facts or lists) or the kinds of features extracted (e.g. semantic or syntactic).

Thorough summaries of the mentioned surveys can be found in the appendix.

4.5 State-of-the-Art Question Answering Systems

In this section we will describe the approach used to analyze the question answering systems found either via references from the surveys, or directly via the literature research. Over the course of this analysis over eighty publications have been included and analyzed. As previously established, this thesis concentrates on the natural language understanding parts of the systems, therefore the answer retrieval and presentation techniques were not further analyzed, as they lie beyond the scope of this thesis. The goal of this analysis is to identify methods and tools shared amongst the question answering systems in regard to their natural language understanding pipelines. A method in this context is corresponding to ways how to extract features and how to apply them to aid with natural language understanding within a question answering system. A tool refers to an existing implementation which offers the functionality to apply one or more methods. To achieve this, we relied on the written descriptions of said natural language understanding pipeline elements in the corresponding publications. First, the shared methods were identified. Afterwards a quantitative analysis of the methods and tools was conducted to identify which of them were most often used in the field of question answering systems. No qualitative analysis of the methods or tools was conducted, as it both lies beyond the scope of this thesis, and would be nigh-impossible due to the very different domains as well as often non-available source code and data sets.

4.5.1 Feature Types

To provide the necessary information to interpret a textual natural language input, this information needs to be extracted from the natural language utterances in a repeatable and structured way. These features can be divided into four distinct types: (i) morphological, (ii) syntactic, (iii) lexical and (iv) semantic features. Every system uses at least one of those features, while some use multiple features from every kind.

Morphological features: these set of features concentrates on the surface form of words. These features can be used on a sentence level, were the number or length of words is described, or on a word level, using e.g. normalization, lemmatization or stemming.

Syntactical features: here the “shape” of a sentence is at the center of interest. The syntax of a sentence, the order in which specific words in specific forms are forming a sentence, can be described using different methods. They concentrate on the contextual type of words, their syntactic “role” in the context of the words before and after it, like part-of-speech tags, dependency relations and syntactic chunks or phrases.

Lexical features: in this set of features the connections between words is described. Words can form a hierarchical structure amongst each other, where they share connections of different kinds amongst each other. These can be homonymity/synonymity, hyponymity or hypernymity, depending on their relative position within the hierarchy.

Semantic features: these features try to describe the meaning or semantic role of words or phrases in a sentence. To describe this, several different approaches are used, e.g. named entity recognition, semantic role labeling and word embeddings.

4.5.2 Method Groups

Different methods and tools are used to augment the input text and extract the kinds of features described in the previous section. These features belong to one of the feature types mentioned above, e.g. morphological or semantic. The method groups are based on the chapter structure used in [Jurafsky and Martin, 2018], and new groups were added for methods which were not mentioned in the book. The following groups of methods are used for further analysis (a detailed description of the methods can be found in Section 4.8):

Controlled vocabulary: The use of a defined and finite subset of allowed words. Words outside of this vocabulary are not allowed.

Text normalization: All methods connected to create a cleaned and homogeneous representation of different surface forms of the natural language utterances. This encompasses character mappings, stemming and lemmatization.

Part-of-speech tagging: Assigning labels to tokens depicting their grammatical role within a sentence.

Named entity recognition: The detection and classification of mentions of named entities in an utterance.

Tree parsing: This describes a family of methods using the surface form of a sentence to detect locations and type of segments and their interdependencies within a natural language sentence.

Lexical databases: Manually created databases containing information about relations between words.

Word embeddings: Vector representations of words used to calculate sentence-sentence or word-word similarities.

Handwritten rules: Heuristic rules written by the authors, e.g. mappings of intents to interrogative words or regular expressions to identify common phrases.

4.5.3 Analyzed Systems

In this section we will present the findings of the question answering system analysis. Due to the number of analyzed systems, we will apply an adapted classification following

[Mishra and Jain, 2016], using the application domain and data source type to provide a better overview of the results, as well as ordering the systems by the publishing date of the corresponding publication. The question answering systems were found by using the surveys and the question answering systems that they refer to, and a custom search using scopus.com and Google Scholar. A detailed verbal description of each analyzed system can be found in the appendix. In the following, we will only present the results of the quantitative analysis.

Closed domain systems: This refers to question answering systems which are limited to questions from a specific domain, e.g. pharmacological products or sports results.

Open domain systems: This refers to question answering systems which are not limited to any specific domain, using large-scale knowledge bases to retrieve an answer, e.g. Wikipedia.

Structured data sources: Typically a relational database or triple-store. Basically any knowledge source that can be queried with a structured query language like SQL or SPARQL.

Unstructured data sources: Usually refers to a text-corpus, e.g. documents from the web. Basically any data source which needs some form of natural language preprocessing to be able to be queried.

Table 4.1 shows the method usage in closed domain question answering systems using unstructured data sources.

Table 4.2 shows the method usage in closed domain question answering systems using structured data sources.

Table 4.3 shows the method usage in open domain question answering systems using unstructured data sources.

Table 4.4 shows the method usage in open domain question answering systems using structured data sources.

4.5.4 Findings

As shown in Figure 4.1 and more detailed in Table 4.5, there are 4 method groups that are clearly more popular overall than the others: (i) part-of-speech tagging, (ii) named entity recognition, (iii) tree parsing and (iv) lexical databases. When comparing the relative number of method applications amongst the different categories, especially the categories *"Closed/Unstructured"* and *"Open/Structured"* show big differences. The reasons for that can be seen in (1) the category *"Closed/Unstructured"* contains a relatively large number of very recent systems, as compared to the other categories, and (2) the category *"Open/Structured"* in very much dominated by the *"Question Answering over Linked Data"* (QALD) evaluation campaign hosted by the ISWC[3] and ESWC[4] conferences.

[3] http://swsa.semanticweb.org/content/international-semantic-web-conference-iswc
[4] https://eswc-conferences.org/

System	Year	Contr. vocab.	Text nor.	POS	NER	Tree parser	Lexical DB	Word embed.	Rules
[Burke et al., 1997]	1997		yes				yes	yes	
[Abacha and Zweigenbaum, 2015]	2015			yes	yes	yes			
[Damiano et al., 2017]	2016		yes	yes	yes				
[Schwarzer et al., 2016]	2016			yes			yes	yes	
[Carvalho et al., 2017]	2017		yes	yes				yes	
[Kim et al., 2017]	2017		yes	yes		yes		yes	

Table 4.1: Overview of the usage of the eight methods groups in closed domain question answering systems with unstructured data sources.

System	Year	Contr. vocab.	Text nor.	POS	NER	Tree parser	Lexical DB	Word embed.	Rules
[Green Jr. et al., 1961]	1961	yes		yes		yes			
[Androutsopoulos et al., 1995]	1995					yes	yes		
[Linckels and Meinel, 2005]	2005	yes							
[Wong, 2005]	2005			yes	yes	yes	yes		
[Bernstein et al., 2006]	2006	yes							
[Cimiano et al., 2007]	2007			yes		yes	yes		
[Kaufmann et al., 2007]	2007		yes				yes		
[Lopez et al., 2007]	2007			yes		yes	yes		
[Wang et al., 2007]	2007			yes	yes	yes	yes		
[Damljanovic et al., 2010]	2010			yes		yes	yes		
[Unger and Cimiano, 2011]	2011				yes	yes	yes		
[Pradel et al., 2012]	2012	yes							
[Frost et al., 2014]	2014	yes		yes					yes
[Hamon et al., 2014]	2014		yes	yes	yes	yes			
[Asiaee et al., 2015]	2015		yes	yes	yes	yes			
[Zhang et al., 2016]	2016					yes			
[Saany et al., 2017]	2017			yes		yes		yes	yes
[Šukys et al., 2017]	2017	yes	yes	yes		yes			yes
[Yin et al., 2017]	2017					yes		yes	
[Marginean, 2017]	2017	yes							yes

Table 4.2: Overview of the usage of the eight methods groups in closed domain question answering systems with structured data sources.

System	Year	Contr. vocab.	Text nor.	POS	NER	Tree parser	Lexical DB	Word embed.	Rules
[Moldovan et al., 1999]	1999			yes	yes	yes	yes		
[Harabagiu et al., 2000]	2000			yes	yes	yes	yes		
[Srihari and Li, 2000]	2000			yes	yes	yes			
[Litkowski, 2001]	2000			yes	yes	yes	yes		
[Kwok et al., 2001]	2001			yes		yes	yes	yes	
[Zheng and Arbor, 2002]	2002		yes		yes				yes
[Ferret et al., 2002]	2002			yes	yes	yes	yes		
[Wu et al., 2004]	2004			yes	yes	yes	yes		
[Nyberg et al., 2005]	2005			yes	yes	yes	yes		
[Qiu et al., 2007]	2007		yes	yes		yes			
[Waltinger et al., 2011]	2011		yes	yes	yes	yes			
[Kalyanpur et al., 2012]	2012		yes	yes	yes	yes	yes		yes
[Fader et al., 2013]	2013						yes		
[Berant and Liang, 2014]	2014		yes	yes				yes	
[Freitas and Curry, 2014]	2014			yes		yes			yes
[Tsai et al., 2015]	2015				yes		yes		
[Baudiš and Šedivý, 2015]	2015		yes	yes	yes	yes	yes		
[Sun et al., 2015]	2015								
[Gallagher and Zadrozny, 2016]	2016			yes	yes	yes			
[An et al., 2017]	2017							yes	
[Figueroa, 2017]	2017		yes	yes	yes	yes	yes		
[Hoque and Quaresma, 2017]	2017			yes	yes	yes	yes		
[Romeo et al., 2017]	2017					yes			
[Ruan et al., 2017]	2017							yes	
[Yue et al., 2017]	2017							yes	
[Oh et al., 2017]	2017							yes	
[Belyaev et al., 2017]	2017		yes	yes	yes				
[Khvalchik et al., 2017]	2017		yes	yes		yes			

Table 4.3: Overview of the usage of the eight methods groups in open domain question answering systems with unstructured data sources.

System	Year	Contr. vocab.	Text nor.	POS	NER	Tree parser	Lexical DB	Word embed.	Rules
[Aggarwal and Buitelaar, 2012]	2012			yes	yes	yes	yes	yes	
[Cabrio et al., 2012]	2012				yes				
[Lopez et al., 2012]	2012		yes	yes		yes			yes
[Unger and Bühmann, 2012]	2012			yes	yes	yes	yes		
[Walter et al., 2012]	2012			yes	yes	yes	yes		
[Yahya et al., 2012]	2012			yes	yes	yes	yes		
[Dima, 2013]	2013		yes	yes		yes	yes		
[Giannone et al., 2013]	2013		yes	yes		yes		yes	
[Dima, 2014]	2014		yes	yes	yes	yes	yes		
[He et al., 2014]	2014			yes	yes	yes	yes	yes	
[Höffner and Lehmann, 2014]	2014			yes	yes	yes	yes		
[Park et al., 2014]	2014			yes	yes	yes	yes	yes	
[Xu et al., 2014]	2014			yes	yes				yes
[Zou et al., 2014]	2014			yes	yes	yes	yes		
[Beaumont et al., 2015]	2015			yes	yes	yes	yes		
[Hakimov et al., 2015]	2015			yes		yes	yes		
[Ruseti et al., 2015]	2015		yes	yes	yes	yes	yes		
[Shekarpour et al., 2015]	2015		yes		yes				
[Song et al., 2015]	2015	yes		yes	yes				
[Usbeck et al., 2015]	2015			yes	yes	yes			yes
[Zhu et al., 2016]	2016			yes	yes	yes			
[Nam et al., 2017]	2017		yes		yes	yes			
[Chandurkar and Bansal, 2017]	2017		yes	yes	yes	yes			

Table 4.4: Overview of the usage of the eight methods groups in open domain question answering systems with structured data sources.

Method Group	C/U	C/S	O/U	O/S	Overall
Controlled vocabulary	0%	15%	0%	4%	5%
Text normalization	67%	20%	32%	35%	32%
Part-of-speech tagging	83%	55%	68%	83%	70%
Named entity recognition	33%	25%	57%	83%	55%
Tree parsing	33%	70%	64%	87%	70%
Lexical database	33%	40%	50%	57%	48%
Word embeddings	67%	10%	21%	17%	21%
Handwritten rules	17%	25%	11%	13%	16%
Number of systems	6	20	28	23	78

Table 4.5: Relative number of method group application by QAS category. “C" stands for “Closed domain", “O" for “Open domain", “U" for “Unstructured data source", and “S" for “Structured data source". In the column labeled “Overall” the overall relative application per method group can be found. Notice the differences in word embedding and text normalization usage for the category closed/unstructured, as well as the increased application of named entity recognition methods in the category open/structured.

These challenges require the participating systems to identify and link named entity mentions, therefore all of these systems are using one method or another to reach that goal, explaining the high number of systems applying this method group as compared to the overall number of systems. More recent developments, however, show that especially word embeddings have gained increasing interest. As can be seen in Figure 4.2 and Table 4.6, almost 50% of analyzed question answering systems released in 2017 used word embeddings. The reasons for this can be seen in the increasing popularity of neural networks, which on the one hand use word embeddings of words as inputs, e.g. in [An et al., 2017], while on the other hand are also responsible for the generation of these word embeddings. Especially the release of the word2vec dense vector representations [Mikolov et al., 2013] has led to widespread application in natural language understanding. Another trend that could be observed is the decline in usage of lexical databases in recent years. While one of the most popular methods used in the past, Figure 4.3 and Table 4.6 show a drop in the relative number of systems using lexical databases. Due to the limited sample size, no assumptions towards the statistical significance are made. A possible explanation for this observation is that with increasing focus on multilinguality (e.g. the current QALD-9 challenge, with a task for multilingual question answering[5]), the benefits from using language-specific lexical databases are declining, and therefore the use of such systems is less likely to occur.

[5] https://project-hobbit.eu/challenges/qald-9-challenge/

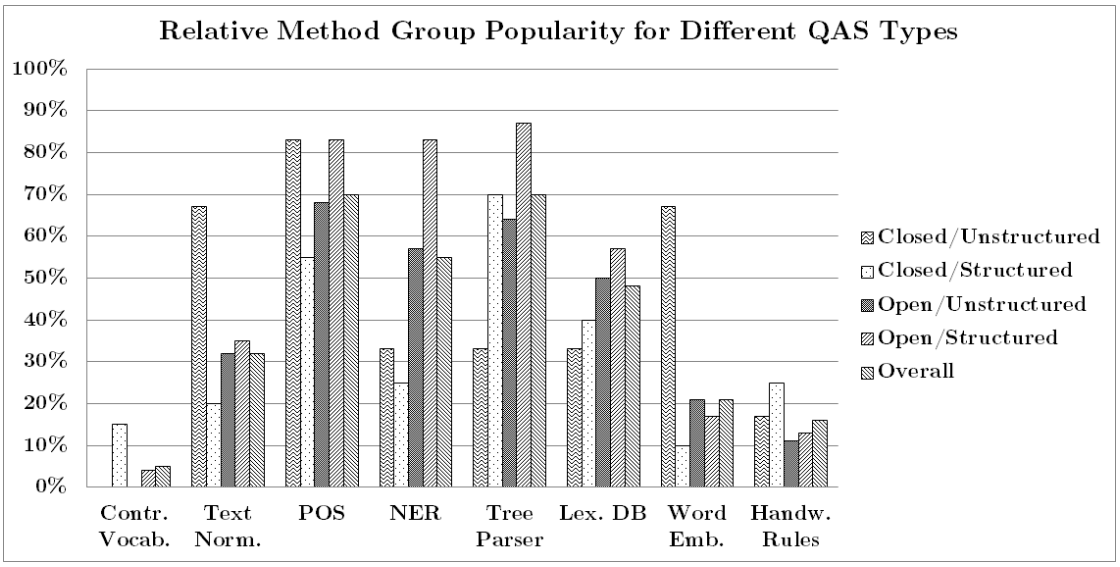


Figure 4.1: Visualization of the relative method group popularity in the different types of question answering systems.

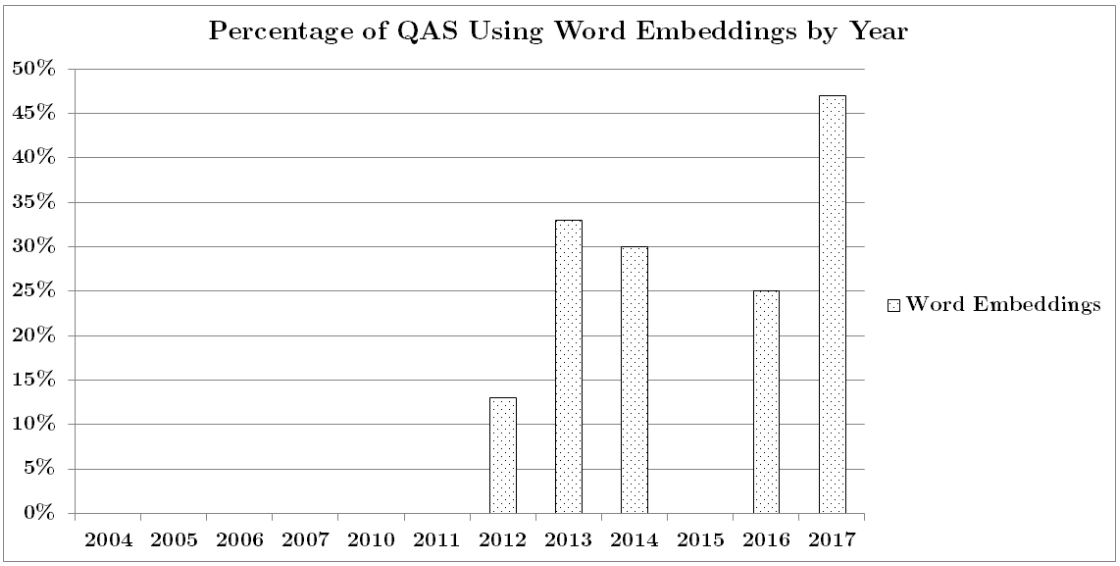


Figure 4.2: Visualization of the trend in relative usage of word embeddings in question answering systems. Note the rise after 2013, which coincides with the release of the word2vec dense word embeddings implementation by [Mikolov et al., 2013]

.

Year	#	Contr. vocab.	Text nor.	POS	NER	Tree parser	Lex. DB	Word embed.	Rules
1961	1	100%	0%	100%	0%	100%	0%	0%	0%
1995	1	0%	0%	0%	0%	100%	100%	0%	0%
1997	1	0%	100%	0%	0%	0%	100%	100%	0%
1999	1	0%	0%	100%	100%	100%	100%	0%	0%
2000	2	0%	0%	100%	100%	100%	50%	0%	0%
2001	2	0%	0%	100%	50%	100%	100%	50%	0%
2002	2	0%	50%	50%	100%	50%	50%	0%	50%
2004	1	0%	0%	100%	100%	100%	100%	0%	0%
2005	3	33%	0%	67%	67%	67%	67%	0%	0%
2006	1	100%	0%	0%	0%	0%	0%	0%	0%
2007	5	0%	40%	80%	20%	80%	80%	0%	20%
2010	1	0%	0%	100%	0%	100%	100%	0%	0%
2011	2	0%	50%	50%	100%	100%	50%	0%	0%
2012	8	13%	25%	75%	75%	75%	63%	13%	25%
2013	3	0%	67%	67%	0%	67%	67%	33%	0%
2014	10	10%	30%	100%	70%	70%	50%	30%	30%
2015	12	8%	33%	67%	83%	75%	42%	0%	17%
2016	4	0%	25%	75%	50%	50%	25%	25%	0%
2017	17	12%	47%	53%	29%	59%	18%	47%	18%

Table 4.6: Development of method group usage over time. The method usage was grouped by year of publication release. Notice the decline of lexical database usage, and the increase in word embedding usage.

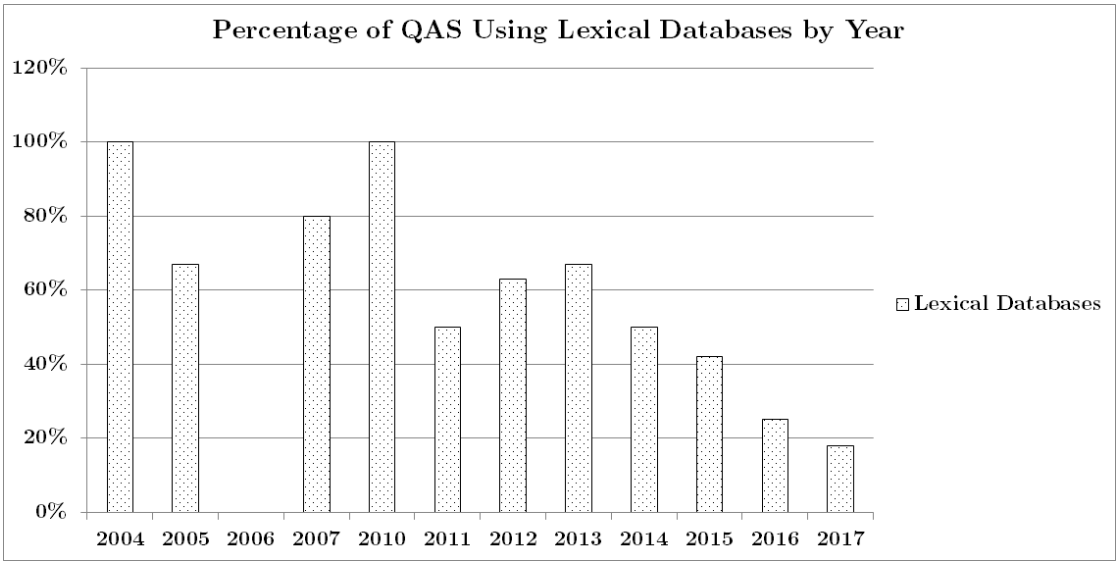


Figure 4.3: Visualization of the trend in relative usage of lexical databases in question answering systems. With increasing popularity of multilingual benchmarks and conference workshops the usage is continuously decreasing.

.

4.6 Question Answering Systems in Other Languages

Only very few of the analyzed systems offer support for languages other than English. Of those system, most of them support German language ([Zheng and Arbor, 2002, Waltinger et al., 2011, Schwarzer et al., 2016]), followed by Italian and Chinese with two systems both ([Zheng and Arbor, 2002, Damiano et al., 2017, Zhang et al., 2017, Ruan et al., 2017]). The other languages were French ([Zheng and Arbor, 2002]), Spanish ([Zheng and Arbor, 2002]), Portuguese ([Zheng and Arbor, 2002]), Russian ([Belyaev et al., 2017]) and Romanian ([Marginean, 2017]), each with one system. Due to the low number of systems supporting other languages, no statements regarding the significance of the difference in relative method group application are being made. Some of the authors regard the lack of mature models as a major issue [Belyaev et al., 2017, Marginean, 2017]. The other publications do not describe any issues regarding the language-specific quality of the methods, tools or services used, therefore any assumption on the impact of the language used on the performance of the system would be speculative at best. In Table 4.7 and Figure 4.4 the distribution of methods can be found.

4.7 Trends and Challenges

One of the findings in the analysis was that the publications regarding natural language understanding and question answering systems are heavily influenced by conferences, workshops and challenges. Especially the *"Text REtrieval Conference"* (TREC) workshop

System	Lang.	Year	Contr. vocab.	Text norm.	POS	NER	Tree pars.	Lex. DB	Word emb.	H.R.
[Zheng and Arbor, 2002]	ENG, GER, FRE, SPA, ITA, POR	2002		yes		yes				yes
[Waltinger et al., 2011]	GER	2011		yes	yes	yes	yes			
[Damiano et al., 2017]	ITA	2016		yes	yes	yes				
[Schwarzer et al., 2016]	GER	2016			yes			yes	yes	
[Zhang et al., 2017]	ENG, CHI	2016					yes			
[Belyaev et al., 2017]	RUS	2016		yes	yes	yes				
[Nam et al., 2017]	KOR	2017		yes		yes	yes			
[Ruan et al., 2017]	CHI	2017						yes	yes	
[Marginean, 2017]	ENG, ROM	2017								yes

Table 4.7: Question answering systems supporting non-English languages and the method groups they are applying. Notice the limited number of only 9, with only 3 of them offering support for German language.

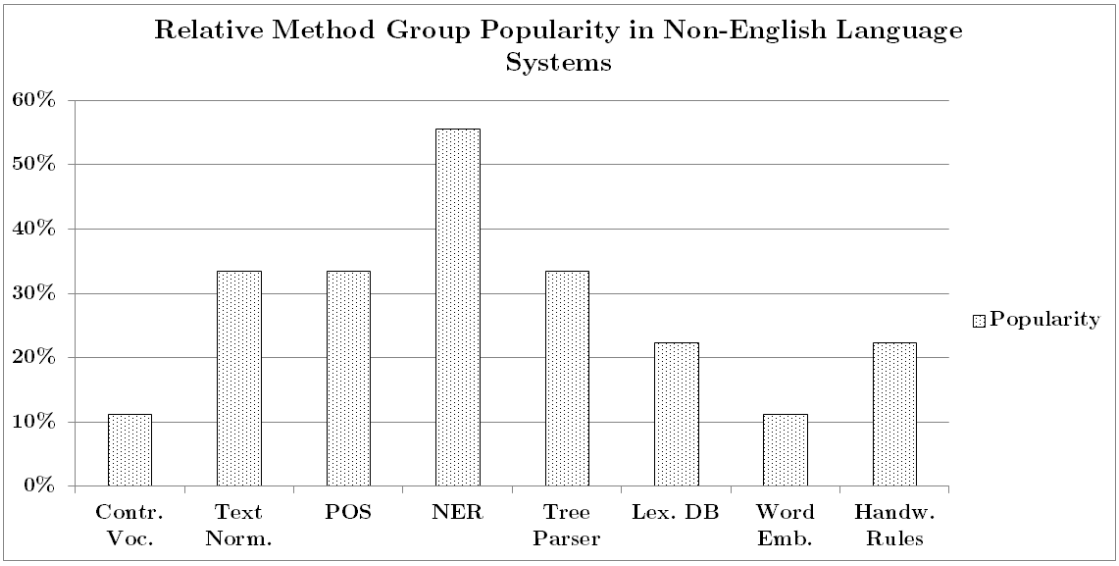


Figure 4.4: Visualization of the relative popularity of the eight method groups in non-English language systems.

series in general, and their question answering track in particular, as well as the *"Question Answering over Linked Data"* (QALD) challenges are connected to a large portion of the publications that were analyzed in the previous sections. Another example would be the *"COmpetition on Legal Information Extraction/Entailment"* (COLIEE), which produced several publications regarding question answering using *Recognition of Textual Entailment* (RTE) methods. Among the conferences that host these challenges and tracks are the *"International Conference on Artificial Intelligence"* (ICAIL), the "International Semantic Web Conference" (ISWC) as well as the *"Extended Semantic Web Conference"* (ESWC). On a more general level targeted towards the field of natural language processing as a whole, the most commonly named are the *"Conference on Computational Natural Language Learning"* (CoNLL), which is especially known for the contributions towards named entity recognition and dependency parsing, and the *"Annual Meeting of the Association for Computational Linguistics"* (ACL) with multiple workshops organized in the field of question answering. The main reason for this heavy influence is the fact that alongside the workshops and tracks that are held at the aforementioned conferences goes the release of high-quality datasets for training and evaluation, providing a shared point of reference. Releases like the CoNLL 2003 dataset for multilingual named entity recognition are to this day a main point of reference for developments in named entity recognition and part-of-speech tagging. Another trend is a shift towards deep learning methods, with complex neural networks being applied to tasks like named entity recognition, word/document embeddings and machine translation. Especially in the field of word and document embeddings the more recent approaches like neural-network-based dense word embeddings like word2vec [Mikolov et al., 2013], FastText [Bojanowski et al., 2016] and GloVe [Pennington et al., 2014], as well as document or sequence embeddings like doc2vec

[Le and Mikolov, 2014], have all but replaced previous approaches such as term frequency - inverse document frequency (TF-IDF) or point-wise mutual information (PMI) weighed vector representations, as well as topic-embeddings like explicit semantic analysis (ESA) [Gabrilovich and Markovitch, 2007]. In the field of named entity recognition, there is a special interest in the application of recurrent neural networks, especially Long-Short-Term-Memory Networks (LSTMN) [Hochreiter and Schmidhuber, 1997], to solve the challenge of correctly labeling sequences of tokens, e.g. in [Chiu and Nichols, 2015]. Another interesting finding was, that with the increased quality web search engines offer nowadays, the application of previously popular approaches using web search engines as data sources for question answering have also gained significant performance boosts. [Tsai et al., 2015] could show that the advances in web search engine performance have rendered most of their natural language processing pipeline useless, if not harmful to the overall performance. Amongst the foremost challenges relevant to natural language understanding currently faced by the question answering systems community are (i) bridging the lexical gap, (ii) dealing with complex queries and (iii) dealing with ambiguity. The lexical gap refers to the situation when different words or phrases have the same meaning, e.g. “automobile” and “car”, or “season following summer” and “autumn”. Different approaches have been used so far, like hand-made lexical databases containing information about synonymitiy relations between words, offering so-called “synsets” for given words. In the example of “automobile” and “car”, the lexical database WordNet [Miller et al., 1990] provides as synset for the word “automobile” the words “car”, “auto”, “automobile”, “machine” and “motorcar”. The issue of complex queries in this context refers to queries which require multiple facts to be resolved. These queries need to be thoroughly understood, requiring complex semantic analysis of the query as well as possibly the data source. Such semantic analysis encompass the identification and correct interpretation of modifier and quantifier terms such as “all”, “non”, “with” or “without”, and their relation to other constituents of the analyzed text,or the resolution of co-references, where typically pronouns such as “she” or “they” are used in place of an entity or object. Ambiguity in this context can be seen as a kind of complementary challenge to the lexical gap, where a single word can have multiple meanings, which is defined by the context the word is used in. An example would be the word “bank”, which can refer to a seating opportunity, a building or a company, the meaning of the word being determined by the context it is used in. The same issue is also present in the challenge of named entity linking, where a mention of a named entity has to be linked to an existing entity in a knowledge base, e.g. in the sentence “Thomas is a striker playing for Munich”, the named entity mentions “Thomas” and “Munich” have to be resolved to the correct entities inside a knowledge base. In the given example, “Thomas” might have hundreds of possible candidate entities in the knowledge base, while only “Thomas Müller” is the correct one, and “Munich” can refer to the location “Munich” as well as the football team “Bayern Munich”. This disambiguation can only be done with some form of knowledge of the context the mentions were used in. This disambiguation is target of ongoing research, e.g. in [Moro et al., 2014].

4.8 Methods and Tools for Natural Language Understanding

In the following section we will present in more detail the most common methods and tools used in the systems analyzed. First, we will start with the most basic approach, bag-of-words, followed by the individual methods and tools used in the methods groups previously established.

4.8.1 Bag-of-Words

In this approach, the input text is considered a set, or "bag", of words, and the presence of a word in the given sentences is used as feature information. The input text is tokenized, and the individual words are added to a set, representing the original text, while punctuation marks are usually ignored. Sometimes the words are also normalized to limit the vocabulary size. Normalization methods are presented in Section 4.8.3. In doing so, any information about the order of the words is lost, as well as possible information contained in the capitalization and surface form of a word, if normalization is applied. An example:

The input text:

E_1 : "$When\ is\ the\ next\ game\ of\ Rapid\ Vienna$?"

will be represented by the set:

$set(E_1)$: ["is", "$game$", "$next$", "of", "$Rapid$", "the", "$Vienna$", "$When$"]

This set representation is then used to calculate similarity to other texts, or to classify a text using a pre-trained classifying algorithm. To calculate those similarities, the set representation is embedded into a high-dimensional vector space, where every word represents a dimension. Going back to the initial example, we consider another use case, in which one would want to classify the utterance *"When is the next game of Rapid Vienna"*. In a very simple setting, we will classify the utterance using only two labeled utterances, in which the labels (in uppercase) represent the class of the utterance:

U_1 : "$When\ is\ the\ next\ game\ of\ Austria\ Vienna$?", $SCHEDULE$

U_2 : "$What\ was\ the\ last\ result\ of\ Austria\ Vienna$?", $RESULT$

The word sets representing those sentences are

$set(U_1)$ = ["$Austria$", "is", "$game$", "$next$", "of", "the", "$Vienna$", "$When$"]

$set(U_2)$ = ["$Austria$", "$last$", "of", "$result$", "the", "$Vienna$", "was", "$What$"]

In the next step, we combine the vocabularies of both examples, resulting in this set containing 12 words:

V = ["$Austria$", "is", "$game$", "$last$", "$next$", ..., "was", "$What$", "$When$"]

Using the vocabulary as the basis for a vector space embedding, the vector representation is built by having each word of the vocabulary representing a dimension, in the example above the first dimension would be representing the word "Austria", the second the word "is", and so on. For every utterance to be embedded into this vector space, for every word that is present in both the the utterance to be embedded and in the vocabulary, the dimension corresponding with the word is set to "1", while all the other dimensions corresponding to words in the vocabulary which were not present in the utterance are set to "0". Therefore the vector representations of the example utterances are:

$vec(U_1) = [1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1]$

$vec(U_1) = [1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0]$

To classify our original example E_1: "When is the next game of Rapid Vienna?", we first also convert it into a set of words, and embed this set into the same vector space as the utterances U_1 and U_2. Notice that the word "Rapid" is not inside the vocabulary, and therefore must be discarded in the embedding process. The vector representation of E1 is:

$vec(E_1) = [0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1]$

Now to classify, the simplest method is to calculate a similarity measure with every labeled utterance, and take the label of the most similar utterance. The most commonly used similarity measures in this context are (i) number of overlapping words, and (ii) cosine similarity, which is the dot product of the vector representation normalized by the product of the magnitudes of the vectors. The closer the value is to 1, the more similar the vectors are considered. In the given example, the number of overlapping words is represented by the function $sim_1(v_1, v_2)$, and the cosine similarity is represented by the function $sim_2(v_1, v_2)$. The cosine of two vectors can be calculated like this:

$$cos(v_1, v_2) = \frac{< v_1, v_2 >}{||v_1||.||v_2||}, with\ ||v|| = \sqrt{< v, v >} \tag{4.1}$$

Using the above equation 4.1, the similarity measures calculated for the input vector $vec(E_1)$ and the vector representations $vec(U_1)$ and $vec(U_2)$ of the labeled examples U_1 and U_2 are:

$sim_1(E_1, U_1) = 7$

$sim_1(E_1, U_2) = 2$

$sim_2(E_1, U_1) = 0.94$

$sim_2(E_1, U_1) = 0.29$

Both similarity measures in this very simple example would lead to the classification of the utterance "When is the next game of Rapid Vienna?" with the label "SCHEDULE", as both similarity measures show a higher value for the similarities between vector $vec(E_1)$ and $vec(U_1)$ than for the similarities between vectors $vec(E_1)$ and $vec(U_2)$. In

a real-world implementation, one would obviously use much more labeled examples to cover more possible inputs. Another possible implementation would be to not calculate the individual similarities between the input and all labeled examples, but to use the labels and vector representations of the labeled examples to train a machine learning algorithm such as Naive Bayes [Maron and Kuhns, 1960] to classify the input vector.

Known Issues: This basic approach, while very simple and easy to implement, does not work well in some use cases, as the order of the words and therefore the semantic information included in this, is lost. An example of such a problem would be these two utterances, and the assumption that lowercase normalization is applied:

$set(\text{“}This\ is\ a\ good\ toy,\ dog.\text{”}) = [\text{“}a\text{”}, \text{“}dog\text{”}, \text{“}good\text{”}, \text{“}is\text{”}, \text{“}this\text{”}, \text{“}toy\text{”}]$

$set(\text{“}Is\ this\ a\ good\ dog\ toy?\text{”}) = [\text{“}a\text{”}, \text{“}dog\text{”}, \text{“}good\text{”}, \text{“}is\text{”}, \text{“}this\text{”}, \text{“}toy\text{”}]$

As can be seen, the set representation of the two words are similar (as would their vector representations be), while the two sentences have very different meanings.

Another issue presented by this approach can be visualized by the following example:

$E_2 : \text{“}How\ much\ is\ this\ car?\text{”}$

$E_3 : \text{“}What\ is\ the\ price\ of\ this\ automobile?\text{”}$

$set(E_2) = [\text{“}car\text{”}, \text{“}how\text{”}, \text{“}is\text{”}, \text{“}much\text{”}, \text{“}this\text{”}]$

$set(E_3) = [\text{“}automobile\text{”}, \text{“}is\text{”}, \text{“}of\text{”}, \text{“}price\text{”}, \text{“}the\text{”}, \text{“}this\text{”}, \text{“}what\text{”}]$

$V = [\text{“}automobile\text{”}, \text{“}car\text{”}, \text{“}how\text{”}, \text{“}is\text{”}, \text{“}much\text{”}, \text{“}of\text{”}, \text{“}price\text{”}, \text{“}the\text{”}, \text{“}this\text{”}, \text{“}what\text{”}]$

As can be easily seen just by looking at the sets and vector representations, the similarity measures above would not be applicable to measure the semantic equivalence of the two utterances. While they have the same meaning, in both cases someone wants to know the price of a car, the vocabulary used is so different, that a bag-of-words approach without further augmentation could not bridge this so-called “lexical gap” (see also Section 4.7). No specific tools were used in the analyzed literature to apply this method.

4.8.2 Controlled Vocabulary

This methods refers to the limitation of the words that are allowed inside the input text. While in an open vocabulary approach, any word is allowed inside the input text, in a controlled vocabulary approach only a limited subset of natural language is allowed. This approach is used to prevent problems that come with the ambiguous nature of natural language, like the previously discussed lexical gap. To refer to the previous example with the words “car” and “automobile”, if only the use of the word “car” was allowed, this particular problem would not be an issue. There are several approaches how this controlled vocabulary is build: (i) using a pre-existing general purpose vocabulary like the “Semantics of Business Vocabulary and Business Rules” (SBVR) [Šukys et al., 2017], (ii) extracting the vocabulary from a knowledge base [Bernstein et al., 2006], or

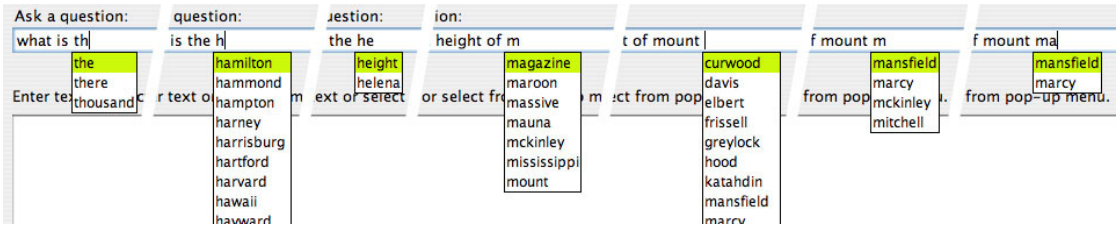


Figure 4.5: Example of the application of a controlled vocabulary approach using a naive auto-complete functionality. Source: [Bernstein et al., 2006]

.

(iii) a combination of both approaches, where a limited set of predefined vocables is combined with the extracted ones [Marginean, 2017]. Different approaches are also used in how the user is presented with this limitation, [Bernstein et al., 2006] use a naive auto-complete functionality, where the user is presented with all the possible vocables that share the same prefix (Figure 4.5), whereas [Song et al., 2015] present a more intuitive and intelligent auto-complete functionality, where only those vocables are suggested, which are allowed following a step of logical inference, where the existing types and relations from the underlying ontology are evaluated against the user input. Another approach is used by [Šukys et al., 2017], where the user's input is ingested as written by the user, and if one or more words in the input could not be mapped to a word from the controlled vocabulary, the user is asked to clarify these words by either defining it as a synonym of an existing vocable, mark it so that the system ignores it, or explicitly state it to be a proper name. No specific tools were used in the literature analyzed to support in the use or generation of a controlled vocabulary except the use of SBVR, which, however, does only refer to a defined standard, not an implementation.

4.8.3 Text Normalization

The goal of this group of methods is to augment and/or annotate the individual words in such a way, as to help reduce the ambiguity that may be faced in following processing steps. But while the previously presented approach attempts to minimize the word ambiguity by reducing the choices a user can make in formulating her intent, normalization methods take free form input. Multiple different methods are used in text normalization, namely (i) character mapping, (ii) stemming, and (iii) lemmatization. Each of these steps either replaces or appends as an annotation to the original word a simplified version of the original word. [Jurafsky and Martin, 2017d] does also include tokenization and sentence segmentation in text normalization.

Character mapping: this refers to a technique in which one or more characters are replaced or removed from the input to make further processing easier. This is usually solved using regular expressions [Jurafsky and Martin, 2017d]. One example would be to replace special characters like ampersand ("&") with the word "and", or umlauts like "ä" or "ö" with either their non-umlaut counterparts like "a" or "o", or with "ae" or "oe".

```
tokens = ["The","cats","are","loose","!","Who","was","that","?"]
stemmer = PorterStemmer()
stems = [stemmer.stem(token) for token in tokens]

print(' '.join(tokens))
print(' '.join(stems))
```

```
The cats are loose ! Who was that ?
The cat are loos ! Who wa that ?
```

Figure 4.6: Visualization of the stemming of an example text using the NLTK Porter Stemmer. Notice that the words "was" and "are" do not share the same stem.

One reason for this is to capture the semantics better (in the case of the ampersand), or to mitigate possible encoding issues when using characters outside the standard ASCII table.

Stemming: this refers to a technique in which a word is reduced to a naive base form, by using simple heuristics to cut off suffixes or words. For example the word "cats" would be reduced to the stem "cat", a stem shared with the word "cat" itself. This is especially useful in morphologically simple languages like English, where nouns know only two surface forms - one for singular, and one for plural. This naive approach, however, does come with some drawbacks, as it relies on naive heuristics, which do not over complex surface forms of the same word, for example the word "be", which can have multiple different forms like "was" or "are". These different surface forms do not share a common stem, and naive stemmers are not able to identify the shared morpheme of those words. To give an example, the popular *"Porter stemmer"* [Porter, 1980] stems the word "was" down to "wa", and the word "are" down to "are". The most popular tools used for stemming in the analyzed literature were Stanford CoreNLP [Manning et al., 2014] and OpenNLP[6].

Lemmatization: this refers to a more complex version of stemming, in which a word is reduced down its basest form, the morpheme. While for the above example of the words "cat" and "cats" the result would be the same as with normal stemming - the morpheme "cat" - proper lemmatization helps also with more complex morphological forms. The words "are", "was" and "am" would all be lemmatized to its morpheme base of "be". This, however, requires both in-depth lexical and grammatical knowledge of the language to which it is applied. The most popular tools used to apply lemmatization in the analyzed literature were Stanford CoreNLP, TreeTagger [Schmid, 1994], LanguageTool[7], NLTK WordNetLemmatizer [Bird and Loper, 2004], OpenNLP and GATE [Cunningham et al., 2001].

[6] https://opennlp.apache.org/
[7] https://languagetool.org/

```
pos = nltk.pos_tag(nltk.word_tokenize("The cats are lose! Who was that?"))
lemmatizer = WordNetLemmatizer()
lemmas = []

for word,postag in pos:

  wn_tag = penn_to_wn(postag)

  if wn_tag != None:
    lemmas.append(lemmatizer.lemmatize(word,penn_to_wn(postag)))
  else:
    lemmas.append(word)

print(' '.join(tokens))
print(' '.join(lemmas))
```

```
The cats are loose ! Who was that ?
The cat be lose ! Who be that ?
```

Figure 4.7: Visualization of the lemmatization of an example text using the NLTK WordNet Lemmatizer [Bird and Loper, 2004]. Notice that the words “was” and “are” share the same stem.

4.8.4 Part-of-Speech Tagging

Part-of-speech tagging refers to a technique which assigns a label to every word within a sentence depicting its syntactic category or word class in the context of the given sentence. The most commonly used word classes are taken from the Penn treebank [Marcus et al., 1993]. In this tagset 45 different word classes and their corresponding and abbreviated tags are defined, e.g. singular proper nouns (NNP), past tense verbs (VBD) or adjectives (JJ). Figure 4.8 depicts the application of part-of-speech tags to an example text using the Stanford CoreNLP part-of-speech tagger. The class of a word is useful information which can be used to infer the neighboring words, as well as their general syntactic structure [Jurafsky and Martin, 2016a]. This information is also used by other feature extraction methods like lemmatization (see Section 4.8.3) or named entity recognition, as the information about the word class helps in the disambiguation of ambiguous word like “land”, which can be both an action (as in “land the account”) or an entity (as in “land of the free”). The most common techniques used to apply part-of-speech tags to a given text are hidden Markov models and maximum entropy models, both trying to apply the most likely sequence of tags to a given sequence of tokens. The most commonly used tools for applying part-of-speech tags in the literature analyzed were the Stanford CoreNLP POS Tagger, TreeTagger, SENNA [Collobert et al., 2011], the HipHep Tagger used by ANNIE (GATE) [Dimitrov, 2002], OpenNLP, clearNLP [Choi and Palmer, 2011] and RASP [Briscoe and Carroll, 2002].

Hidden Markov models: a hidden Markov model (HMM) assumes that an unobserved stochastic process is the underlying cause for an observed series of events. In the case of part-of-speech tagging, the observed events are the individual words of a sentence, and

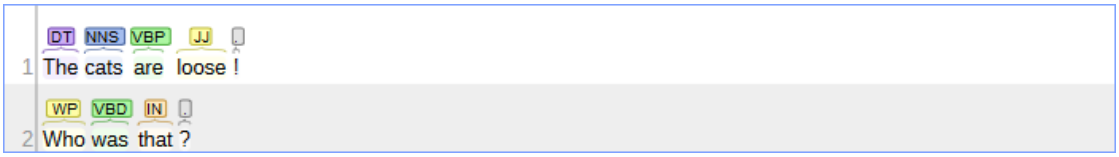


Figure 4.8: Visualization of the application of the Stanford CoreNLP part-of-speech tagger to an example text. The tags are following the Penn treebank tagset.

the unobserved - or hidden - events are the part-of-speech tags, which explain why a certain words are following each other. Another, more obvious example of such a causal connection between observed and hidden events is given by [Jurafsky and Martin, 2016a], in which the observed events are a diary, containing information about the number of ice creams eaten, and the unobserved event being the temperature, which, while being unobserved, can be deduced by assuming that a higher temperature will result in a larger quantity of ice creams eaten, while a lower temperature will result in a lower amount. The basic goal of a HMM-based part-of-speech tagger is to infer the most likely label series for a given series of words. They also make two simplifying assumptions: the (i) independence and (ii) bigram assumption. The first assumes that the probability of a word appearing only depends on its own tag, and not on anything else, and the latter states that a tag (or label) is only dependent on its immediate predecessor, and nothing else. An annotated corpus is used to train the emission probabilities $P(w_i|t_i)$, which denote the probability of observing word w_i given the hidden state t_i, and the transition probabilities $P(t_i|t_{i-1})$, which denote the probabilities of the hidden process transitioning to state t_i given the previous state t_{i-1}. These emission and transition probabilities are used to calculate the joint distribution of a sequence of observations (e.g. a sequence of words W) and a sequence of hidden states (e.g. a sequence of tags T) (see Equation 4.2). This joint distribution can then be used to calculate the optimal sequence of hidden states $\hat{T}$ best explaining the observed sequence (see Equation 4.3.

$$P(W,T) = P(W|T) \times P(T) = \prod_{i=1}^{n} P(w_i|t_i) \times \prod_{i=1}^{n} P(t_i|t_{i-1}) \tag{4.2}$$

$$\hat{T} = \arg\max_{T} P(W,T) \tag{4.3}$$

Maximum entropy Markov model: a maximum entropy Markov model (MEMM) is an adaption of the MaxEnt classifier [Adwait Ratnaparkhi, 1996]. In contrast to a HMM, a MEMM computes the posterior probabilities (i.e. how likely is sequence of tags T given a sequence of words W) directly. The following Equation 4.4 and Equation 4.5 depict how a MEMM discriminates the best possible tag sequence $\hat{T}$:

$$P(T|W) = \prod_{i=1}^{n} P(t_i|t_{t-1}, w_i) \tag{4.4}$$

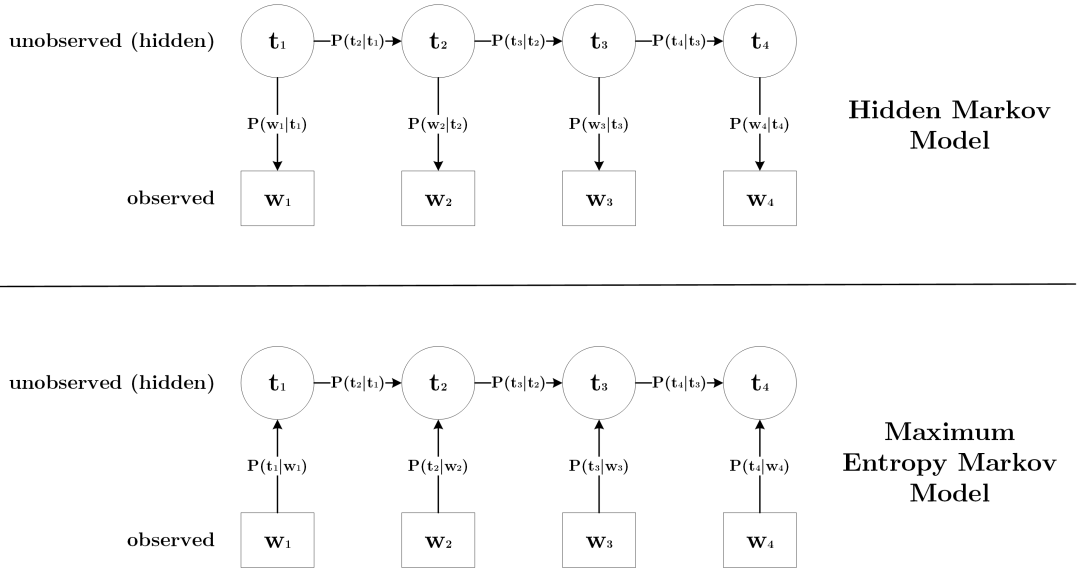


Figure 4.9: Trellis diagram visualizations of the main difference between a hidden Markov model (HMM) and a maximum entropy Markov model (MEMM). The edges correspond to the terms that define the joint/conditional probabilities of the hidden and observed states. Notice the direction of the transition between tags and words: in contrast to HMMs, MEMMs are able to calculate the posterior probabilities directly, i.e. it is a discriminative model, as opposed to the generative HMM, which "generates" the observed states using the hidden states. $P(t_{k+1}|t_k)$ is the transition probability of transitioning from a specific state t_k to another state t_{k+1}, $P(w_k|t_k)$ and $P(t_k|w_k)$ are the emission probabilities for a specific observed state w_k given a specific hidden state t_k or vice versa.

$$\hat{T} = \arg\max_{T} P(T|W) \tag{4.5}$$

In practice however, a MEMM-based tagger does not only include the current word and the previous tag (like a HMM-based tagger), but usually looks at all the words and tokens within given windows sizes, backward looking and forward looking for words, and backward looking for tokens. A MEMM, like a HMM, is trained using an annotated training set. Figure 4.9 visualizes the basic difference between a hidden Markov model and maximum entropy Markov model.

4.8.5 Tree Parsing

This section refers to a family of methods using tree parsing to produce tree-like representations of an input sentence based on its syntactic structure. Tree parsing in this context refers to the identification and classification of segments containing valuable information and their interdependencies [Jurafsky and Martin, 2017b]. There are two types of tree representations that such parsers usually produce: (i) constituency based trees and (ii)

dependency based trees. The type of representation a parser produces is defined by the type of grammar that is used to create the tree structure, constituency trees are usually defined by context free grammars, while dependency trees are defined by dependency grammars.

Constituency based trees: they represent the phrases - or chunks - of a sentence, and the hierarchy within those phrases. The most simple approach is to divide a sentence into non-overlapping segments and classify them. This approach is called "chunking". The classes usually used in such a representation are verb-phrases (VP) and noun phrases (NP) and such phrases can contain one or more words. An example for a noun phrase would be "the cat", which consists of a determiner ("the") and a noun ("cat"). A verb phrase in the context of a chunker is one or more verbs, an example would be "has awoken". In the complete example sentence "The cat has awoken" the basic chunker output could be represented using a simple bracket notation, e.g. `[(NP) The cat] [(VP) has awoken]`. More complex implementations of constituency based representations allow for overlapping, and so result in a tree-like representation. These chunks are built based upon a predefined grammar consisting of rules how - based on their word class - words can be combined to form valid chunks. An example of such a rule would be $NP \rightarrow Det\ Nominal$ as well as $Nominal \rightarrow Noun$ which are the rules that define the noun phrase "a cat", consisting of a determiner and a noun. Figure 4.10 depicts a constituency tree of the example sentence "The cat has awoken". Constituency trees are especially helpful in named entity recognition, as entity mentions are most often reflected by noun phrases, and therefore can help identifying or validating mention detection. The tools most often used to create constituency trees in the literature analyzed were Stanford CoreNLP, ASSERT [Pradhan et al., 2004], TreeTagger and Cymfony [Srihari and Li, 1999].

Dependency based trees: based on dependency grammar they represent the connections and dependencies between the individual words within a sentence. These dependencies are defined by the parser, the Stanford CoreNLP dependency parser uses the Stanford Dependencies [De Marneffe and Manning, 2008], consisting of 50 grammatical relations like *"nominal subject"* (nsubj) and *"determiner"* (det). The dependencies follow a predefined formalism, guaranteeing that the resulting graph is acyclic and directed, with only one root and only one path from the root to each vertex. This tree can be created using a "shift-reduce parser" [Jurafsky and Martin, 2017b], in which each word is shifted onto a stack, and for the last two words the parser asks a so-called "oracle" to check if an action can be applied (e.g. if a grammatical relation is defined for those two words). If an action can be applied, the two words are replaced by the word with the outgoing edge representing the relation, and the next word is pushed onto the stack, repeating the cycle until no further words can be pushed onto the stack and no more action can be applied. To illustrate, see Table 4.8 for a simplified trace of the parse of the example sentence "The cat has awoken.". Figure 4.11 illustrates the resulting tree. The previously mentioned oracle, which defines the actions to be taken by the parser is trained using supervised machine learning and an annotated training corpus to create transition mappings which define which actions the oracle advises the parser

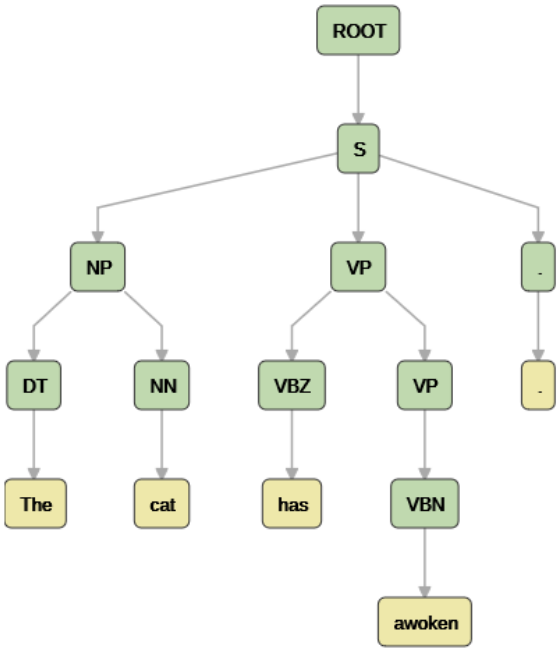


Figure 4.10: Visualization of the application of the Stanford CoreNLP constituency parser to an example sentence. Notice that all words are terminal nodes, and their immediate parent nodes are their corresponding part-of-speech tags.

to take. The most popular tools used in the analyzed literature to create dependency trees were Stanford CoreNLP basic [De Marneffe and Manning, 2008] and universal dependencies [de Marneffe et al., 2014], MiniPar [Lin, 2003], Collins parser [Koo et al., 2008], TreeTagger, CHAOS parser [Basili et al., 1998], MaltParser [Nivre et al., 2007] and OpenNLP.

4.8.6 Named Entity Recognition

Named entity recognition (NER) refers to the identification and classification of mentions of named entities in a given text. Named entities are, generally speaking, objects that can be given a proper name. There is no formal definition as to what constitutes a named entity, however, most general purpose NER systems support between 3 and 7 classes of entities that can be identified e.g. persons, locations, organizations and numerical values like monetary amounts, dates, times and percentages. Much like part-of-speech tagging, named entity recognition is a sequence labeling problem, in which the tagger tries to apply an optimal sequence of labels to a sequence of words. The tagging system needs to be either hand-written, e.g. using regular expressions, or trained using an annotated training corpus. The annotations in this corpus define which classes of entities the tagger will support. An important training corpora for German language natural language

Step	Stack	Words	Action	Relation
0	[root]	[the,cat,has,awoken]	SHIFT	
1	[root,the]	[cat,has,awoken]	SHIFT	
2	[root,the,cat]	[has,awoken]	LEFTARC	(the←cat)
3	[root,cat]	[has,awoken]	SHIFT	
4	[root,cat,has]	[awoken]	SHIFT	
5	[root,cat,has,awoken]	[ ]	LEFTARC	(has←awoken)
6	[root,cat,awoken]	[ ]	LEFTARC	(cat←awoken)
7	[root,awoken]	[ ]	RIGHTARC	(root→awoken)
8	[root]	[ ]	done	

Table 4.8: This is a simplified example parse of the example sentence "The cat has awoken". The parser shifts asks an oracle what actions to take providing the two latest additions to the stack as parameters. The possible actions are SHIFT, LEFTARC and RIGHTARC. The parser then applies the action to the words in the stack until there are no more words in the word list and the stack only consists of the root node.

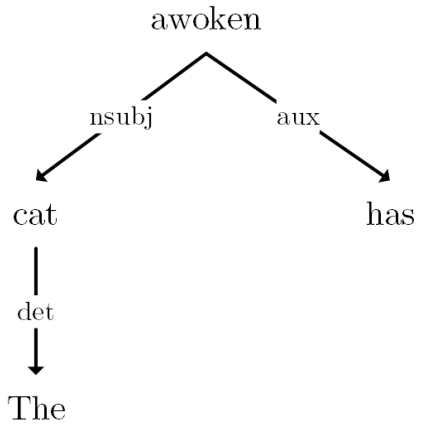


Figure 4.11: Visualization of the application of the Stanford CoreNLP dependency parser to an example sentence. Notice that the relations are a product of a parse similar to the one in Table 4.8

Word	POS-Tag	Named-Entity-Tag
<word>	NE	I-ORG
<word>	NE	I-ORG
<word>	$.	O

Table 4.9: An example sentence from the dataset provided for the CoNLL 2003 shared task. The first column contains placeholders for any word, the second column denotes the part-of-speech tag the word needs to have, and the last column contains tags for named entities, in this case denoting an organization.

processing is the dataset provided for the CoNLL 2003 language independent named entity recognition task. It provides training data sets for both German and English language, and covers 4 classes: `PERSON`, `LOCATION`, `ORGANIZATION` and `MISC`. The data are annotated using a specific annotation style, called *"IOB"* (inside, outside, begin), which is used to mark each word as being outside a named entity mention (O), being the beginning word of a multi-word mention (B), or the only word, or on the inside or at the end of a multi-word mention (I). Extensions like BILOU (begin, inside, last, outside, unit) also give the annotator the possibility to mark the last word of a mention covering multiple words (L), as well as explicitly annotating single word mentions as a singular unit (U). To differentiate the different classes, each annotation, except the outside (O) annotation, is given a suffix denoting the class of the mention, e.g. B-PERS could stand for the beginning of a mention of a person. Table 4.9 shows an excerpt of the CoNLL 2003 German named entity recognition dataset.

Named entity recognition needs to solve two separate problems: (i) identifying mentions of named entities (mention detection), and (ii) classifying the mention (mention classification). Figure 4.12 shows the visualization of the those two steps using the Stanford CoreNLP named entity recognizer. More advanced systems also try to disambiguate and link the identified and classified mentions to items in an existing knowledge base. This task is called named entity recognition, disambiguation and linking (NERDL). Amongst the fields where this method is applied are knowledge base population (KBP) or question answering over linked data (QALD), where the former is used to populate a knowledge base with facts and artifacts from a natural language text, and the latter uses linked data to answer natural language questions. To solve the first two parts of named entity recognition (identifying and classifying mentions in a natural language text), any method capable of solving a sequence labeling problem can be applied, like the previously presented hidden Markov models, or maximum entropy Markov models. The method used by the most prominently used named entity recognizer tool, Stanford CoreNLP [Finkel et al., 2005], however, is conditional random fields, which are a approach very similar to MEMMs, while solving a major issue that MEMMs are suffering from, namely label bias. In recent years however, neural networks have established themselves as a viable competitor to conventional conditional random field approaches. Especially long-short-term-memory (LSTM) networks have proven to achieve state-of-the-art perfor-

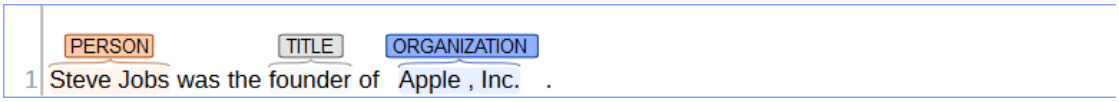


Figure 4.12: Visualization of the application of the Stanford CoreNLP named entity recognizer to an example sentence. Notice the three distinct classes of identified entity mentions.

mance [Lample et al., 2016]. The most popular tools used for named entity recognition in the literature analyzed were Stanford CoreNLP, FOX [Speck and Ngomo, 2017], SENNA, MetaMap [Aronson and Lang, 2009] and GATE. It has to be noted, that many systems use custom implementations, especially when combining mention named entity recognition with disambiguation and linking. One of the most common approaches is to naively take sequences of nouns up to a predefined number of tokens as mention candidates, and use a local search engine like Apache Lucene[8] or DBPedia Spotlight [Mendes et al., 2011] to find candidate entities in the underlying knowledge base. Disambiguation is then custom implemented using contextual features to rank the candidates and link to the candidate with the highest score.

Conditional random fields (CRF): a CRF [Lafferty et al., 2001] models a full sequence of labels given a sequence of inputs (in the case of named entity recognition, this input is the words). Like a maximum entropy Markov model (MEMM), a CRF is used to identify the optimal combination of labels given a sequence of words, using a pre-learnt model. These models are learnt using supervised machine learning and datasets usually containing millions of annotated words. The main advantage between CRF and MEMM lies in the way that they apply normalization[9]: MEMM applies a normalization term on a per-state basis, while CRF normalizes over the entire sequence in one step. Normalizing on a per-state basis can lead to the situation that states with a low number of transitions are preferred over states with a larger number of transitions, since the individual transition probabilities will be smaller due to the per-state normalization. Imagine a state with two transitions, the average probability of each transition is going to be 0.5, as compared to a state with five transitions, which will have an average probability of only 0.2. Extending Equation 4.2, Equations 4.6 shows the application of the normalization term Z in a MEMM. ω is the weight vector applied to the values returned by feature function f. Equation 4.7 shows how the normalization term is applied in a CRF. Note that in a CRF the normalization term only applied once. Also note, that in a CRF the entirety of the observed sequence is taken into account, not only the current word.

$$P(T|W) = \prod_{i=1}^{n} P(t_i|t_{t-1}, w_i) = \prod_{i=1}^{n} \frac{\exp(\omega^T f(t_i, t_{i-1}, w_i))}{Z(t_{i-1}, w_i)} \tag{4.6}$$

[8] https://lucene.apache.org/core/

[9] Applying a normalization term ensures that the sum of the probabilities equals 1.

$$P(T|W) = \frac{1}{Z(w_{1:n}, \omega)} \prod_{i=1}^{n} \exp(\omega^T f(t_i, t_{i-1}, w_{1:n})) \tag{4.7}$$

Long-short-term-memory networks (LSTM): LSTM networks [Hochreiter and Schmidhuber, 1997] are a special kind of neural network which have seen a large increase in interest in the last couple of years. Generally speaking, there are several major families of neural networks that are referred to when one speaks about "deep learning": (i) convolutional neural networks (CNN), (ii) recurrent networks (RNN), and, more recently, (iii) generative adversarial neural networks (GAN). Each of those families of networks have different domains in which they excel, CNNs for example are used for image classification, GANs can be used for machine translation, and one of the major fields of application of RNNs is sequence labeling, like part-of-speech tagging or named entity recognition. One of the major differences between RNNs and the other neural networks is that recurrent networks retain information about past outputs, which is used for the calculation of future outputs. An LSTM network is a recurrent network composed of LSTM memory cells, which do not retain information statically, but learn what kind of information to retain, and what kind of information to forget depending on the inputs during training of the network. This kind of retained information is especially helpful when dealing with sequences, where prior (or following) items can give additional information about the nature of the currently processed item.

4.8.7 Lexical Databases

Lexical databases contain information about the relation between words, or the nature of a word. As already established in prior sections, one of the major challenges in natural language understanding is the ambiguity of words. Not only can different words share the same meaning (synonymity), but one word can also have different meanings, depending on the context (polysemity). Without additional knowledge, a natural language understanding system would struggle to infer that the words "automobile" and "car" share the same meaning, and what a user's intent might be in the question "Where is the next bank?", as the user could be referring to bank as in the monetary institution, or bank as in sandbank. Lexical databases provide information to help with these issues, as they contain information about related words and concepts. This additional information might help in the disambiguation of ambiguous terms, or to enrich a text to e.g. better find corresponding items and concepts in an underlying knowledge base. Figure 4.13 visualizes the response from the lexical database WordNet when searching for information about the word "automobile". As can be seen, the word "car" is amongst the so-called synset of the word "automobile", which means that these are words with the same meaning. Figure 4.14 shows the response for the the word "bank", a word with many different possible meanings. Notice that the database also provides short descriptions for the different word meanings. The most popular tools used in the analyzed literature were WordNet [Miller et al., 1990], Patty [Nakashole et al., 2012], BOA [Gerber and Ngomo, 2011], ReVerb [Fader et al., 2011], LexInfo [Cimiano et al., 2011] and Wortschatz [Quasthoff et al., 2006]. ReVerb is actually a tool for Open Information Extraction

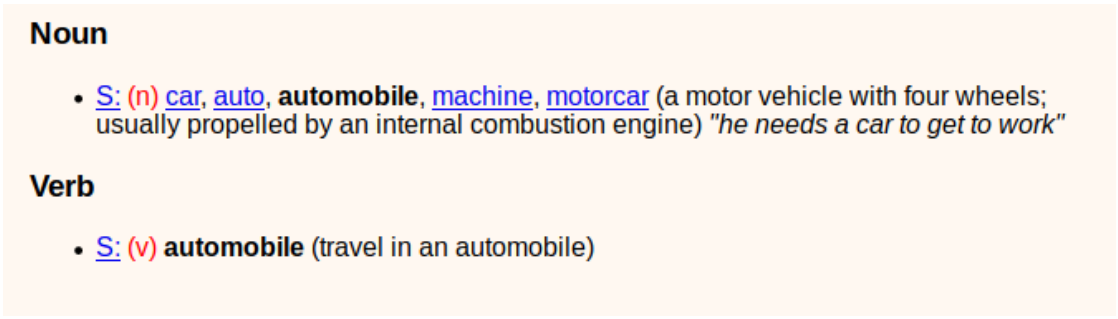


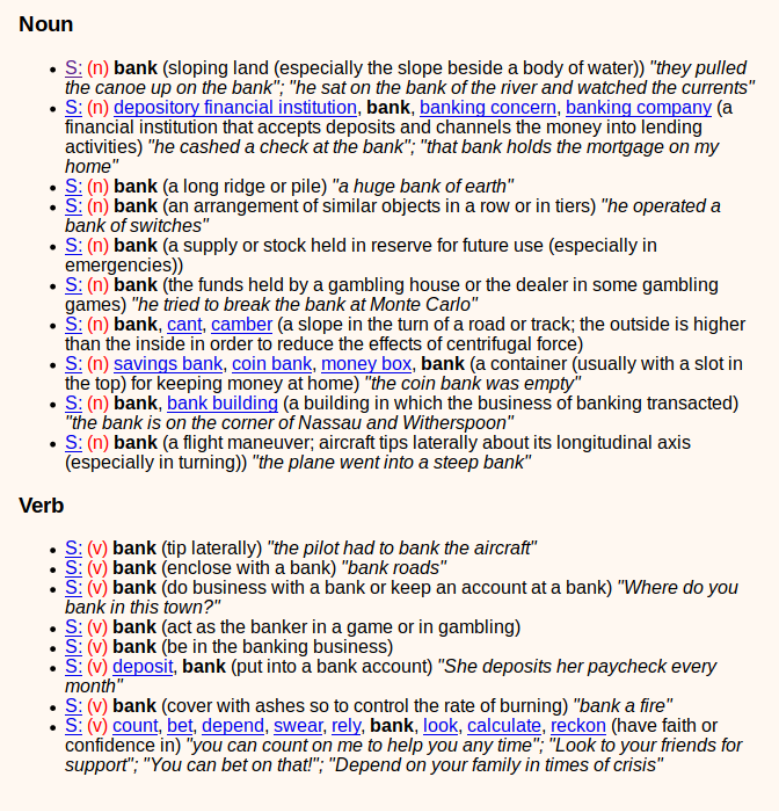
Figure 4.13: Response from the lexical database WordNet for the search term "automobile". The database responds with a set of words with similar meaning, like "car" or "auto".

(OIE), but includes lexical information for relation mapping. Besides these tools, many systems use either pre-prepared gazetteers, containing terms, most often proper names, of a defined type. This can be seen as a mixture between named entity recognition and lexical databases, as the presence of a word in such a list gives the system additional information about the nature of the word and its relation to other, similar, words.

4.8.8 Word Embeddings

Natural language understanding tasks often involve the use of machine learning algorithms, like decision trees or support vector machines. As input, however, they do not expect characters, but vectors of fixed length. So one of the issues of natural language understanding is the transformation of words into a numerical representation without losing its morphological and semantic information. This transformation can be seen as the embedding of a word into a predefined vector space, therefore the term "word embeddings". The result of such an embedding is a numerical vector of fixed length, which can be used in further processing steps. The most naive approach would be to convert all the characters of a word into numerical representations, e.g. their ASCII code. But this representation would not result in a vector of fixed length, as words can have different numbers of characters, and any attempt to apply padding (i.e. defining a fixed maximum length and filling up all non-used dimensions with zeros) would result in very sparse vectors, as the length of the vector is dependent on the longest word in the vocabulary.

One-hot vector representation: The most commonly used naive approach is the use of a one-hot representation. In such a vector representation each word in the vocabulary represents exactly one dimension in the vector resulting in a very sparse, but fixed length vector representation unique for every word, in which only the dimension associated with the word is set to "1", and all other dimensions are set to "0". Besides the high dimensionality of such a vector representation (the number of dimensions is equal to the size of the vocabulary), no semantic information is contained in these representations, as the position of the single "1" is only depending on the order of the words in the



Noun

- S: (n) **bank** (sloping land (especially the slope beside a body of water)) *"they pulled the canoe up on the bank"; "he sat on the bank of the river and watched the currents"*
- S: (n) depository financial institution, **bank**, banking concern, banking company (a financial institution that accepts deposits and channels the money into lending activities) *"he cashed a check at the bank"; "that bank holds the mortgage on my home"*
- S: (n) **bank** (a long ridge or pile) *"a huge bank of earth"*
- S: (n) **bank** (an arrangement of similar objects in a row or in tiers) *"he operated a bank of switches"*
- S: (n) **bank** (a supply or stock held in reserve for future use (especially in emergencies))
- S: (n) **bank** (the funds held by a gambling house or the dealer in some gambling games) *"he tried to break the bank at Monte Carlo"*
- S: (n) **bank**, cant, camber (a slope in the turn of a road or track; the outside is higher than the inside in order to reduce the effects of centrifugal force)
- S: (n) savings bank, coin bank, money box, **bank** (a container (usually with a slot in the top) for keeping money at home) *"the coin bank was empty"*
- S: (n) **bank**, bank building (a building in which the business of banking transacted) *"the bank is on the corner of Nassau and Witherspoon"*
- S: (n) **bank** (a flight maneuver; aircraft tips laterally about its longitudinal axis (especially in turning)) *"the plane went into a steep bank"*

Verb

- S: (v) **bank** (tip laterally) *"the pilot had to bank the aircraft"*
- S: (v) **bank** (enclose with a bank) *"bank roads"*
- S: (v) **bank** (do business with a bank or keep an account at a bank) *"Where do you bank in this town?"*
- S: (v) **bank** (act as the banker in a game or in gambling)
- S: (v) **bank** (be in the banking business)
- S: (v) deposit, **bank** (put into a bank account) *"She deposits her paycheck every month"*
- S: (v) **bank** (cover with ashes so to control the rate of burning) *"bank a fire"*
- S: (v) count, bet, depend, swear, rely, **bank**, look, calculate, reckon (have faith or confidence in) *"you can count on me to help you any time"; "Look to your friends for support"; "You can bet on that!"; "Depend on your family in times of crisis"*

Figure 4.14: Response from the lexical database WordNet for the search term "bank". Notice the number of different word meanings, and the short descriptions provided by the database.

vocabulary, which could be alphabetically, by length or just random, and the euclidean distance between all words is equal. One approach solving the problem with the lack of semantic information contained in such a naive representation would be to follow the assumption of context-dependency of meaning, subsummized in the famous bon-mot uttered by J. R. Firth: "You shall know a word by the company it keeps".

Co-occurrence matrix: A naive approach to convert a words into a fixed length vector representation with regard to its context is to define a co-occurrence matrix, in which every row is a word, and every column is a word used in its context, resulting in a matrix of size NxN, where N is the size of the vocabulary. Taking this co-occurrence matrix, every row would be the context in which a word has been used in a given text. Having captured the contexts of a word, one can apply distance measures like cosine similarity (the dot product between two vectors) to measure the semantic relatedness between two words. One problem with this however, is that not all words carry equal amounts of information. Some words are used very frequently, as are e.g. determiners like "the" or "a", while others are used less frequently, like proper names. So the information that

a word has a lot of co-occurrences with a word like “a” or “the” carries less semantic information as compared to its co-occurrence with the word “phonology”.

Pointwise mutual information: To capture this concept of relative importance of words, one can replace the number of co-occurrences by the pointwise mutual information (PMI) [Fano and Hawkins, 1961] between two words. It is, simply put, the ratio between the actually observed co-occurrences and the expected co-occurrences between words under the assumption of independence. It is defined by the following equation:

$$PMI(w_1, w_2) = \log_2 \frac{P(w_1, w_2)}{P(w_1)P(w_2)}, w_1, w_2 \in V \tag{4.8}$$

The following illustrates the differences between these approaches given three sentences:

```
A mouse is a small mammal.
A goose is a big bird.
A rhino is a big mammal.
```

Resulting vocabulary:

$V = [a, mouse, is, small, mammal, goose, big, bird, rhino]$

$|V| = 9$

Looking at the vector representations of the words “mouse”, “goose” and “rhino”:

One-hot:

$vec(mouse) = [0, 1, 0, 0, 0, 0, 0, 0, 0]$

$vec(goose) = [0, 0, 0, 0, 0, 1, 0, 0, 0]$

$vec(rhino) = [0, 0, 0, 0, 0, 0, 0, 0, 1]$

Naive co-occurrence, context window is the co-occurrence within the same sentence:

$vec(mouse) = [2, 1, 1, 1, 1, 0, 0, 0, 0]$

$vec(goose) = [2, 0, 1, 0, 0, 1, 1, 1, 0]$

$vec(rhino) = [2, 0, 1, 0, 1, 0, 1, 0, 1]$

To get the PMI-weighed vector representation, we need to calculate the pointwise mutual information for each dimension, taking the co-occurrences of the word “mouse” as an example:

$cooc(mouse) = (a(2), mouse(1), is(1), small(1), mammal(1), ..., rhino(0))$

From that we can calculate $P(w_1, w_2)$ for each $(mouse, w_2)$ tuple, where $w_2 \in V$ by taking the co-occurrences and dividing them by the number of words in the text ($N = 18$):

$P(mouse, a) = 2/18$, $P(mouse, mouse) = 1/18$, $P(mouse, is) = 1/18$, $P(mouse, small) = 1/18$, $P(mouse, mammal) = 1/18$, $P(mouse, goose) = 0/18$, $P(mouse, big) = 0/18$, $P(mouse, bird) = 0/18$, $P(mouse, rhino) = 0/18$

Now we need the values for $P(w_i)$, which is the number of occurrences of each word divided by N, the total number of words in the text:

$P(a) = 6/18$, $P(mouse) = 1/18$, $P(is) = 3/18$, $P(small) = 1/18$, $P(mammal) = 2/18$, $P(goose) = 1/18$, $P(big) = 2/18$, $P(bird) = 1/18$, $P(rhino) = 1/18$

With those values we can calculate each value for pointwise mutual information (see Equation 4.8), e.g. for the tuple $(mouse, a)$:

$PMI(mouse, a) = \log \frac{P(mouse,a)}{P(mouse)P(a)}$

$PMI(mouse, a) = \log_2 \frac{\frac{2}{18}}{\frac{1}{18}\frac{6}{18}} = \log_2 6 = 2.58$

Applying this to all tuples, we can calculate the final vectors:

$vec(mouse) = [2.58, 4.17, 2.58, 4.17, 3.17, 0, 0, 0, 0]$

$vec(goose) = [2.58, 0, 2.58, 0, 0, 4.17, 3.17, 4.17, 0]$

$vec(rhino) = [2.58, 0, 2.58, 0, 3.17, 0, 3.17, 0, 4.17]$

Calculated similarities using cosine similarity:

$$sim(v_1, v_2) = \frac{v_1.v_2}{|v_1||v_2|} \tag{4.9}$$

Based on naive co-occurrence vectors:

$sim(vec(mouse), vec(goose)) = 0.63$

$sim(vec(mouse), vec(rhino)) = 0.75$

Based on PMI-weighed co-occurrence vectors:

$sim(vec(mouse), vec(goose)) = 0.23$

$sim(vec(mouse), vec(rhino)) = 0.43$

As can be seen in this (somewhat synthetic) example, the PMI-weighed vectors were able to better capture the semantic similarity between the words “mouse” and “rhino”. This is due to the fact that the weights corresponding to the co-occurrences with more common words were reduced as compared to the weights for less common words. The co-occurrences with the words “a” and “is” were treated with relatively less emphasis as the co-occurrence with the word “mammal”, leading to a larger difference in similarities.

Document embeddings: A similar approach is also used to embed documents into a vector space, using the number occurrences of words as weights for an vector with the

dimensionality of the size of the vocabulary. With this embedding, semantic similarities between documents can be calculated. In this approach similar issues emerge, as some words are very common and therefore their occurrences within a text bear little information as compared to the occurrence of a less-common word. This is also solved using weighted values based on the frequencies of a word occurring in a given document. This approach is called "TF-IDF", as in "term frequency - inverse document frequency", in which the the frequency of a term within a document is multiplied by the inverse frequency of the word occurring in all documents, effectively dampening the impact of words that occur in many (or all) documents:

$$tfidf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \log \frac{N}{n_t} \tag{4.10}$$

with $f_{t,d}$ being the raw count of term t in document d, N the number of documents, and n_t the number of documents containing term t.

Dense word embeddings: A shared problem of all these embeddings, however, is their high dimensionality and sparsity. High dimensionality will lead to increased effort in machine learning applications, as the number of weights increase with the number of input features, and sparsity can lead to overfitting models to the training data [Jurafsky and Martin, 2016b]. To solve those issues, one needs to reduce the dimensionality of the word embeddings. Several approaches exists, like latent semantic analysis [Deerwester et al., 1990], which applies singular value decomposition to high dimensional co-occurrence matrices, resulting in a more dense representation where the columns do no longer represents individual context words, but a latent, "semantic" features. Another, now very popular, approach is the application of neural networks to create dense word vectors. The general idea behind it is that words which share similar contexts, should also have similar vector representations. Given that expectation a neural network is trained on very large text corpora to assign vectors of a predefined length to words in such a way, that assigned vectors of words with similar contexts have large dot products, indicating semantic relatedness, whereas the dot products with words outside of the context should be minimized, indicating semantic non-relatedness. The most popular word in this field is word2vec [Mikolov et al., 2013], which optimized the training process to enable the learning of dense word representations on huge training corpora, containing billions of words. In recent years other neural network-based dense vector embeddings have emerged, like GloVe [Pennington et al., 2014] or FastText [Bojanowski et al., 2016]. Beside the obvious property of the aforementioned word embeddings, capturing semantic relatedness between words, by doing so they also capture relational meanings. As vectors for words used in similar contexts are similar, so are the offsets created by relations between concepts, e.g. country - capital. As the vectors for the words "Berlin", "Rome" and "London" are similar - so are the vectors for "Germany", "Italy" and "England". This leads to the offsets between related terms, e.g. vec(Rome) - vec(Italy), to also be similar, capturing, in this case, the relationship "capital city". One could use this to approximate vectors either not in the vocabulary, or to provide evidence for the plausibility of an

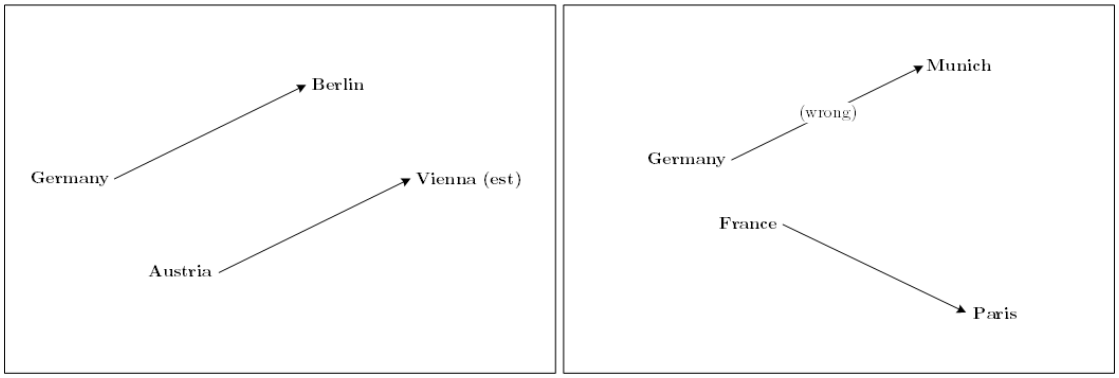


Figure 4.15: Illustration of possible uses of relationship information provided by vector offsets. In the left panel, the known offset between "Germany" and "Berlin" is used to estimate the word vector for "Vienna". In the right, the known offset between "France" and "Paris" is used as evidence that the assumption that "Munich" is the capital city of "Germany" is wrong.

assumed relationship. An example for the former would be, if for some reason the word "Vienna" does not exist in the vocabulary the embeddings were trained on, but the word "Austria" does, one could use the knowledge that Vienna is the capital of Austria to estimate a fitting vector representation by using an existing offset:

$$vec(Vienna) \approx vec(Austria) + (vec(Germany) - vec(Berlin))$$

In the latter use case, one could use the offsets as evidence that an assumed relationship is wrong, e.g. when assuming that Munich was the capital of Germany, the offsets between a known correct relationship of the same kind could be used as a comparison:

$$vec(Germany) - vec(Munich) \approx vec(France) - vec(Paris)$$

Figure 4.15 provides a visual illustration of this behavior.

The most popular tools used in the literature analyzed were pretrained models by word2vec and FastText, as well as the original implementation of word2vec provided by Google[10]. Several libraries exist to provide easy access to pretrained models, e.g. gensim for Python.

4.8.9 Handwritten Rules

This refers to those methods, which are not covered by the previous method groups, and are manually implemented for very specific use cases. One example of such a method would be the application of regular expressions or grammar rules to directly classify a question based on the interrogative word, see Table 4.10 and Table 4.11 for examples.

[10] https://code.google.com/archive/p/word2vec/source/default/source

ID	Regular expression	Intent
1	"Ŵhen .+"	TIME
2	"Ŵhere .+"	LOCATION
3	"Ŵho .+"	PERSON

Table 4.10: A basic ruleset consisting of regular expressions and intents. When an input is matched by one of the regular expressions, the input is classified with the corresponding intent.

Input	Matching rule ID	Applied intent
Where is the next gas station?	2	LOCATION
When does the next train leave?	1	TIME
Who is responsible for the tickets?	3	PERSON

Table 4.11: Example inputs and the rules that match them. The third column shows the intent the input was classified with.

Another approach would be grammatical rules, like they are used in tree parsing tasks (see Section 4.8.5), or rules written in the *"Java Annotation Pattern Engine"* (JAPE) [Cunningham et al., 2000] which uses both patterns applied on the string-level, as well as patterns applied to annotations. These kinds of approaches lead to quick and reliable results, however, they do not scale well, as every rule has to be manually prepared by qualified people. Approaches like this have been mostly replaced by machine learning based methods, where some of these techniques are incorporated, e.g. in feature engineering (e.g. applying rules to inputs to augment the feature vectors) or tree parsers. The tools used in the analyzed literature were GATE (applying JAPE grammars), as well as the Grammatical Framework (GF) [Ranta, 2004].

4.8.10 Intent Classification

Intent classification is not only a family of methods, but an essential, and mostly final, step in a natural language understanding pipeline. In this step, the input text together with all the information extracted by the aforementioned groups of methods are used to classify the general intent of the input, i.e. what the user wants. There are multiple different concepts that are used synonymously, like expected answer type (EAT) detection or question classification. For the remainder of this thesis however, we will use the term "intent classification" to refer to the generic task of classifying the input following a defined set of types. Such a taxonomy can be created by hand, or automatically using lexical databases [Jurafsky and Martin, 2017c]. The most famous general-purpose taxonomy was defined in [Li and Roth, 2002], consisting of 56 classes in a two-layer hierarchy. Table 4.12 shows an excerpt of these classes. The most important role of intent classification is that it enables efficient filtering of candidate responses [Li and Roth, 2006]. Usually the intent

Class	**Class**
HUMAN	*LOCATION*
group	city
individual	country
title	mountain
description	other
	state

Table 4.12: An excerpt of the taxonomy defined by [Li and Roth, 2002]. The taxonomy consists of two layers in a hierarchical structure, with six coarse classes, like *ENTITY*, *HUMAN* or *LOCATION*, and 50 fine-grained classes, like *HUMAN*/individual. The above example shows two of those six coarse classes and their fine-grained "child"-classes.

classification is done using machine learning approaches, but, as mentioned in the previous section, naive approaches might use manually created sets of rules. Any machine learning algorithm can be applied to this task, as the previous steps in the natural language pipeline usually provide the intent classification step with a numerical vector of fixed length. Amongst the algorithms applied in the literature analyzed were support vector machines (SVM) [Romeo et al., 2017], sparse networks of winnows (SNOW) [Carlson et al., 1999, Li and Roth, 2006] or convolutional neural networks (CNN) [Kim et al., 2017]. These algorithms all rely on labeled training sets where the vector representation of an input is labeled with the corresponding class taken from the taxonomy.

4.9 Summary

In this section we have analyzed the current state-of-the-art of methods and tools used in natural language understanding, focusing on question answering systems as our main source of literature. This was done upon the consideration that question answering systems can be viewed as the most basic version of a chat-bot, and also because of the large number of available literature. We have seen that the majority of systems are using English as the language of choice, and that authors describe a similar set of issues when creating a system in a language other than English, specially the lack of mature models and tools. One of the reasons for this lack of models and tools in other languages is due to the fact that development in the field of question answering (and natural language processing in general) is driven by the availability of large-scale training and evaluation corpora. Many of the tools and models currently in use in state-of-the-art tools are trained and fine-tuned using publicly available corpora created in the course of conferences and workshops like CoNLL. As many of these corpora have to be manually created, and the necessary number of training examples for modern machine learning based methods is very large, the effort necessary to produce high-quality corpora is enormous, they are therefore mainly focused on the smallest common denominator in the field of natural language processing: the English language. Another interesting observation was the

shift towards deep-learning models starting around 2016. Especially methods using sequence tagging are now dominated by deep neural networks, but also many current systems are implemented as complete end-to-end neural network architectures. The majority of systems, however, are using a similar ensemble of methods and tools, leading us with a set of popular methods and tools which can be considered the state-of-the-art. Using the relative popularity of methods as a ranking mechanism, we have also given a detailed overview of the most popular methods and their most commonly used implementations, i.e. tools. Coming back to our second research question ("What is the current state-of-the-art regarding language-dependent methods, tools and services involved in creating chat-bots?"), we could identify a set of the most popular methods and tools currently in use, with some methods being far more popular than others - for example part-of-speech tagging, named entity recognition, tree parsing and use of lexical databases being the most popular. The complete list can be found in Table 4.5, a visualization of the popularity distribution in Figure 4.1. The most popular tools used to apply these methods are listed in the detailed descriptions of the methods. In the next chapter we will present the results of a case study using these most popular tools to implement a custom natural language understanding module, and will evaluate it against publicly available commercial systems, as well as a naive baseline, using only a basic bag-of-words approach.

CHAPTER 5

Case Study

In the previous chapter we have presented the current state-of-the-art regarding methods and tools used in natural language understanding. Most of the analyzed systems only support English language inputs, and only two recent systems explicitly support German. One of these systems is a open domain question answering system using Wikipedia articles as data source [Waltinger et al., 2011], the other is a question answering system in the e-government domain [Schwarzer et al., 2016]. Both use methods and tools from very different method groups, the only overlap is the support for German language and the use of part-of-speech tagging. The lack of a robust number of existing systems presents a problem, as one can not rely on the findings presented in these publications to be relevant for a different domain. Neither has there been any interest - academic or otherwise - to approach the question of applicability of natural language processing methods and tools in a German language natural language understanding system. If one decides to implement an English language natural language understanding system, she is presented with both a robust number of existing implementations, both academic and commercial, as well as with comparative studies [Braun et al., 2017]. To help alleviate this a bit, we will present a custom German language natural language understanding based on the most popular methods and tools, identified in the previous chapter. First, we will present the findings of a survey of existing natural language understanding services which offer support for German language, then we will describe the domain and structure of the custom created parallel English and German language training and evaluation corpus, as well as the process of creating it. In the following section, we will present the chosen methods and tools with the reasoning for why they were selected. In the next section, we will present the architecture of the custom natural language pipeline, followed by a description of the feature engineering process and subsequent training of the custom implementation and the existing services.

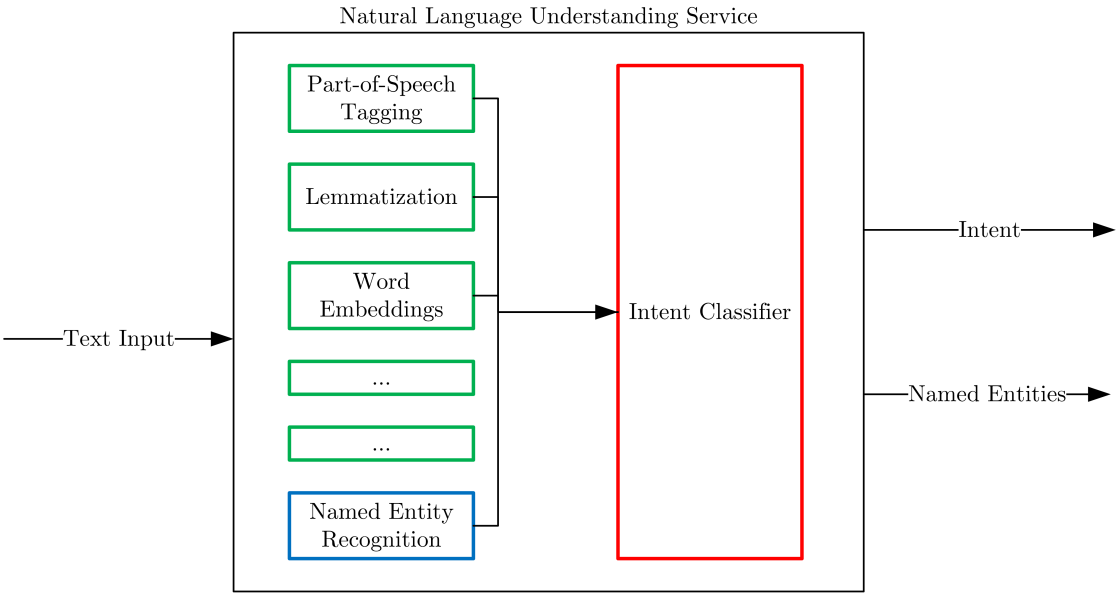


Figure 5.1: Mandatory features in a basic natural language understanding service. Marked in red is the "Intent Classification" submodule, which classifies the intent of an input text based upon the output of the feature extraction submodules (marked in green and blue). These are affected by the explicit language support, as many of them (e.g. part-of-speech tagging, word embeddings) depend on pre-trained language-specific models. Named entity recognition is different from the other feature extractors as their output depends on the application domain and is most often provided side by side with the intent as the output of a natural language understanding service.

5.1 Natural Language Understanding Services

To have a point of reference for the performance of a custom implementation of a natural language understanding pipeline, we performed a survey of existing and publicly available implementations. Using the systems mentioned in [Braun et al., 2017] as a basis, the systems were analyzed with regard to specific features that have to be fulfilled. These features make it possible to properly compare a custom, German language implementation with other systems. These features are: (i) the intent classification has to be trainable, (ii) the system must support trainable named entity recognition, (iii) German language must be explicitly supported and (iv) a free evaluation version must be accessible. Figure 5.1 depicts which parts of a natural language understanding service are connected to the mandatory features.

Trainable intent classification: this has to be fulfilled as the main goal of a natural language understanding system is to find out what the intent of a user's utterance is.

These intents, however, are closely related to the application domain. To give an example, a service application for public transport might be able to provide the user with train schedules, but does not support users ordering a pizza. Therefore the natural language understanding system of a public transportation application should be able to properly classify utterances with the intent to get a train schedule, and a food ordering system should be able to classify utterances revolving around ordering a pizza. To properly classify the user queries, one could create custom rules, e.g. regular expressions. But this requires a lot of manual labor and does not scale well (see Section 4.8.9). Therefore the classification process should be solved using machine learning approaches, teaching the system how to properly classify using a manually created dataset consisting of a large number of examples. All systems must provide an interface to train the intent classification with a custom dataset.

Trainable named entity recognition: besides classifying the intent of an utterance, extracting and classifying mentions of named entities is another main purpose of a natural language understanding system. To properly react to a user's query, the intent alone might not be enough. To use the same example as before, a public transportation application can not properly answer a question only knowing that the user wants to know a train schedule. It must also be provided with additional information like a train line or a station name. The natural language understanding system must be able to find and classify mentions of such named entities. As the class of entities, much as the intent, depends on the application domain, the system must provide an interface to train the named entity recognition.

Explicit support for German language: the classification of intent, as well as the identification and classification of named entities depend either directly or indirectly on large annotated training corpora. Directly in the case when the system can learn directly from a given dataset, or indirectly, when the system needs to use methods that themselves need to be trained on large datasets, like e.g. part-of-speech tagging. As the system should be trainable using only a dataset with the domain-specific annotations about the intents and named entities, they must explicitly support German language. This means that that they either learn directly from the provided dataset, or they come with pre-trained models, already supporting German language input when internally applying methods like part-of-speech tagging.

Free evaluation version: for practical reasons within the scope of this thesis, the systems must either be free, or provide a free evaluation version.

Candidate systems: taking the existing comparative study [Braun et al., 2017], which compared the performances of several natural language understanding systems in English language as a point of reference, the following systems were used as candidates for comparison in German language: (i) wit.at[1], (ii) Microsoft LUIS[2], (iii) IBM Watson

[1] https://wit.ai/
[2] https://www.luis.ai/

System/Feature	Custom Intents	Custom NEs	German	Free
wit.ai	yes	yes	yes	yes
Microsoft LUIS	yes	yes	yes	yes
IBM Watson	yes	yes	yes	yes
Amazon Lex	yes	yes	no	yes
API.ai	yes	yes	no*	yes
rasa	yes	yes	yes	yes
Google Cloud NL	no	no	no*	yes

Table 5.1: Comparison of the candidate natural language understanding systems with regard to the mandatory features they must support to be considered a viable system for the case study (custom trainable intents and named entities, support for German language, free evaluation copy) as of November 2017. Features marked with (*) have since changed, but were not considered.

Conversational Service (now Watson Assistant)[3], (iv) Amazon Lex[4], (v) API.ai (now Dialogflow)[5] and (vi) rasa[6]. Additionally, Google Cloud Natural Language[7] was also taken into account. All systems were analyzed with regard to the mandatory features established above. The results of the analysis can be found in Table 5.1. It has to be noted, that the analysis was performed in November of 2017, and that some systems have expanded their functionalities since that time. Especially API.ai (now Dialogflow), and Google Cloud Natural Language have now official support for German language, but were not considered viable candidates since that support was only added sometime after November 2017. One of the systems also changed their official name, from IBM's Watson Conversational Services to IBM Watson Assistant. However, the functionality relevant for the evaluation performed did not change.

As can be seen in Table 5.1, only four of the seven systems provide all of the necessary features, namely (i) wit.ai, (ii) Microsoft LUIS, (iii) IBM's Watson Conversational Services and (iv) rasa. It also has to be noted that of all the candidates, only rasa is completely free of charge and open source.

5.2 Selection of Methods and Tools

For the implementation of the custom German language natural language understanding system, a set of methods and tools needs to be chosen which will serve as the systems base. As previously introduced, one way to implement a natural language understanding system is to extract features from the input text, and to use these features to train a

[3] https://www.ibm.com/blogs/watson/2018/03/the-future-of-watson-conversation-watson-assis
[4] https://aws.amazon.com/de/lex/
[5] https://dialogflow.com/
[6] https://nlu.rasa.com/
[7] https://cloud.google.com/natural-language/

Rank	Method	Tools
1	Part-of-speech tagging	Stanford CoreNLP [Manning et al., 2014] TreeTagger [Schmid, 1994] SENNA [Collobert et al., 2011]
2	Tree parsing	Stanford CoreNLP TreeTagger
3	Named entity recognition	Stanford CoreNLP Fox [Speck and Ngomo, 2017] SENNA
4	Lexical databases	WordNet [Miller et al., 1990] Patty [Nakashole et al., 2012] BOA [Gerber and Ngomo, 2011] Wortschatz [Quasthoff et al., 2006]
5	Text normalization	Stanford CoreNLP TreeTagger
6	Word embeddings	Word2vec [Mikolov et al., 2013] GloVe [Pennington et al., 2014] FastText [Bojanowski et al., 2016]
7	Handwritten rules	JAPE [Cunningham et al., 2000] Grammatical Framework [Ranta, 2004]
8	Controlled vocabulary	-

Table 5.2: A summarized listing of the most popular methods and their corresponding tools with regard to the literature analysis performed in Section 4. As can be seen, Stanford CoreNLP is by far the most prominently used tool, offering implementations for several different methods. No tools were found for the application of controlled vocabulary.

machine learning algorithm. Following the research of the state-of-the-art regarding language natural language processing methods and tools, only the most popular feature extraction methods and their corresponding most popular tools were considered for further analysis. To be considered a viable choice, a method/tool combination must have the following features: (i) the tool must be publicly available and (ii) German language must be explicitly supported (i.e. explicit German language models must be available). As a quick recap of Section 4, Table 5.2 summarizes the most popular methods together with their most popular tools considering the literature analyzed.

To minimize the overhead and to limit the scope of this thesis, only one viable tool per method will be used in further evaluation steps. To find appropriate tools, the most popular were analyzed with regard to the mandatory featured established above. The results of this evaluation can be seen in Table 5.3.

Following the findings of the above evaluation, the following set of tools was chosen for

Tool (Method)	Publicly Available	Supporting German
Stanford CoreNLP (POS)	**yes**	**yes**
TreeTagger (POS)	yes	yes
SENNA (POS)	yes	no
Stanford CoreNLP (parse trees)	**yes**	**yes**
TreeTagger (parse trees)	yes	yes
Stanford CoreNLP (NER)	**yes**	**yes**
Fox (NER)	yes	yes
SENNA (NER)	yes	no
WordNet (lexical DB)	yes	no
Patty (lexical DB)	yes	no
BOA (lexical DB)	no	-
Wortschatz (lexical DB)	no*	yes
Stanford CoreNLP (text norm.)	**yes**	**yes**
TreeTagger (text norm.)	yes	yes
Word2vec	**yes**	**yes****
GloVe	yes	no
FastText	yes	yes

Table 5.3: Overview of the availability and German language capabilities of the candidate tools. Bold text denotes those tools, which were used later on in the custom implementation. *: word relations like synonymity are not publicly available. **: available at `https://devmount.github.io/GermanWordEmbeddings/`

the implementation of the custom natural language understanding service:

Stanford CoreNLP: this tool will be used to apply the following methods: (i) sentence separation and tokenization (text normalization), (ii) part-of-speech tagging, (iii) constituency and dependency trees (tree parsing) and (iv) named entity recognition. This tool was chosen because of its popularity, the well-documented Java library and its wide coverage of a large number of different natural language processing methods.

Word2Vec + gensim: this tool will be only be used indirectly. A pre-trained German model will be used in combination with the Python library gensim[8]. The model was trained using a dump of the German language version of Wikipedia, and consists of a set of word-vector tuples, which can be loaded and looked up using the gensim library.

Since none of the lexical databases publicly available support German language in a way comparable to WordNet, no such tool will be used. The use of handwritten rules and grammars is inherently language-independent, but they have to be manually created by a native speaker, and the quality of the rules depends on both the qualification and effort put in by the person writing them. To limit the scope of this thesis and to

[8] `https://pypi.org/project/gensim/`

avoid introducing an unwanted bias, no handwritten rules will be applied. As no tool is available to provide a controlled vocabulary, this method group will also not be covered.

5.3 Test and Training Data

To be able to compare the performances of the most popular methods and tools in both English and German, we need a shared evaluation dataset which provide comparable content in both languages. Specialized, multilingual datasets are common in natural language processing, e.g. the CoNLL 2003 English-German dataset for named entity recognition [Tjong Kim Sang and De Meulder, 2003]. However, parallel datasets, where every datapoint is present in more than one language, are much harder to procure, and are especially driven by machine translation use cases, e.g. the Europarl parallel corpora [Koehn, 2005]. These texts usually do not include named entity annotations, and neither do they include sentence-level discourse information like intents. Therefore, there currently is no English-German parallel corpus to evaluate the performances of natural language understanding services on. To still be able to make some form of evaluation, a new sports-themed corpus is manually created and used for further evaluation. Following the basic idea of bootstrap learning [Mintz et al., 2009], a pre-existing database consisting of scraped competition, team and player names, as well as a set of 85 handwritten patterns created by the author were used to emulate a larger data set. To provide parallelity, each German pattern was translated verbatim by the author into English. Table 5.4 shows examples of such patterns, which are emulating football-related user questions. The actual training and evaluation datasets were created using random named entities from the pre-existing database and using them as replacements for the placeholders in the manually created patterns. Figure 5.2 depicts such a replacement process from the basic unfilled pattern to the final labeled datapoint in a format used by the rasa system. This way training datasets of slightly below 12000 labeled utterances were created for every system, as well as a shared evaluation dataset consisting of 288 labeled utterances, created from a separate set of patterns and named entities. This size was chosen due to training size limitations in the free evaluation version of Watson Conversational Services, limiting training dataset sizes to 12000 utterances. Figure 5.3 shows some of the different data formats for each of the external natural language understanding services participating in the evaluation. The patterns were classified into 6 distinct intents: (i) schedule, (ii) result, (iii) standing, (iv) squad, (v) player statistics and (vi) player information. The aforementioned existing database was used as a basis for three types of named entities: (i) competitions, (ii) teams and (iii) players. In the following each of the intents and named entity types is presented with a short description.

Schedule (intent): the user wants to know the schedule of upcoming matches, either for a given team or a given competition. An example question and a corresponding answer would be:

```
Q: When does Real Madrid play next?
```

```
A: Dec. 12th, 9pm against Barcelona
```

Result (intent): the user wants to know the results of one or more finished or currently running matches, either for a given team or a given competition. An example of such a question and a corresponding sample answer would be:

```
Q: How did Manchester United play yesterday?
A: Yesterday Manchester United won 3-0 against Stoke City.
```

Standing (intent): the user wants to know the current standings of a given competition and (optionally) a given team. An example question-answer pair would be:

```
Q: What is the current position of RB Leipzig in the German Bundesliga?
A: RB Leipzig is currently placed 4th in the German Bundesliga.
```

Squad (intent): the user wants to know the current roster of a given team. An example question and its corresponding (abbreviated) answer would be:

```
Q: Who is currently playing for Bayern Munich?
A: 1: Manuel Neuer, 2: [...]
```

Player statistics (intent): the user wants to know performance statistics of a given player, optionally restricted to a given competition. An example would be:

```
Q: How many goals did Marko Arnautovic score in the Champions League?
A: None.
```

Player information (intent): the user wants to know biographical and contractual information about a given player. An example would be:

```
Q: How old is Diego Costa?
A: Diego Costa is 29 years old.
```

Note: the generation of answers like the ones provided is not within the scope of this thesis. They are only presented to illustrate the nature of the intents.

Competition (named entity): a named entity of type competition, also known as league. An example of such a named entity would be *"German Bundesliga"* or *"Premier League"*.

Pattern	Intent	Language
Wer ist Tabellenführer in der [COMP]?	standings	German
Wer spielt bei [TEAM]?	squad	German
Which team is leader in the [COMP]?	standings	English
Who is playing for [TEAM]?	squad	English

Table 5.4: Examples of patterns used to generate a larger dataset by populating the placeholder slots identified by square brackets and the named entity type that can be filled with. The [COMP] slot can be filled with a random named entity of the type "competition", a [TEAM] slot can be filled with a random named entity of type "team".

Team (named entity): a named entity of type team depicts a football team, or club. An example of such an entity would be *"Real Madrid"* or *"Rapid Wien"*.

Player (named entity): a named entity of type player depicts a person playing for a football club. An example of such a named entity would be *"Marko Arnautovic"* or *"Lionel Messi"*.

5.4 Implementation

To evaluate the performance of the identified most popular methods and tools, as well as the external natural language understanding systems, in both English and German language, the following software artifacts were implemented: (i) a custom natural language understanding system combining the most popular methods and tools called *GermaNLU*, (ii) a training framework to generate data in a format fitting for the external services and to execute the training process, (iii) a naive bag-of-words based custom natural language understanding system to serve as a baseline, and (iv) an evaluation framework measuring the performances of each system using a shared evaluation dataset. The implementation was done using the programming languages Java[9] and Python[10], as well as the popular frameworks Spring Boot[11], flask[12] and gensim[13]. For intent classification, the machine learning toolkit Weka[14] was used for feature selection and model training. Additionally, the multi-container tool docker compose[15] as well as multiple BASH-scripts[16] were used for deployment and execution. The sources can be found on GitHub[17].

[9] https://java.com/en/
[10] https://www.python.org/
[11] https://spring.io/projects/spring-boot
[12] http://flask.pocoo.org/
[13] https://radimrehurek.com/gensim/
[14] https://www.cs.waikato.ac.nz/ml/weka/
[15] https://docs.docker.com/compose/
[16] https://www.gnu.org/software/bash/
[17] https://github.com/nathaniel-boisgard

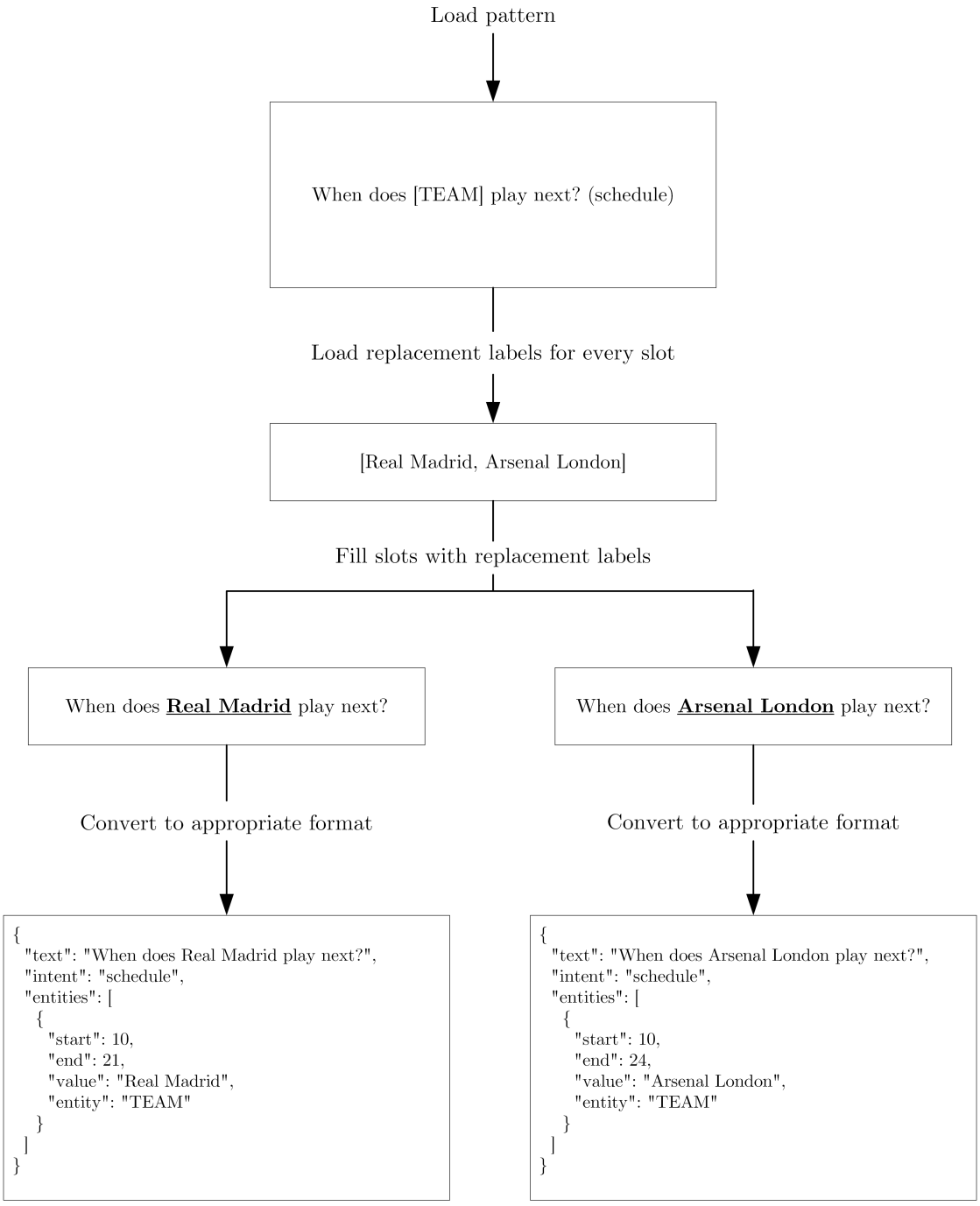


Figure 5.2: Visualization of the replacement process used to generate the dataset. For every pattern the slots are filled with replacement labels, generating multiple outputs, which are in turn converted to a training format expected by one of the natural language understanding services.

Microsoft LUIS

```
{
  "text":"Which team is leading the German Bundesliga table?",
  "intentName":"standings",
  "entityLabels":[
    {
      "startCharIndex":26,
      "endCharIndex":43,
      "entityName":"COMP"
    }
  ]
}
```

Rasa NLU

```
{
  "text":"Which team is leading the German Bundesliga table?",
  "intent":"standings",
  "entities": [
    {
      "start":26,
      "end":43,
      "value":"German Bundesliga",
      "entity":"COMP"
    }
  ]
}
```

Wit.ai

```
{
  "text":"Which team is leading the German Bundesliga table?",
  "entities":[
    {
      "value":"standings",
      "entity":"intent"
    },
    {
      "value":"German Bundesliga",
      "entity":"COMP",
      "start":26,
      "end":43
    }
  ]
}
```

Figure 5.3: Examples of the training formats used to train some of the natural language understanding services, namely Microsoft LUIS, rasa NLU and wit.ai. Notice that every format is slightly different, as there is no defined public standard for tasks like this.

Custom natural language understanding system (GermaNLU): this tool was implemented using Java and Python as the main implementation languages. As laid out in Section 5.2, the tools to be used are Stanford CoreNLP and word2vec-based word embeddings with a custom Python wrapper to make it accessible from the main Java application. This wrapper is a flask web-application which returns the word2vec word vector in text/plain format for a given word. The main application can be accessed via command line, or via a Spring Boot web-application. The command line version can be used for bulk-classification, which is especially useful when evaluating, the web-application can only be used with only one input sentence. The pre-trained models for named entity recognition provided by Stanford CoreNLP, however, do not cover the named entity types used by the training set, therefore a custom model needed to be trained before any other implementation steps could be made. The training dataset was converted into an IOB-tagged format, and the Stanford CoreNLP conditional random field (CRF) based implementation was used to train a new model. Listing 5.1 shows some examples of the IOB-tagged training data in TSV (tab-separated values) format.

```
Which       O
team        O
is          O
leading     O
the         O
German      B-COMP
Bundesliga  I-COMP
Table       O
?           O
```

Listing 5.1: An IOB- style tagged example sentence. The "O" tag denotes all tokens outside a named entity mention, tags starting with "B" denote the start of an entity, and the second part of the tag, separated by a hyphen, denotes the type of an entity, in the above case "COMP", a competition. "I"-tags denote all tokens following the first, that are part of an entity mention spanning more than one token.

The main application was written in Java using the Spring Boot framework as a basis. The base version of the application reads in CSV files containing the labeled training data, extracts all possible features and creates a numerical vector representation of the labeled input text based on the extracted features, saving them in the Weka file format "ARFF". While some features already provide numerical representations (e.g. the vectors provided by the word2vec model), others only provide categorical information, like part-of-speech tags, pase trees and named entities. These features need to be converted to a numerical representation before they can be added to the feature vector. One such method is the so-called "one hot" vector representation (see Section 4.8.8 on sparse word embeddings), where each categorical type is represented by a separate dimension. For example, to encode the information whether a word is a noun or an adjective, a 2-dimensional vector representation can be used, where the first dimension represents the category "noun", and the second dimension represents the category "adjective". To encode the information

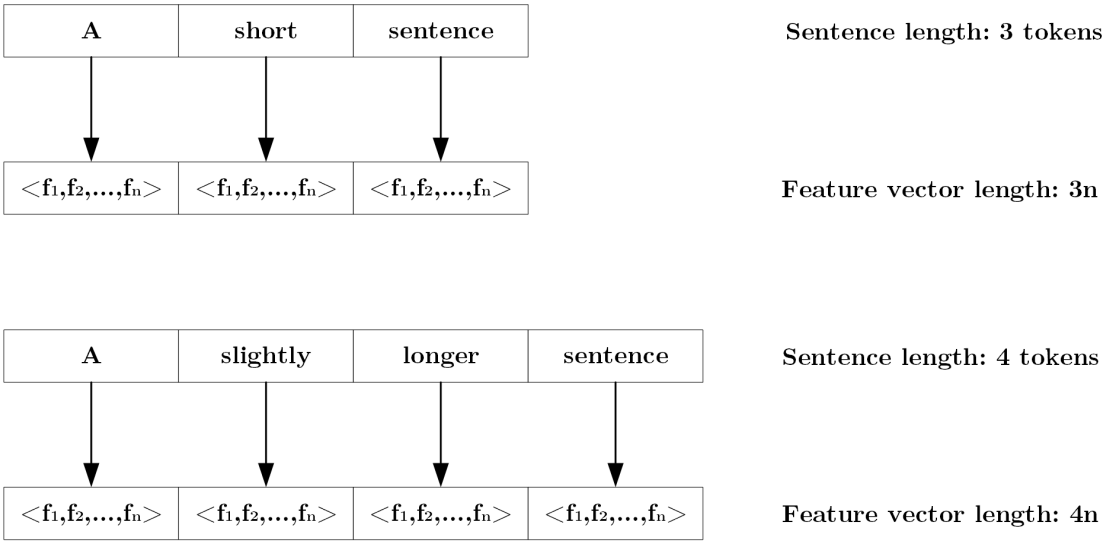


Figure 5.4: Visualization of the problem of converting sentences of different lengths into vectors on a per-token base, resulting in vectors of different lengths. As illustrated, every token is converted into n-dimensional feature vectors, which are concatenated to represent the sentence. This results in vectors of different lengths, 3n and 4n, respectively.

that an observed word is a noun, the corresponding one-hot encoded vector would be [1,0], and the vector corresponding to the fact that a word is an adjective would be [0,1]. Another way to encode categorical data is to use an integer encoding, where each different category is assigned a distinct numerical value. To use the example from before, "noun" could be represented by the number 1, and "adjective" by the number 2. While this is a less sparse representation, no approximate mappings can be done, where e.g. a machine learning classifier outputs a probability distribution over the categorical space. Converting a sentence into a vector that can be used by a machine learning algorithm provides an additional challenge: sentences are not always of the same length, therefore fixed representations of each word feature do not work, as the resulting feature vectors would have different lengths. Figure 5.4 shows how such an encoding would look like.

Most machine learning algorithms, however, demand the training data input to be of equal length. One way to solve this problem is to use padding, i.e. to take the largest dimensionality that any of the training data would produce, and use this as the standard dimensionality. Any feature vector shorter than that would be filled up with fixed values, most often 0 - this is called "zero padding". This padding can be added to the front or the back of the vector. While this is the most straightforward approach, this results in very large, and often very sparse, vectors. This can have negative implications regarding the further use of such vectors, described as the "curse of dimensionality" in [Bellman, 1957]. This refers to dynamic programming, but the basic problems described there also apply to machine learning approaches, where any level of significance has to be ensured. Also, the number of parameters to be estimated during learning phase scales with the

number of dimensions, as does the necessary number of training data [Theodoridis and Koutroumbas, 2009]. This goes hand in hand with the remarks provided by [Jurafsky and Martin, 2016b], where the authors point to possible overfitting issues when using high-dimensional vectors in machine learning. To avoid this, a set of custom generated featured closely following the literature were designed for this implementation. These custom features are:

The average of all word vector for all verb phrases in the input sentence: verb phrases often describe the relation between objects, and are especially used in relation extraction. For this reason, this feature was chosen.

The word vector of the word identified as root of the dependency tree: the dependency tree root can naively be considered the most important, as all other words are directly or indirectly depending on it. For this reason, this feature was chosen.

The average of the word vectors for the first 1,2 or 3 words of the sentence: in questions the first words often define the nature of the question, e.g. questions starting with "When is" might expect a date in the future, while questions starting with "When was" might expect a date from the past. Likewise, questions starting with "Who" might expect a person name. It can therefore be assumed that the first words of a question are important to the nature of the expected answer, and for this reason this feature was chosen.

The number of named entities of type "competition": in the context of the training data used for this thesis, the number of named entities of competition type might give an indication as to which intent is most likely, as questions concerning the intent "squad" are less likely to contain a competition name as are questions of the intent "standing". This is the reason why this feature was chosen.

The number of named entities of type "team": analogous to the number of "competition" entities, this might give hints about the intent. This is why this feature was chosen.

The number of named entities of type "player": analogous to the number of "competition" entities, this might give hints about the intent. This is why this feature was chosen.

The average word vector over all words: this is a very naive approach to create a representation of the whole sentence. It can be best described as a *"bag of embeddings"*. This is the reason why this feature was added.

The word vector with the smallest magnitude: an outlier might provide information about the nature of the intent. This is the reason why this feature was added.

The word vector with the largest magnitude: analogous to the vector with the smallest magnitude, this feature is added under the assumption that outliers might provide some additional insight.

The resulting feature vectors for the input sentences are produced by concatenating all the individual feature vectors used, plus one dimension for the intent. In the next step, a machine learning algorithm was trained using these vector representations as a basis. As mentioned above, high dimensional vectors can cause problems when used in

machine learning, therefore further analysis was done to evaluate the benefit of reducing the number of dimensions. [Yang and Pedersen, 1997] report that information gain is a valuable measure to indicate the importance of a feature in the context of text classification. Information gain is defined as the difference in entropy if a feature is present or not. The built-in information gain calculation feature of the machine learning tool Weka was used to calculate which features provide the best discriminative power towards the intents the feature vectors are labeled with. This was done separately for each language.

Using the results from the information gain calculation to train several machine learning algorithms with several different hyperparameter settings, the best performing models were then saved and used in the resulting end-to-end natural language pipeline to classify the intent of a new utterance.

Figure 5.5 visualizes the process of generating and evaluating the optimal models for intent classification. Figure 5.6 shows the general architecture of the final end-to-end natural language understanding system, while Listing 5.2 shows some example inputs and their corresponding outputs.

```
Input:
Wer spielt bei Real Madrid

Response:
{
  "query": "Wer spielt bei Real Madrid",
  "intent": "squad",
  "entities": [
    {
      "start": 15,
      "end": 19,
      "value": "Real",
      "entity": "B-TEAM"
    },
    {
      "start": 20,
      "end": 26,
      "value": "Madrid",
      "entity": "I-TEAM"
    }
  ]
}


Input:
```

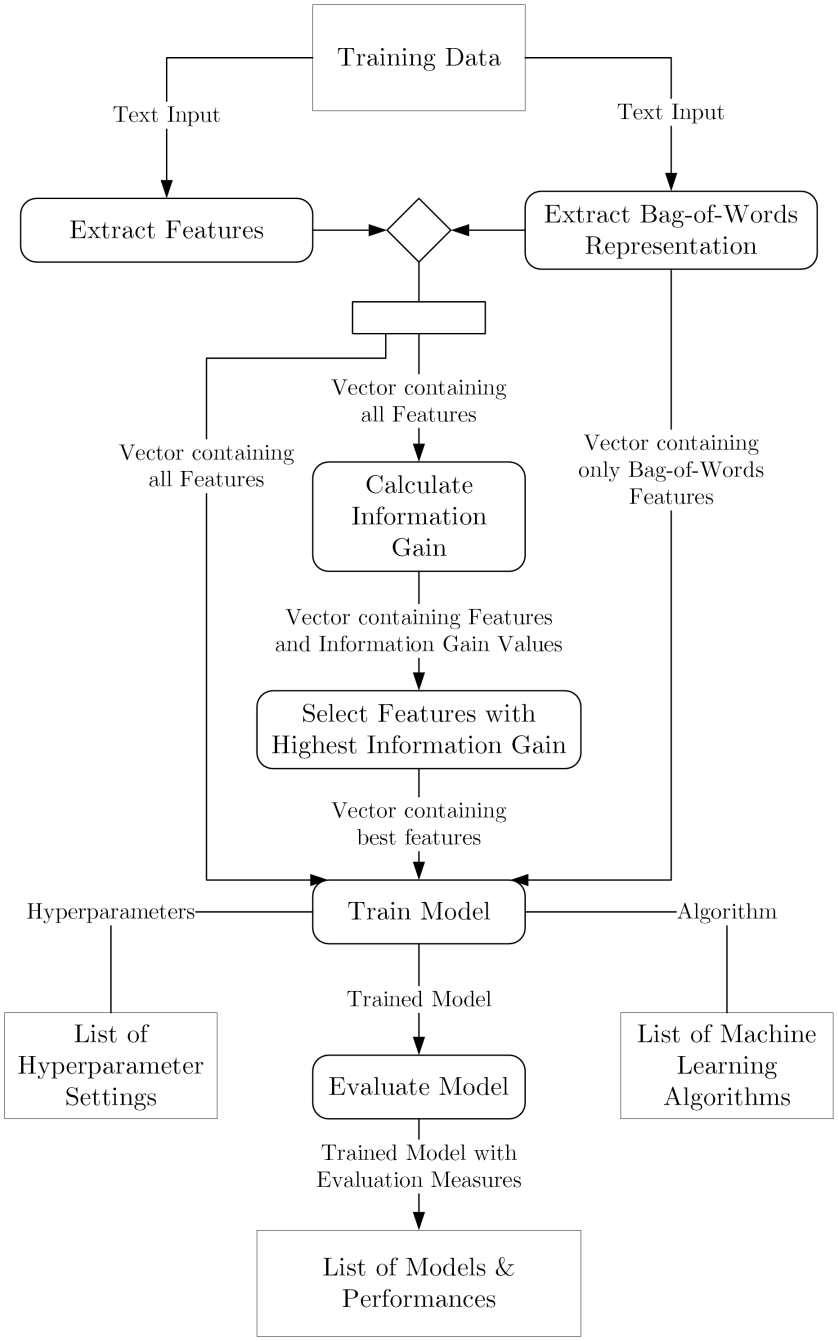


Figure 5.5: Visualization of the feature and algorithm evaluation and training process. First, we extract all features using the methods and tools identified in Section 5.2. Then, we calculate the information gain for each of the features to identify those which have the most discriminative power. Then, we train a multitude of different machine learning algorithms with different hyperparameter settings to identify the optimal feature/algorithm/hyperparameter combination. The optimal model is then used for further evaluations.

Figure 5.6: General architecture of the custom natural language understanding service implementation. The input text is normalized and converted into a feature vector. Based on this feature vector the intent is classified and a response is generated consisting of the intent and the identified named entities.

```
Wie ist die Tabelle der Bundesliga

Response:
{
  "query": "Wie ist die Tabelle der Bundesliga",
  "intent": "standings",
  "entities": [
    {
      "start": 24,
      "end": 34,
      "value": "Bundesliga",
      "entity": "B-COMP"
    }
  ]
}
```

Listing 5.2: Examples of input data and corresponding responses of the implemented webservice wrapper.

Training framework: the training framework is a combination of an application generating and converting the training data into the formats fitting for every application, as well as a set of bash scripts using the programmable interfaces provided by the services to upload the training data and train the systems. The base application is written in Java, using the Spring Boot framework as basis. The application takes a set of CSV files containing patterns and named entities as input, and creates training data for every service as output. Figure 5.7 depicts the basic architecture of the application. As the training formats are different for each system, the application first converts generates the training data in an internal, homogeneous data structure, as described in the UML diagram depicted in Figure 5.8. From there, the individual training formats are generated out of this homogeneous data structure. The patterns and some of the named entities are different in English and German language, and training data for each language are generated in separate steps and saved in separate locations. The actual training process of the external natural language systems is done via bash scripts using the provided programmable interfaces for batch training, if such are provided. Watson Conversational Services for example provide a web interface via which the generated training data can be uploaded. Before the training of the services can be done, however, each service needs to be set up. For each service two separate service instances were initialized, one for German language, the other for English language. Each of the service instances was trained with the appropriate data sets. Listing 5.3 shows an example of a training script, uploading the training data to the training endpoint provided by the service. Rasa had to be treated separately, as it is self hosted and was initialized and executed using the docker image provided by the authors[18].

[18] https://hub.docker.com/r/rasa/rasa_nlu/

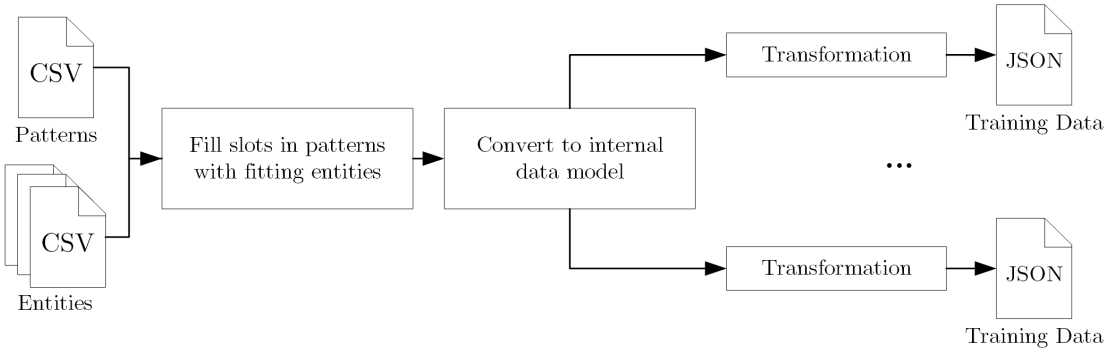
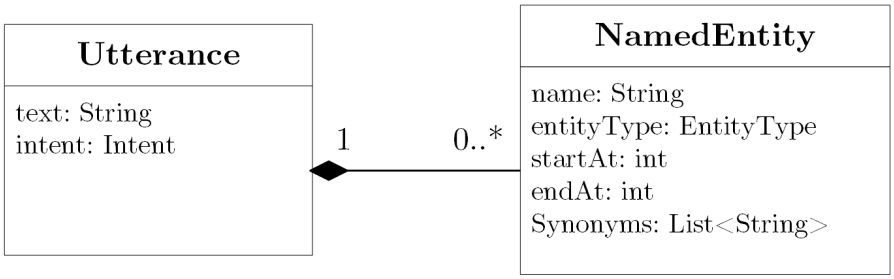


Figure 5.7: Visualization of the basic architecture of the data-generator application. Using basic CSV files containing patterns and entities, the application first fills all slots in the patterns with fitting entity values, generating a large number of permutations. These permutations are enriched with additional information during the conversion into the internal, homogeneous data model, from which all training data for the specific applications are generated in a data transformation step.



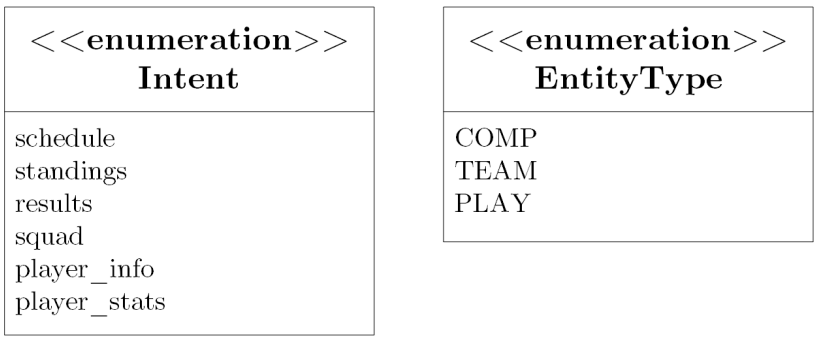


Figure 5.8: The internal homogeneous data format used by the data-generator application.

```
#!/bin/bash

FILES=./data/luis-training-*.json
LUIS_SUBKEY=SOMEKEY
LUIS_APPID=SOMEAPPID
LUIS_APPVERSION=0.1

for f in $FILES
do
printf "\nPROCESSING $f\n"
curl -s -o /dev/null -w "%{http_code}" \
-H "Content-Type: application/json" \
-H "Ocp-Apim-Subscription-Key:$LUIS_SUBKEY" \
-XPOST \
https://westus.api.cognitive.microsoft.com/\
luis/api/v2.0/apps/$LUIS_APPID/versions/\
$LUIS_APPVERSION/examples -d @$f
done

curl -H "Ocp-Apim-Subscription-Key:$LUIS_SUBKEY" \
--data "" \
https://westus.api.cognitive.microsoft.com/\
luis/api/v2.0/apps/$LUIS_APPID/versions/\
$LUIS_APPVERSION/train
```

Listing 5.3: Bash script used for uploading the prepared training data to the training API provided by Microsoft LUIS.

It has to be noted, that wit.ai could not be trained in English language, despite multiple tries. No error message was provided and wit.ai support did not respond to inquiries regarding this issue.

Naive baseline: to provide a baseline to compare all natural language understanding services (external and also the custom implemented) to, a basic naive natural language understanding service was implemented, relying only on a bag-of-words approach. As approaches like this do not use any semantic features provided by the text or tools, it can be considered language-independent, as it only relies on proper tokenization, a problem that can be considered solved for English and German language on an intra-sentence level. The reasoning behind this is, that a more sophisticated implementation using additional syntactic and semantic information should perform at least equally well, and any performance below the one provided by the baseline should be investigated thoroughly.

Evaluation framework: the evaluation framework was implemented as a Java application, using the programmable interfaces provided by the services to classify intents from

an evaluation dataset and compare the returned classification with the actual one. All systems are evaluated using the same evaluation dataset. Listing 5.4 shows examples of such evaluation utterances, Listing 5.5 shows an example of how the service interfaces are called within the application.

```
{
  "text":"Which team is leading the German Bundesliga?",
  "intent":"standings"
}
{
  "text":"Who is playing for Rapid?",
  "intent":"squad"
}
{
  "text":"The next match between Austria Vienna and Austria Vienna?",
  "intent":"schedule"
}
```

Listing 5.4: Examples of the utterances used to evaluate the intent classification performance.

```
public class WatsonService {
  private final Conversation service;

  public WatsonService(){

    service = new Conversation(
          Conversation.VERSION_DATE_2017_05_26
        );
        service.setUsernameAndPassword(
          [USERNAME],
          [PASSWORD]
        );
  }

  public MessageResponse request(String inputString){

    InputData input = new InputData.Builder(
          inputString
        ).build();

        MessageOptions options = new MessageOptions
          .Builder([APPID])
          .input(input)
          .build();
```

```
        MessageResponse response = null;
        int maxTries = 20;
        while(response == null
      && maxTries-- > 0){

          try{
            response = service
                  .message(options)
                  .execute();
                break;
          }catch(Exception e){
            System.err.println(
                  "Oooops!"
                );
                continue;
      }
    }

    return response;
  }
}

WatsonService watsonService = new WatsonService();
DataService dataService = new DataService();
testIntents = dataService
  .getWatsonData()
  .getCommonExamples();
List<RuntimeIntent> watsonIntents = new ArrayList<>();
for(Intent i:testIntents){
  MessageResponse watsonResponse = watsonService
    .request(i.getText());
  watsonIntents.add(watsonResponse.getIntents());
}
```

Listing 5.5: Code excerpt showing how the service is called within the testing framework. First, a class is defined wrapping the communication with the IBM Watson Assistant service.Then a data service class is instantiated providing the evaluation data. Then for each of the evaluation sentences the trained Watson application is called and and the response is added to a list, which is formatted and saved in CSV format.

5.5 Summary

In this chapter we have introduced our custom natural language understanding service implementation and the necessary infrastructure tools. We have described how a custom, sports-themed English/German parallel dataset was created following an adapted approach taken from bootstrapping, in which we generate additional data by replacing parts of a limited seed. This was necessary as there exist no fitting publicly available dataset which could be used to train English and German language natural language understanding services in parallel to compare their performances. We presented a total of six services: *Microsoft LUIS*, *IBM Watson Assistant*, *wit.ai*, *rasa NLU*, a custom natural language understanding service using the most popular tools (*GermaNLU*), and a naive bag-of-words baseline. We also described in detail which tools were selected and how they were used in the internal architecture of the custom implementation. In the next chapter we will present the results of the evaluation of the feature selection, the machine learning/parameter combination selection to define which features are used internally by the custom service to classify the input text. Finally, we will present and compare the evaluation results of all systems.

CHAPTER 6

Evaluation

The aforementioned implementation, together with the training and evaluation datasets, were used to train and evaluate all services. In the following section we present the results of the evaluations of all trained models and services involved.

First, we evaluate the models used for named entity recognition. We have trained the named entity recognizer provided by Stanford CoreNLP in both German and English language in the previous chapter. We then compare the performances of the two models. The results are presented as confusion matrices, comparing the classification results with the actual ground truth in tabular form, as well as the common classification performance measures precision, recall and F-score. Each of the mentioned evaluation methods is shortly introduced below.

We then evaluate the intent classification model used at the heart of the *GermaNLU* system. This evaluation consists of two steps: (i) the evaluation of the methods and tools (i.e. the features used for classification), and (ii) the evaluation of the machine learning algorithm and the hyperparameter values used. The methods and tools were identified in Section 4.5.4, and subsequently implemented in Section 5.4.

Finally, we compare the performance achieved by the best-performing model within the *GermaNLU* system with the performances of state-of-the-art publicly available commercial natural language understanding system and a naive bag-of-words based baseline, all of which have been introduced in the previous chapter.

6.1 Methodology

This quantitative evaluation is done by comparing the expected labels from the labeled evaluation dataset with the results from the classification results provided by the evaluated service. To evaluate the quality of the results, following the evaluations presented in

[Braun et al., 2017] these methods and measures are used: (i) confusion matrices, (ii) precision, (iii) recall and (iv) F-score.

Confusion matrix: this is the tabular visualization of the comparison of the predictions generated by a model and the pre-labeled data. Usually, rows are used to indicate the expected results, and columns the predictions. The intersection of rows and columns, i.e. the matrix cells, are filled with the number of items for which the expected and predicted results match the row and column indices. The most important of these values are: (i) true positives, (ii) false positives, (iii) true negatives and (iv) false negatives.

True positives: the number of data points of a given class = X, correctly classified as class X

True negatives: the number of data points of given class $\neq$ X, not classified as X

False positives: the number of data points of a given class $\neq$ X, incorrectly classified as X

False negatives: the number of data points of a given class = X, incorrectly classified as not X

Table 6.1 shows a simple example of a multi-class confusion matrix with 3 classes.

	Class A (pred)	**Class B (pred)**	**Class C (pred)**	
Class A (exp)	Correctly classified as A = **TP of A**	A, misclassified as B	A, misclassified as C	Row-sum of misclassifications = **FN of A**
Class B (exp)	B, misclassified as A	Correctly classified as B = **TP of B**	B, misclassified as C	Row-sum of misclassifications = **FN of B**
Class C (exp)	C, misclassified as A	C, misclassified as B	Correctly classified as C = **TP of C**	Row-sum of misclassifications = **FN of C**
	Column-sum of misclassifications = **FP of A**	Column-sum of misclassifications = **FP of B**	Column-sum of misclassifications = **FP of C**	

Table 6.1: An example of a confusion matrix for a three-class classification. The rows depict the expected classification results, i.e. as defined by the labels provided by the evaluation dataset. The column depict the classifications predicted by the classifier to be evaluated. The fields provide an explanation how the values for true positives, false negatives and false positives are calculated for each class, as these numbers are necessary for the calculation of precision, recall and F-score. Legend: TP = True Positives, FP = False Positives, FN = False Negatives

Precision: this measure (also defined in [Ting, 2010]) shows how precise the classification results of an evaluated classifier are. It is defined as the ratio between correctly classified instances to all classifications predicted by the classifier. For example, if a classifier is supposed to classify images as showing people or not showing people, the precision of this classifier would be calculated as follows:

Expected results:

- 5 images showing people, 3 not (total of 8 images)

Classification results:

- 4 out of 5 images showing people were correctly classified as showing people
- 1 image showing people was incorrectly classified as not showing people

- 1 image not showing people was incorrectly classified as showing people
- the rest was classified as not showing people

The resulting confusion matrix is illustrated in Table 6.2.

	People (pred)	Not people (pred)	
People (exp)	4 (=TP)	1	FN=1
Not people (exp)	1	2	
	FP=1		

Table 6.2: Example of a confusion matrix showing the classification performance of a fictional image classifier. Legend: True Positive = TP, False Positive = FP, False Negative = FN

Again, precision is defined as the ratio of correctly classified instances compared to all classifications given. In the above example, 4 out of 5 "people" classifications were correct, resulting in a precision value of 0.8. The following Equation 6.1 shows the general formula how to calculate precision:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{6.1}$$

Precision in a multi-class classification environment is calculated in a similar fashion, with the exception that values for false positives are the column-sums of all misclassifications (see Table 6.1), resulting in a per-class precision value. Overall precision is then calculated by taking the average over all per-class precision values, a method also known as "macro average precision", or alternatively via a method called "micro-average precision", where a each classes contribution to the overall precision value is weighed by the number of classified instances. This method is especially helpful if there is a large imbalance in class distribution. In the following, the value calculated by "macro-averaging" is used, as the class distribution of the evaluation dataset is close to even.

Recall: this measures (also defined in [Ting, 2010]) how good a classifier is at finding instances of a specific class. Is is defined as the ratio between correctly classified instances to all actual instances that the classifier should have identified. Reusing the example provided to explain precision, recall would be calculated in the following way: Of the 5 pictures that show people, the classifier was only able to find 4, therefore the recall value ids 4/5, or 0.8. The following Equation 6.2 shows the general formula how to calculate recall:

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{6.2}$$

Analogous to precision, also recall can be calculated for a multi-class classification setup, but in this case the false negatives (FN) are the row-sums of all misclassifications. Also, the calculations of overall recall can be calculated using the same macro- or micro-averaging method, and like with precision, for the evaluation done in this thesis "macro-average recall" will be used.

F-score: this measure (also defined in [Sammut and Webb, 2010]) takes into account both the precision and recall. It is a commonly used measure to assess the quality of a classification. The basic idea behind it is to evaluate both how exact and how exhaustive a classification is. It is used to devalue those classification models that are either highly precise, but with low recall, or vice versa. An example would be that a named entity recognizer model correctly labels two out of ten possible person mentions and nothing else, which would result in a perfect precision, since all the labels depicting a token to be a person were correct. The recall value, however, of this named entity recognizer would be a woeful 2/10. The F-score would take into account both values, providing us with a more realistic quality measure as either precision or recall on their own. The F-score value in this example would be 1/3, a value more consistent with the experienced quality of the labeling. Its most basic version the F1-score is the harmonic average between precision and recall. The following Equation 6.3 describes how the F1-Score is calculated:

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{6.3}$$

Within the scope of this evaluation, the term "F-score" refers to the "F1-score" measure.

6.2 Evaluation of Custom-Trained Named Entity Recognizer

To identify the custom named entity types used in both the training and evaluation datasets, a custom named entity recognition model was trained. Using the provided conditional random fields - based implementation from Stanford CoreNLP, two models have been trained and evaluated - one for German language, and another for English language. Listing 6.1 shows the commands for training and testing, Table 6.3 shows the confusion matrix of the evaluation of the German model, Table 6.4 the confusion matrix for the English model. Figure 6.1 shows the per-type and macro-averaged precision and recall values for the German model, Figure 6.2 for the English model.

```
# TRAIN GERMAN
echo "Training German NER model"
java -cp "*" -Xmx8g edu.stanford.nlp.ie.crf.CRFClassifier \
 -prop corenlp-ner-training.de.prop \
 -readerAndWriter \
 edu.stanford.nlp.sequences.CoNLLDocumentReaderAndWriter \
 -inputEncoding UTF-8 -outputEncoding UTF-8
```

```
# TEST GERMAN
echo "Testing German NER model"
java -cp "*" -Xmx8g edu.stanford.nlp.ie.crf.CRFClassifier \
 -loadClassifier ner-model.sport.de.ser.gz \
 -testFile corenlp-ner-test.de.tsv -readerAndWriter \
 edu.stanford.nlp.sequences.CoNLLDocumentReaderAndWriter \
 -entitySubclassification iob2 -inputEncoding UTF-8 \
 -outputEncoding UTF-8
```

Listing 6.1: Training and evaluation command for the custom-trained German language Stanford CoreNLP named entity recognizer.

	COMP	**TEAM**	**PLAY**	**NONE**
COMP	252	0	0	0
TEAM	0	427	2	88
PLAY	0	0	176	0
NONE	0	0	0	2390

Table 6.3: Confusion matrix for the German named entity recognition model evaluation. “COMP”, “TEAM”, and “PLAY” are the different classes of named entities. “NONE” includes any token that is not part of a named entity.

	COMP	**TEAM**	**PLAY**	**NONE**
COMP	244	0	0	8
TEAM	0	463	0	54
PLAY	0	0	176	0
NONE	0	0	0	2417

Table 6.4: Confusion matrix for the English named entity recognition model evaluation. “COMP”, “TEAM”, and “PLAY” are the different classes of named entities. “NONE” includes any token that is not part of a named entity.

As can be seen, the per-class performances of the models are quite similar, and especially the overall macro-averaged precision and recall values are virtually identical. No indication for a language-induced bias could be identified during training and evaluation.

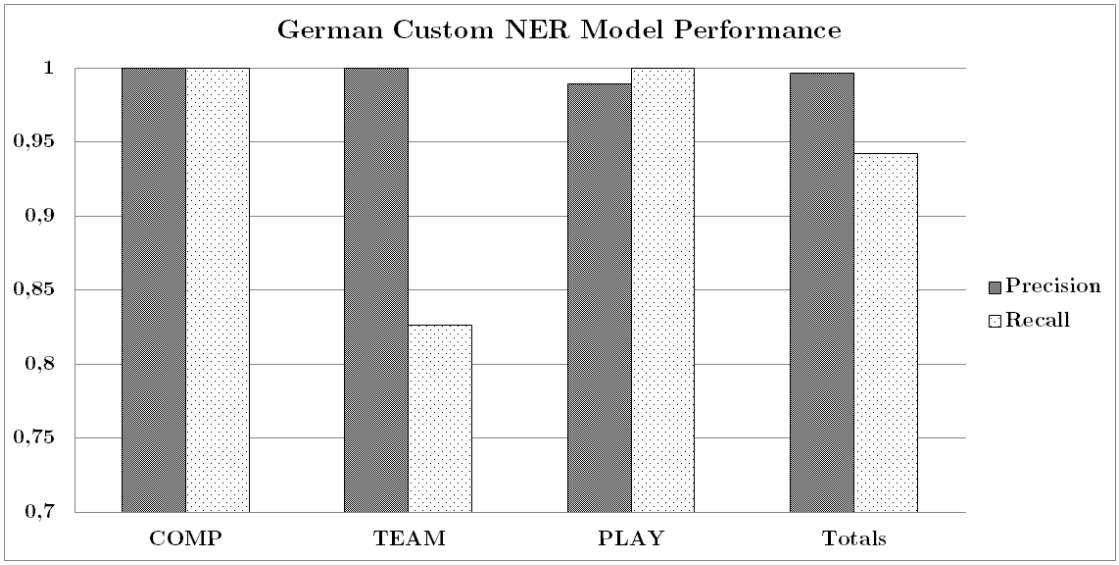


Figure 6.1: Per-class and macro-averaged precision and recall of the German named entity recognition model evaluation.

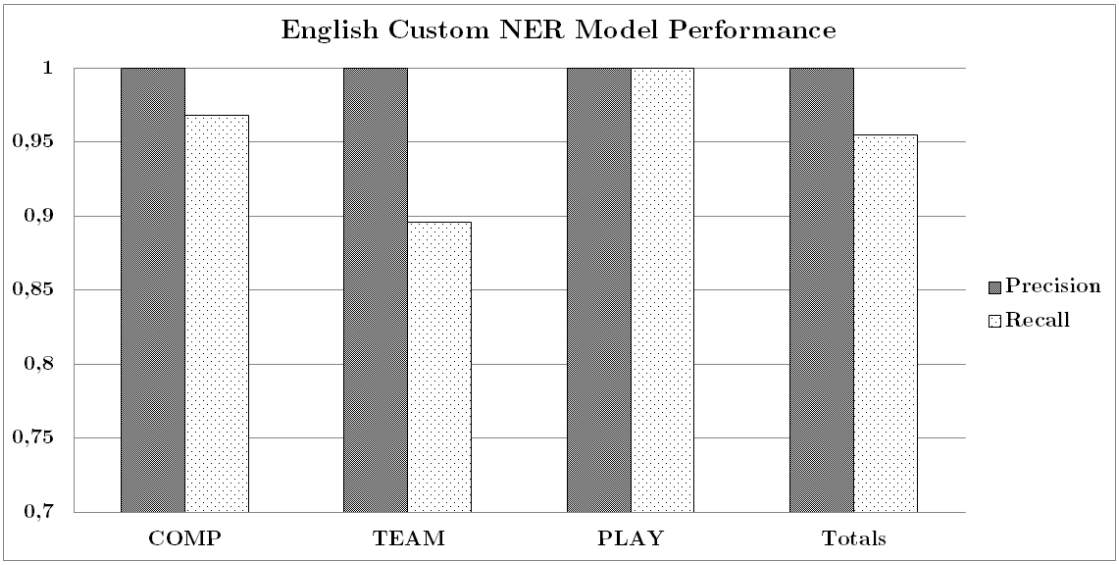


Figure 6.2: Per-class and macro-averaged precision and recall of the English named entity recognition model evaluation.

Feature	Dimensionality	AIG (German)	AIG (English)
Smallest Word Vector	300	1.26	0.77
Biggest Word Vector	300	1.29	0.60
Average Word Vector	300	1.80	1.63
Dependency Root Word Vector	300	1.88	1.82
Average Verb Word Vector	300	0.00	1.53
Average Word Vector of First 3 Words	300	1.80	1.63
Bag-of-Words Vector	251/254*	0.06	0.06
Number of COMP entities	1	0.48	0.48
Number of TEAM entities	1	0.86	0.84
Number of PLAYER entities	1	0.83	0.80

Table 6.5: Results of the information gain analysis of the calculated features of the German and English language training sets. *: English and German language dataset vocabularies have slightly different sizes, with the German vocabulary containing 3 words less compared with the English dataset vocabulary. Legend: AIG = Average Information Gain

6.3 Evaluation of Feature Selection for Custom NLU Pipeline

As previously described (see Section 5.4), this step was taken to limit the potential impact of the so-called "curse of dimensionality". To identify a set of features to best predict the intent, the information gain for each of the features was calculated using the vector representations calculated internally by the custom NLU service implementation for the German and English language training datasets. As most of the features are n-dimensional vectors, the average information gain for each of the features was used as the deciding factor. The calculation was done using the in-built functionality provided by the machine learning tool Weka and some post-processing in Microsoft Excel. Table 6.5 shows the results of the information gain evaluation of both the German and English language features.

The biggest difference between the two languages is the impact of the average verb word vector, which can be explained by the shortcomings of the German language word2vec model, which exclude many of the basic verbs like "hat", "ist" or "sein". As words outside the vocabulary are represented by either a completely random vector, an all-zero vector

or the overall average vector of the whole vocabulary, this feature should arguably have no positive impact on the classification, an intuition that can be verified by the calculated information gain values. Another difference can be observed in the differences between the averages information gain of the biggest and smallest word vectors. This can be also interpreted as a result of the differences in the natures of the word vector models.

6.4 Evaluation of Machine Learning Algorithms for Intent Classification

To infer the intent from a natural text utterance, we train a machine learning algorithm to predict the intent of a text utterance based on a set of training data. Several different algorithms can be used, and many of these algorithms have parameters affecting the way that their predictions are built using the training data. To find the algorithm/parameter combination providing the best results, we will train several combinations and evaluate the performance of the classifier using a shared evaluation dataset. The algorithms and their tunable parameters are listed in Table 6.6.

Algorithm	**Kernels**	**Parameters**
Random forests (ensemble)	n/a	n/a
Decision tree	n/a	Confidence factor [0.05, 0.10, 0.15]
Support vector machine	[Linear, Radial base function]	Complexity parameter [0.01, 0.1, 1, 2, 5, 10, 50, 100]

Table 6.6: List of the machine learning algorithms and the parameters with values used in the evaluation process.

Following the findings presented in the evaluation of machine learning algorithms in text classification in [Sebastiani, 2002], we chose three different approaches, which showed the best performances: (i) ensemble methods, (ii) decision trees and (iii) support vector machines.

Ensemble methods: technically not a algorithm, but a method combining multiple classifiers using one or more algorithms. We chose to use random forests [Breiman, 2001] as the ensemble method of choice. Random forests operate by creating a multitude of decision trees, combining the results. The machine learning tool Weka offers a mature implementation, and it is also often referred to in the literature we analyzed, e.g. in [Baudiš and Šedivý, 2015] or [Braslavski et al., 2017].

Decision trees: in this approach an algorithm is used to create tree structures, where the leaf nodes are class labels, and the non-terminal node are if-then-else style choices,

e.g. if y<=1 then go left, else right. We chose to use the C4.5 algorithm [Quinlan, 1993], as the Weka machine learning tool provides a mature implementation.

Support vector machines: a linear classifier, the main advantage lies in applying so-called kernel functions, mapping the non-linearly separable inputs into a high-dimensional space where the data points can then be linearly separated [Cristianini and Shawe-Taylor, 2000]. support vector machines have been used in many of the analyzed systems, e.g. in [Zhang and Lee, 2003], [Carvalho et al., 2017] or [Asiaee et al., 2015]. Weka also offers an implementation offering linear and radial base function-based kernel functions [Keerthi and Lin, 2003].

The tunable parameters "confidence factor" and "complexity parameter" affect the algorithm behavior in such a way that the "confidence factor" is used to decide when to prune nodes during training a decision tree model, smaller values incurring more pruning, while the "complexity parameter" is used to weigh hyperplane optimization against allowing misclassifications when training a Support Vector Machine, smaller values giving more weight towards hyperplane optimization, allowing more misclassifications during the training process. Both parameters can be interpreted in such a way that smaller values should result in better generalization, while bigger values will result in a more precise fit regarding the training set. The evaluation was done on four different feature combinations from the training set: (i) including all features, (ii) including all features except the bag-of-words features, (iii) only the best features with regard to the information gain evaluation results, and (iv) only the bag-of-words features, the latter representing the model used for the naive bag-of-words baseline. The reasoning behind this is to find out what the impact of feature selection according to the information gain value is compared to the use of all features, and to see what the performance would be if we combined the naive bag-of-words features with the custom high-level features. Table 6.7 shows a detailed description of the different training set variations.

Table 6.8 presents the results of the evaluation, showing the best performing algorithm/-parameter combination for every dataset version, with the model achieving the highest F-score considered the best. If more than one algorithm/parameter combination share the same F-score, the model with the highest precision is considered best. If still more than one combinations are considered best, then all the models are considered equally good and are included in this listing.

Label	Language	Features	Dimensions	Description
DE-ALL	German	All	2054	All features calculated by the custom NLU implementation.
DE-SEM	German	All but bag-of-words	1803	All features except the bag-of-words vector. This is the "full" set of semantic and syntactic information extracted. This was used to compare the classification performances of high-level features compared to a naive baseline.
DE-BEST	German	Best only	1500	Best features according to information gain calculation. This was used to find out if lower dimensionality has an impact on classification. Heuristically, all features with an information gain value above 1 were included. Note: an alternative version with 1200 dimensions was also tested, excluding the smallest word vector feature This dataset variation did, however, not show any different results, and is therefore omitted from the remainder of this work.
DE-BOW	German	Bag-of-words only	251	Only bag-of-words vector. This is the base for the "naive" baseline implementation.
EN-ALL	English	All	2057	All features calculated by the custom NLU implementation.
EN-SEM	English	All but bag-of-words	1803	All features except the bag-of-words vector. This is the "full" set of semantic and syntactic information extracted. This was used to compare the classification performances of high-level features compared to a naive baseline.
EN-BEST	English	Best only	1200	Best features according to information gain calculation. This was used to find out if lower dimensionality has an impact on classification. Heuristically, all features with an information gain value above 1 were included.
EN-BOW	English	Bag-of-words only	254	Only bag-of-words vector. This is the base for the "naive" baseline implementation.

Table 6.7: Listing and description of the datasets used in machine learning algorithm evaluation. Each of the datasets was used to train and evaluate all of the machine learning algorithm-parameter combinations shown in Table 6.6. The best combination was used to calculate a model for the final language understanding pipeline.

Dataset	Algorithm	Parameter	F-Score (Pr.)
DE-ALL	SVM with linear Kernel	C=0.01	0.82 (0.84)
DE-SEM	SVM with linear Kernel	C=0.01	0.77 (0.81)
DE-BEST	SVM with linear Kernel	C=0.01	0.78 (0.83)
DE-BOW	SVM with linear Kernel, SVM with RBF Kernel	C=[0.01, 0.1, 1, 2, 5, 10, 50, 100]	0.71 (0.86)
EN-ALL	Random Forests	n/a	0.71 (0.81)
EN-SEM	SVM with linear Kernel	C=0.1	0.70 (0.80)
EN-BEST	SVM with linear Kernel, SVM with RBF Kernel	C=[0.01, 0.1, 1, 2, 5, 10, 50, 100]	0.72 (0.78)
EN-BOW	SVM with linear Kernel	C=0.01	1.00 (1.00)

Table 6.8: Performance evaluation of the machine learning algorithms on different variations of the training and test data. Note the perfect score achieved for the English language bag-of-words dataset, and the German language variation with only the best custom features outperforming the German language naive bag-of-words approach.

Looking at the result, immediately the perfect score achieved for the English language naive bag-of-words baseline comes to attention. Comparably high scores could be achieved by all SVM kernel/parameter combinations. Evaluation of the German language dataset variations has shown that neither the custom nor the bag-of-words features show the best results on their own, but the combination of both. Interestingly, the dataset variation using only the best custom features performs better than the naive baseline, a result which, if at all, was expected for the English language datasets. Regarding the approach to mitigate the impact of the "curse of dimensionality", slightly better results could be achieved for both languages when reducing the dimensions. The reduction was 303 dimensions in the case of German language, and 603 dimensions in the case of English language. For the final custom natural language service, both language versions used only the best custom features and their best performing algorithm/parameter combinations. The generated models were then used in the comparison with the commercial natural language understanding services to evaluate the quality of our extracted features and the classification models.

6.5 Evaluation of Natural Language Understanding Services

Until now we have only evaluated the performances of different feature and classification algorithm combinations within the custom implementation. Using the results from the best sets of features and algorithm/parameter combinations, we will compare their performance with that of commercial and publicly available natural language systems which have been trained with the same training data, and evaluated using the same evaluation data as our custom service. As the named entity recognition approaches of the systems differ too much, no fair overall comparison is possible, and is therefore not part of this evaluation. We will evaluate based on the macro-averaged F-score and precision values, like in previous section. The systems used in this evaluation are (the systems implemented for this thesis are emphasized):

- Microsoft LUIS
- IBM Watson Assistant
- wit.ai
- rasa NLU
- *GermaNLU*, the custom NLU system using the most popular state-of-the art methods and tools
- *Bag-of-words baseline*, a custom baseline NLU system using a naive bag-of-words approach

Table 6.9 shows the raw results of the evaluation, Figure 6.3 a visualization. The corresponding confusion matrices can be found in the appendix.

What all commercial services had in common, was that their English language performances are better than their corresponding performances in German language. The intent classification performance they provided in English language was consistently above that of our custom implementation, but below the naive baseline. Interestingly, our custom implementation was consistently better in German language intent classification performance, which show comparable performance to that of the naive baseline. The promising results provided by rasa NLU in the evaluation done in [Braun et al., 2017] could not be reproduced.

6.6 Summary

In this chapter we have presented the results of the evaluation of the case study. First, we had a look at the performance of the custom-trained named entity recognition models in German and English language, which showed virtually identical results for

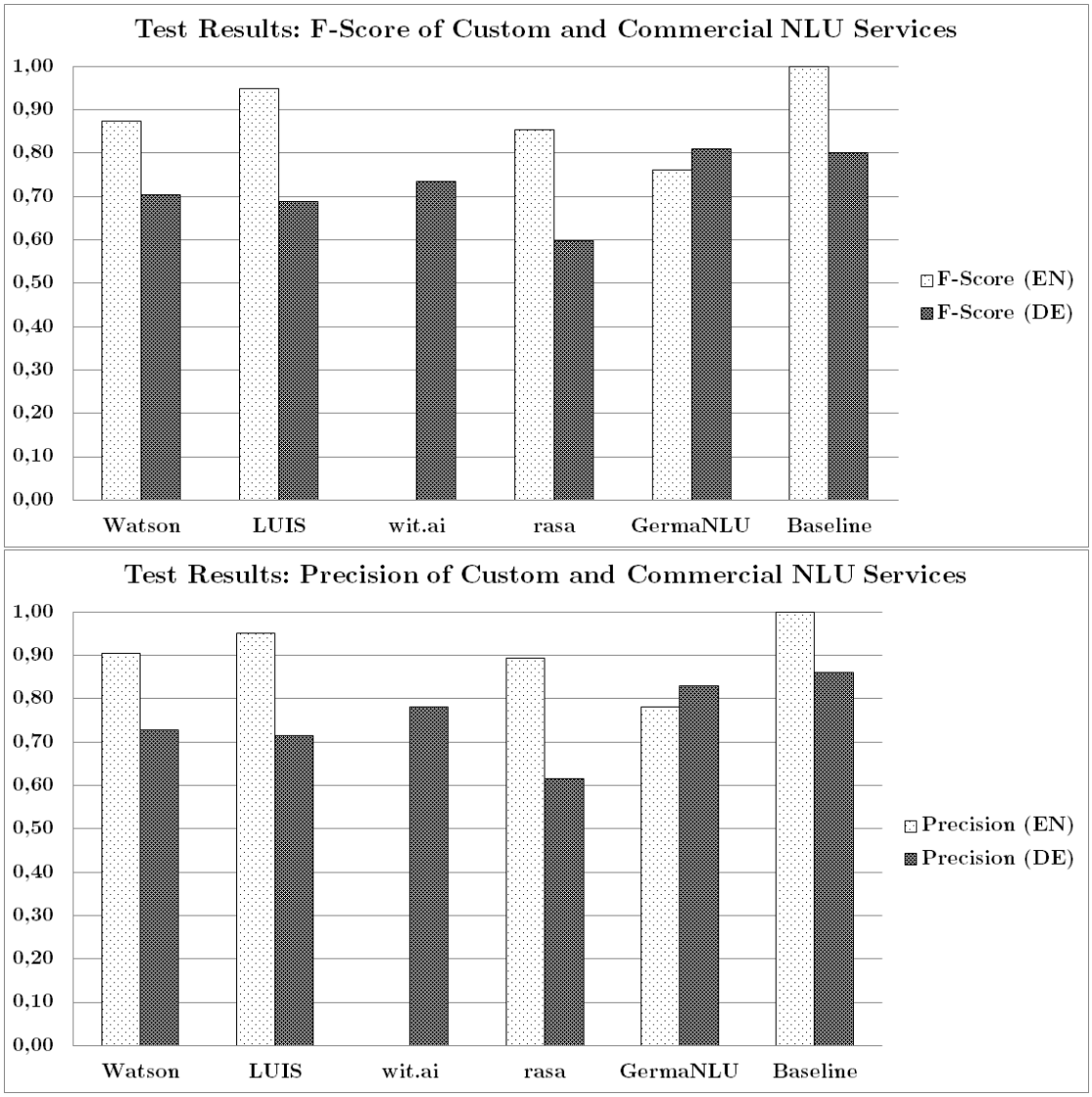


Figure 6.3: F-scores and precision values of evaluation results of all systems. The majority of German language evaluation results are below their English language counterparts, with the exception of the custom implementation. Note: wit.ai could not be trained in English language. Also notice that precision values are closer together than F-scores, as recall values are all slightly worse than their corresponding precision values.

Service	Language	F-Score	Precision	Recall
IBM Watson	English	0.87	0.90	0.85
IBM Watson	German	0.70	0.73	0.68
Microsoft LUIS	English	0.95	0.95	0.94
Microsoft LUIS	German	0.69	0.71	0.66
wit.ai	English	n/a	n/a	n/a
wit.ai	German	0.73	0.78	0.69
rasa NLU	English	0.85	0.89	0.82
rasa NLU	German	0.60	0.61	0.58
GermaNLU	English	0.76	0.78	0.75
GermaNLU	German	0.81	0.83	0.78
Baseline	English	1.00	1.00	1.00
Baseline	German	0.80	0.86	0.75

Table 6.9: The macro-averaged f-score, precision and recall values measured during evaluation of the custom implementations and the commercial natural language understanding systems. Note: wit.ai could not be trained in English language.

either language. Then, we evaluated the impact feature engineering could have on the classification performance. With the assumption that high-dimensional feature vectors could have negative impact on the predictive performance of a model, we first calculated the information gain measure as a ranking metric on the features to find out which features promise the strongest discriminative impact. Then, we trained different version of the dataset, each with different sets of features used, against several combinations of machine learning algorithms and their corresponding training parameters. The findings presented in Table 6.8 showed that selecting the features with the highest average information gain did show a slight positive impact on the quality of the model. The biggest impact on predictive quality, however, was the choice of machine learning algorithm and their training parameters. A total of 20 different combinations have been evaluated, showing that in the context of this dataset support vector machines with a linear kernel function and a complexity parameter of 0.01 had the best overall performance. Using the best sets of features and intent classification models, we then evaluated the custom natural language understanding system against the other systems, which had been trained using the same datasets, and evaluated using the same evaluation dataset. The results showed that our custom implementation could not compete with commercial systems in English language, in German language, however, the performance of our custom system was better than that of any of the commercial systems, suggesting that the methods and tools identified during the state-of-the-art research can indeed be successfully applied to German language systems.

With regard to the third research question (*"Is there a measurable difference in the performances of current methods, tools and services when using German language, as*

compared to English?") and its corresponding sub-question (*"If so, are there methods, tools or services for which these differences are smaller or non-existent?"*), yes, we could identify some methods that show differences when applied in either language. Word embeddings and lexical databases showed differences in their behaviors, the former because the available German language models are of inferior quality as compared to their English language counterparts, the latter because German language equivalents for English language lexical databases are either not usable in the same manner, or simply do not exist. As can be seen in Table 6.5, the other methods and their corresponding tools have not shown any language-dependent differences in the context of the evaluation of this case study, assuming that any such difference would manifest itself in a large discrepancy in average information gain provided by the resulting features. In the next chapter, we will discuss these findings in more details, and following that, present a final summary.

CHAPTER 7

Summary and Discussion

In this chapter we will discuss the combined findings presented in this thesis. The findings will be organized around the research questions.

7.1 RQ1: Is there a common definition of the term chat-bot, and if so, what is it?

To find an answer to this question, we first did a literature analysis to find out what definitions are used in publications using "chat-bots" either within their titles, their abstracts or their keywords. Also, the definitions provided in books with dedicated chapters on the subject were included. The analysis of the definitions provided in the literature quickly showed that no definitive definition for the term "chat-bot" exists. It is often used synonymously with other terms, like conversational systems, and the definitions provided in the literature often contradict each other. To be exact, [Jurafsky and Martin, 2017a] explicitly state that chat-bots are "non-task-oriented", while other sources - e.g. [Kincaid and Pollock, 2017, Dale, 2016] - use it synonymously with the term "virtual assistant", which is a direct contradiction. However, when looking at a larger number of definitions, an adapted taxonomy can be deduced, which is more in lieu with the terminology used in most publications. It can be seen in Figure 7.1

It is unclear if the authors are unaware of the definition postulated by [Jurafsky and Martin, 2017a], or if it is simply outdated. Another possible explanation for this divergence is the fact that the term "chat-bot" saw such a big hype, that many researcher felt compelled to include the term in their work even when only remotely relevant, or just labeled their implementations "chat-bots" to benefit from the heightened attention. This, however, is pure speculation, and can not be proven. When considering use of the term in the majority of publications analyzed, the updated taxonomy provides a much better fit than any other definition or classification attempt so far. By splitting

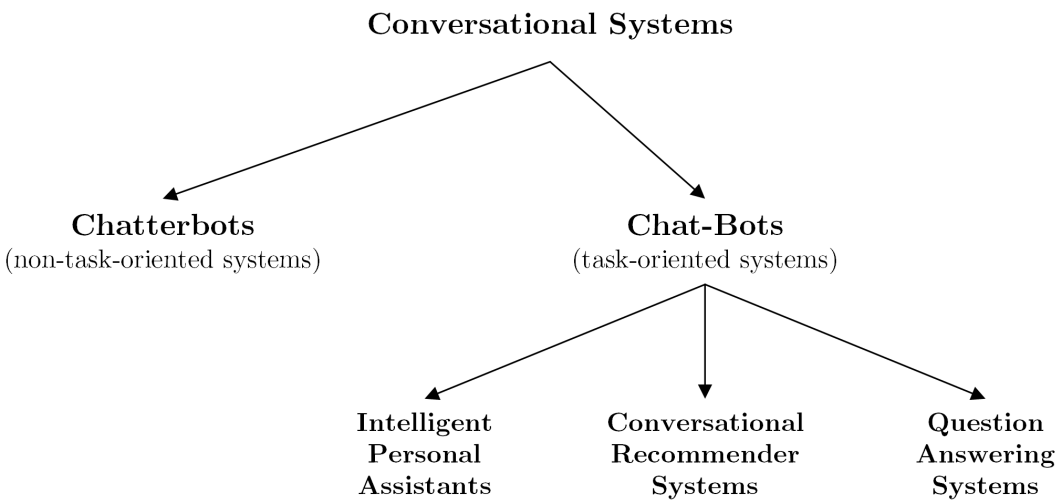


Figure 7.1: The deduced taxonomy locating the terminology within the field of conversational systems and defining their interconnections.

up the hitherto synonymously used terms "chatterbot" and "chat-bot", and defining the former as non-task-oriented, effectively taking over the definition of the term "chat-bot" postulated in [Jurafsky and Martin, 2017a], and defining the latter as synonymous to all task-oriented conversational systems, we provide a better fit between the definition of the term and its actual usage in the literature. Another addition was the emergence of three distinct subclasses of chat-bots/task-oriented conversational systems, which were also included in the proposed taxonomy. We postulate that this updated taxonomy better reflects the status quo as presented by the current literature, as well as provides a relative positioning of the most common terms between each other, which should help in future searches for fitting literature. To circle back to the research question, while no official definition of the term exists, it can be deduced from analyzing the current literature, that the best fit is the definition of task-oriented conversational systems, i.e.:

Chat-bots are software agents capable of natural language based interactions, supporting the user in fulfilling a specific task.

7.2 RQ2: What is the current state-of-the-art regarding language-dependent methods, tools and services involved in creating chat-bots?

First we analyzed the architectures of chat-bots described in the literature, to find which parts of a chat-bot are actually affected by the language used. Deriving a common, basic architecture, four distinct modules could be identified: (i) a dialog management module, responsible for handling text input and output, (ii) a natural language understanding

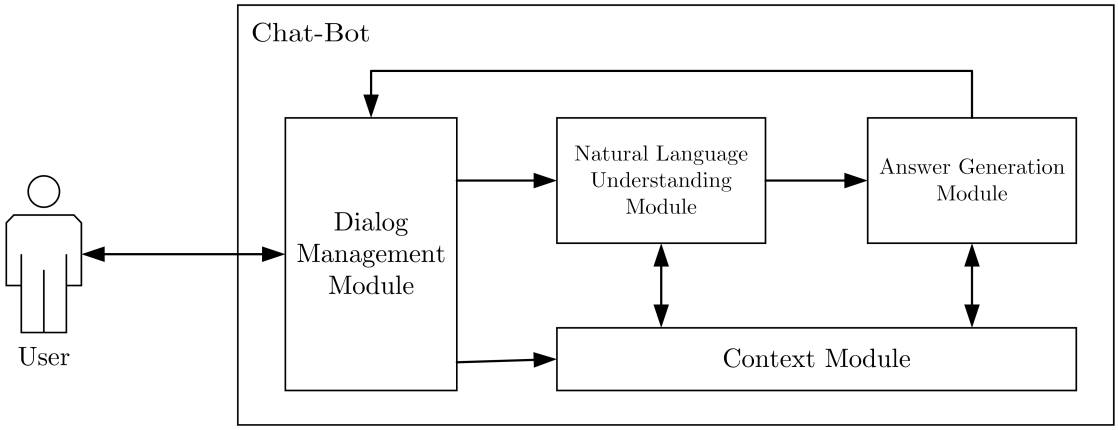


Figure 7.2: Basic unified architecture of a chat-bot system. It is derived from the architectures presented in the cited works. Four distinct modules could be identified, as well as their interactions.

module, responsible for extracting information from the natural language input, as well as classifying the intent of it, (iii) a context module responsible for handling the state of the chat-bot application, and (iv) an answer generation module, responsible for producing a fitting answer depending on the state and the extracted data from the input. The architecture can be seen in Figure 7.2. As only the natural language understanding module is interpreting the natural language utterances, the search for state-of-the-art methods and tools affected by the language used was focused on that field. Methods used in I/O handling, holding conversation state or generating answers from an attached knowledge base were considered out of scope, as they are not directly affected by the type of language used.

During the initial research phase it became apparent that for one, question answering systems can be considered one of the basic types of chat-bot, and that since question answering systems carry a long history, the term is well established and a large number of literature can be found using it as a search term. Therefore a large number of publications describing question answering systems was analyzed with a focus on the methods and tools used in natural language understanding. Following the structure of the work of [Jurafsky and Martin, 2018], eight method groups were defined, and assuming that methods and tools that are more commonly used present a good approximation of the current state-of-the-art, a quantitative analysis was done using the question answering systems and the eight method groups. As a result of this analysis, the most commonly used methods could be identified, and were described in detail. The most commonly used tools were also listed in the detailed description of the methods. Figure 7.3 shows the relative number of question answering systems applying the methods in their natural language understanding modules.

In reference to the research question, the presented distribution of methods used can be considered the state-of-the-art in natural language understanding, with the exception of

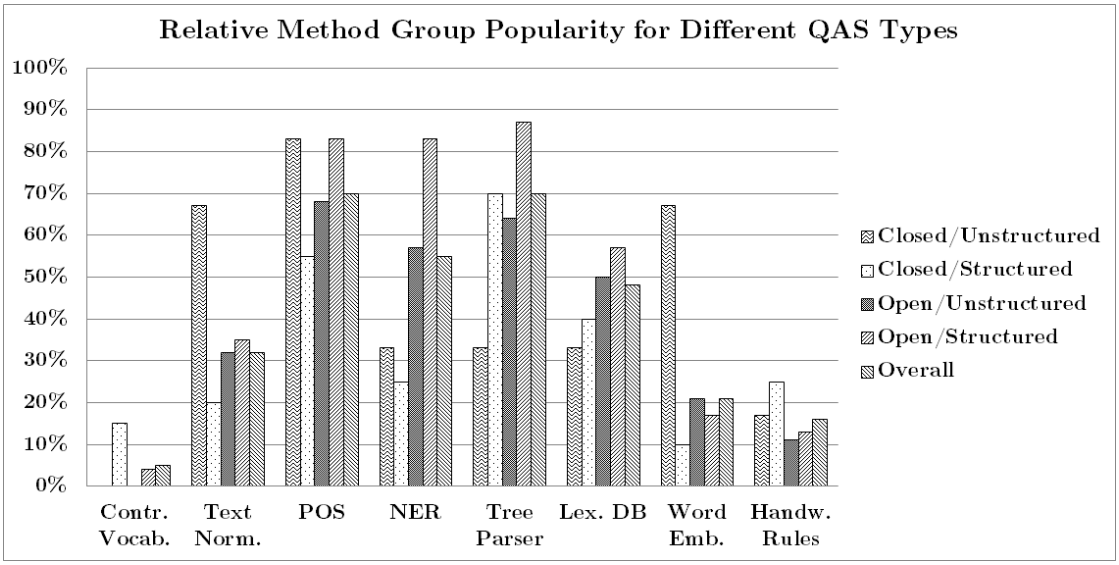


Figure 7.3: Relative number of question answering systems using the eight method groups in their natural language understanding modules. Note that the question answering systems themselves have been split into several different groups, depending on the domain and type of data source to find out if some bias exists towards specific types of question answering systems.

word-embedding, which, as part of a large-scale trend that could be identified, can now be considered one of the most important methods used in natural language understanding. This large scale trend is the now ubiquitous application of deep neural nets to practically every sub-field in natural language understanding. Especially sequence labeling tasks such as part-of-speech tagging, named entity recognition and dependency parsing are now dominated by deep neural network implementations. There is no indication that this trend will change, in fact it can be assumed that within the next years practically "conservative" approaches will be replaced by deep neural networks. Some implementations go so far as to use end-to-end neural network implementations to replace and merge their natural language understanding, answer generation and context modules. If this will become the state-of-the-art, however, is left to be seen.

7.3 RQ3: Is there a measurable difference in the performances of current methods, tools and services when using German language, as compared to English?

Taking the most popular tools described in the analyzed literature, we first evaluated if the tool supports German language. During that evaluation phase it became obvious that

especially lexical databases pose a problem in German language. Most lexical databases used in the literature have either no publicly available German language pendant (e.g. WordNet), or the German language version that exist, can not be used in a similar fashion (e.g. Wortschatz, which provides detailed lexical information only on its website). While some methods are language independent, like regular expressions or grammars, tools or models explicitly supporting German language could be found for all of the most popular methods. The most popular methods and tools that were identified as potentially viable for use in a German language natural language understanding implementation are: (i) Stanford CoreNLP (used for text normalization, part-of-speech tagging, named entity recognition and tree parsing) and (ii) word2vec (used for word embeddings).

Using these tools, a custom natural language service was implemented. The implementation works as follows: First, the natural language input is tokenized, and all methods are applied to annotate the input text. These annotations (e.g. part-of-speech tags or word vector representations) are used to calculate a fixed length vector representation of the input text. This fixed length vector representation is the concatenation of multiple features, like the number of identified named entities, or the word vector of the dependency tree root. This vector is then used as input for a intent classification model, which predicts the intent type of the text.

To optimize intent classification performance, we evaluated several different feature combinations, as well as several machine learning algorithm/parameter combinations. The feature selection was done using the information gain measure to rank the discriminatory power of a feature, and the selection of the machine learning algorithm and their parameters was done by training 20 different combinations with four different combinations of features. The evaluation showed that the best performance for English language could be achieved with a 1500 dimension feature vector and a support vector machine with linear kernel function. For German language, a 1200 dimension feature vector showed the best results, also in combination with a support vector machine with linear kernel function. The evaluation results for those two best combinations were then compared with the evaluation results of four commercial systems, IBM Watson, Microsoft LUIS, wit.ai and rasa NLU, as well as with a naive baseline using a bag-of-words approach. The results of this evaluation can be seen in Figure 7.4.

The results showed that within the context of the case study training and evaluation data, our custom implementation reached comparable, even slightly better results in German language as compared with the commercial system. English language performance, however, was clearly inferior. Interestingly, the naive baseline showed a perfect result for English language, and also the second-best result for German language. The difference in English language performance was to be expected, as the main focus of all the commercial systems is clearly the English language, and competing performance from an implementation with limited engineering effort would seem unlikely. German language performance differences, however, showed that even with limited effort, using off-the-shelf tools, a competitive performance was reachable. This could indicate that either the commercial systems do not focus on German language, a fact that would reflect the

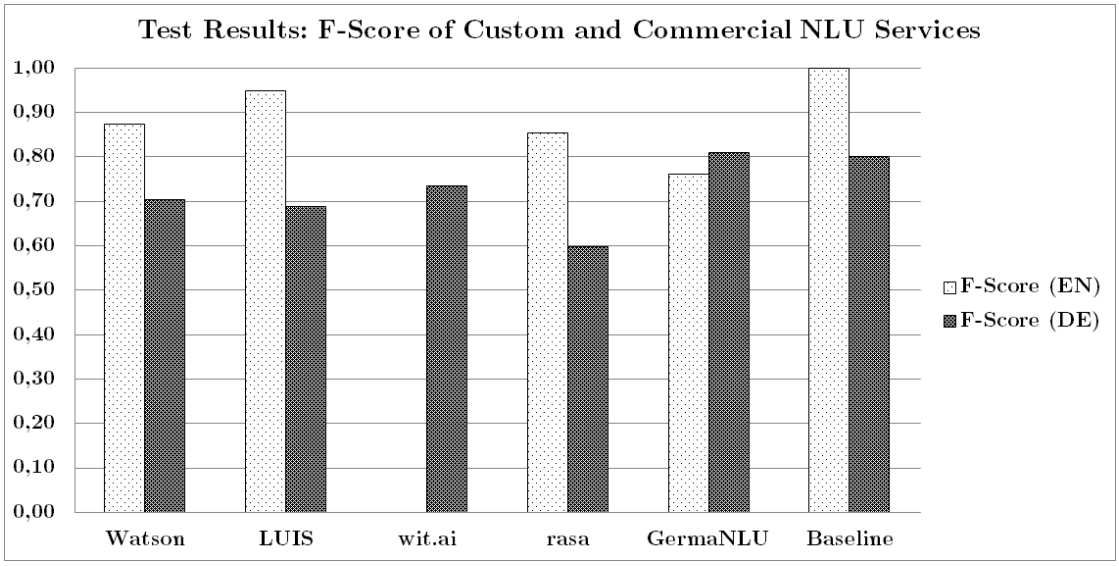


Figure 7.4: The comparison of the performances of four commercial services with the custom implementation using the most popular natural language understanding methods and tools and a naive bag-of-words baseline.

status-quo in natural language processing as a whole. The performance of the naive baseline has shown, that for simple use cases like our sports-themed corpus consisting of short questions, a bag-of-words approach is viable and can show good results.

As to the research question: yes, there are differences in the methods when comparing English and German languages. For one, there are some methods that are inherently language independent, like regular expressions and grammars. The quality of the results can be attributed to the author of the rules, and not the language. Then there are methods for which explicit and mature support for German language is provided, e.g. for part-of-speech tagging, named entity recognition and tree parsing. Then there is word embeddings, for which at the time of authoring the implementation, no publicly available high quality models for German language could be found. During feature selection we could identify that the German model had large gaps in the supported vocabulary, which rendered the use of some features impractical. A problem the English language model did not have. It has to noted however, that in the meantime alternative word embeddings like FastText have published official German models. As we did not use FastText, no comments about the quality of the German model can be made, but it can be assumed that the quality is at least comparable to the word2vec model. Lastly, there is the use of lexical databases which is prevalent in almost every English language system analyzed. Sadly, there are no publicly available German language alternatives for the most widely used English databases, which means that approaches using lexical databases can not easily be translated to German language systems.

7.4 Contributions

In the scope of this thesis, we have provided an overview of the current terminology used in the field of chat-bots, and conversational systems in general. We have shown that the official definition of the term chat-bot does not match with the usage in academic and commercial publications. With some adaptations however, a fitting taxonomy can be defined, covering the most important branches in current chat-bot development.

To identify the language-dependent components of a chat-bot, we have also derived a general, low-level architecture. It shows that in the context of chat-bot development, a dedicated natural language understanding module can be identified. This module is the only component of a chat-bot that needs to be adapted to the language used.

Conducting a broad literature review, we have identified the most commonly applied methods in natural language understanding. The main focus of this literature review was on the natural language understanding methods applied in question answering systems, due to the popularity of the field and the well-definedness of the term, providing us with a large number of publications for analysis.

We then implemented a custom natural language understanding system called "GermaNLU", using only the most popular methods and tools. We have shown that most of the methods and tools used in the development of English language natural language understanding systems can be applied to German language systems. The "GermaNLU" system can be adapted to other domains, given appropriate training data.

We have also identified those methods and tools which will need additional work to provide equivalent functionality in German language, and should be either avoided or used with care. Evaluating this system using a sports-themed parallel English/German dataset against the some of the most popular commercial natural language understanding systems, we have also shown that even with limited implementation effort and using only off-the-shelf tools and models, state-of-the-art performance could be achieved.

CHAPTER 8

Lessons Learned and Future Work

The most interesting learning was that there is a good coverage for German language considering the availability of high-quality tools and models, and that the basic approaches used in English language systems could be reused in a German language implementation. Another interesting aspect is the recent trend towards deep learning. With the advent of more and more sophisticated neural network architectures, capable of capturing high-level semantic features using only unlabeled raw data as input, it can be expected that the gap between languages might soon be closed, or close to it.

This leads directly to another interesting aspect, the availability of training data. As supervised machine learning approaches depend on manually labeled data, or, as shown in our case, on external knowledge bases to produce usable training and evaluation data, the number of publicly available high-quality datasets is limited. Without a common baseline however, results presented in the scope of a publication can not easily be verified or reproduced. This dependence on a limited number of datasets has lead to the the situation that advances in the individual subfields of natural language processing were often triggered by the emergence of usable datasets. Shifting towards unsupervised, or distantly supervised training methods, however, might solve this issue, as the manual effort will be limited to the production of baseline evaluation sets. Therefore it can be expected, that the evolution cycles of advances in the field of natural language processing are getting shorter and shorter. Combined with the recent advances in automated machine translation, differences in languages might possibly disappear in the near future.

Regarding the learnings gathered during the implementation of the case study, it has turned out that Java is not an optimal choice. Especially when it comes to machine learning, Python provides both a much larger community, as well as mature and well documented libraries, something that cannot be said about Java in that field. Also, Python, in combination with Jupyter notebooks, provides a much more flexible approach towards data-driven experiments as any Java-based alternative. If tasked again with

the implementation of a natural language understanding system, Python would be the obvious choice.

With regard to future works, this thesis provides a multitude of potential extensions. For one, an evaluation of the intent classification performance based on several different datasets could lead to more sophisticated insights regarding the behavior of the evaluated tools and methods. Another addition would be the creation of a structured mapping survey of the field of conversational systems, based on the literature analysis already done. Also, the existing system "GermaNLU" while needing some additional work, could serve as a basis for a domain-independent, German language natural language system alternative. And finally, an integration and evaluation of deep-learning approaches would provide the chance to evaluate their language-independence, and point towards future adaptations in that field.

List of Figures

List of Tables

Bibliography

[Abacha and Zweigenbaum, 2015] Abacha, A. and Zweigenbaum, P. (2015). MEANS: A medical question-answering system combining NLP techniques and semantic Web technologies. *Information Processing and Management*, 51(5):570–594.

[Adwait Ratnaparkhi, 1996] Adwait Ratnaparkhi (1996). A maximum entropy model for part-of-speech tagging. *In Proceedings of the Empirical Methods in Natural Language Processing Conference*, 1(49):133–142.

[Aggarwal and Buitelaar, 2012] Aggarwal, N. and Buitelaar, P. (2012). A system description of natural language query over DBpedia. *CEUR Workshop Proceedings*, 913(Ild):96–99.

[Aha et al., 2001] Aha, D. W., Breslow, L. A., and Muñoz-Avila, H. (2001). Conversational Case-Based Reasoning. *Applied Intelligence*, 14(1):9–32.

[Al-Zubaide and Issa, 2011] Al-Zubaide, H. and Issa, A. A. (2011). OntBot: Ontology based ChatBot. *2011 4th International Symposium on Innovation in Information and Communication Technology, ISIICT'2011*, pages 7–12.

[Allen et al., 1999] Allen, J., Guinn, C., and Horvtz, E. (1999). Mixed-initiative interaction. *IEEE Intelligent Systems and their Applications*, 14(5):14–23.

[Allen and Hayes, 1989] Allen, J. F. and Hayes, P. J. (1989). Moments and points in an interval-based temporal logic. *Computational Intelligence*, 5(3):225–238.

[Ameixa et al., 2014] Ameixa, D., Coheur, L., Fialho, P., and Quaresma, P. (2014). Luke, I am your father: Dealing with out-of-domain requests by using movies subtitles. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8637 LNAI:13–21.

[Amit et al., 2017] Amit, P., Marimuthu, K., Nagaraja, R., and Niranchana, R. (2017). Comparative study of cloud platforms to develop a chatbot. *International Journal of Engineering & Technology*, 6(3):57–61.

[An et al., 2017] An, C., Huang, J., Chang, S., and Huang, Z. (2017). Question similarity modeling with bidirectional long short-term memory neural network. *Proceedings -*

2016 IEEE 1st International Conference on Data Science in Cyberspace, DSC 2016, (2006):318–322.

[Androutsopoulos et al., 1995] Androutsopoulos, I., Ritchie, G. D., and Thanisch, P. (1995). Natural Language Interfaces to Databases - An Introduction. *Journal of Natural Language Engineering*, (709):50.

[Aronson and Lang, 2009] Aronson, A. R. and Lang, F.-M. (2009). The Evolution of MetaMap, a Concept Search Program for Biomedical Text. In *AMIA Annual Symposium Proceedings*, page 1990.

[Asiaee et al., 2015] Asiaee, A. H., Minning, T., Doshi, P., and Tarleton, R. L. (2015). A framework for ontology-based question answering with application to parasite immunology. *Journal of Biomedical Semantics*, 6(1):1–25.

[Auer et al., 2007] Auer, S., Bizer, C., Kobilarov, G., Lehman, J., Cyganiak, R., and Ives, Z. (2007). DBpedia: A Nucleus for a Web of Open Data. *The Semantic Web. Lecture Notes in Computer Science*, 4825:722–735.

[Avula, 2017] Avula, S. (2017). Searchbots: Using Chatbots in Collaborative Information-seeking Tasks. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '17*, (2010):1375–1375.

[Babar et al., 2017] Babar, Z., Lapouchnian, A., and Yu, E. (2017). Chatbot design - Reasoning about design options using i* and process architecture. In *CEUR Workshop Proceedings*, pages 73–78.

[Baizal et al., 2016a] Baizal, Z. A., Widyantoro, D. H., and Maulidevi, N. U. (2016a). Factors Influencing User's Adoption of Conversational Recommender System Based on Product Functional Requirements. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 14(4):1575.

[Baizal et al., 2016b] Baizal, Z. K. A., Widyantoro, H., and Maulidevi, U. (2016b). Design of Knowledge for Conversational Recommender System Based on Product Functional Requirements. In *Design of knowledge for conversational recommender system based on product functional requirements." Data and Software Engineering (ICoDSE), 2016 International Conference on*, pages 1–6.

[Baker et al., 1998] Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley FrameNet Project. In *Proceedings of the 17th international conference on Computational linguistics*, pages 86–90.

[Bang et al., 2015] Bang, J., Noh, H., Kim, Y., and Lee, G. G. (2015). Example-based chat-oriented dialogue system with personalized long-term memory. *2015 International Conference on Big Data and Smart Computing, BIGCOMP 2015*, pages 238–243.

[Basili et al., 1998] Basili, R., Pazienza, M. T., and Zanzotto, F. M. (1998). Efficient Parsing for Information Extraction. In *13th European Conference on Artificial Intelligence (ECAI98)*.

[Baudiš and Šedivý, 2015] Baudiš, P. and Šedivý, J. (2015). Modeling of the question answering task in the YodaQA system. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9283:222–228.

[Beaumont et al., 2015] Beaumont, R., Grau, B., and Ligozat, A.-L. (2015). Sem-graphqa@ qald5: Limsi participation at qald5@ clef. In *CLEF (Working Notes)*.

[Bellman, 1957] Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press.

[Belyaev et al., 2017] Belyaev, S. A., Kuleshov, A. S., and Kholod, I. I. (2017). Solution of the answer formation problem in the question-answering system in Russian. In *Proceedings of the 2017 IEEE Russia Section Young Researchers in Electrical and Electronic Engineering Conference, ElConRus 2017*, pages 360–365.

[Berant and Liang, 2014] Berant, J. and Liang, P. (2014). Semantic Parsing via Paraphrasing. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425.

[Bernstein et al., 2006] Bernstein, A., Kaufmann, E., Kaiser, C., and Kiefer, C. (2006). Ginseng: A guided input natural language search engine for querying ontologies. *2006 Jena User Conference*, (May):2–4.

[Bickmore et al., 2016] Bickmore, T. W., Utami, D., Matsuyama, R., and Paasche-Orlow, M. K. (2016). Improving access to online health information with conversational agents: a randomized controlled experiment. *Journal of medical Internet research*, 18(1).

[Bird and Loper, 2004] Bird, S. and Loper, E. (2004). NLTK : The natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions.*, pages 31–34.

[Blackburn and Bos, 2005] Blackburn, P. and Bos, J. (2005). Representation and inference for natural language. *A first course in computational semantics. CSLI*.

[Bodoff and Raban, 2016] Bodoff, D. and Raban, D. (2016). Question Types and Intermediary Elicitations. *Journal of the Association for Information Science and Technology*, 67(2):289–304.

[Bojanowski et al., 2016] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching Word Vectors with Subword Information. Technical report.

[Bouziane et al., 2015] Bouziane, A., Bouchiha, D., Doumi, N., and Malki, M. (2015). Question Answering Systems: Survey and Trends. *Procedia Computer Science*, 73(Awict):366–375.

[Braslavski et al., 2017] Braslavski, P., Savenkov, D., Agichtein, E., and Dubatovka, A. (2017). What Do You Mean Exactly? *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval - CHIIR '17*, pages 345–348.

[Braun et al., 2017] Braun, D., Hernandez-Mendez, A., Matthes, F., and Langen, M. (2017). Evaluating Natural Language Understanding Services for Conversational Question Answering Systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, number August, pages 174–185.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[Bridge, 2002] Bridge, D. G. (2002). Towards Conversational Recommender Systems: A Dialogue Grammar Approach. *ECCBR Workshops*, pages 9–22.

[Briscoe and Carroll, 2002] Briscoe, T. and Carroll, J. (2002). Robust Accurate Statistical Annotation of General Text. *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)*, pages 1499–1504.

[Burke et al., 1997] Burke, R. D., Hammond, K. J., Kulyukin, V. a., Lytinen, S. L., Tomuro, N., Schoenberg, S., and Burke, R. D. (1997). Question answering from frequently-asked question files: Experiences with the FAQ finder system. *AI Magazine*, 18(2):57–66.

[Cabrio et al., 2012] Cabrio, E., Cojan, J., Aprosio, A. P., Magnini, B., Lavelli, A., and Gandon, F. (2012). QAKiS: An open domain QA system based on relational patterns. In *CEUR Workshop Proceedings*, volume 914, pages 9–12.

[Carlson et al., 1999] Carlson, A., Cumby, C., Rosen, J., and Roth, D. (1999). The SNoW Learning Architecture. page 24.

[Carvalho et al., 2017] Carvalho, D. S., Nguyen, M. T., Tran, C. X., and Nguyen, M. L. (2017). Lexical-morphological modeling for legal text analysis. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10091 LNCS:295–311.

[Chandurkar and Bansal, 2017] Chandurkar, A. and Bansal, A. (2017). Information Retrieval from a Structured KnowledgeBase. In *Proceedings - IEEE 11th International Conference on Semantic Computing, ICSC 2017*, pages 407–412.

[Chiu and Nichols, 2015] Chiu, J. P. C. and Nichols, E. (2015). Named Entity Recognition with Bidirectional LSTM-CNNs. Technical report.

[Choi and Palmer, 2011] Choi, J. D. and Palmer, M. (2011). Getting the Most out of Transition-based Dependency Parsing. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL '11): shortpapers*, 2:687–692.

[Chung et al., 2017] Chung, H., Iorga, M., Voas, J., and Lee, S. (2017). Alexa, Can I Trust You? *Computer*, 50(9):100–104.

[Cimiano et al., 2011] Cimiano, P., Buitelaar, P., McCrae, J., and Sintek, M. (2011). LexInfo: A declarative model for the lexicon-ontology interface. *Journal of Web Semantics*, 9(1):29–51.

[Cimiano et al., 2007] Cimiano, P., Haase, P., and Heizmann, J. (2007). Porting natural language interfaces between domains. *Proceedings of the 12th international conference on Intelligent user interfaces - IUI '07*, pages 180–189.

[Collobert et al., 2011] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

[Cowan et al., 2017] Cowan, B. R., Pantidi, N., Coyle, D., Morrissey, K., Clarke, P., Al-Shehri, S., Earley, D., and Bandeira, N. (2017). "What can i help you with?": Infrequent Users' Experiences of Intelligent Personal Assistants. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services - MobileHCI '17*, pages 1–12.

[Cristianini and Shawe-Taylor, 2000] Cristianini, N. and Shawe-Taylor, J. (2000). An introduction to support vector machines.

[Cunningham et al., 2001] Cunningham, H., Maynard, D., Bontcheva, K., and Tablan, V. (2001). GATE: an Architecture for Development of Robust HLT Applications. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, (July):168–175.

[Cunningham et al., 2000] Cunningham, H., Maynard, D., and Tablan, V. (2000). JAPE: a Java Annotation Patterns Engine. *2000*, page Technical Report.

[Dale, 2016] Dale, R. (2016). The return of the chatbots. *Natural Language Engineering*, 22(05):811–817.

[Damiano et al., 2017] Damiano, E., Spinelli, R., Esposito, M., and Pietro, G. D. (2017). Towards a Framework for Closed-Domain Question Answering in Italian. *Proceedings - 12th International Conference on Signal Image Technology and Internet-Based Systems, SITIS 2016*, pages 604–611.

[Damljanovic et al., 2010] Damljanovic, D., Agatonovic, M., and Cunningham, H. (2010). Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. *Lecture Notes in Computer Science (in-*

cluding subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6088 LNCS(PART 1):106–120.

[De Carolis et al., 2017] De Carolis, B., de Gemmis, M., Lops, P., and Palestra, G. (2017). Recognizing users feedback from non-verbal communicative acts in conversational recommender systems. *Pattern Recognition Letters*, 99:87–95.

[de Marneffe et al., 2014] de Marneffe, M.-C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., and Manning, C. D. (2014). Universal Stanford Dependencies: A cross-linguistic typology. *In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4585–4592.

[De Marneffe and Manning, 2008] De Marneffe, M.-C. and Manning, C. D. (2008). Stanford typed dependencies manual. Technical report.

[Deerwester et al., 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

[Dewey, 2016] Dewey, C. (2016). Meet Tay, the creepy-realistic robot who talks just like a teen.

[Diefenbach et al., 2017] Diefenbach, D., Lopez, V., Singh, K., and Maret, P. (2017). Core techniques of question answering systems over knowledge bases: a survey. *Knowledge and Information Systems*, pages 1–41.

[Dima, 2013] Dima, C. (2013). Intui2: A prototype system for question answering over linked data. In *CLEF (Working Notes)*.

[Dima, 2014] Dima, C. (2014). Answering natural language questions with Intui3. *CEUR Workshop Proceedings*, 1180:1201–1211.

[Dimitrov, 2002] Dimitrov, M. (2002). *A Light-weight Approach to Coreference Resolution for Named Entities in Text*. PhD thesis.

[Fader et al., 2011] Fader, A., Soderland, S., and Etzioni, O. (2011). Identifying relations for open information extraction. *Proceedings of the Conference on . . .*, pages 1535–1545.

[Fader et al., 2013] Fader, A., Zettlemoyer, L., and Etzioni, O. (2013). Paraphrase-Driven Learning for Open Question Answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1608–1618.

[Fadhil and Villafiorita, 2017] Fadhil, A. and Villafiorita, A. (2017). An Adaptive Learning with Gamification & Conversational UIs. *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization - UMAP '17*, pages 408–412.

[Fano and Hawkins, 1961] Fano, R. and Hawkins, D. (1961). Transmission of information: A statistical theory of communications. *American Journal of Physics*, 29:793–794.

[Ferrara et al., 2016] Ferrara, E., Varol, O., Davis, C., Menczer, F., and Flammini, A. (2016). The Rise of Social Bots. *Communications of the ACM*, 59(7):96–104.

[Ferret et al., 2002] Ferret, O., Grau, B., Hurault-Plantet, M., Illouz, G., Monceaux, L., Robba, I., and Vilnat, A. (2002). Finding an answer based on the recognition of the question focus. *NIST Special Publication*, pages 362–370.

[Ferrucci et al., 2010] Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J., Schlaefer, N., and Welty, C. (2010). Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31(3):59–79.

[Figueroa, 2017] Figueroa, A. (2017). Automatically generating effective search queries directly from community question-answering questions for finding related questions. *Expert Systems with Applications*, 77:11–19.

[Finkel et al., 2005] Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics - ACL '05*, (1995):363–370.

[Følstad and Brandtzæg, 2017] Følstad, A. and Brandtzæg, P. B. (2017). Chatbots and the new world of HCI. *Interactions*, 24(4):38–42.

[Fonte et al., 2016] Fonte, F. A., Nistal, M. L., Rial, J. C., and Rodriguez, M. C. (2016). NLAST: A natural language assistant for students. *IEEE Global Engineering Education Conference, EDUCON*, 10-13-Apri(April):709–713.

[Frank et al., 2007] Frank, A., Krieger, H. U., Xu, F., Uszkoreit, H., Crysmann, B., Jörg, B., and Schäfer, U. (2007). Question answering from structured knowledge sources. *Journal of Applied Logic*, 5(1):20–48.

[Freitas and Curry, 2014] Freitas, A. and Curry, E. (2014). Natural Language Queries over Heterogeneous Linked Data Graphs: A Distributional-Compositional Semantics Approach. *Proceedings of the 19th international conference on Intelligent User Interfaces - IUI '14*, pages 279–288.

[Freitas et al., 2011] Freitas, A., Oliveira, J. G., O'Riain, S., Curry, E., and Pereira Da Silva, J. C. (2011). Treo: Best-effort natural language queries over linked data. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6716 LNCS:286–289.

[Frost et al., 2014] Frost, R. A., Donais, J., Mathews, E., Agboola, W., and Stewart, R. (2014). A demonstration of a natural language query interface to an event-based semantic web triplestore. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8798:343–348.

[Gabrilovich and Markovitch, 2007] Gabrilovich, E. and Markovitch, S. (2007). Computing semantic relatedness using wikipedia-based explicit semantic analysis. *IJCAI International Joint Conference on Artificial Intelligence*, pages 1606–1611.

[Gallagher and Zadrozny, 2016] Gallagher, S. and Zadrozny, W. (2016). Leveraging Large Corpora using Internet Search for Question Answering. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence, Proceedings*, pages 532–535.

[Genc and O'Sullivan, 2017] Genc, B. and O'Sullivan, B. (2017). Improving navigation in critique graphs. *Proceedings - 2016 IEEE 28th International Conference on Tools with Artificial Intelligence, ICTAI 2016*, pages 134–141.

[Gerber and Ngomo, 2011] Gerber, D. and Ngomo, A.-C. N. (2011). Bootstrapping the linked data web. In *1st Workshop on Web Scale Knowledge Extraction@ ISWC*, volume 2011.

[Giannone et al., 2013] Giannone, C., Bellomaria, V., and Basili, R. (2013). A HMM-based approach to question answering against linked data. *CEUR Workshop Proceedings*, 1179.

[Gianvecchio et al., 2011] Gianvecchio, S., Xie, M., Wu, Z., and Wang, H. (2011). Humans and bots in internet chat: Measurement, analysis, and automated classification. *IEEE/ACM Transactions on Networking*, 19(5):1557–1571.

[Graf et al., 2015] Graf, B., Krüger, M., Müller, F., Ruhland, A., and Zech, A. (2015). Nombot – Simplify Food Tracking. In *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia - MUM '15*, pages 360–363.

[Graus et al., 2016] Graus, D., Bennett, P. N., White, R. W., and Horvitz, E. (2016). Analyzing and Predicting Task Reminders. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization - UMAP '16*, pages 7–15.

[Green Jr. et al., 1961] Green Jr., B. F., Wolf, A. K., Chomsky, C., and Laughery, K. (1961). Baseball: an automatic question-answerer. In *AFIPS Joint Computer Conferences*, pages 219–224.

[Gunaratna et al., 2017] Gunaratna, K., Yazdavar, A. H., Thirunarayan, K., Sheth, A., and Cheng, G. (2017). Relatedness-based multi-entity summarization. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 1060–1066.

[Hakimov et al., 2015] Hakimov, S., Unger, C., Walter, S., and Cimiano, P. (2015). Applying Semantic Parsing to Question Answering Over Linked Data: Addressing the Lexical Gap. In *Proceedings of the International Conference on Applications of Natural Language to Information Systems 2015*, pages 103–109.

[Hamon et al., 2014] Hamon, T., Grabar, N., Mougin, F., and Thiessard, F. (2014). Description of the POMELO system for the task 2 of QALD-4. *CEUR Workshop Proceedings*, 1180:1212–1223.

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

[Harabagiu et al., 2000] Harabagiu, S., Moldovan, D., and Pasca, M. (2000). Falcon: Boosting knowledge for answer engines. In *Proceedings of the Ninth Text REtrieval Conference (TREC 2000)*, volume 1, pages 479–488.

[Hauswald et al., 2016] Hauswald, J., Laurenzano, M. A., Zhang, Y., Li, C., Rovinski, A., Khurana, A., Dreslinski, R. G., Mudge, T., Petrucci, V., Tang, L., and Mars, J. (2016). Sirius Implications for Future Warehouse-Scale Computers. *IEEE Micro*, 36(3):42–53.

[He et al., 2014] He, S., Zhang, Y., Liu, K., and Zhao, J. (2014). CASIA@V2: A MLN-based question answering system over Linked Data. *CEUR Workshop Proceedings*, 1180(61272332):1249–1259.

[Hearst, 2011] Hearst, M. A. (2011). 'Natural' search user interfaces. *Communications of the ACM*, 54(11):60–67.

[Hill et al., 2015] Hill, J., Randolph Ford, W., and Farreras, I. G. (2015). Real conversations with artificial intelligence: A comparison between human-human online conversations and human-chatbot conversations. *Computers in Human Behavior*, 49:245–250.

[Hirzel et al., 2017] Hirzel, M., Mandel, L., Shinnar, A., Siméon, J., and Vaziri, M. (2017). I Can Parse You : Grammars for Dialogs. In *LIPIcs-Leibniz International Proceedings in Informatics, vol. 71. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik*, number 71, pages 1–15.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1–32.

[Höffner and Lehmann, 2014] Höffner, K. and Lehmann, J. (2014). Towards question answering on statistical linked data. *Proceedings of the 10th International Conference on Semantic Systems - SEM '14*, pages 61–64.

[Höffner and Lehmann, 2017] Höffner, K. and Lehmann, J. (2017). Survey on Challenges of Question Answering in the Semantic Web. *Semantic Web*, 8(6):895–920.

[Hoque and Quaresma, 2015] Hoque, M. M. and Quaresma, P. (2015). Semontoqa. In *Proceedings of the Forum for Information Retrieval Evaluation on - FIRE '14*, pages 10–20.

[Hoque and Quaresma, 2017] Hoque, M. M. and Quaresma, P. (2017). A content-aware hybrid architecture for answering questions from open-domain texts. *19th International Conference on Computer and Information Technology, ICCIT 2016*, pages 293–298.

[Intelligence, 2016] Intelligence, B. (2016). Messaging apps are now bigger than social networks. *https://www.businessinsider.de/the-messaging-app-report-2015-11?r=US&IR=T, visited 2018-07-07.*

[Jaya Kumar et al., 2017] Jaya Kumar, A., Schmidt, C., and Köhler, J. (2017). A knowledge graph based speech interface for question answering systems. *Speech Communication*, 92:1–12.

[Jijkoun and de Rijke, 2007] Jijkoun, V. and de Rijke, M. (2007). The task first, please. In *Proceedings of the SIGIR 2007 Workshop on Focused Retrieval*, pages 23–27.

[Joho et al., 2017] Joho, H., Cavedon, L., Arguello, J., Shokouhi, M., and Radlinski, F. (2017). First International Workshop on Conversational Approaches to Information Retrieval (CAIR'17). *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '17*, pages 1423–1424.

[Juang et al., 2015] Juang, J., Li, Q., Xue, Y., Cheng, T., Xu, S., Jia, J., and Feng, L. (2015). Teenchat: a chatterbot system for sensing and releasing adolescents' stress. In *International Conference on Health Information Science*, pages 133–145.

[Jurafsky and Martin, 2017a] Jurafsky, D. and Martin, J. (2017a). Dialog Systems and Chatbots. In *Speech and Language Processing.*, chapter 28, pages 418–440. 3rd edition.

[Jurafsky and Martin, 2016a] Jurafsky, D. and Martin, J. H. (2016a). Part-of-speech tagging. In *Speech and Language Processing*, chapter 10. 3rd edition.

[Jurafsky and Martin, 2016b] Jurafsky, D. and Martin, J. H. (2016b). Semantics with Dense Vectors. In *Speech and Language Processing, 3rd edition*, volume 3. 3rd edition.

[Jurafsky and Martin, 2017b] Jurafsky, D. and Martin, J. H. (2017b). Dependency Parsing. In *Speech and Language Processing*, chapter 14. 3rd edition.

[Jurafsky and Martin, 2017c] Jurafsky, D. and Martin, J. H. (2017c). Question Answering. In *Speech and Language Processing*, chapter 28, pages 400–417. 3rd edition.

[Jurafsky and Martin, 2017d] Jurafsky, D. and Martin, J. H. (2017d). Regular Expressions, Text Normalization, Edit Distance. In *Speech and Language Processing*, chapter 2, pages 10–33. 3rd edition.

[Jurafsky and Martin, 2018] Jurafsky, D. and Martin, J. H. (2018). *Speech and Language Processing.* 3rd edition.

[Kalyanpur et al., 2012] Kalyanpur, A., Boguraev, B. K., Patwardhan, S., Murdock, J. W., Lally, A., Welty, C. a., Prager, J. M., Coppola, B., Fokoue-Nkoutche, A., Zhang, L., Pan, Y., and Qui, Z. M. (2012). Structured data and inference in DeepQA. *IBM Journal of Research and Development*, 56(3):351–364.

[Kaufmann et al., 2007] Kaufmann, E., Bernstein, A., and Fischer, L. (2007). NLP-Reduce: A "naïve" but Domain-independent Natural Language Interface for Querying Ontologies. *4th European Semantic Web Conference ESWC 2007*, pages 1–2.

[Keerthi and Lin, 2003] Keerthi, S. S. and Lin, C.-J. (2003). Asymptotic behaviors of support vector machines with gaussian kernel. *Neural computation*, 15(7):1667–1689.

[Khvalchik et al., 2017] Khvalchik, M., Pithyaachariyakul, C., and Kulkarni, A. (2017). Answering the Hard Questions. In *International Conference on Language, Data and Knowledge*, pages 253–261.

[Kim et al., 2017] Kim, M. Y., Xu, Y., and Goebel, R. (2017). Applying a convolutional neural network to legal question answering. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10091 LNCS:282–294.

[Kincaid and Pollock, 2017] Kincaid, R. and Pollock, G. (2017). Nicky: Toward a Virtual Assistant for Test and Measurement Instrument Recommendations. *Proceedings - IEEE 11th International Conference on Semantic Computing, ICSC 2017*, pages 196–203.

[Kiseleva et al., 2016] Kiseleva, J., Williams, K., Jiang, J., and Crook, A. C. (2016). Understanding User Satisfaction with Intelligent Assistants. In *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*, volume 1, pages 121–130.

[Klopfenstein et al., 2017] Klopfenstein, L. C., Delpriori, S., Malatini, S., and Bogliolo, A. (2017). The Rise of Bots: A Survey of Conversational Interfaces, Patterns, and Paradigms. *Proceedings of the 2017 Conference on Designing Interactive Systems - DIS '17*, pages 555–565.

[Knox et al., 2011] Knox, C., Law, V., Jewison, T., Liu, P., Ly, S., Frolkis, A., Pon, A., Banco, K., Mak, C., Neveu, V., Djoumbou, Y., Eisner, R., Guo, A. C., and Wishart, D. S. (2011). DrugBank 3.0: A comprehensive resource for 'Omics' research on drugs. *Nucleic Acids Research*, 39(SUPPL. 1):1035–1041.

[Koehn, 2005] Koehn, P. (2005). Europarl : A Parallel Corpus for Statistical Machine Translation. *MT Summit*, 11:79–86.

[Kolomiyets and Moens, 2011] Kolomiyets, O. and Moens, M. F. (2011). A survey on question answering technology from an information retrieval perspective. *Information Sciences*, 181(24):5412–5434.

[Konstantinova and Orasan, 2013] Konstantinova, N. and Orasan, C. (2013). Interactive Question Answering. *Emerging Applications of Natural Language Processing: Concepts and New Research*, (October):149 —- 169.

[Koo et al., 2008] Koo, T., Carreras Pérez, X., and Collins, M. (2008). Simple semi-supervised dependency parsing. *46th Annual Meeting of the Association for Computational Linguistics*, (June):595–603.

[Kumar and Joshi, 2017] Kumar, V. and Joshi, S. (2017). Incomplete Follow-up Question Resolution using Retrieval based Sequence to Sequence Learning. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '17*, pages 705–714.

[Kwok et al., 2001] Kwok, C., Etzioni, O., and Weld, D. S. (2001). Scaling question answering to the web. *ACM Transactions on Information Systems*, 19(3):242–262.

[Lafferty et al., 2001] Lafferty, J., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML '01 Proceedings of the Eighteenth International Conference on Machine Learning*, 8(June):282–289.

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

[Le and Mikolov, 2014] Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

[Li et al., 2016] Li, H., Min, M. R., Ge, Y., and Kadav, A. (2016). A Context-aware Attention Network for Interactive Question Answering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 927–935.

[Li et al., 2017] Li, J., Liu, H., Zhang, Y., and Xing, C. (2017). A Health QA with Enhanced User Interfaces. In *Proceedings - 13th Web Information Systems and Applications Conference, WISA 2016 - In conjunction with 1st Symposium on Big Data Processing and Analysis, BDPA 2016 and 1st Workshop on Information System Security, ISS 2016*, pages 173–178.

[Li and Roth, 2002] Li, X. and Roth, D. (2002). Learning Question Classifiers. *COLING '02 Proceedings of the 19th international conference on Computational linguistics*, pages 1–7.

[Li and Roth, 2006] Li, X. and Roth, D. (2006). Learning question classifiers: The role of semantic information. *Natural Language Engineering*, 12(3):229–249.

[Lin, 2003] Lin, D. (2003). Dependency-Based Evaluation of Minipar. *Treebanks - Building and Using Parsed Corpora*, pages 317–329.

[Linckels and Meinel, 2005] Linckels, S. and Meinel, C. (2005). a Simple Solution for an Intelligent Librarian System. In *IADIS AC*, pages 495–503.

[Litkowski, 2001] Litkowski, K. C. (2001). Syntactic clues and lexical resources in question-answering. *NIST SPECIAL PUBLICATION SP*, (249):157–166.

[Liu et al., 2016] Liu, Y., Yi, X., Chen, R., and Song, Y. (2016). A Survey on Frameworks and Methods of Question Answering. In *Proceedings - 2016 3rd International Conference on Information Science and Control Engineering, ICISCE 2016*, pages 115–119.

[Lopez et al., 2012] Lopez, V., Fernández, M., Motta, E., and Stieler, N. (2012). PowerAqua: Supporting users in querying and exploring the Semantic Web. *Semantic Web*, 3(3):249–265.

[Lopez et al., 2013] Lopez, V., Unger, C., Cimiano, P., and Motta, E. (2013). Evaluating Question Answering over Linked Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 21:3–13.

[Lopez et al., 2007] Lopez, V., Uren, V., Motta, E., and Pasin, M. (2007). AquaLog: An ontology-driven question answering system for organizational semantic intranets. *Web Semantics*, 5(2):72–105.

[Madhu et al., 2017] Madhu, D., Jain, C. J., Sebastain, E., Shaji, S., and Ajayakumar, A. (2017). A novel approach for medical assistance using trained chatbot. *Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2017*, (Icicct):243–246.

[Mahmood and Ricci, 2007] Mahmood, T. and Ricci, F. (2007). Towards learning user-adaptive state models in a conversational recommender system. *Proceedings of the 15th Workshop on Adaptivity and User Modeling in Interactive Systems, ABIS*, 7:373–378.

[Manning et al., 2014] Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

[Marcus et al., 1993] Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

[Marginean, 2017] Marginean, A. (2017). Question answering over biomedical linked data with Grammatical Framework. *Semantic Web*, 8(4):565–580.

[Maron and Kuhns, 1960] Maron, M. E. and Kuhns, J. L. (1960). On relevance, probabilistic indexing, and information retrieval. *Journal of the Association for Computing Machinery*, 7:216–244.

[Mazur et al., 2012] Mazur, M., Rzepka, R., and Araki, K. (2012). Chatterbots with occupation - Between non task and task oriented conversational agents. *AISB/IACAP World Congress 2012: Linguistic and Cognitive Approaches to Dialogue Agents, Part of Alan Turing Year 2012*, pages 61–66.

[McCarthy et al., 2004] McCarthy, K., Reilly, J., McGinty, L., and Smyth, B. (2004). On the Dynamic Generation of Compound Critiques in Conversational Recommender Systems. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, volume 3137, pages 176–184.

[McGinty and Smyth, 2003] McGinty, L. and Smyth, B. (2003). On the role of diversity in conversational recommender systems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2689:276–290.

[Mcginty and Smyth, 2003] Mcginty, L. and Smyth, B. (2003). Tweaking Critiquing. In *Proceedings of the Workshop on Personalization and Web Techniques at the International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 20–27.

[Mendes et al., 2011] Mendes, P. N., Jakob, M., García-Silva, A., and Bizer, C. (2011). DBpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems - I-Semantics '11*, volume 95, pages 1–8.

[Mikolov et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, pages 3111–3119.

[Miller et al., 1990] Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. J. (1990). Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244.

[Mintz et al., 2009] Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. *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*, 2(2005):1003.

[Mishra and Jain, 2016] Mishra, A. and Jain, S. K. (2016). A survey on question answering systems with classification. *Journal of King Saud University - Computer and Information Sciences*, 28(3):345–361.

[Moldovan et al., 1999] Moldovan, D., Harabagiu, S., Pasca, M., Mihalcea, R., Goodrum, R., Girju, R., and Rus, V. (1999). Lasso: A tool for surfing the answer net. In *TREC*, volume 8, pages 65–73.

[Moore et al., 2017] Moore, R. J., Arar, R., Ren, G.-J., and Szymanski, M. H. (2017). Conversational UX Design. *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA '17*, pages 492–497.

[Moro et al., 2014] Moro, A., Raganato, A., and Navigli, R. (2014). Entity Linking meets Word Sense Disambiguation: a Unified Approach. *Transactions of the Association for Computational Linguistics (TACL)*, 2(0):231–244.

[Nakashole et al., 2012] Nakashole, N., Weikum, G., and Suchanek, F. M. (2012). PATTY: A Taxonomy of Relational Patterns with Semantic Types. *EMNLP-CoNLL*, (July):1135–1145.

[Nam et al., 2017] Nam, S., Choi, G., and Choi, K.-s. (2017). SRDF: A Novel Lexical Knowledge Graph for Whole Sentence Knowledge Extraction. In *First International Conference, LDK 2017*, volume 10318, pages 315–329.

[Neidhardt et al., 2015] Neidhardt, J., Seyfang, L., Schuster, R., and Werthner, H. (2015). A picture-based approach to recommender systems. *Information Technology and Tourism*, 15(1):49–69.

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

[Nyberg et al., 2005] Nyberg, E., Frederking, R., Mitamura, T., Bilotti, M., Hannan, K., Hiyakumoto, L., Ko, J., Lin, F., Lita, L., Pedro, V., and Schlaikjer, A. (2005). JAVELIN I and II Systems at TREC 2005 JAVELIN I : Main Track Run. *Analysis*, pages 1–12.

[Oh et al., 2017] Oh, J.-H., Torisawa, K., Kruengkrai, C., Iida, R., and Kloetzer, J. (2017). Multi-column convolutional neural networks with causality-attention for why-question answering. *WSDM 2017 - Proceedings of the 10th ACM International Conference on Web Search and Data Mining*, pages 415–424.

[Palmer et al., 2005] Palmer, M., Gildea, D., and Kingsbury, P. (2005). The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

[Papaioannou and Lemon, 2017] Papaioannou, I. and Lemon, O. (2017). Combining Chat and Task-Based Multimodal Dialogue for More Engaging HRI. *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction - HRI '17*, pages 365–366.

[Park et al., 2014] Park, S., Shim, H., and Lee, G. G. (2014). ISOFT at QALD-4: Semantic similarity-based question answering system over linked data. *CEUR Workshop Proceedings*, 1180:1236–1248.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

[Porcheron et al., 2017] Porcheron, M., Fischer, J. E., McGregor, M., Brown, B., Luger, E., Candello, H., and O'Hara, K. (2017). Talking with Conversational Agents in Collaborative Action. *Companion of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing - CSCW '17 Companion*, pages 431–436.

[Porter, 1980] Porter, M. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137.

[Pradel et al., 2012] Pradel, C., Haemmerlé, O., and Hernandez, N. (2012). A semantic web interface using patterns: the SWIP system. *Graph Structures for Knowledge ...*, pages 172–187.

[Pradhan et al., 2004] Pradhan, S., Ward, W., Hacioglu, K., Martin, J., and Jurafsky, D. (2004). Shallow Semantic Parsing using Support Vector Machines. *Proceedings of the Human Language Technology Conference/North American chapter of the Association for Computational Linguistics annual meeting (HLT/NAACL-2004)*.

[Qiu et al., 2007] Qiu, X., Li, B., Shen, C., Wu, L., Huang, X., and Zhou, Y. (2007). FDUQA on TREC2007 QA Track.

[Quasthoff et al., 2006] Quasthoff, U., Richter, M., and Biemann, C. (2006). Corpus portal for search in monolingual corpora. *Proceedings of the fifth international conference on language resources and evaluation*, pages 1799–1802.

[Quinlan, 1993] Quinlan, J. (1993). C4. 5, programs for machine learning. In *In Proc. of 10th International Conference on Machine Learning*, pages 252–259.

[Radlinski and Craswell, 2017] Radlinski, F. and Craswell, N. (2017). A Theoretical Framework for Conversational Search. *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval - CHIIR '17*, pages 117–126.

[Ramshaw and Marcus, 1995] Ramshaw, L. A. and Marcus, M. P. (1995). Text Chunking using Transformation-Based Learning. *Natural language processing using very large corpora*, pages 1–13.

[Ranta, 2004] Ranta, A. (2004). Grammatical Framework: A Type-Theoretical Grammar Formalism. *The Journal of Functional Programming*, 14(2):145–189.

[Razzaghnoori et al., 2018] Razzaghnoori, M., Sajedi, H., and Jazani, I. K. (2018). Question classification in Persian using word vectors and frequencies. *Cognitive Systems Research*, 47:16–27.

[Reilly and Reilly, 2004] Reilly, J. and Reilly, J. (2004). Thinking Positively-Explanatory Feedback for Conversational Recommender Systems. *Proceedings of the European Conference on Case-Based Reasoning (ECCBR-04) Explanation Workshop*, pages 115–124.

[Romeo et al., 2017] Romeo, S., da San Martino, G., Barrón-Cedeño, A., and Moschitti, A. (2017). A multiple-instance learning approach to sentence selection for question ranking. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10193 LNCS:437–449.

[Rosenberg, 2016] Rosenberg, S. (2016). How To Build Bots for Messenger. *https://developers.facebook.com/blog/post/2016/04/12/bots-for-messenger/, visited 2018-06-08*.

[Ruan et al., 2017] Ruan, H., Li, Y., Wang, Q., and Liu, Y. (2017). A Research on Sentence Similarity for Question Answering System Based on Multi-feature Fusion. *Proceedings - 2016 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2016*, pages 507–510.

[Rubin et al., 2010] Rubin, V. L., Chen, Y., and Thorimbert, L. M. (2010). Artificially intelligent conversational agents in libraries. *Library Hi Tech*, 28(4):496–522.

[Ruseti et al., 2015] Ruseti, S., Mirea, A., Rebedea, T., and Trausan-Matu, S. (2015). Qanswer-enhanced entity matching for question answering over linked data. In *CLEF (Working Notes)*.

[Saany et al., 2017] Saany, S. I. A., Mamat, A., Mustapha, A., Affendey, L. S., and Rahman, M. N. A. (2017). Syntax and Semantics Question Analysis Using User Modelling and Relevance Feedback. *International Journal on Advance Science Engineering Information Technology*, 7(1):329–337.

[Sammut and Webb, 2010] Sammut, C. and Webb, G. I., editors (2010). *F1-Measure*, pages 397–397. Springer US, Boston, MA.

[Sansonnet et al., 2006] Sansonnet, J.-P., Leray, D., and Martin, J.-c. (2006). Architecture of a Framework for Generic Assisting Conversational Agents. In *International Workshop on Intelligent Virtual Agents*, number January 2014, pages 145–156.

[Schmid, 1994] Schmid, H. (1994). Probabilistic part-o-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 1–9.

[Schwarzer et al., 2016] Schwarzer, M., Düver, J., Ploch, D., and Lommatzsch, A. (2016). An interactive e-government question answering system. In *CEUR Workshop Proceedings*, volume 1670, pages 74–82.

[Sebastiani, 2002] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

[Setiaji and Wibowo, 2017] Setiaji, B. and Wibowo, F. W. (2017). Chatbot Using a Knowledge in Database: Human-to-Machine Conversation Modeling. *Proceedings - International Conference on Intelligent Systems, Modelling and Simulation, ISMS*, pages 72–77.

[Shekarpour et al., 2015] Shekarpour, S., Marx, E., Ngonga Ngomo, A. C., and Auer, S. (2015). SINA: Semantic interpretation of user queries for question answering on interlinked data. *Journal of Web Semantics*, 30:39–51.

[Shiga et al., 2017] Shiga, S., Joho, H., Blanco, R., Trippas, J. R., and Sanderson, M. (2017). Modelling Information Needs in Collaborative Search Conversations. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '17*, pages 715–724.

[Shimazu, 2002] Shimazu, H. (2002). ExpertClerk: A conversational case-based reasoning tool for developing salesclerk agents in e-commerce webshops. *Artificial Intelligence Review*, 18(3-4):223–244.

[Song et al., 2017] Song, D., Oh, E. Y., and Rice, M. (2017). Interacting with a Conversational Agent System for Educational Puroposes in Online Courses. In *10th International Conference on Human System Interactions (HSI)*, pages 78–82.

[Song et al., 2015] Song, D., Schilder, F., Smiley, C., Brew, C., Zielund, T., Bretz, H., Martin, R., Dale, C., Duprey, J., Miller, T., and Harrison, J. (2015). TR Discover: A Natural Language Interface for Querying and Analyzing Interlinked Datasets. *Proceedings of the ISWC 2015, Part II*, pages 21–37.

[Speck and Ngomo, 2017] Speck, R. and Ngomo, A.-C. N. (2017). Ensemble Learning of Named Entity Recognition Algorithms using Multilayer Perceptron for the Multilingual Web of Data. *Proceedings of the Knowledge Capture Conference on - K-CAP 2017*, pages 1–4.

[Srihari and Li, 1999] Srihari, R. and Li, W. (1999). Information Extraction Supported Question Answering. *CYMFONY NET INC WILLIAMSVILLE NY*, pages 1–6.

[Srihari and Li, 2000] Srihari, R. and Li, W. (2000). A question answering system supported by information extraction. *Proceedings of the sixth conference on Applied natural language processing* -, pages 166–172.

[Šukys et al., 2017] Šukys, A., Nemuraitė, L., and Butkienė, R. (2017). SBVR based natural language interface to ontologies. *Information Technology and Control*, 46(1):118–137.

[Sun et al., 2015] Sun, H., Ma, H., Yih, W.-t., Tsai, C.-T., Liu, J., and Chang, M.-W. (2015). Open Domain Question Answering via Semantic Enrichment. *Proceedings of the 24th International Conference on World Wide Web - WWW '15*, pages 1045–1055.

[Sun et al., 2017] Sun, Y., Yuan, N. J., Xie, X., McDonald, K., and Zhang, R. (2017). Collaborative Intent Prediction with Real-Time Contextual Data. *ACM Transactions on Information Systems*, 35(4):1–33.

[Theodoridis and Koutroumbas, 2009] Theodoridis, S. and Koutroumbas, K. (2009). *Pattern Recognition*. Academic Press, 4th edition.

[Ting, 2010] Ting, K. M. (2010). *Precision and Recall*, pages 781–781. Springer US, Boston, MA.

[Tintarev et al., 2016] Tintarev, N., O'donovan, J., and Felfernig, A. (2016). Introduction to the Special Issue on Human Interaction with Artificial Advice Givers. *ACM Transactions on Interactive Intelligent Systems*, 6(4):1–12.

[Tjong Kim Sang and De Meulder, 2003] Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

[Trippas et al., 2015] Trippas, J. R., Sanderson, M., Spina, D., and Cavedon, L. (2015). Results Presentation Methods for a Spoken Conversational Search System. *Proceedings of the First International Workshop on Novel Web Search Interfaces and Systems*, pages 2–4.

[Trippas et al., 2017] Trippas, J. R., Spina, D., Cavedon, L., and Sanderson, M. (2017). How Do People Interact in Conversational Speech-Only Search Tasks. *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval - CHIIR '17*, pages 325–328.

[Tsai et al., 2015] Tsai, C.-t., Yih, W.-T., and Burges, C. J. C. (2015). Web-based Question Answering : Revisiting AskMSR. Technical report.

[Unger and Bühmann, 2012] Unger, C. and Bühmann, L. (2012). Template-based question answering over RDF data. *Proceedings of the 21st international conference on World Wide Web*, pages 639–648.

[Unger and Cimiano, 2011] Unger, C. and Cimiano, P. (2011). Pythia: Compositional meaning construction for ontology-based question answering on the semantic web. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6716 LNCS:153–160.

[Usbeck et al., 2015] Usbeck, R., Ngomo, A. C. N., Bühmann, L., and Unger, C. (2015). HAWK - hybrid question answering using linked data. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9088:353–368.

[van Weijen, 2012] van Weijen, D. (2012). The Language of (Future) Scientific Communication. *Research Trends*, (31).

[Vtyurina et al., 2017] Vtyurina, A., Savenkov, D., Agichtein, E., and Clarke, C. L. A. (2017). Exploring Conversational Search With Humans, Assistants, and Wizards. *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 2187–2193.

[Walter et al., 2012] Walter, S., Ung, Cimiano, P., and Bär, D. (2012). Evaluation of a Layered Approach to Question Answering over Linked Data. In *International Semantic Web Conference*, pages 362–374.

[Waltinger et al., 2011] Waltinger, U., Breuing, A., and Wachsmuth, I. (2011). Interfacing virtual agents with collaborative knowledge: Open domain question answering using wikipedia-based topic models. *IJCAI International Joint Conference on Artificial Intelligence*, pages 1896–1902.

[Wang et al., 2007] Wang, C., Xiong, M., Zhou, Q., and Yu, Y. (2007). PANTO: A Portable Natural Language Interface to Ontologies. *The Semantic Web: Research and Applications*, pages 473–487.

[Webber and Webb, 2010] Webber, B. and Webb, N. (2010). Question Answering. In Clark, A., Fox, C., and Lappin, S., editors, *The Handbook of Computational Linguistics and Natural Language Processing*, chapter 22, pages 630–654. John Wiley & Sons, reprint edition.

[Weeratunga et al., 2016] Weeratunga, A. M., Jayawardana, S. A. U., Hasindu, P. M. A. K., Prashan, W. P. M., and Thelijjagoda, S. (2016). Project Nethra - An intelligent assistant for the visually disabled to interact with internet services. *2015 IEEE 10th International Conference on Industrial and Information Systems, ICIIS 2015 - Conference Proceedings*, pages 55–59.

[Wei et al., 2017] Wei, H., Zhang, F., Yuan, N. J., Cao, C., Fu, H., Xie, X., Rui, Y., and Ma, W.-Y. (2017). Beyond the Words: Predicting User Personality from Heterogeneous Information. *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining - WSDM '17*, pages 305–314.

[Wong, 2005] Wong, W. (2005). *Practical Approach to Knowledge-based Question Answering with Natural Language Understanding and Advanced Reasoning*. PhD thesis, KOLEJ UNIVERSITI TEKNIKAL KEBANGSAAN MALAYSIA.

[Wu et al., 2004] Wu, M., Zheng, X., Duan, M., Liu, T., and Strzalkowski, T. (2004). Question Answering By Pattern Matching, Web-Proofing, Semantic Form Proofing. *Proceedings of TREC 2003*, pages 578–585.

[Xie et al., 2017] Xie, Z., Zeng, Z., Zhou, G., and Wang, W. (2017). Topic enhanced deep structured semantic models for knowledge base question answering. *Science China Information Sciences*, 60(11):110–103.

[Xu et al., 2014] Xu, K., Feng, Y., and Zhao, D. (2014). Xser@QALD-4: Answering Natural Language Questions via Phrasal Semantic Parsing. In *Proceedings of the CLEF 2014 Conference*, pages 1260–1274.

[Xu et al., 2017] Xu, Z., Liu, B., Wang, B., Sun, C., and Wang, X. (2017). Incorporating loose-structured knowledge into conversation modeling via recall-gate LSTM. *Proceedings of the International Joint Conference on Neural Networks*, 2017:3506–3513.

[Yahya et al., 2012] Yahya, M., Berberich, K., Elbassuoni, S., Ramanath, M., Tresp, V., and Weikum, G. (2012). Natural language questions for the web of data. *EMNLP -CoNLL '12*, (July):379–390.

[Yang and Pedersen, 1997] Yang, Y. and Pedersen, J. O. (1997). A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of the 14th International Conference on Machine Learning (ICML)*, pages 412–420.

[Yin et al., 2017] Yin, J., Zhao, W. X., and Li, X.-M. (2017). Type-Aware Question Answering over Knowledge Base with Attention-Based Tree-Structured Neural Networks. *Journal of Computer Science and Technology*, 32(4):805–813.

[Yosef et al., 2011] Yosef, M. A., Hoffart, J., Bordino, I., Spaniol, M., and Weikum, G. (2011). AIDA: An Online Tool for Accurate Disambiguation of Named Entities in Text and Tables. *Proceedings of the VLDB Endowment*, 4(12):1450–1453.

[Yue et al., 2017] Yue, C., Cao, H., Xiong, K., Cui, A., Qin, H., and Li, M. (2017). Enhanced question understanding with dynamic memory networks for textual question answering. *Expert Systems with Applications*, 80:39–45.

[Yuksel et al., 2017] Yuksel, B. F., Collisson, P., and Czerwinski, M. (2017). Brains or Beauty. *ACM Transactions on Internet Technology*, 17(1):1–20.

[Zamora, 2017] Zamora, J. (2017). Rise of the Chatbots: Finding a Place for Artificial Intelligence in India and US. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces Companion - IUI '17 Companion*, pages 109–112.

[Zhang and Lee, 2003] Zhang, D. and Lee, W. S. (2003). Question classification using support vector machines. *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 26–32.

[Zhang et al., 2017] Zhang, W., Wang, H., Ren, K., and Song, J. (2017). Chinese sentence based lexical similarity measure for artificial intelligence chatbot. *Proceedings of the 8th International Conference on Electronics, Computers and Artificial Intelligence, ECAI 2016*, pages 1–4.

[Zhang et al., 2016] Zhang, Y., He, S., Liu, K., and Zhao, J. (2016). A Joint Model for Question Answering over Multiple Knowledge Bases. *Proceedings of the 30th Conference on Artificial Intelligence (AAAI 2016)*, pages 3094–3100.

[Zheng and Arbor, 2002] Zheng, Z. and Arbor, A. (2002). AnswerBus Question Answering System. *Proceedings of the second international conference on Human Language Technology Research*, pages 399–404.

[Zhu et al., 2016] Zhu, C., Ren, K., Liu, X., Wang, H., Tian, Y., and Yu, Y. (2016). A graph traversal based approach to answer nonaggregation questions over DBpedia. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9544:219–234.

[Zou et al., 2014] Zou, L., Huang, R., Wang, H., Yu, J. X., He, W., and Zhao, D. (2014). Natural Language Question Answering over RDF — A Graph Data Driven Approach. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 313–324.

Appendix A: Question Answering System Surveys

"A Survey on Frameworks and Methods of Question Answering" ([Liu et al., 2016]) give an overview of different approaches commonly used in QAS development. They group the methods and tools into different groups of application: (i) question classification, (ii) question similarity, (iii) answer extraction, (iv) answer scoring and (v) question translation.

Question classification: the authors state that classic approaches stemming from pregenerated linguistic patterns are not flexible enough to provide comparable results to current machine learning based approaches, which rely on extracting features from an input text and a subsequent classification based upon that features using machine learning algorithms like support vector machines, k-nearest-neighbour or Bayesian classifiers.

Question similarity: if the QAS is based upon FAQs or upon a community answering questions (community based question answering, or CQA), then a common approach is to calculate a similarity measure between a users question and the already answered questions in the data source. If an already answered question can be found with a similarity measure above a certain threshold, the existing answer is presented to the user. The authors present different approaches, based upon the extraction and comparison of (i) syntactic features and (ii) semantic features. Syntactic features include word frequencies, word order, sentence length and dependency structure, semantic features include latent topic distributions, synonymous relations, is-a relations and semantic paths. These features can be evaluated using statistical or machine learning approaches.

Answer extraction: depending on the data source used by the QAS, different strategies are used to extract a fitting answer. The authors compare the tradeoffs between small, closed domain data sources and big, open domain data sources like Wikipedia, with the latter providing higher recall but lower precision, and the former providing good precision but potentially lower recall. Also depending on the data source and the type of question, techniques like text retrieval and information extraction are used to extract either facts or similar questions with connected answers.

Answer scoring: if the QAS identifies more than one potential answer, they are called "answer candidates" and must be scored, with the highest scoring candidate being

presented to the user. The authors list different approaches like ranking based on ordinal similarity measures, relevance of the data source, occurrence frequency of the answer candidate and professionality level of the answer.

Question translation: if the QAS uses a structured data source like a relational database or linked data, the question must be translated into a formal language like SQL or SPARQL. These translated questions are used to query the data source for a fitting answer. The commonly used technology is semantic parsing.

"Survey on Challenges of Question Answering in the Semantic Web" ([Höffner and Lehmann, 2017]) give an overview of the most common challenges faced when implementing a QAS, and the most common ways to confront these challenges. The authors suggest that the implementation of a QAS is a non-trivial task, as it combines and mixes "NL techniques on top of traditional IR systems". They identify two main stages which semantic question answering system are typically composed of: (i) query analyzer and (ii) retriever. The former "generates or formats the query that will be used to recover the answer at the retrieval stage" [Höffner and Lehmann, 2017]. Most of the challenges identified fall in the field of natural language understanding.

Lack of mature and/or commonly available methods and tools: for some of the techniques applied in the query analysis phase, such as part-of-speech tagging, mature methods and tools exist, while for others, such as answer candidate disambiguation, they are not commonly available or domain-specific. Therefore high quality systems have to implement a lot of the necessary components from scratch.

Lexical gap: the same meanings can be expressed in several ways, using disjunct vocabularies. Common solutions are: query expansion: by using synonyms from a lexical database (e.g. WordNet [Miller et al., 1990]), alternative versions of the original queries are generated and used in processing steps further down the pipeline pattern matching: by using pattern libraries like PATTY [Nakashole et al., 2012] or BOA [Gerber and Ngomo, 2011], common semantic patterns in the original query can be identified and mapped to semantic frames, which can be used to easily identify RDF triples in a knowledge base. string similarity functions: after normalization of the input words (e.g. stemming, special character transliteration), similarity functions can be applied to identify matching entities or relations in a knowledge base. Such similarity functions are Jaro Winkler, largest common substring or Levenshtein Automatons.

inferring the meaning via textual entailment: an existing knowledge base of world knowledge is used as the source for entailment, where synonyms or hyponyms are used to extend the facts from the world KB. An example would be that the word KB holds the fact "birds fly". Using a lexical database, the system can answer the question "can sparrows fly?" by inferring from the lexical relation of "sparrows" being a hyponym of "bird", that sparrows can indeed fly.

Ambiguity: contrary to the problem raised by the lexical gap, where morphologically different phrases or words can have the same meaning, ambiguity is the issue of morpho-

logically similar phrases or words having different meaning, depending on the context. The authors present the following common solutions:

- **Corpus-based methods:** based upon the distributional hypothesis (i.e. that words that share similar contexts also share similar meanings), an existing text-corpus is used to calculate contextual features. This contextual information is used to calculate similarities between phases or words. Methods used to calculate those contextual features are, amongst others, words within a defined window, part-of-speech tags and parse-tree structures. More complex approaches use context from outside the question, such as a user's previous utterances

- **Resource-based methods:** taking advantage of RDF-based data sources, the resources to be disambiguated are compared to candidate concepts from the knowledge base, and are assigned similarity scores. Methods used to identify the most fitting candidates are, amongst many others, hidden Markov models or Markov logical networks

Multilingualism: knowledge is expressed in different languages, which might be different from the language the question is formulated in. Linked data can be queried using language tags, but unstructured data sources are usually available in a single language only. To bridge this language gap, the authors report of two approaches used in question answer systems: multilingual lexical databases: using a database to translate the query keywords into the language of the data source, e.g. the German language lexical database GermaNet [Hamp and Feldweg, 1997], as part of the multilingual lexical database EuroWordNet provides word- and concept-level mappings to several languages like English, French, Spanish and Italian. wikipedia-based: use existing language mappings from Wikipedia automated translation of query parts: only translates parts of a query and use those to infer the translations of the remaining entities using a knowledge base

Complex queries: questions where multiple facts have to have to be found to correctly answer the question, such as nested questions, or questions with modalities, filters or aggregations often cause problems. Approaches to solve this are, amongst others:

- **Using ontology-specific lexica:** a prebuilt lexicon to resolve quantifiers, comparisons and superlatives focus extraction: the identification of the question focus to predict the lexical answer type. This can be used to narrow down the search space

- **Subtree-matching:** matching subtrees of the questions parse-tree to concepts from the knowledge base translation into logic form: translating the query into an intermediary logical form and translating that into a query in a structured query language

Distributed knowledge: sometimes the facts necessary to answer a question are distributed amongst several different linked knowledge bases. To efficiently query multiple knowledge bases and merge the results, the authors list, amongst others, those approaches:

- **Query decomposition and substitution:** complex queries are decomposed into subqueries using co-reference resolution, aligned using custom ontologies and used to query the separate knowledge bases
- **Result consolidation:** using similarity metrics to rank candidate entities from different knowledge bases and consolidating them to formulate an answer

Procedural, temporal and spatial questions: heavily summarized as "how"-questions (procedural), "when/how long"-questions (temporal) and "where/how far"-questions (spatial), these types of questions require processing beyond fact retrieval from a data source. Some approaches used to answer questions of these types are presented by the authors:

- **Procedural question answering:** using predefined patterns of part-of-speech tags to identify procedural texts from an unstructured data source
- **Temporal question answering:** (i) the application of Allen's Interval Based Temporal Logic [Allen and Hayes, 1989] to answer questions by having an understanding of the concepts "before" and "after", (ii) using the implicit spatial and temporal context of the user to resolve ambiguities
- **Spatial question answering:** (i) using commonly used RDF schema, expressing locations as 2-dimensional geo-coordinates, and modeling relationships on top of them, (ii) enriching of named entity recognition with metadata such as nearness, inclusion or crossing

Templates: to answer complex questions which result in complex SPARQL queries, a question answering system needs to use sophisticated approaches. The authors distinguish two kinds of approaches: (i) template based and (ii) template-free approaches. The former maps the input to either manually or automatically created query templates, whereas the latter try to build the SPARQL queries based on the syntactic structure of the input. The authors present, amongst others, the following approaches to this challenge:

- **Graph pattern templates:** using information about the question type, the named entities and part-of-speech tags present in the question, a graph pattern template is generated and mapped to resources using lexical and pattern databases like WordNet and PATTY, as well as similarity measures. The graph pattern combinations are then converted to SPARQL queries and evaluated against the knowledge base

- **Manually created templates:** for closed, narrow domains, query templates are manually created and questions are mapped onto them
- **Assignment of semantic roles:** assigning semantic labels like “variable”, “entity” or “relation” to words in the query, and creating SPARQL queries from the resulting set of labeled words

“Question Answering Systems: Survey and Trends" ([Bouziane et al., 2015]) give an overview of current trends and existing question answering systems to lay down a foundation to build an Arabic language question answering system of their own. The authors claim that the type of data source is the main influencer on the quality and architecture of a question answering system, and that they “are most effective to interact with structured knowledge bases”. They distinguish between question answering systems for the web of documents and unstructured text on the one side, and question answering systems for the web of data on the other. While the former is a shared issue between information retrieval and natural language processing, the latter is mainly focused on transforming a natural language input into a query in a structured language. The authors provide an extensive list of different question answering systems, grouping them into natural language interfaces to databases (NLIDB) and IR-based systems in one group, and ontology and web-of-data-based systems in the other, giving a very basic introduction to the techniques used by each system. These techniques listed are, amongst others:

- **Question classification:** identifying the type of question defines the way an answer is searched for and presented
- **Intermediate language representation:** convert a natural language input into an intermediate formal representation, which is used to create queries in a structured query language
- **Named entity recognition:** identifying tokens within the text that represent a certain class or concept, e.g. persons, locations or dates
- **Expected answer type:** identifying the lexical type of the answer and mapping it to existing classifications of retrieved named entities
- **Lexical databases:** using the classification information and relations to other words to extend the initial query

In the context of their goal of implementing an Arabic language question answering system, the authors come to the conclusion that maturity of natural language research for a specific language is a primordial factor for the development of a question answering system.

“Core techniques of question answering systems over knowledge bases: a survey" ([Diefenbach et al., 2017]) give a very thorough and detailed overview of

question answering over linked data, especially focusing on question answering systems that participate in the QALD challenges [Lopez et al., 2013]. They describe the datasets that are used in these challenges, like DBPedia [Auer et al., 2007] or Drugbank [Knox et al., 2011], and what metrics are used to rank the participants. An total number of 26 question answering systems participating in a QALD challenge is presented and the techniques used are listed in tabular form. The authors split the question answering process into four distinct tasks: (i) question analysis, (ii) phrase mapping, (iii) disambiguation and (iv) query construction. For each of the analyzed systems the techniques used in each tasks in the question answering process are listed.

Question analysis: the syntactic analysis of the user utterance, this task determines the segmentation of a question, as well as identifies if such segments correspond to a certain subject or object instance and if and how these segments form any kind of dependency structure. The authors list the following key techniques used in this task:

- **Named entity recognition (NER):** it is described the identification of "contiguous spans of tokens that refer to a resource". The authors mention several strategies that are implemented by different question answering systems, these strategies are: (i) using natural language processing NER tools, e.g. the Stanford CoreNLP NER tool [Manning et al., 2014], to identify named entities in the question, (ii) an n-gram strategy, i.e. trying to map n-grams from the question to entities in the underlying knowledge base, or (iii) using entity linking tools, which do not only identify spans of tokens in the question, but also link them to existing entities in the underlying knowledge base, e.g. DBPedia Spotlight [Mendes et al., 2011] or AIDA [Yosef et al., 2011]
- **Part-of-speech tagging:** it is described as the mapping of phrases to subject or object instances, properties or classes, which enables for better mapping to resources inside a knowledge base. The authors mention two strategies to implement POS-tagging, (i) handmade rules, i.e. manually created patterns like regular expressions and predefined question templates (e.g. using the GATE NLP tool [Cunningham et al., 2001]), or (ii) learning rules using machine learning, i.e. using an manually annotated corpus to train a POS tagging model, using annotation like the CoNLL IOB format [Ramshaw and Marcus, 1995] and tools like the Stanford CoreNLP POS tagger
- **Dependency-parsing:** this is the identification of the relations of chunks of a question with each other. Different strategies towards that goal are listed by the authors, like, amongst others, (i) phrase-structure-grammar based strategies, where the parser breaks down a sentence into its constituent parts and chunking them together, (ii) dependency-grammar based strategies, where the parser identifies and classifies the kind of the dependencies of each word. Dependency trees can be used to extract relations by (a) searching for the biggest connecting subtree which can be mapped to a property, or (b) searching for the shortest path between named entities. Stanford CoreNLP provides tools for both these strategies. Additionally,

the authors also mention a strategy merging the former two approaches, identifying dependencies between phrases, resulting in a directed acyclic graph, rather than a tree

Phrase mapping: in this step tries to find sets of resources from the underlying knowledge base which correspond to one or more words in the question with a high probability. The authors present an overview of techniques to provide this functionality:

- **Knowledge base labels:** using the RDF[1] (Resource Description Framework) property "rdfs:label" to find entities in the knowledge base. These labels are human-readable versions of of a resource and can be used to find matches using search engines like Virtuoso[2] or Apache Lucene[3]. While this approach is not without problems, all question answering systems the authors have analyzed are using some form of phrase mapping.

- **String similarity:** in case of misspelled words, string similarity measures can be applied to still find viable labels in the knowledge base. There are many different distance or similarity measures like Levenshtein distance or Jaccard distance, and some search engines like Apache Lucene offer this functionality via fuzzy searches. Stemming is also a possible solution to this issue, allowing different lexeme of the same word being correctly matched

- **Semantic similarity:** also referred to as the "lexical gap", an issue arises when two words share the same meaning, but have different forms, i.e. they only share a semantic relationship. The authors present different approaches to solve this problem, amongst others via (i) lexical databases like WordNet, which store semantic relationships between words, like synonymy, hyponymy and hypernymy, (ii) via redirects in the RDF schema, e.g. "owl:sameAs" links, via (iii) pattern databases like PATTY, which store different forms of the same relations, or via (iv) using big text corpora, e.g. by extracting facts using tools like ReVerb [Fader et al., 2011], TEXTRUNNER or WOE, or by extracting distributional information with word embeddings like word2vec [Mikolov et al., 2013] or Explicit Semantic Analysis (ESA) [Gabrilovich and Markovitch, 2007], where a metric like the cosine similarity can be used to infer semantic similarity

Disambiguation: the authors describe two distinct ambiguity problems that can arise. First, the case that segmentation in the question analysis task could lead to different results, where for example the segmentation and dependency parsing can return different results. The second type describes the case when during the phrase mapping phase several different candidate resources are found in the underlying knowledge base, and the

[1] `https://www.w3.org/RDF/`
[2] `https://github.com/openlink/virtuoso-opensource`
[3] `https://lucene.apache.org/`

question answering system needs to decide which resource is the best fit. The authors list, amongst others, the following approaches to solve these problems:

- **Local disambiguation:** to decide which resource is the best fit, the system needs to rank them. Two features are mainly used for this task: (i) string or semantic similarity of the phrase to the resource label, and (ii) a type consistency check between properties and their attributes. While the first feature is used to rank the results, the second is used to exclude those resources that are not of the correct type. All systems the authors analyzed use this kind of disambiguation
- **Graph Search:** in this approach the graph structure of the underlying knowledge base is used to disambiguate the candidate resources. The authors write about two main strategies: (i) tries to disambiguate the resources under the assumption that a question can be translated into a graph, and that the correct resources from the knowledge base have to form a graph of similar shape, while the second strategy (ii) is only applying phrase mappings to named entities, and searches the underlying knowledge base for all relations and properties that are connected to the candidate mappings, looking for string similarities in the original natural language query. If a connected relation matches a relational phrase from the query, the candidate resource is assumed to be a correct mapping. The authors claim that while he first approach has higher precision, the second has a higher recall, as more potential resources from the underlying knowledge base are used for comparison, and that for both of the strategies performance becomes an issue if ambiguity becomes too high
- **Hidden Markov models (HMM):** this approach assumes that the words of a question resemble an observable stochastic process, while the candidate resources from the underlying knowledge base are the states of a hidden stochastic process, explaining the observations. The correct mappings form the hidden process in such a way, that they produce the most probable explanation of the observations. This can be calculated using the Viterbi algorithm
- **Integer linear programming (ILP):** in this approach the assignment of the correct resources to their corresponding phrases is formulated as an optimization problem, and solved using an integer linear programming solver.
- Markov logical network (MLN): in this approach the assignment of the correct resources is - similar to the ILP approach - formulated in such a way, that a MLN can be trained and applied to form phrases and find fitting resources.
- **Neural nets:** training a perceptron using different features like label similarity and resource popularity, the output of this neural network is maximal for the correct mapping

Query construction: this is the task where the question answering system needs to construct a SPARQL query to retrieve answers from the underlying knowledge base.

The authors describe a problem they call the "semantic gap", which refers to the fact that information in the underlying knowledge base might be encoded differently as could be deduced from the question alone. A question answering system therefore needs to implement a strategy to bridge this gap to be able to give answers in such cases. The authors present several different approaches to solve this:

- **Templates:** this refers to the use of predefined queries with slots to be filled. They can range from fully predefined queries covering a specific question type, or smaller templates, which are combined to generate the final query. These approaches are mostly limited to a small number of linguistic input triples

- **Using information from question analysis:** using the information gathered during segmentation, phrase matching, and disambiguation, intermediary representations as linguistic triples can be constructed from the input, using, amongst other features, (i) the associated resources, (ii) the order of the resources and relations, (iii) part-of-speech tags or (iv) dependency trees. These features are used to deduce SPARQL queries. The authors present many different approaches to achieve this, like, amongst others, (i) creating graphs where the arguments are represented by the vertices, and relations are represented by edges, using this graph to create the SPARQL query, or (ii) very much like in disambiguation, searching the dependency tree for sub-trees corresponding to resources. According to the authors, all of these approaches suffer from the same shortcoming, as they assume that it is possible to construct a working SPARQL query from the structure of the query alone, without knowing how the information is encoded in the underlying knowledge base

- **Semantic parsing:** this refers to a particular kind of parser, providing semantic interpretations of an input sentence. All of the approaches listed by the authors are grammar-based, but differ in the kind of grammar they use, like, amongst others, (i) GF grammars, (ii) context-free grammars, (iii) combinatory categorical grammars or (iv) lexical tree-adjoint grammars. These grammars are used to convert the input into e.g. lambda calculus, from which the SPARQL query is generated. The authors argue that while these approaches are very powerful in the sense that one can immediately generate a query from a given input, they suffer from the limitation that for every item a corresponding semantic representation must be available, which results in a large number of rules and different semantic representations. Question answering systems using this approach try to mitigate this issue by using machine learning approaches to train representations using an annotated corpus, or to use part-of-speech tags to create default-representations for lexical items not covered by specific rules

- **Machine learning:** using the features extracted during the question analysis and phrase mapping phase, machine learning approaches can be used to disambiguate the candidate resources and build valid SPARQL query representations of a question.

- **Semantic information:** using only the semantic information retrieved during phrase mapping, in this approach the system tries to create a valid graph using all mapped classes and resources as vertices, and all mapped relations as edges. All the possible graphs are translated into SPARQL queries and evaluated upon the underlying knowledge base

Querying distributed knowledge: this refers to a scenario where knowledge bases are not already interlinked and might refer to the same entities using different URIs. The question answering system has to identify equivalent resources in the different knowledge bases to able to retrieve connected information which might be available only in a specific knowledge base. According to the authors this is usually solved by retrieving results from all available knowledge bases and comparing the labels, creating an aligned set of URIs which can be used to query for additional information. The authors also note that in these scenarios scalability is a major problem.

The authors conclude that most systems analyzed share many common techniques, and the teams behind the development of those systems usually concentrate on only a few components, leaving out the others. They also make the point that the used techniques can hardly be compared, as it is impossible to fairly judge the performance of a small component in a monolithic pipeline, as its impact on the overall performance is hard to measure and its local performance is highly dependent on the other components in the pipeline, and - as an example - high recall or precision measures are also dependent on the purpose of a component. To provide better comparability, the authors suggest the use of a modular architecture.

"A survey on question answering technology from an information retrieval perspective" ([Kolomiyets and Moens, 2011]) gives an overview of information retrieval techniques commonly used in question answering, as well as a general introduction into the field. The authors provide a short historic overview of the developments in the question answering field in the last 50+ years, as well as a typical architecture of a question answering system. The main methods presented by the authors are:

Bag-of-words representations: with this approach the question is considered a - sometimes preprocessed - set of words, without taking into accord any grammatical features. Multiple ways of using this kind of representation to identify matches are presented: (i) Boolean models, where an exact match is searched for, (ii) algebraic models where the bag-of-words representation is interpreted as a p-dimensional vector, where p is the size of the vocabulary, and similarity measures like cosine similarity can be calculated between two representations, or (iii) probabilistic models, such as the language model. The authors suggest that using lexical databases like WordNet can improve the performance of such approaches.

Morpho-syntactic analysis: this subsummizes a whole group of approaches where the word form and syntactic structure of a sentence are used to gather information about the input. The authors present the following approaches: (i) stemming and lemmatization,

where the former reduces a word to a "stem" which does not change during flexation, while the latter reduces words to its base lexeme, (ii) n-grams to represent the structure of a sentence, (iii) part-of-speech tags to detect the syntactic word class, (iv) chunking, where sentences are split up into base noun- and verb-phrases, and (v) dependency trees, where a sentence is broken up into its constituents and their dependencies are represented in tree form. These approaches can be used to calculate similarities between sentences, e.g. by using syntactic tree kernels or tree edit models.The authors also note that the application of morpho-syntactic analysis approaches has been proven to results in an increased performance, while also increasing computational complexity.

Semantic classification of the expected answer type: in this approach one assumes that the question provides additional information about the type of information it requests. The task of this approach is to identify this type. The authors present several implementations of this, mostly based on taxonomies which predefine the classes an expected answer type (EAT) can be classified as. These taxonomies range from 27 classes in flat hierarchy to taxonomies with 200 different classes and others with multiple hierarchies. The most famous taxonomy for expected answer type classification is the hierarchical taxonomy by Li and Roth [Li and Roth, 2006], with 6 coarse-grained main classes, and 50 fine-grained subclasses. Methods to identify the EAT class are also presented: (i) manually created grammars, or (ii) machine learning methods like tree learners, support vector machines, maximum entropy classifiers and conditional random fields. The authors note that machine learning approaches have become more popular, as hand-crafting of complex grammars is time consuming and requires rule-writing skills, while annotating a training dataset is also time-consuming but requires less qualification. The classification itself can be interpreted in a deterministic or probabilistic way, where the latter uses a distribution over the possible classes. Furthermore, the application of EAT classification has shown to significantly improve question answering performance.

Semantic classification of the constituents: in this approach the constituents of a question are mapped to slots within a semantic frame, a so-called "case", mostly determined by the main verb of the sentence. The authors mention two tools that provide the functionality for using semantic frames, namely FrameNet [Baker et al., 1998] and PropBank [Palmer et al., 2005]. The mappings created using this this technique can be used to constrain the number of potential answer sentences, and the mappings can be used to find matches without the actual lexical instantiations. The authors also suggest the use of a logical retrieval model to infer the answer to a query. The application of semantic frames has shown to increase performance.

Identification of discourse relationships: the information need of the user might not be properly defined by only one question, but my multiple consecutive ones, where the user refines or broadens the initial question. But discourse relationships are not only limited to the user input, but also the data source, where information might be split across several sentences, and it has to be identified in a very similar way. The authors refer to "noun phrase co-reference resolution", where pronouns are replaced by the noun they are referring to. This approach can be extended by not only taking into account equivalence,

but also concepts like hyper- or hyponymy and spatial or temporal references. The authors note that at the time of writing the survey, the field of co reference resolution still needs substantial research to obtain accurate results.

Translation into a Structured Language: this describes the task of the translation of a natural language input into a query in a structured query language, which is used to gather an answer from a structured data source like a relational database. Techniques like (i) semantic role labeling or (ii) sets of symbolic rules and mappings can be used to create such queries.

Translation into and reasoning with a logical representation: in this approach, which is similar to the translation into a structured query language, the input is translated into a logical representation, like, amongst others, (i) first order logic, (ii) meaning representation language (MRL) [Blackburn and Bos, 2005], (iii) lambda calculus or (iv) direct natural language representations. To translate natural language sentences into these representations often integrates semantic role labeling, which relies on (i) handwritten symbolic rules, or (ii) machine learning, and there especially on probabilistic relational learners. A major advantage of such a representation is that if the question as well as the answers are represented using a logical formalism, the relevance of an answer can easily be deduced using current theorem prover models, even if the information is spread across several data sources. The authors note that this kind of approaches have a large potential to increase performance, but at the cost of computational complexity.

The authors conclude that with the advances made from moving from simple bag-of-words models towards complex structured or logical queries obtained from natural language input, the differences between closed-domain systems and open-domain systems become less pronounced.

"A survey on question answering systems with classification ([Mishra and Jain, 2016])": in this survey the authors use criteria deduced from a literature survey to classify existing question answering systems to provide a basis upon which the current successes and future needs towards question answering systems should be identified. These criteria are: (i) application domain, (ii) question type, (iii) applied text analysis, (iv) data source type, (v) data source characteristics, (vi) matching functions and retrieval models, and (vii) generated answer forms. Based upon these criteria, the authors propose the following classification:

Application domain: this classifies a question answering system based on the application domain for which the system can answer questions. The authors propose the distinction into:

- **General domain question answering systems:** such systems are able to answer questions from practically any domain. The authors note that such systems are more suitable for casual users, as they do not require the use of domain-specific vocabulary, nor do they require any prior preparation by the user, however, the answer quality is usually low and unsatisfactory for expert users

- **Restricted domain question answering systems:** such systems answer questions from one specific domain, e.g. medicine or patents, using only domain specific data sources. The authors note that such systems offer more utility for expert users, and the quality of answers is usually high. This also means however, that the such a system is only able to answer a limited number of questions. The authors also note that while attempts exist to integrate restricted domain systems into general domain systems, the problem of identifying when to hand a question over to a specific sub-system remains unsolved

Question type: this differentiates question answering systems by the types of questions they are built to answer. The correctness of the classification of the question type has a massive impact on the question answering system's performance. The authors propose five different categories:

- **Factoid questions:** this refers to questions that can be answered with a fact of a certain type. Usually these are questions starting with "what", "when", "which" or "who". These systems usually provide a satisfactory performance, as, for one, the answers are usually named entities which can be identified via named entity recognition (NER), and large and structured data sources like Wikipedia can be used as a fitting data source. However, these systems rely on the correct classification of the expected answer types, which on itself is a research topic in the field of question answering systems
- **List questions:** this refers to questions that expect a set of entities of facts as an answer. They can be interpreted as a series of consecutive factoid questions, where the previous answers are removed from the set of possible answers. A lot of the techniques used in factoid question answering can also be applied here„ however, it remains a problem to find a threshold to define the number of items that should be returned
- **Hypothetical questions:** this refers to questions that ask for information related to a hypothetical event, e.g. "what would happen if...". But while expert users might like to use such a system to find optimal answers for hypothetical questions, the implementation of a question answering system capable of answering these types of questions is cumbersome, as it requires a data source upon which the system has to be able to infer the answers, rather than just extract is
- **Causal questions:** this refers to the type of questions which ask for the cause for a certain fact or event, e.g. "why did...". The authors note that answering those kinds of questions is quite problematic, as for one it is hard to find a the correct reason(s) to complex questions, and that current models suffer from problems originating from their bag-of-words retrieval models
- **Confirmation questions:** this refers to questions expecting a "yes" or "no" answer. Similar to answering hypothetical questions, this requires a question

answering system to be able to reason over their knowledge base, which requires a higher level of knowledge acquisition and retrieval techniques, which are still under development

Applied text analysis: the authors propose a classification based upon the types of analysis done on questions in a question answering system. They present the following categories:

- **Morphological analysis:** this is the type of analysis which focuses on the surface forms of words. Through stemming and lemmatization morphemes and lexeme are extracted. While this analysis is required for effective searches, it also takes away semantic information when words share the same stemmed word, but are semantically different

- **Syntactic analysis:** this is the analysis of the grammatical construction of words in a sentence. An examples for such an analysis is the generation of parse trees. The authors note that syntactic analysis can improve performance by reducing the search space when looking for a fitting answer, as it takes into account not only the surface form of a word but also extends it with information about its role within the sentence, however, Incorrect assignments are an issue here

- **Semantic analysis:** this is the deduction of meaning of questions based on the words. According to the authors, usually the parse trees from the syntactic analysis are used to interpret the possible meaning. One task in the semantic analysis is semantic role labeling. The authors note that while the implementation of such techniques results in more effective searches and solves the problem of finding the expected answer type, the techniques (e.g. named entity recognition, part-of-speech tagging) used to gather semantic information are limited to the sentence level, and are also a source of potential problems.

- **Pragmatic and discourse analysis:** this is the kind of analysis that takes into account the context of an utterance, where previous utterances are also taken into account. The authors list some techniques used in the course of this analysis, (i) anaphora resolution, where, e.g. the relations of pronouns to proper nouns are resolved, and (ii) discourse structure recognition, which identifies logical connections between parts of a text. According to the authors this kind of analysis is necessary to be able to answer complex questions and to deduce the meaning of a text, but current implementations are still far from ideal, and problems from previous analysis steps like named entity recognition and part-of-speech tagging make it even harder to correctly analyze

- **Expected answer type analysis:** in this analysis the question answering system determines the entity type of an answer based on the category of the question. The authors note that this is helpful for factoid questions, but does not provide helpful insights in causal and procedural questions

- **Focus recognition:** in this step the question answering system tries to identify the parts of the question that are most relevant for finding the correct answer

Data source type: the authors propose three categories, (i) structured data sources, (ii) semi-structured data sources and (iii) unstructured data sources.

- **Structured data sources:** data is structured into semantic sets of entities, connected with relations. The description of all entities and relations is called a schema, and a structured data source can be queried using a corresponding query language. The authors note that while structured data sources are more reliable and do not require complex natural language processing on their end, but are also usually limited and hard to construct, and there are also many different formats and query languages
- **Semi-structured data sources:** this refers to data formats where the schema and data is not separated. According to the authors this kind of data representation provides a high level of flexibility, but building such data sources is labor intensive, and references are hard to reconcile
- **Unstructured data sources:** this refers to all data sources without any rules. Usually this means to heaps of text documents. Using this kind of data source requires the application of natural language processing and information retrieval techniques. The authors note that while data sources like this are easy to update, they also cause problems with paraphrasing and reliability.

Data source characteristics: data sources can further be classified by other features besides their structure. The authors propose five such characteristics, (i) the source size, (ii) the language the data is available in, e.g. German, English, etc., (iii) the heterogeneity of the data, meaning the number of different data formats used, (iv) the genre and (v) the media, i.e. text, image, video, etc..

Matching functions and retrieval models: here the authors propose a classification based on the types of matching functions used in the different retrieval models. They propose six different categories, (i) set theoretic models, (ii) algebraic models, (iii) probabilistic models, (iv) feature based models, (v) expected answer type analysis, and (vi) conceptual graph based models.

- **Set theoretic models:** treats documents and sentences as sets of words or phrases, and the matching is done by carrying out operations on sets
- **Algebraic models:** refers to models using vector or matrix representations of texts, calculating matching scores as scalar values
- **Probabilistic models:** these models treat "documents and questions in terms of probability relevance"

- **Feature models:** refers to models texts are represented as vectors consisting of different features extracted by feature functions, matching is done via calculation of a scalar matching score
- **Conceptual graph based models:** representing a sentence as a graph, matching is done by means of graph similarity measures

Forms of generated answers: The authors differentiate between (i) extracted answers, and (ii) generated answers, where the former refers to the retrieval of whole passages of text, where the latter refers to the construction of answer sentences based upon the findings of the question answering system, e.g. through reasoning.

The authors conclude that not all factors can be categorized, as they are hidden, like the skill of the user who is asking the questions. This could be solved by using conversational capabilities, which are still in development, or by using contextual information like browsing history.

Appendix B: Analyzed Question Answering Systems

Note: The referenced sources can be found in the thesis bibliography.

[Green Jr. et al., 1961] (BASEBALL): One of the first question answering systems ever created, it uses a database-like structured data source to answer baseball-related questions. The methods used to extract the necessary information from the users utterance are the use of controlled vocabulary, part-of-speech tagging, tree parsing and using interrogative words to classify the users questions.

[Androutsopoulos et al., 1995] (MASQUE/SQL): A natural language frontend for relational databases, it transforms a user's utterance into an SQL query, retrieves facts from a database and presents them to the user. The methods used to extract the necessary information from the users utterances are tree parsing and the use of a lexical database.

[Burke et al., 1997] (FAQ Finder): A question answering system using text files containing frequently asked questions (FAQs) as data source. It uses the lexical database WordNet to bridge the lexical gap and enhance the retrieval quality. Besides the lexical database it uses stemming and TF-IDF word embeddings to calculate a similarity score between a users question and an existing question, and returns the answer corresponding to the question with the highest score.

[Moldovan et al., 1999] (LASSO): A question answering system using documents as data source. It introduces so-called "paragraph indexing", in which paragraphs are the main focus of indexing, and not the documents, allowing for more fine-granular search results when looking for an answer. The natural language understanding methods used are part-of-speech tagging, tree parsing, named entity recognition and use of a lexical database.

[Harabagiu et al., 2000] (FALCON): Also a question answering system relying on paragraph indexing, the methods concerning natural language understanding are part-of-speech tagging, tree parsing, named entity recognition and use of a lexical database.

[Srihari and Li, 2000] (TextractQA): A question answering system built atop an information extraction engine called "Textrect". The methods used for natural language understanding are part-of-speech tagging, tree parsing and named entity recognition.

[Litkowski, 2001] (DIMAP-QA): A question answering system using text documents as data source. The NLU-relevant methods are part-of-speech tagging, tree parsing, named entity recognition and using a lexical database.

[Kwok et al., 2001] (Mulder): A question answering system using a web search engine to search for text documents as data sources. The users query is analyzed, expanded and relayed to the web search engine, the results provided by it are parsed and a fitting answer is extracted. The natural language understanding related methods used are part-of-speech tagging, tree parsing, the use of a lexical database and using TF-IDF weighted word embeddings.

[Zheng and Arbor, 2002] (AnswerBus): Similar to MULDER, it is a question answering system using a web search engine to retrieve documents potentially containing a fitting answer to a user's question. It also uses a simple coreference resolution in adjacent sentences to improve the retrieval of answer candidates. The natural language understanding methods used are lemmatization and named entity recognition.

[Ferret et al., 2002] (QALC): A question answering system using text documents as data source. It relies on recognizing the focus words of a question to retrieve a fitting answer. The natural language understanding methods used are part-of-speech tagging, tree parsing, named entity recognition and use of a lexical database.

[Wu et al., 2004] (ILQUA): A question answering system using text documents and a web search engine as data sources. It retrieves potential answers from relevant text passages, and uses as web search to try to confirm the answer candidates. The relevant natural language understanding methods used are tree parsing, named entity recognition and using a lexical database.

[Nyberg et al., 2005] (JAVELIN): A question answering system using text documents as data source. The natural language understanding methods used are part-of-speech tagging, tree parsing, named entity recognition and use of a lexical database.

[Linckels and Meinel, 2005] (CHESt): A question answering system for retrieving documents in a library setting. The methods used in understanding the natural language questions are the use of a controlled vocabulary and a lexical database.

[Wong, 2005] (NaLURI): A framework capable of answering natural language questions using an underlying ontology. The methods used in natural language understanding involve part-of-speech tagging, tree parsing, named entity recognition and the use of a gazetteer.

[Bernstein et al., 2006] (GINSENG): A natural language interface for OWL knowledge bases, it differs from similar systems in the way it deals with the lexical gap. It relies on the vocabulary used by the underlying knowledge base, suggesting fitting terms to the user on the fly. This use of controlled vocabulary is the method used for natural language understanding by this system.

[Qiu et al., 2007] (FDUQA): A question answering system using text documents as data source. The natural language understanding methods used are lemmatization, part-of-speech tagging and tree parsing.

[Cimiano et al., 2007] (ORAKEL): Also a natural language interface for knowledge bases, it focuses on portability between different domains. The methods used in terms of natural language understanding are the use of domain-aligned lexica, part-of-speech tagging and tree parsing

[Kaufmann et al., 2007] (NLP-Reduce): Another natural language interface for ontologies trying to avoid complex linguistic and semantic methods, offering a "naive" approach. The methods used by the system are stemming and the use of automatically generated lexica with the help of a lexical database.

[Lopez et al., 2007] (AquaLog): A portable question answering system using one or more knowledge bases as data source. The methods employed to solve natural language understanding are part-of-speech tagging, tree parsing and using a lexical database.

[Wang et al., 2007] (PANTO): A portable natural language interface for ontologies with focus on nominal phrases, the methods used for natural language understanding are using custom lexica, part-of-speech tagging, tree parsing and the use of a lexical database.

[Damljanovic et al., 2010] (FREyA): A natural language interface for ontologies, capable of asking clarification questions if the system failed to retrieve an answer. The methods used for natural language understanding are part-of-speech tagging, tree parsing and usage of a lexical database.

[Unger and Cimiano, 2011] (Pythia): An ontology-based question answering system. This system uses custom generated lexica, tree parsing and named entity recognition for natural language understanding.

[Waltinger et al., 2011]: A German language question answering system using a Wikipedia dump as data source. The natural language understanding methods used are lemmatization, part-of-speech tagging, tree parsing and named entity recognition.

[Kalyanpur et al., 2012] (DeepQA/Watson): A very famous question answering system using a multitude of different data sources, both unstructured text and structured knowledge bases. It applies a large number of different methods relevant to the task of natural language understanding: stemming and lemmatization, part-of-speech tagging, tree parsing, named entity recognition, using lexical databases and custom weighted word embeddings.

[Pradel et al., 2012] (SWIP): Another natural language interface for ontologies, this system differs in such a way that it translates the natural language utterance provided by the user into several natural language queries aligned with the vocabulary of the underlying ontology, and lets the user decide which query best fits his or her information need. This query is subsequently used to query the underlying knowledge base. The only relevant method used for natural language understanding is the use of controlled vocabulary.

[Aggarwal and Buitelaar, 2012]: A question answering system using the linked data as data source. The natural language understanding methods used are part-of-speech tagging, tree parsing, named entity recognition, dense word embeddings and the use of a lexical database.

[Cabrio et al., 2012] (QAKiS): Another question answering system using linked data as data source. The natural language understanding methods used are named entity recognition and the use of a pattern library.

[Lopez et al., 2012] (PowerAqua): A question answering system using linked data as data source. The natural language understanding methods used are part-of-speech tagging, tree parsing and named entity recognition, disambiguation and linking.

[Unger and Bühmann, 2012] (TBSL): A question answering system using linked data as data source. The natural language understanding methods used are part-of-speech tagging, tree parsing, named entity recognition and the use of a lexical database.

[Walter et al., 2012] (BELA): Also a question answering system using linked data as data source. The natural language understanding-relevant methods used are part-of.speech tagging, tree parsing, named entity recognition and the use of a lexical database.

[Yahya et al., 2012] (DEANNA): A question answering system using linked data as data source. The natural language understanding methods used are part-of-speech tagging, tree parsing, named entity recognition and the use of a lexical database.

[Dima, 2013] (Intui2): A question answering system using linked data as data source. The natural language understanding methods used are lemmatization, part-of-speech tagging, tree parsing and the use of a lexical database.

[Giannone et al., 2013] (RTV): A question answering system using linked data as data source. It relies on a hidden Markov model approach to identify fitting answer candidates. The natural language understanding methods used are lemmatization, part-of-speech tagging, tree parsing and dense pointwise mutual information weighted word embeddings.

[Fader et al., 2013] (PARALEX): A paraphrase-driven question answering system using question-answer pairs as data source. The natural language understanding methods used is the use of a lexical database.

[Berant and Liang, 2014] (PARASEMPRE): A question answering system using both unstructured text and a knowledge base as data sources. It uses a large text corpus to build paraphrases of the question which are better aligned to the knowledge base. The natural language understanding methods used are lemmatization, part-of-speech tagging, tree parsing, use of a lexical database and dense word embeddings.

[Freitas and Curry, 2014] (Treo): A question answering system using linked data as data source. The authors use a special vector space embedding based on "Explicit Semantic Analysis" (ESA), and a reference text corpus, to embed the knowledge base into, and to retrieve the answers from. The natural language methods used are part-of-speech tagging, tree parsing, named entity recognition and word embeddings.

[Frost et al., 2014] (DEV-NLQ): A natural language query interface for event-based triplestores, it uses internally represents a user's query as lambda calculus expressions to evaluate them. The methods used for natural language understanding are part-of-speech tagging and the use of a controlled vocabulary.

[Hamon et al., 2014] (POMELO): A question answering system for the biomedical domain, it was one of the participants of the QALD-4 challenge. To solve natural language understanding, the authors used lemmatization, part-of-speech tagging, tree parsing and named entity recognition.

[Dima, 2014] (Intui3): A question answering system using linked data as data source. The successor to Intui2, the natural language understanding methods used are lemmatization, part-of-speech tagging, tree parsing, named entity recognition and use of a lexical database.

[He et al., 2014] (CASIA): A Markov Logic Network based question answering system using linked data as data source. The natural language understanding methods used are part-of-speech tagging, tree parsing, custom implemented named entity recognition, dense word embeddings and the use of a lexical database.

[Höffner and Lehmann, 2014] (CubeQA): A question answering system specializing on statistical data using a knowledge base as data source. The natural language understanding methods used are part-of-speech tagging, tree parsing, named entity recognition and the use of a lexical database.

[Park et al., 2014] (ISOFT): A question answering system using linked data as data source. The natural language understanding methods used are part-of-speech tagging, tree parsing, named entity recognition, dense word embeddings and the use of a lexical database.

[Xu et al., 2014] (Xser): Another question answering system using linked data as data source. The relevant natural language understanding methods used are part-of-speech tagging and named entity recognition.

[Zou et al., 2014]: A question answering system using linked data as data source. The natural language understanding methods used are part-of-speech tagging, tree parsing, named entity recognition and the use of a lexical database.

[Beaumont et al., 2015] (SemGraphQA): Also a question answering system using linked data as data source. The natural language understanding methods used by the authors are part-of-speech tagging, tree parsing, named entity recognition and the use of a lexical database.

[Hakimov et al., 2015]: Also a question answering system using linked data as data source. The natural language understanding methods used are part-of-speech tagging, tree parsing and the use of a lexical database.

[Ruseti et al., 2015] (QAnswer): A question answering system using linked data as data source. The natural language understanding methods used are stemming, part-of-speech

tagging, tree parsing, named entity recognition and the use of a lexical database.

[Shekarpour et al., 2015] (SINA): A question answering system using linked data as data source. The natural language understanding methods used are lemmatization and named entity recognition.

[Song et al., 2015] (TR Discover): A question answering system using a knowledge base as data source, it maps English language fragments to first order logic statements using a feature-based grammar. The natural language understanding methods used is the use of a controlled vocabulary, tree parsing and named entity recognition.

[Usbeck et al., 2015] (HAWK): A question answering system using both unstructured text and linked data. The natural language understanding methods used are part-of-speech tagging, tree parsing and named entity recognition.

[Abacha and Zweigenbaum, 2015] (MEANS): A medical question answering system extracting answers from MEDLINE articles. The articles are converted to semantic graphs and queried using SPARQL, while the queries are produced by converting the natural language questions from the users. The methods used to extract the information from both the source documents and the users questions are similar and involve part-of-speech tagging, named entity recognition and tree parsing.

[Asiaee et al., 2015] (OntoLNQA/AskCuebee): A framework for question answering in the biomedical field, applied in a system called AskCuebee for questions concerning parasite immunology data. The methods used for natural language understanding are stemming, part-of-speech tagging, tree parsing and named entity recognition.

[Tsai et al., 2015]: A revitalization of AskMSR, an older web search-based question answering system. The authors evaluate how the improvements made by search engine providers have affected systems built around older, less precise versions. The natural language understanding methods used are named entity recognition and the use of a lexical database.

[Baudiš and Šedivý, 2015] (YodaQA): A question answering system framework using unstructured text as data source. The natural language understanding methods used are lemmatization, part-of-speech tagging, tree parsing, named entity recognition and using a lexical database.

[Sun et al., 2015] (QuASE): A question answering system using both unstructured text and structured knowledge bases as data sources, the unstructured text is retrieved via a web search engine. The only relevant method used for natural language understanding is bag-of-words embeddings.

[Gallagher and Zadrozny, 2016] (Watsonism): A question answering system modeled after the basic structure used by IBM's Watson question answering system. The natural language understanding methods used by this system are part-of-speech tagging, named entity recognition and tree parsing.

[Damiano et al., 2017] : An Italian language question answering system using textual data sources capable of answering factual questions in the field of cultural heritage. It uses deep neural nets trained to classify the questions. The techniques used to extract the features used for classification involve stemming, lemmatization, part-of-speech tagging and named entity recognition.

[Schwarzer et al., 2016] A German language question answering system using internal documents from public administration as a data source to answer questions concerning governmental services. The methods used for natural language understanding are TF-IDF word embeddings, part-of-speech tagging and the use of a German lexical database.

[Zhang et al., 2016]: A question answering system capable of using multiple aligned knowledge bases as data source. The method used relevant to natural language understanding is tree parsing.

[Zhu et al., 2016]: A question answering system using linked data as data source, the methods used for natural language understanding are part-of-speech tagging, tree parsing and named entity recognition.

[Nam et al., 2017] (OKBQA extended): A framework for creating question answering systems using knowledge bases as data source. The natural language understanding methods used are part-of-speech tagging, tree parsing and named entity recognition.

[Chandurkar and Bansal, 2017]: A question answering system using a knowledge base as data source. The NLU methods used are part-of-speech tagging, tree parsing and named entity recognition.

[Carvalho et al., 2017]: A question answering system in the legal domain, using legal texts as data sources. It uses recognition of textual entailment to to provide evidence to decide upon the correct answer. The methods relevant to natural language understanding are lemmatization, part-of-speech tagging and dense word embeddings.

[Kim et al., 2017]: Also a question answering system in the legal field using recognition of textual entailment, the methods used here are lemmatization, part-of-speech tagging, tree parsing and dense word embeddings.

[Saany et al., 2017] (QAUF): A natural language interface for knowledge bases, it especially considers the use of modifier terms. It also incorporates user modeling and relevance feedback to improve the answer quality. The methods used for natural language understanding are part-of-speech tagging, tree parsing and custom-weighed word embeddings.

[Šukys et al., 2017]: A natural language interface for ontologies based on the controlled vocabulary SBVR. The natural language understanding methods used are the use of a controlled vocabulary, lemmatization, tree parsing and part-of-speech tagging.

[Yin et al., 2017]: A question answering system relying heavily on the use of deep neural nets, based on the constituency trees of user utterances. The NLU-relevant methods used are tree parsing and dense word embeddings.

[Marginean, 2017] (GFMed): A question answering system for using linked biomedical data as data source, it uses custom grammars to evaluate controlled language queries. The method used for natural language understanding is the use of controlled vocabulary and language.

[An et al., 2017]: A question answering system approach using an unstructured corpus of questions and corresponding answers. Neural nets are used to compute similarities between a user's question and existing questions in the corpus. The only methods used relevant in this context of natural language understanding is the use of dense word embeddings.

[Figueroa, 2017]: A question answering system in the field of community Question-Answering platforms (cQA), where users post questions and other users with more domain knowledge answer them. One of the challenges is to identify questions which have already been satisfactorily answered and present those already existing answers to newly posted questions. The natural language understanding methods used are part-of-speech tagging, tree parsing, named entity recognition and the use of a lexical database.

[Hoque and Quaresma, 2017] (SEMONTOQA extended): A question answering system using unstructured text, web search engines and knowledge bases as data sources. It is also capable of integrating user feedback to improve the answer candidate ranking. The natural language understanding methods used are tree parsing, named entity recognition and the use of a lexical database.

[Romeo et al., 2017]: A question answering system using cQA forum entries as data source. The natural language understanding relevant method used is tree parsing.

[Ruan et al., 2017]: A question answering system using unstructured text as data source. It concentrates on feature merging to calculate sentence similarity and subsequently train a neural net with the extracted and merged features. The natural language understanding methods used are dense word embeddings and use of a lexical database.

[Yue et al., 2017]: A question answering system using unstructured text as data source bases on an end-to-end neural net implementation. The natural language understanding relevant method used is dense word embeddings.

[Oh et al., 2017]: A question answering system concentrated on causal questions using unstructured text as data source and an end-to-end neural network. The natural language understanding methods used is dense word embeddings to represent the natural language inputs.

[Belyaev et al., 2017]: A Russian language question answering system using web search engines as data source. The system extracts fitting passages from the results returned by the search engines and converts them to an internal RDF-triple representation, which is used for answer candidate extraction. The natural language understanding methods used are lemmatization, part-of-speech tagging and named entity recognition.

[Khvalchik et al., 2017]: A question answering system concentrating on non-factoid questions, using commercial web search engines as data source. The natural language understanding methods used are stemming, part-of-speech tagging and tree parsing.

Appendix C: Training Data Formats

Microsoft LUIS

Training API expects JSON documents containing training objects such as this:

```
{
  "text":"Which team is leading the German Bundesliga table?",
  "intentName":"standings",
  "entityLabels":[
    {
      "startCharIndex":26,
      "endCharIndex":43,
      "entityName":"COMP"
    }
  ]
}
```

IBM Watson Assistant

Training API expects separate CSV files for intents and named entities, format as detailed below:

Intents:

```
TEXT,INTENT
Which team is leading the German Bundesliga table?,standings
```

Entities

```
TYPE,NAME,ALTERNATIVE_NAME_1,...,ALTERNATIVE_NAME_N
COMP,Erste Liga,Austria Erste Liga,2nd League Austria
```

wit.ai

Training API expects JSON documents containing training objects such as this:

```
{
  "text":"Which team is leading the German Bundesliga table?",
  "entities":[
    {
          "value":"standings",
      "entity":"intent"
    },
    {
      "value":"German Bundesliga",
      "entity":"COMP",
      "start":26,
      "end":43
    }
  ]
}
```

rasa NLU

Training API expects training objects such as this:

```
{
  "text":"Which team is leading the German Bundesliga table?",
  "intent":"standings",
  "entities": [
    {
      "start":26,
      "end":43,
      "value":"German Bundesliga",
      "entity":"COMP"
    }
  ]
}
```