BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Ta Van Duy**	Personal ID number:	**456906**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Open Informatics**		
Branch of study:	**Computer and Information Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Entity Linking and Disambiguation Using a Dialogue

Bachelor’s thesis title in Czech:

Vyhledávání entit a významové rozlišování pomocí dialogu

Guidelines:

1. Research current state-of-the-art methods for Entity Linking.
2. Propose suitable recurrent neural network for Entity Linking based on the previous research.
3. Propose a dialogue based on user utterance and given knowledge base (Freebase, Wikidata) to disambiguate the entities.
4. Implement proposed entity linking model and dialogue disambiguation.
5. Run experiments on publicly available data and dialogue oriented data.
6. Discuss your results and compare the Entity Linking part of the task with state-of-the-art methods.

Bibliography / sources:

[1] CHIU, Jason PC; NICHOLS, Eric. Named entity recognition with bidirectional lstm-cnns. arXiv preprint arXiv:1511.08308, 2015.
[2] VAN ERP, Marieke, et al. Evaluating Entity Linking: An Analysis of Current Benchmark Datasets and a Roadmap for Doing a Better Job. In: LREC. 2016. p. 2016.
[3] MORO, Andrea; RAGANATO, Alessandro; NAVIGLI, Roberto. Entity linking meets word sense disambiguation: a unified approach. Transactions of the Association for Computational Linguistics, 2014, 2: 231-244.

Name and workplace of bachelor’s thesis supervisor:

Ing. Jan Pichl, Big Data and Cloud Computing, CIIRC

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **09.01.2018** Deadline for bachelor thesis submission: **25.05.2018**

Assignment valid until: **30.09.2019**

Ing. Jan Pichl Supervisor’s signature	doc. Ing. Tomáš Svoboda, Ph.D. Head of department’s signature	prof. Ing. Pavel Ripka, CSc. Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Bachelor's Thesis

Entity Linking and Disambiguation Using a Dialogue

Ta Van Duy

Supervisor: Ing. Jan Pichl

Study Programme: Open Informatics

Field of Study: Computer and Information Science

May 25, 2018

Acknowledgements

I would like to thank Ing. Jan Pichl for his advice and patience with me throughout my first big project. Secondly, I would like to thank Ing. Jan Šedivý, CSc. for giving me an opportunity to work in eClub. I would like to also thank Bc. Hoang Long Nguyen who was the main reason I have met this awesome group and that I have decided to work in this field. Finally, I would like to thank everyone who was supporting me during my studies, family, friends and class mates.

Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

Declaration

Abstract

Named entity linking is a task, where we try to link entity mentions to the proper entities from knowledge base like Wikipedia or Freebase. However, generally there could be more then one entity candidate for entity mention. The task where we choose the correct entity given the entity candidates is called entity disambiguation.
The aim of this thesis was to create entity linking and disambiguation system using dialogue. In our work we experimented with recurrent neural networks in entity recognition part. Furthermore, we analysed two approaches that used Freebase and Microsoft Concept Graph for generating clarifying question in entity disambiguation part. Finally, we compared our entity linking and disambiguation system with the current state-of-the-art methods.

Keywords Entity Recognition, Entity Disambiguation, Entity Linking, Natural Language Processing, Neural Networks, Knowledge Base, Freebase, Dialogue

Abstrakt

Propojování jmenných entit je úloha, ve které se snažíme propojit entity zmíněné v textu s jejich reprezentací ve znalostních databázích jako Wikipedia nebo Freebase. Nicméně, v obecném případě může rozpoznané entitě v textu odpovídat více kandidátů. Úlohu, která se zabývá rozpoznáním správné entity z dané množiny potencionálních kandiátů, nazýváme významové rozlišování entit.
Cílem naší práce bylo vytvořit systém na vyhledávání a významové rozlišování entit pomocí dialogu. V části zabývající se vyhledáváním entit jsme experimentovali s rekurentními neuronovými sítěmi. V části zabývající se významovým rozlišováním entit jsme analyzovali dva přístupy využívající databázi Freebase a graf konceptů od Microsoftu k generaci otázek sloužících k rozlišení kandidátů. Na závěr jsme porovnali náš přistup s existujícími systémy na rozlišování entit.

Klíčová slova Vyhledávání entit, Významové rozlišování entit, Propojování entit, Zpracování přirozeného jazyka, Neuronové sítě, Znalostní databáze, Freebase, Dialog

Contents

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 Our Goal . . . 2
- 1.3 Thesis Structure . . . 2

2 Related work **3**
- 2.1 Current Trends . . . 3
- 2.2 Named Entity Disambiguation Approaches . . . 4
  - 2.2.1 Entity Linking meets Word Sense Disambiguation: a Unified Approach 4
  - 2.2.2 Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation . . . 5
  - 2.2.3 Did you mean A or B? Supporting Clarification Dialog for Entity Disambiguation . . . 5

3 Problem Formulation **7**
- 3.1 Problem Statement . . . 7
- 3.2 Knowledge Bases . . . 8
  - 3.2.1 Freebase . . . 8
  - 3.2.2 Wikipedia . . . 9
  - 3.2.3 DBPedia . . . 9
  - 3.2.4 Wikidata . . . 9
  - 3.2.5 YAGO . . . 10
- 3.3 RDF . . . 10
- 3.4 SPARQL . . . 11
- 3.5 Datasets . . . 12
  - 3.5.1 CoNLL-2003 Dataset . . . 12
  - 3.5.2 AIDA CoNLL-YAGO Dataset . . . 12
- 3.6 Evaluation Metrics . . . 13

4 Theorethical Analysis **15**
- 4.1 Neural Networks . . . 15
  - 4.1.1 Perceptron . . . 16
  - 4.1.2 Fully Connected Neural Networks . . . 18
  - 4.1.3 Training . . . 19
    - 4.1.3.1 Gradient Descent . . . 19

4.1.3.2 Backpropagation 20
4.1.3.3 Stochastic Gradient Descent 20
4.1.3.4 Vanishing and Exploding Gradients 20
4.1.3.5 Overfitting 21
4.1.4 Convolutional Neural Networks 21
4.1.5 Recurent Neural Networks 23
4.1.5.1 Gated Recurrent Units 24
4.1.5.2 LSTM 25

5 Implementation and Experiments 27
5.1 Named Entity Recognition 27
5.2 Candidate Generation 30
5.3 Named Entity Disambiguation 31
5.3.1 Freebase approach 31
5.3.2 Microsoft Concept Graph Approach 33

6 Conclusion 35
6.1 Future Work 36

A CD/DVD Contents 43

List of Figures

2.1 Directed graph of semantic interpretations 4

3.1 RDF triple diagram 10
3.2 Example SPARQL query with resutls 11
3.3 CoNLL-2003 example sentece 13
3.4 AIDA CoNLL-YAGO example sentece 13

4.1 Biological neuron 15
4.2 Artificial neuron 16
4.3 Non-linear activation functions 17
4.4 Fully connected neural network 18
4.5 Backpropagation diagram 20
4.6 Convolutional neural network in NLP 22
4.7 Recurrent neural network diagram 23
4.8 Gated recurrent unit 24
4.9 Long short-term memory unit 25

5.1 Full process of question generation 32
5.2 Freebase approach on the second type of outcome 32
5.3 Microsoft Concept Graph approach on the first type of outcome 33
5.4 Microsoft Concept Graph approach on the second type of outcome 33

List of Tables

2.1 Examples of central and peripheral representatives given a type 6

5.1 Baseline RNN model with IOB tags 27
5.2 Baseline RNN model results with a different tag scheme 28
5.3 Baseline GRU model 28
5.4 Baseline LSTM model 28
5.5 Models utilizing CNNs 28
5.6 Models utilizing bidirectional layer 29
5.7 Models after applying dropout 29
5.8 Models after adding pre-trained embeddings 29
5.9 Models with varIous hyperparameters 29
5.10 Results of all models 30

Chapter 1

Introduction

1.1 Background

Entity linking (EL) [53] is a task in natural language processing (NLP) where an entity mention is linked to the corresponding entity in a knowledge base (KB) such as Wikipedia[1], Wikidata[2], DBpedia[3], Freebase[4] or YAGO[5]. The task itself can be divided into three parts. The first part is called named entity recognition (NER) [44], where the elements in a text with the meaning of names of persons, locations, organisations, are detected. The detected strings are called entity mentions. The second part consists of generating possible entities from knowledge base which could be associated with the found entity mention. This process is often referred to as entity candidate generation. In the third part, the correct entity from candidate entities has to be linked to the detected entity mention. Since the mention is typically ambiguous, which means that it refers to more than one entity candidate, we call the last part named entity disambiguation (NED) [17].

For example in a sentence:

"Washington is the father of the nation".

The entity mention would be a string *"Washington"*. The meaning of this mention could be

- *Washington, D.C., the capital city of the USA*
- *the State of Washington in the USA*
- *George Washington, the first president of the USA*

In this example, people can quickly tell that the ground truth is *George Washington*. However, it is not that simple for a computer to automatically disambiguate between possible entity candidates.

[1] <https://www.wikipedia.org/>
[2] <https://www.wikidata.org/>
[3] <http://wiki.dbpedia.org/>
[4] <https://developers.google.com/freebase/>
[5] <https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/>

For the past few years, the importance of entity linking has been rising by a notable amount. The possible reason could be that the devices using NLP are used considerably more than they used to. Starting from the Apple's Siri in 2011, continuing to home voice assistants like Amazon Echo or Google Home in the present.

According to [58] the possible applications of EL are information extraction and retrieval, content analysis, knowledge base population and question answering. As we can see the usage of EL is widespread and current technological trends are calling for improvements and new approaches in NLP field. It seems like the smartphone era is coming to an end and the new age of voice is just beginning.

1.2 Our Goal

The aim of this work is to create entity linking and disambiguation system using a dialogue. To do so, we firstly analyse the current state-of-the-art approaches in entity linking and disambiguation field. Afterwards, we focus on the goal itself, which is the implementation of the entity linking and disambiguation system. Finally, we compare our proposed method based on an interaction with a user with the researched state-of-the-art methods.

1.3 Thesis Structure

In chapter 2 we give the brief overview of the current trends and the related work. In chapter 3, we formulate our problem from the theoretical aspect. Furthermore, we describe and explain our choice of a knowledge base, datasets and evaluation metrics. Chapter 4 explains the fundamental theoretical concepts of algorithms used in our proposed model. Chapter 5 is devoted to the implementation of the proposed model and the experiments performed during the process. Finally, in chapter 6 we sum up our work and discuss possible future improvements.

Chapter 2

Related work

2.1 Current Trends

Concerning the NER part of EL, ER systems generally used to rely on hand-crafted features and domain-specific knowledge or resources such as gazetteers [36]. However, in 2013 Mikolov et al. [40, 41] proposed a new and more efficient way to represent words in a continuous space using simple neural network model. Moreover, their model automatically extracted features from a text as authors reported that learned embeddings captured syntactic and semantic regularities in language. For example, to find a word that is similar to "women" in the same sense as a "king" is similar to "men", we can compute

$$vector(X) = vector(\text{“}king\text{”}) - vector(\text{“}men\text{”}) + vector(\text{“}women\text{”})$$

The result is a word closest to the X using cosine similarity, which yields $vector(\text{“}queen\text{”})$ [42].

Nowadays, we can observe a shift from using hand-crafted features to automatic extraction of features using neural networks. People started to use word embeddings to leverage their systems, and also they started to experiment with all the different kinds of embeddings. For example, character embeddings [57] are now used in most of the ER systems. The current state-of-the-art results in NER are achieved by the combination of embeddings, bidirectional long short-term memory networks (BLSTMs) and convolutional neural networks (CNNs) [11, 38].

When it comes to NED, the trend is pretty similar. There is a broad range of hand-crafted features ranging from exact string match or textual context up to part-of-speech (POS) tags as is described in [58]. Furthermore, it should also be mentioned that not only features are created but also whole data structures dedicated particularly to NED. The examples of these structures are mention-entity graph [30] and referent graph [25]. Besides introducing mention–entity graph, Hoffart et al. also developed AIDA CoNNL-YAGO dataset in which we will be particularly interested. This dataset, which we will be using, is based on CoNLL-2003 NER dataset and it is manually annotated with proper entities. Another work concerning datasets is [64] where the current benchmark EL datasets are compared and tested in depth.

Similarly to ER, in last years people are trying to move from creating handcrafted features to end to end systems using neural networks. In the following sections, we will introduce three articles which apply a different approach to the problem of NED.

2.2 Named Entity Disambiguation Approaches

2.2.1 Entity Linking meets Word Sense Disambiguation: a Unified Approach

Entity Linking and Word Sense Disambiguation (WSD) are both tasks that aim to solve ambiguity of the language. Even though these tasks are similar in some ways, there are some significant differences. The difference will be best seen in an example. Given the sentence

"Thomas and Mario are striker playing in Munich"

In this example, the entity linking problem would be to link mentions *"Thomas"* and *"Mario"* to the corresponding entities in a knowledge base. In this case, *Thomas Müller* and *Mario Gómez* are both football players of Bayern Munich. The word sense disambiguation problem can be seen in words *"striker"* and *"playing"*, where the correct sense of the words had to be selected from the dictionary. Thus, the main difference is that EL uses knowledge base while WSD use dictionary and that in EL the mention could be partial while in the WSD, there is a perfect match between word form and word sense. However, other than that, tasks are pretty similar as they both focus on disambiguation of the text.

This article [43] presents an innovative approach to the EL and WSD because they tackle both problems jointly instead of separately as rest of community does. They present a hypothesis that the knowledge used in WSD is useful for EL and vice versa. They propose Babelfly, which is unified graph-approach to both WSD and EL. With this algorithm, they reached 82.1% accuracy on AIDA CoNLL-YAGO dataset 3.5.2.

The base of their algorithm is BabelNet which is a directed multigraph automatically constructed from Wikipedia and Wordnet. In the first step of the algorithm, they assign set of related vertices to each vertex in a semantic network. This is done firstly by giving higher weights to the edges in more densely connected areas. Afterwards given the starting vertex v they perform random walks with a restart to return set of related vertices for v, where conditional probabilities are based on previously mentioned edge weights.

Given a text, they extract all the linkable fragments from it. For each of linkable fragment, they list the possible meanings based on the previous step. Now, Given both identified fragments and related vertices, they build directed graph of semantic interpretations as can be seen in figure 2.1. Afterwards, the densest subgraph heuristic is applied, thanks to which the subgraph of most coherent semantic interpretations is what remains. Each possible candidate meaning in the densest subgraph is given a score equal to a normalized weighted degree. The final step is to link each identified fragment to the highest ranked candidate meaning if its score exceeds a certain fixed threshold.

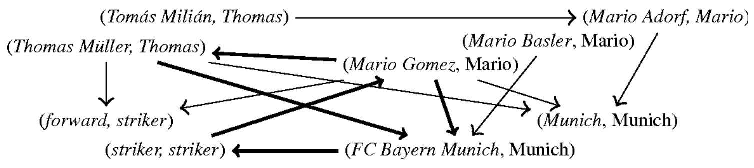

Figure 2.1: Directed graph of semantic interpretations [43]

2.2.2 Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation

In this article [69] the neural network approach is selected to disambiguate entities. Proposed method jointly maps word and entities into the same continuous vector space, and it is an extension of the skip-gram model [40].

The proposed model consists of three parts:

- 1) the original skip-gram model which predicts neighbouring words around the given target word
- 2) the KB graph model which estimates neighbouring entities given the target entity in the link graph of the KB
- 3) the anchor context model that learns to predict neighbouring words given the target entity using anchors and their context words in the KB

The idea is to optimize all three models simultaneously which results in the desired mapping of words and entities into one continuous space. These embeddings are later used in candidate ranking, but firstly a method for candidate generation has to be chosen. Authors use two public entity candidate datasets created particularly for EL on AIDA CoNLL-YAGO dataset and compare their results afterwards. The first one is by Pershina et al. [50] and the second one by Hoffart et al. [30].

As was said earlier, learned embeddings are used as the primary feature in candidate ranking. To be more precise two features are derived from proposed embeddings. The first one, the textual context similarity, is a cosine similarity between an entity and words in a document. The second one, coherence, is measured based on the relatedness between the target entity and other entities in a document. With these two features and several others including string similarity, entity prior $P(e)$, prior probability $P(e|m)$ and the number of entity candidates, they use gradient boosted decision trees [24] and according to our research, their approach is current state-of-the-art with 93.1% micro and 92.6% macro accuracy on AIDA CoNLL-YAGO.

2.2.3 Did you mean A or B? Supporting Clarification Dialog for Entity Disambiguation

So far we have introduced two articles on NED. One with a graph-based approach and the second one was using neural networks. The last article [14] that we will cover in this section is tackling NED from the dialogue perspective which is what we were are precisely looking for. It proposes and discusses the following three ways to construct clarification question.

- **Type-based** This is the most simplistic approach. It supposes that given a word we have its possible entity types. For example, given a word *"orange"*, let's say that the types would be a *"colour"* and a *"fruit"*. The question would be asking whether the *"orange"* was meant as a *"fruit"* or as a *"colour"*. However, the most significant problem with this approach is that it cannot distinguish between the entities of the same type. Moreover, it highly depends on the availability of human-readable and descriptive labels for entity types. Therefore, other types of questions are introduced.

- **Example-based** In cases where the entity class labels are not available or useful the next most straightforward way to clarify the meaning is using other example entities that are similar to the possible meanings. Let's suppose that we are trying to disambiguate mention *"orange"* again. In this approach instead of asking for types, the question would be: *"Do you mean 'orange' like banana and apple, or 'orange' like yellow?"*. To ask such question, we have to choose how many and which examples to provide. Authors state that just showing the alphabetically first few members or a random subset of an entity type is not sufficient, especially in cases where a list of type members may run to the thousands.

  They propose to use word2vec [40] continuous space for choosing examples of the type. One approach is to select the closest vector measured in cosine distance. The second proposed approach is to compute the cosine distance between all term pairs in a type and then for each member compute the sum of the distance to all other terms. Those with the smallest total sum are in a sense the "median" of the terms of that entity type. We can think of this as finding stereotypical terms from each type, which reflects our intuition of what would make good example terms for a clarifying question. Terms that have multiple common meanings besides the one under consideration end up with a larger distance and thus are more "peripheral".

Type	**Central**	**Peripheral**
Color	mauve, lilac, pink, taupe	coal, sage, bordeaux, coffee
Fruit	apricots, cheries, plums, mellon	mulberry, berry, mandarin, orange

  Table 2.1: Examples of central and peripheral representatives given a type [14]

- **Usage-based** The last proposed method shows language usage of a term in context. For example: *"Do you mean 'orange' as in 'I would like an orange juice.' or 'I love the 4G speed Orange now offers.'?"* We can find the context by analysing a large corpus where the entities of the target type occur many times. We look at these occurrences to identify contexts that are fairly specific to the type in question. This results in a set of patterns where a type has a high probability to occur. For example, for the type colour: *'black / *'; '* in colour.'; 'red, * and blue'; 'shades of * and'; '* and purple';*

  Patterns are scored as to the specificity of the context to the given type, and in the most discriminating patterns, the * is replaced with the word of interest which in our case is *"orange"*. We then query a corpus for occurrences of such patterns. If we get a match, the sentence in which it appears is selected. Otherwise, the process is repeated with the next best pattern and so forth until a match is found. Afterwards, only the clause of interest is selected from the founded sentence and only this part is used in a question. Hence, selected sentence: "Orange and purple crayon streaks rainbowed across his briefcase surface." results in the question: "Do you mean 'orange' like 'orange and purple crayon streaks'?"

Chapter 3

Problem Formulation

3.1 Problem Statement

As described in section 1.1, entity linking is a task where we first find named entity mentions in given text, then generate entity candidates from selected KB and afterwards link founded entity mention to the appropriate entity.

NER can be formulated as sequence labelling problem where we label each word in a sentence by specific tag from given set of labels. NED can be taken as a ranking problem where we order entity candidates and afterwards, we choose the highest ranked (most probable) entity candidate.

In practice, when dealing with NED, researchers expect a pre-tagged input, or they use existing external annotators so that a full focus is put on NED. Typically, Stanford CoreNLP[1] [39] is widely used in NLP community. However, in our work, we experiment with neural networks in NER part as they had reached state-of-the-art results which we discussed in chapter 2.

Concerning candidate generation, we will use label-lookup[2] which is used in Alquist [51] and YodaQA [4]. Nonetheless, two public entity candidate dataset for AIDA CoNNL-YAGO provided by Hoffart et al. [30] respectively Pershina et al. [50] should be taken into account. However, both of these provides only Wikipedia mapping. Additionally, they are created particularly for EL on AIDA CoNLL-YAGO while label-lookup can be used more universally.

In the NED part, we experiment with Freebase and Microsoft Concept Graph[3] to generate questions so that we can disambiguate using dialogue. Our approach will be more thoroughly discussed in the section devoted to implementation and experiments 5. However, before the implementation, several things should be clarified first. In this section, we will specify the choices of the KB, datasets and evaluation metrics.

[1]<`https://stanfordnlp.github.io/CoreNLP/`>
[2]<`https://github.com/brmson/label-lookup`>
[3]<`https://concept.research.microsoft.com/`>

3.2 Knowledge Bases

In general, a knowledge base is storage of information. In our case, a knowledge base can be viewed as a database of entities, information about the entities and mutual relations between them. Since a knowledge base is needed in every EL pipeline and each of KB is different in some aspects, the choice for us as designers of EL system is the important one.

In this section, we will primarily focus on Freebase because it is the KB we will be using. Then we will describe Wikipedia from which many features are derived and finally we will give a brief overview of other known knowledge bases.

3.2.1 Freebase

Freebase [6] is a graph based database of general human knowledge. From a historical point, Freebase was launched in 2007 by Metaweb Technologies company. The company was acquired by Google in 2010. In December 2014, Google announced that it would shut down Freebase in mid-2015 and help with the move of the data from Freebase to Wikidata. Freebase was officially shut down on 2 May 2016 [67]. Nowadays, only RDF dump of Freebase is what is left. Additionally, Google Knowledge Graph and Wikidata are meant to serve as a substitution for Freebase.

The data in Freebase were collaboratively created, structured and maintained. What is meant by that, is that Freebase was publicly accessible through an HTTP-based graph-query API and public users could work with the database as they wanted to. Information in Freebase was collected from multiple sources. The main source was Wikipedia, but there were a lot of other sources [22] including the following:

- Wikimedia Commons
- Open Library Project
- Stanford University Library
- TVRage
- MusicBrainz
- National Register of Historic Places
- OurAirports
- ITIS - Taxonomy of plants and animals,
- World of Spectrum
- WordNet

Thanks to the broad coverage from the multiple sources, Freebase was and still is one of the largest currently existing knowledge bases, covering the most of world existing entities, even though it was shut down. Since the last Freebase database dump included over 42 million objects and 1,9 billion relationships and it is free to use, we will be using FB as our KB. In the following part, we will describe what RDF is because Freebase is now only available as RDF dump. Afterwards, we will also look on the way of querying RDF data.

3.2.2 Wikipedia

Wikipedia is a free multilingual encyclopedia launched on January 15, 2001. Wikipedia founders, Jimmy Wales and Larry Sanger, aimed to create the most comprehensive, free, widely-available collection of human knowledge. The goal seems to be accomplished pretty well as it is currently the 5th most popular website according to Alexa rank [1], and it contains over 5.6 million articles in English [16]. These statistics make Wikipedia not only the most popular Internet encyclopedia in the world but also the largest one.

Similarly to Freebase, the content of Wikipedia exists mainly thanks to the collaborative effort of the community around the world. The primary entry in Wikipedia is an article, which defines and describes an entity or a topic. Each article in Wikipedia is uniquely referenced by an identifier. This URL identifiers are often used for mappings, which can be seen for example in section 3.5.2. The article itself serves as valuable entity linking feature. For example, mention, context or entity candidate embeddings can be created from a text which is describing entity [62]. Some of other EL features provided by Wikipedia are article categories, redirect pages, disambiguation pages, and hyperlinks in Wikipedia articles [58].

3.2.3 DBPedia

Even though Wikipedia is comprehensive and popular among people, its structure is semi-structured, therefore not entirely machine-understandable. This issue is tackled by DBpedia [2, 5, 37] project which is a collaboration of the Free University Berlin, the University of Leipzig and OpenLink Software. The project aims to create a structured knowledge base from Wikipedia which would be freely available on the Web. To be more specific, DBpedia tries to extract structured information which is embedded in the articles and map it to a single ontology. The most of structured information on Wikipedia is in the form of infoboxes. An infobox is a table which is located in the top right corner of the Wikipedia article and lists article's most relevant facts. Extracted data are stored in the RDF triplets and can be queried with SPARQL. The possibility to query structured data is the main advantage in comparisons with the Wikipedia, where only the full-text searches can be performed.

3.2.4 Wikidata

Wikidata [65] is a collaborative knowledge base launched by Wikimedia foundation in 2012. Its goal is to store the information in a structured way to help other projects including Wikipedia or the question answering systems.

Since Freebase authors thought of their project as "Wikipedia for structured data" and with quickly growing Wikidata community which was around 6000 active contributors in mid-2015, Google decided to down Freebase in mid-2015. Freebase authors believed that supporting Wikidata was the best decision to support the development of an open, collaborative knowledge base. Wikidata became Freebase successor, and the mapping from Freebase to Wikidata was created [48]. Nevertheless, not all information was migrated because Wikidata community was very eager to have references for their statements, which Freebase was often lacking. Additionally, the licences under which the datasets were published differed, which reduced the number of possible mappings too.

3.2.5 YAGO

YAGO [30] – Yet Another Great Ontology is a huge semantic knowledge base, derived from Wikipedia WordNet and GeoNames. Researchers from the Max Planck Institute for Computer Science in Saarbrücken developed YAGO as well as our dataset. Currently, YAGO contains more than 10 million entities, and it also includes more than 120 million facts about these entities. The interesting point is that the fact correctness of YAGO has been manually evaluated, proving a confirmed accuracy of 95 %

3.3 RDF

RDF [33] which stands for Resource Description Framework is a standardised framework for representing information in the Web. It has been recommended as a standard by The World Wide Web Consortium (W3C) which is the leading international standards organisation for the World Wide Web. Data in RDF are stored in the form of triplets. Each triple consists of a subject, a predicate and an object. Such triples are called statements. A simple statement can be visualised by a node and directed-arc diagram as can be seen in 3.1, in which each triple is represented as a node-arc-node link.

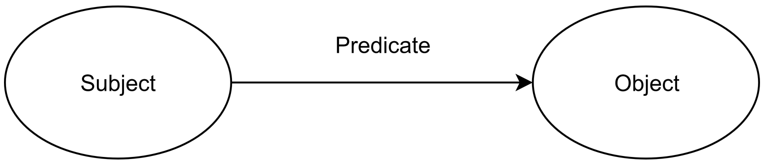

Figure 3.1: RDF triple diagram

A set of statements is called an RDF graph. The nodes of an RDF graph are its subjects and objects. Oriented edges are predicates which denotes a relationship between nodes and they point from the subject to the object of the predicate. Everything except blank nodes and literals, which are atomic values from which property cannot lead, is unambiguously identified with internationalised resource identifier (IRI).

RDF statements using full IRIs are lengthy and unclear. XML namespaces which point to the specific vocabularies of the local property names are used to help readability of RDF statements. They are defined at the beginning of the document where we define a mapping between prefix and a namespace IRI. Thanks to the XML namespaces, individual IRIs can be afterwards written in the form of *prefix:localname*, which makes RDF statements shorter and more readable. An example is shown in figure 3.2 Moreover, namespaces have to be uniquely identified with IRIs too. Therefore, same local resource name with different namespace means different resource in the whole graph while with same namespace it points globally to the same resource.

3.4 SPARQL

Now that we have a framework for storing information, we would like to have a way to query the data. W3C suggests standardised language for querying data in RDF format called SPARQL [52] which is a recursive acronym for SPARQL Protocol and RDF Query Language.

SPARQL is a graph-matching query language. Therefore querying, given data source D and query q, is essentially just the process of mapping patterns which occur in q against D. The results of such query are the values obtained from this matching [49]. The syntax of SPARQL is very similar to the SQL syntax which can be seen in the following example.

```
DATA:
@prefix foaf:       <http://examplens.com/foaf/2018/> .

_:a  foaf:name       "John"@EN.
_:a  foaf:name       "Jan"@CZ.
_:a  foaf:mbox       <mailto:john@sparqlexample.com> .

QUERY:
PREFIX foaf: <http://examplens.com/foaf/2018/>
SELECT ?name ?mbox
    WHERE {
        ?x  foaf:name  ?name .
        ?x  foaf:mbox  ?mbox .
             FILTER ( lang(?name) = "EN" )
          }

RESULT:
{
"head": {"vars": [ "name" , "mbox" ]} ,
"results": {
  "bindings": [
      {
    "name": { "type": "literal" , "value": "John"@EN },
    "mbox": { "type": "literal" , "value": "<mailto:john@sparqlexample.com>" }
      }
    ]
  }
}
```

Figure 3.2: SPARQL example

3.5 Datasets

In our work, We used two English datasets, CoNLL-2003 and AIDA CoNLL-YAGO. Here, we will briefly describe both of them.

3.5.1 CoNLL-2003 Dataset

Originally ConLL-2003 is a dataset provided for participants of the CoNLL-2003 shared task [63] which concerned language-independent named entity recognition. However, nowadays it serves as a standard benchmark dataset for NER. It consists of Reuters articles between August 1996 and Augst 1997. The dataset is split into three files, a training file, a development file and a testing one. The learning methods should be training with the training part, while the hyperparameters should be learned with the development one. The testing part is used for evaluation.

A data format is consistent as each file consists of lines with four fields: the word, its part-of-speech tag, its chunk tag and its named entity tag. Individual sentences are separated by empty lines. Out of three given word labels, we will be concerned only about the last one, named entity tag. Named entity tag is formed of an IOB1 tag and an additional tag signalising whether an entity has a meaning of person (PER), organisation (ORG), location (LOC) or miscellaneous name (MISC). Inside-outside-beginning (IOB) format is a standard labelling scheme for named entity chunks. In IOB1, 'I' is a token inside a chunk, 'O' is a token outside a chunk and 'B' is the beginning of chunk immediately following another chunk of the same Named Entity. We will also mention IOB2 format because it is used in the second dataset. IOB2 is same as IOB1, except that a 'B' tag is given for every token, which exists at the beginning of the chunk [34].

3.5.2 AIDA CoNLL-YAGO Dataset

As can be seen from the name, AIDA CoNLL-YAGO [30] named entity linking dataset is based on the original CoNLL-2003 named entity recognition dataset. It was created by Hoffart et al. because standard benchmark dataset for EL was missing. It contains similar Reuters articles and even though dataset is a single file it still preserves split into the three parts thanks to the numbering of documents. Each document starts with a line: "-DOCSTART- (<docid>)". Similarly to the CoNNL-2003 dataset, each line contains one word and sentences are separated by empty line too. The format of a line is following.

- column 1 is the token
- column 2 IOB2 tag, blank if O
- column 3 is the full mention used to find entity candidates
- column 4 is the corresponding YAGO2 entity if there is one or –NME–
- column 5 is the corresponding Wikipedia URL of the entity
- column 6 is the corresponding Wikipedia ID of the entity
- column 7 is the corresponding Freebase MID if there is one

```
EU          NNP   I-NP  I-ORG
rejects     VBZ   I-VP  O
German      JJ    I-NP  I-MISC
call        NN    I-NP  O
to          TO    I-VP  O
boycott     VB    I-VP  O
British     JJ    I-NP  I-MISC
lamb        NN    I-NP  O
.           .     O     O
```

Figure 3.3: CoNLL-2003 example sentece

```
EU    B     EU     --NME--
rejects
German      B      German     Germany    http://en.wikipedia.org/wiki/Germany   11867 /m/0345h
call
to
boycott
British     B      British    United_Kingdom   http://en.wikipedia.org/wiki/United_Kingdom 31717 /m/07ssc
lamb
.
```

Figure 3.4: AIDA CoNLL-YAGO example sentece

3.6 Evaluation Metrics

For the evaluation of our task, we use the following standard metrics where TP is the number of predicted entities that are in fact entities, FP is the number of predicted entities that should not be labelled as entities, TN is the number of words that are not predicted as entities and in fact are not entities and finally FN is the number of words which should be labelled as entities but are not.

- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$
- $Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$

It should be noted that there are several options on what to take as entity match. We could either take in account only correct tag and disregard a span of the entity or vice versa, or we can consider both span and tag of the entity. For the comparability with current results in NER, we will use the exact match where we consider both factors.

In NED part, the manual tagging is needed to evaluate the performance of our model. Due to the limited time, we decided not to manually tag the whole dataset but rather estimate our performance using smaller sample dataset. In this part, we used accuracy as an evaluation metric since it is used for evaluation of the methods with which we would like to compare our method.

Chapter 4

Theorethical Analysis

In this section, we will cover neural networks [26] (NN), because we will utilize them in our approach. Furthermore, most of the state-of-the-art results in both NER and NED are using neural networks too. We will describe main principles of NN, and we will introduce basic types of NN with their usage in NLP.

4.1 Neural Networks

One of the main things that people are trying to understand is the way how human brain works. This persistent effort to understand how we think was one of the reasons why neural networks were developed. A neural network is a machine learning algorithm inspired by a human brain. It takes a basic building block of the brain, biological neuron, and models it mathematically to an artificial neuron. A comparison of a biological neuron and an artificial neuron can be seen in figures 4.1 and 4.2.

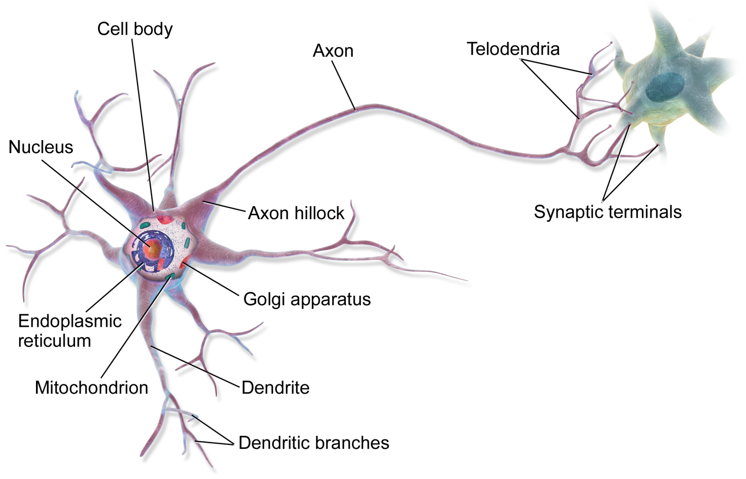

Figure 4.1: Biological neuron [8]

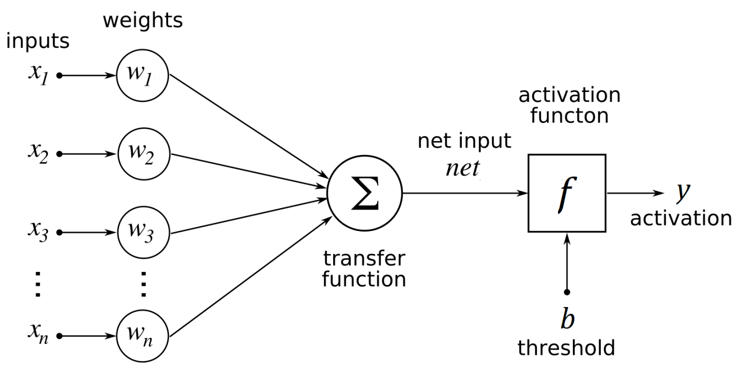

Figure 4.2: Artificial neuron [12]

4.1.1 Perceptron

Perceptron [54] is equivalent to a single neuron. Similarly to a biological neuron, an artificial perceptron neuron either fires a signal or not. Thus, it can be viewed as a binary classifier. The weighted sum of inputs is taken as an input to the activation function. In case of the perceptron, the activation function is called step function, and it has the following form:

$$f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

However, sometimes we would like a neuron to activate only if the input sum is greater than a certain threshold, not just to be limited to a zero threshold. That is why a bias term b is introduced as another input to the perceptron. For mathematical convenience, we often add input $x_0 = 1$ and weight $w_0 = b$ instead of bias term b so that input to the activation function can be taken as a dot product of weight vector $\mathbf{w}$ with input vector $\mathbf{x}$, see the following equation.

$$\mathbf{w}^T\mathbf{x} + b = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b = w_01 + w_1x_1 + \cdots + w_nx_n = \tilde{\mathbf{w}}^T\tilde{\mathbf{x}}$$

As was mentioned earlier, perceptron can be viewed as a binary classifier. Hence, it can be interpreted as a yes-no answer to a question. Some types of more difficult questions can be answered by layering the perceptrons into a network which is called multilayer perceptron [55] (MLP). However, we cannot model everything with just perceptrons.

In order to have more complex models that are capable of solving complicated tasks, we would like to measure the strength of a neuron activation, not just to be limited strictly to 0 and 1. Therefore, new non-linear functions 4.3 are introduced. In our thesis, we will work with the following non-linear functions.

Sigmoid

$$\sigma(x) = \frac{e^x}{1 + e^x}$$

Hyperbolic tangent

$$\tanh(x) = 2 \cdot \sigma(2x) - 1$$

Rectified Linear Unit (ReLU)

$$relu(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Softmax

$$softmax(x)_j = \frac{e_j^x}{\sum_{k=1}^{K} e_k^x} \text{ for } j \in \{1, \ldots, K\}$$

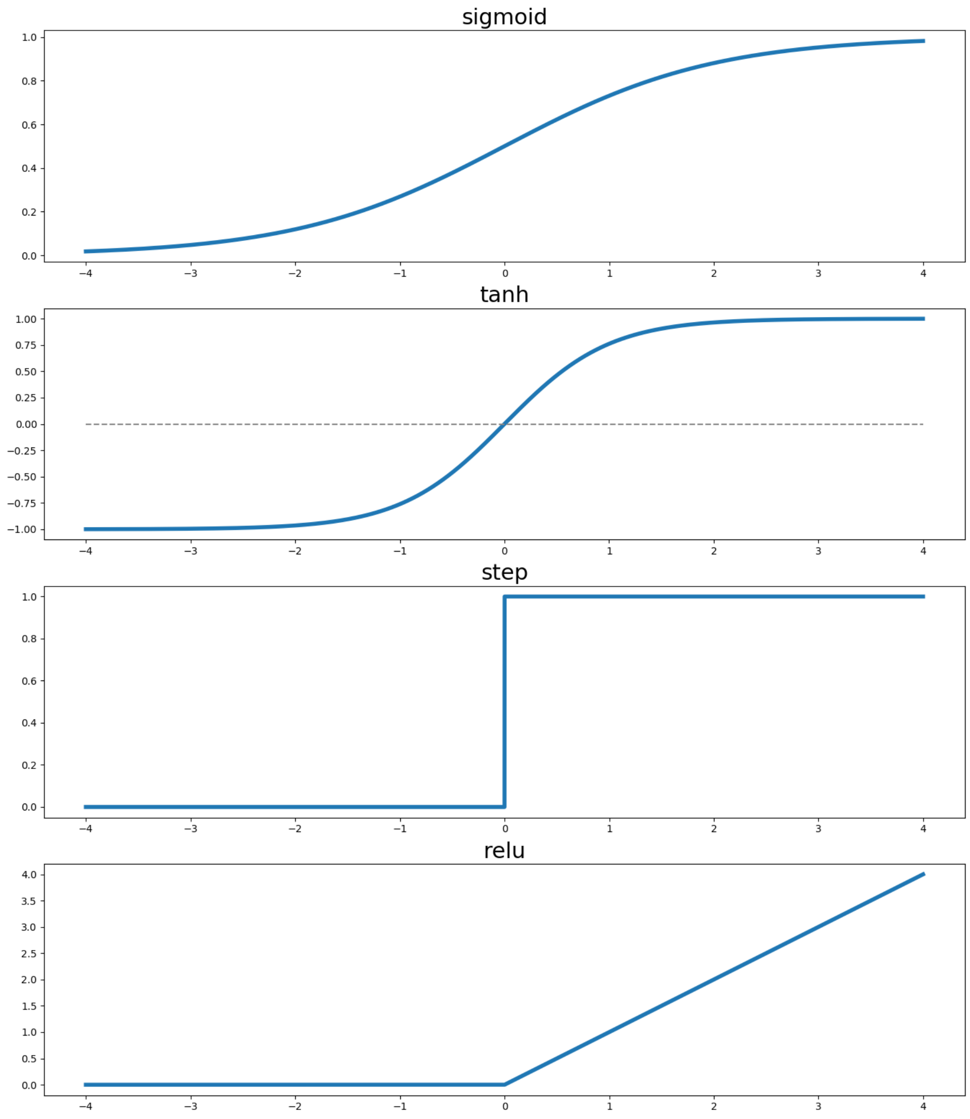

Figure 4.3: Activation functions

4.1.2 Fully Connected Neural Networks

A basic neural network has a form of several stacked layers of neurons. The first layer is called input layer, the last one is called output layer, and the layers in between are referred to as hidden layers. A simple architecture is visualized in figure 4.4.

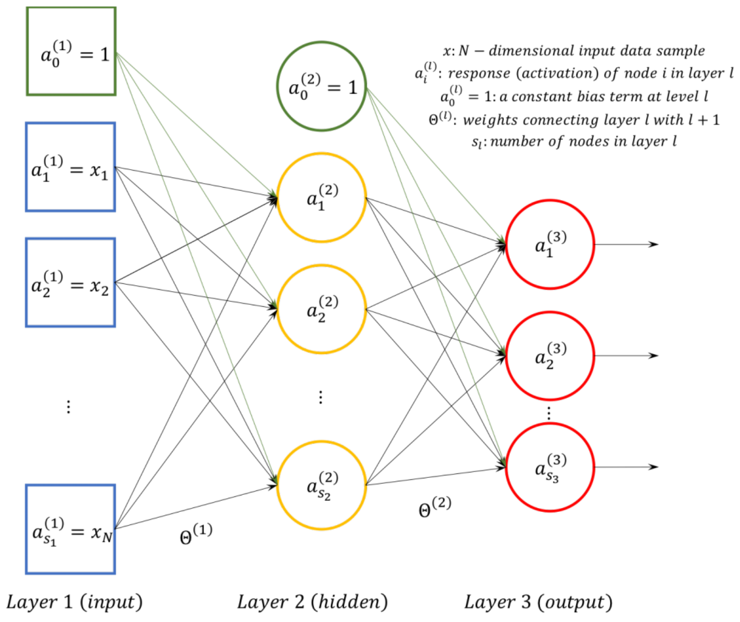
x: N – dimensional input data sample
$a_i^{(l)}$: response (activation) of node i in layer l
$a_0^{(l)} = 1$: a constant bias term at level l
$\Theta^{(l)}$: weights connecting layer l with $l+1$
s_l: number of nodes in layer l
$a_0^{(1)} = 1$
$a_1^{(1)} = x_1$
$a_2^{(1)} = x_2$
$a_{s_1}^{(1)} = x_N$
$a_0^{(2)} = 1$
$a_1^{(2)}$
$a_2^{(2)}$
$a_{s_2}^{(2)}$
$a_1^{(3)}$
$a_2^{(3)}$
$a_{s_3}^{(3)}$
$\Theta^{(1)}$
$\Theta^{(2)}$
Layer 1 (input)
Layer 2 (hidden)
Layer 3 (output)

Figure 4.4: Fully connected neural network [47]

As we can see, each neuron in one layer is connected to all neurons in the following layer. This is the reason why we refer to this kind of network as a fully connected neural network. Also, the neurons in one layer are only connected to the following layers, and there is no connection backwards, this type of NN is called feedforward NN.

In the previous section, we showed that input of non-linear function in case of a single neuron could be written conveniently as a dot product. Generally, we can write the output of the neural networks only in terms of matrices, vectors. Given the following definition

$$f(\mathbf{z}) = (f(z_1), f(z_2), \ldots, f(z_n))$$

the output of our network in example could be written like:

$$\mathbf{a}^{(3)} = f(\theta^{(2)}\mathbf{a}^{(2)}) = f(\theta^{(2)}[1, f(\theta^{(1)}\mathbf{a}^{(1)})])$$

Additionally, an important fact that the activation function has to be non-linear can be clearly seen in this written form. If the activation function were linear, the whole output would be just result of nested matrix multiplications. However, thanks to the linearity we would be able to rewrite multi-layer network to single-layer one with the same results.

4.1.3 Training

So far we have just explained how to calculate the output of neural networks. In this section, we focus on their training. The desired state is that given the input, the correct output neuron would activate. We can influence this behaviour by changing the weights in each layer. The whole training is in fact just a process of learning the appropriate weight matrices. This can be formulated as an optimization problem as we are trying to minimize an objective function, sometimes also called cost function, which measures how much the current output is different from the desired one. Ideally, we would like to have equality between network output and desired output. Therefore, we are minimizing an objective function, and ideal state is when the value is 0. The most known objectives functions are a mean square error, information divergence, categorical cross entropy.

4.1.3.1 Gradient Descent

The most widely used algorithm for training neural networks is gradient descent. Gradient descent is a first order iterative algorithm for finding local minima. An iteration is calculated as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \lambda_k \nabla f(\mathbf{x}_k)$$

Where $\nabla f(\mathbf{x}_k)$ is the gradient which determines the direction of the next step and λ_k is a learning rate which determines the size of the step in the direction of the negative gradient.

Gradient descent is a representative of descent methods where objective function monotonically decreases. Therefore it holds that

$$f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$$

The main advantage is that this algorithm always converges to the local minima because we always have step in a descent direction which can be observed from

$$f'(\mathbf{x}_k)(-\nabla f(\mathbf{x}_k)) = -f'(\mathbf{x}_k)f'(\mathbf{x}_k)^T < 0$$

On the other hand, it can handle only differentiable functions which is the reason why we require activation functions to be differentiable. Moreover, it often converges relatively slowly.

When it comes to the speed of the convergence, the natural question is why not to use second-order methods for finding local minima. However, these methods require a calculation or at least approximation of a Hessian matrix and also a matrix inverse is needed. These calculations are computationally expensive and also the memory requirements for storing Hessian are $O(N^2)$. Thus, the second-order methods are unfeasible for large neural networks with many parameters. Other possibilities for training are, for example, EM algorithm or genetic algorithms. However, gradient descent is still a go-to algorithm when it comes to the training mainly because of backpropagation algorithm which will be described in the following section.

4.1.3.2 Backpropagation

Backpropagation [27] is an algorithm which efficiently computes the gradient of the given objective function with respect to all weights and biases. However, instead of a naive approach of calculating every partial derivative separately, it uses previously calculated results to compute the new ones. In fact, it is just a repetitive use of the chain rule in derivation. Intuitively, given an objective function, we know how we want to change the final output to minimize the cost. Now, given how we want to change the output of the final layer, we look at how the input of the previous layer influence this one. This idea is used repetitively starting from the end until we get to the first layer. That is where the name of the algorithm came from.

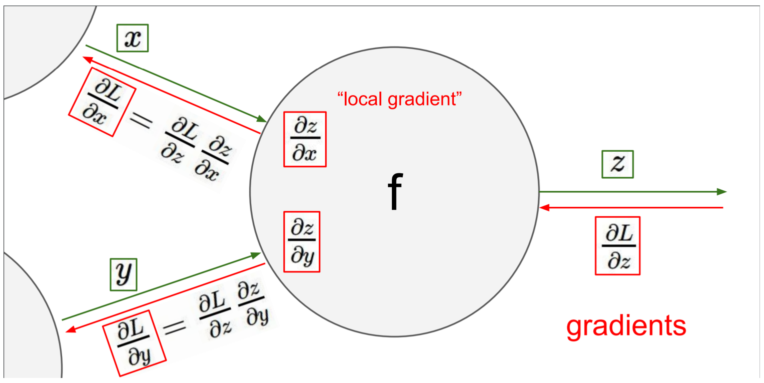
x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
$\frac{\partial z}{\partial y}$
z
$\frac{\partial L}{\partial z}$
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$
gradients

Figure 4.5: Backpropagation diagram [23]

4.1.3.3 Stochastic Gradient Descent

Despite having an efficient algorithm for calculating gradient, we have to compute gradients for all the training examples to make one weight update. Thus, the more training examples, the more time it takes to make one update of the weights in the descending direction. However, to be able to model difficult problems require we generally require large training sets which makes a standard gradient descent inapplicable in practice.

This problem is tackled by a modification of gradient descent called stochastic gradient descent [7]. The main idea behind the stochastic gradient descent is to take a smaller portion of the data to estimate the gradient instead of calculating the gradient over all training examples. This estimate cannot guarantee that we will always make descent step, but it generally converges.

4.1.3.4 Vanishing and Exploding Gradients

Calculating a gradient is not the only challenge that we face during the training of neural networks. The other big problem related to gradient is a vanishing gradient [28]. It is a problem where gradients with respect to weights get too small to effect weight update at all.

The reason why it occurs is that while propagating error through n layers, multiplication with a derivative of the activation function $f'(z_i)$ is applied n times. However, traditional activation functions such as the sigmoid or the hyperbolic tangent have gradients in the range (0, 0.25) and (0,1) respectively. Hence, a repetitive multiplication with a number smaller than 1 has the effect of exponential decrease with respect to the number of layers n while the front layers train very slowly.

One possible solution for the vanishing gradient is ReLU activation function. The derivative of ReLU for positive inputs is always one. Therefore, the backpropagated error does not change while multiplying by the derivative. Other solutions GRUs and LSTMs which will be described later.

In contrast to vanishing gradient, there is also an exploding gradient [45]. Here, the gradient with respect to weights get too large which results in very large updates to neural network model weights during training. This makes a model unstable and unable to learn from the training data as a large update overshoot a downhill step. In the worst case, a value of the weight update approaches infinity which makes further learning impossible. However, in this case, there is a relatively simple solution which is called gradient clipping. The main idea is to limit gradient values so that they cannot exceed a certain threshold.

4.1.3.5 Overfitting

The last problem concerning training which we will examine is overfitting [9]. Overfitting occurs when the model does not generalize well enough and does not perform well in practice, despite fitting the training data well. This is a natural tendency caused by way of the neural network training.

Overfitting is usually detected by having a validation set. Usually, if the classification error and loss in validation set raises while it drops in the testing set, the overfitting has occurred. This directly leads us to the technique called early stopping [71]. The idea is to evaluate the network performance on the validation set and save the copy of the model only if it outperforms the previous best one.

Another technique which is widely used to reduce overfitting is dropout [60]. This simple technique randomly drops units which speed up the training while making the model more robust. There are plenty other techniques such as regularization, cross-validation or ensembling [35], which are used to reduce the overfitting, yet the simple idea as dropout generally reduces overfitting well.

4.1.4 Convolutional Neural Networks

Convolutional neural networks [15] (CNNs) are networks that take into account the spatial structure of the input. Unlike in fully connected neural network, we connect only several input neurons to the neuron in the following layer. Moreover, we do not have separate weights for every single neuron, but we share weights for the same filter. Furthermore, we also apply pooling layers, which reduce the dimension of input while it preserves the most relevant information. All these ideas rapidly reduce the number of parameters we have to learn in contrast to fully connected networks. Hence, we are enabled to build deeper networks.

A convolutional filter is easily interpretable for images where we can think of it as a sliding window function over the matrix with pixel values which represents an image. In NLP our matrix is going to be a sentence represented by word embeddings where each word is one row.

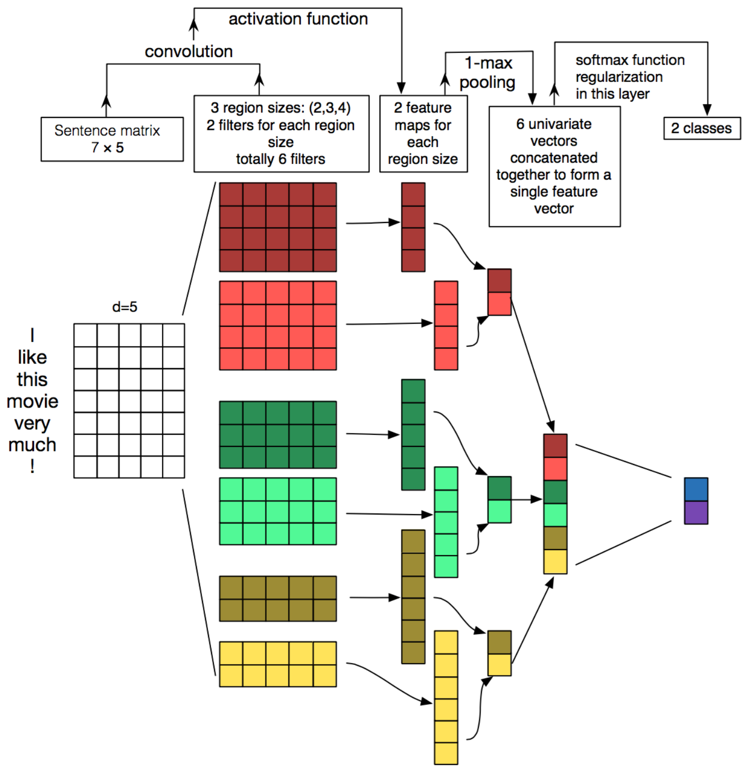
activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

Figure 4.6: Convolutional neural network in NLP [72]

CNNs are in nature more suitable for computer vision because the concept of the local detection of some feature in the image seems to be a logical idea. Local filters, shared weights and pooling helps to capture the local information but does not care that much about where something appeared. Thus, the use of CNN is not that straightforward when it comes to NLP, where we typically care about the word order [31] since just a change in position of a word in the sentence can change the meaning. However, words close to each other are often semantically related, but it should be noted that it is not always the case.

In NLP CNNs are typically used in classifications tasks [32], such as sentiment analysis[21] or topic categorization [66] because here we rather focus on the content than on order. On the contrary, in the tasks where we put a great emphasis on the word order like sequence tagging, part-of-speech (PoS) tagging or entity recognition, pure CNN architectures are generally not used. Instead of that, CNNs are utilized as a part of the architectures in such problems, and they serve as a feature extractor [56, 11].

4.1.5 Recurrent Neural Networks

Traditional language models [61] for sequential data are based on predicting the current word on n previous words. This assumption that current word relies only on a window of previous words instead of all previous words is incorrect. However, this simplification was working pretty well in practice. On the other hand, the memory requirements grow very big with the number of previous words that a current prediction depends on.

Recurrent neural networks (RNNs) were introduced as a model with a theoretical capability of taking all previous information in the account. RNN are a natural architecture of neural networks for NLP problems since text or speech have sequential nature. Simple RNN model is depicted in figure 4.7. We can see that in RNN output does not depend only on current input but also on the previous outputs, whereas in previously mentioned fully connected and convolutional networks, output relies solely on current input. An important fact is that weights are shared through different time steps. Moreover, the memory requirements do not grow with the number of previous words but only depends on the number of words we are trying to predict.

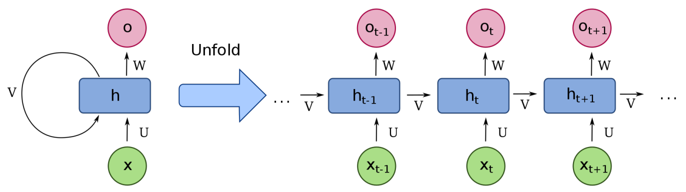

Figure 4.7: Recurrent neural network diagram [18]

The following equations hold for calculation of the hidden state and the output.

$$\mathbf{h}_t = \tanh(U\mathbf{x}_t + V\mathbf{h}_{t-1})$$
$$\mathbf{o}_t = softmax(W\mathbf{h}_t)$$

The main problem of RNNs is a struggle to learn long-term dependencies [46]. The problem is vanishing gradient again. This is because unfolded RNN is in fact just a traditional fully connected network as can be seen in figure 4.7. Hence, the problem of backpropagating the error to the layers further from the end is persisting in RNNs too. That is the reason why RNN architectures like GRUs and LSTMs, which are specially designed to persist the long-term dependencies, were introduced.

4.1.5.1 Gated Recurrent Units

Firstly we will introduce Gated Recurrent Units (GRUs) which are a simplification of LSTMs. They are a relatively new idea as they were introduced in 2014 by Chun et al [13]. In contrast to simple RNNs where a unit contains only the hyperbolic tangent nonlinearity, GRUs introduce two additional gates that influence how a neuron deals with current input and previous information. See figure 4.8.

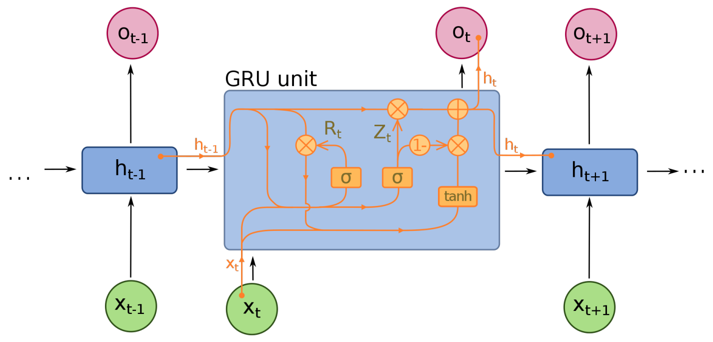

Figure 4.8: Gated recurrent unit [20]

The two gates are called update gate and reset gate, and they have following almost similar equations.

$$\mathbf{z}_t = \sigma(W^z \mathbf{x}_t + U^z \mathbf{h}_{t-1})$$

$$\mathbf{r}_t = \sigma(W^r \mathbf{x}_t + U^r \mathbf{h}_{t-1})$$

The gates differ in weight matrices, but more importantly, they differ in usage. Their usage can be explained because the following equations for the calculation of new memory content and final memory content holds.

$$\tilde{\mathbf{h}}_t = \sigma(W\mathbf{x}_t + \mathbf{r} \circ U\mathbf{h}_{t-1})$$

$$\mathbf{h}_t = \mathbf{z}_t \circ \mathbf{h}_{t-1} + (\mathbf{1} - \mathbf{z}_t) \circ \tilde{\mathbf{h}}_t$$

Reset gate is used to control how much past information should be forgotten. Intuitively if all values in r get close to zero, we ignore the information from the previous hidden state because it would be irrelevant in the future. Update gate has a function of determining how much of the past information needs to be passed to the output. We can see that if all values in z get close to one, we copy the information from the previous step and there is no problem with vanishing gradient. Furthermore, it can be easily seen that GRUs are a general model of RNNs. That is because if r is 1 and z is 0 then the h_t equation for GRUs simplifies to the h_t equation for RNNs.

4.1.5.2 LSTM

Long short-term memory [29] (LSTMs) networks are an even more complex recurrent model than GRUs. The surprising fact is that they were introduced already in 1997 which is much earlier then GRUs which is a simplification of LSTMs.

LSTMs introduce three gates instead of two which were introduced in GRUs. We refer to these gates as input, forget and output gate. Moreover, they also have one additional state which is called a cell state. Following equations holds for the LSTMs.

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(W^i\mathbf{x}_t + U^i\mathbf{h}_{t-1})\\
\mathbf{f}_t &= \sigma(W^f\mathbf{x}_t + U^f\mathbf{h}_{t-1})\\
\mathbf{o}_t &= \sigma(W^o\mathbf{x}_t + U^o\mathbf{h}_{t-1})\\
\tilde{\mathbf{c}}_t &= \tanh(W^c\mathbf{x}_t + U^c\mathbf{h}_{t-1})\\
\mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i_t} \circ \tilde{\mathbf{c}}_t\\
\mathbf{h}_t &= \mathbf{o}_t \circ \tanh(\mathbf{c}_t)
\end{aligned}
$$

Forget gate controls how much the network cares about the previous results. Input gate tells us how much we should care about the current state. This is a stronger mechanism than update gate in GRUs as we have two separate mechanisms to control the memory cell instead of a single update gate.

Another difference between LSTMs and GRUs is that LSTMs have additional cell state. The reason is that LSTMs also control how much information they want to expose to the output. In GRUs the output is equal to the final memory state, whereas in LSTMs the output is controlled by the output gate which decides what information is relevant for the current state.

If we continue with the comparison of GRUs with LSTMs from the performance aspect, there is no clear winner. Researchers have tried both models on various problems. Sometimes GRUs outperform LSTMs, but sometimes it is another way around. Generally, the performance is pretty similar while GRUs have the advantage of the simpler model which often results in quicker training. On the other side, LSTMs are more flexible and have more modelling power with more gates. Furthermore, they are more thoroughly explored as GRUs are relatively new.

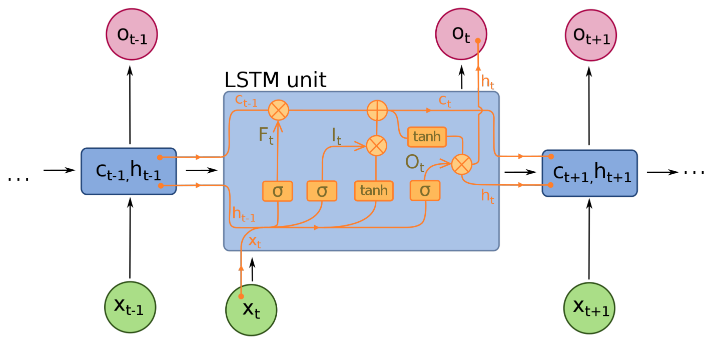

Figure 4.9: Long short-term memory unit [19]

Chapter 5

Implementation and Experiments

In this section, we go through the implementation of the proposed approach. Furthermore, we describe the experiments which had been done, as we move along the system architecture.

5.1 Named Entity Recognition

As we discussed in section 3.1, we decided to experiment with neural networks in NER. RNNs had reached state-of-the-art results in NER, and their popularity in NLP task is growing. Therefore, we tried to implement own recurrent neural approach in order to get a better understanding of the power of RNNs, instead of blindly using existing tagger.

We used Python3 programming language throughout the whole work as it is one of the most popular languages for machine learning. Python provides us powerful libraries for neural networks, and it can also handle SPARQL which we need later. The main library which we use is called keras which is a high-level deep learning library.

Our first task in EL pipeline is to recognize which entity mentions we should disambiguate at all. Thus, our first simplistic approach was to use IOB scheme described in section 3.5.1 and tag the input text.

Our baseline model consisted of untrained 100-dimensional word embeddings, one layer with 100 classic RNN units and an output layer with 3 neurons where each corresponded to one of the IOB tags. The activation function was softmax, starting learning rate was default 0.001 which was adapted during the training by the rmsprop optimizer. Furthermore, we clipped the gradients at the value of 5. The objective function was categorical cross entropy. The training was done one sentence by sentence as a natural solution of different sentence lengths. Additionally, we used early stopping during our training to reduce the overfitting.

Throughout the experiments with neural networks in NER, we incrementally expanded this baseline model and observed the changes in the results. In the following table, we can see the performance of our baseline model.

Model	**Precision**	**Recall**	**F1**
EMB(100) + RNN(100)	72.79%	74.76%	73.76%

Table 5.1: Baseline RNN model with IOB tags

These results looked promising, and we believed that adding additional layers, switching RNN units for GRUs or LSTMs and using pre-trained embeddings would improve the current results. Even though the sole IOB tags would be sufficient for the continuation of our EL model, we decided to switch to the same tagging model which was used in the original CoNLL2003 task so that the result of our system would be comparable. The following table shows the comparison o the results of the models labelled by sole IOB tags (I, O, B) and CoNLL tags (I-PER, I-LOC, I-ORG, I-MISC, B-PER, B-LOC, B-ORG, B-MISC, O).

Model	**Precision**	**Recall**	**F1**
EMB(100) + RNN(100) (IOB)	72.79%	74.76%	73.76%
EMB(100) + RNN(100) (CoNLL)	63.35%	48.97%	55.24%

Table 5.2: Baseline RNN model results with a different tag scheme

The decision to change the tagging scheme made the sequence tagging much more difficult because now instead of three tags we had nine. With more difficult task, the scores in evaluation metrics naturally dropped. We decided to update our network by using more complex GRU units instead of classic RNN units.

Model	**Precision**	**Recall**	**F1**
emb(100) + GRU(100)	62.64%	54.03%	58.02%

Table 5.3: Baseline GRU model

We had observed a slight improvement which is not surprising since GRUs are a general version of RNNs. Afterwards, we decided to use LSTMs to see whether there would be the difference in performance between GRUs and LSTMs which was discussed in section 4.1.5.2

Model	**Precision**	**Recall**	**F1**
EMB(100) + LSTM(100)	60.85%	58.42%	59.61%

Table 5.4: Baseline LSTM model

We observed that there is no clear winner regarding our evaluation metrics which supports what was said in theoretical section 4.1.5.2. We stopped experimenting with simple RNN units and continued with a comparison of stronger GRUs and LSTMs while we kept changing our model. Next update which we made was adding a convolutional layer with 64 filters and the filter length of 5 words.

Model	**Precision**	**Recall**	**F1**
EMB(100) + CNN(64,5) + GRU(100)	65.07%	62.04%	63.52%
EMB(100) + CNN(64,5) + LSTM(100)	63.79%	61.27%	62.5%

Table 5.5: Models utilizing CNNs

We noticed that GRUs got better results than LSTMs this time. We assumed that it was because of GRUs having fewer parameters to train resulting in quicker training. We decided to add a bidirectional layer which is used in the most of the NED neural approaches. Thanks to the bidirectional layers the current state in the network can reach both the past input but also the future input information as we pass sentence in left-to-right and right-to-left word order.

Model	**Precision**	**Recall**	**F1**
EMB(100) + CNN(64,5) + BGRU(100)	64.61%	58.97%	61.66%
EMB(100) + CNN(64,5) + BLSTM(100)	66.94%	64.08%	65.48%

Table 5.6: Models utilizing bidirectional layer

Additional GRU layer did not improve the performance of the current model. On the other hand, we observed opposite effect after adding additional LSTM layer. We supposed that higher modelling power of LSTMs was the reason why the difference in this particular experiment occurred. Moreover, we think that this might be the reason why bidirectional lstms are used in most state-of-the-art approaches while bidirectional grus not.

In the next update, we applied dropout layer with the 50% dropout probability to reduce the overfitting of our model. Besides that, we noticed that training slowed down after using bidirectional layer so we thought dropout would speed up the training.

Model	**Precision**	**Recall**	**F1**
EMB(100) + CNN(64,5) + DROP(0.5) + BGRU(100)	67.32%	66.14%	66.73%
EMB(100) + CNN(64,5) + DROP(0.5) + BLSTM(100)	66.01%	58.08%	61.79%

Table 5.7: Models after applying dropout

The next update was to use pre-trained embeddings. We used 100-dimensional Skip-gram embeddings provided by Lample et al. [36] that fitted to our previously designed model without a need to change it.

Model	**Precision**	**Recall**	**F1**
SKIP(100) + CNN(64,5) + DROP(0.5) + BGRU(100)	67.44%	70.56%	68.97%
SKIP(100) + CNN(64,5) + DROP(0.5) + BLSTM(100)	64.52%	61.66%	63.06%

Table 5.8: Models after adding pre-trained embeddings

This time we tried changing the number of convolutional filters and changing the number of neurons in recurrent layers to see whether the problem was in chosen hyperparameters.

Model	**Precision**	**Recall**	**F1**
SKIP(100) + CNN(32,5) + DROP(0.5) + BGRU(100)	62.13%	59.13%	60.59%
SKIP(100) + CNN(64,5) + DROP(0.5) + BGRU(100)	66.82%	63.5%	65.12%
SKIP(100) + CNN(128,5) + DROP(0.5) +BLSTM(200)	63.88%	57.55%	60.55%
SKIP(100) + CNN(64,5) + DROP(0.5) + BLSTM(200)	64.52%	61.66%	63.06%

Table 5.9: Models with varIous hyperparameters

After no further progress in performance together with limited computational and time resources, we decided to use an existing NER model in order to progress with EL task. The reason behind our choice was that NED is highly dependent on the results of the previous parts in EL pipeline. We used implementation of a neural model described by Lample et al. [36] which reached F1 score **90.94**% on AIDA CoNLL-YAGO dataset using previously examined embeddings, CNNs and BLSTMs. The final comparison of our architectures and results can be seen in the following table.

Model	Precision	Recall	F1
EMB(100) + RNN(100)	63.35%	48.98%	55.24%
EMB(100) + GRU(100)	62.64%	54.03%	58.02%
EMB(100) + CNN(64,5) + GRU(100)	65.07%	62.04%	63.52%
EMB(100) + CNN(64,5) + BGRU(100)	66.61%	58.97%	61.66%
EMB(100) + CNN(64,5) + DROP(0.5) + BGRU(100)	67.32%	66.14%	66.73%
SKIP(100) + CNN(64,5) + DROP(0.5) + BGRU(100)	66.44%	70.56%	**68.97**%
SKIP(100) + CNN(32,3)+ DROP(0.5) + BGRU(100)	62.13%	59.13%	60.59%
EMB(100) + LSTM(100)	60.85%	58.42%	59.61%
EMB(100) + CNN(64,5) + LSTM(100)	63.79%	61.27%	62.50%
EMB(100) + CNN(64,5) + BLSTM(100)	66.94%	64.08%	65.48%
EMB(100) + CNN(64,5) + DROP (0.5) + BLSTM(100)	66.01%	58.08%	61.79%
SKIP(100) + CNN(64,5) + DROP (0.5) + BLSTM(100)	60.89%	59.65%	60.26%
SKIP(100) + CNN(128,5) + DROP(0.5) +BLSTM(200)	63.88%	57.55%	60.55%
SKIP(100) + CNN(64,5) + DROP(0.5) + BLSTM(200)	64.52%	61.66%	63.06%

Table 5.10: Results of all models

In the end, our best neural model reached **68.97**% F1 score while the neural model that we decided to use reached **90.94**% F1 score. The are many possible factors that influenced the performance of our models. We argue that one of the possible reasons for such different results using almost similar techniques was our choice to train model sentence by sentence. This straightforward approach to solve the problem of various sentence length probably slowed down training too much. The other approach that could have selected was padding sentences to the same length so that we could choose the batch size. However, the disadvantage of the padding is bigger memory consumption since there would be many sentences with additional zeros inside.

5.2 Candidate Generation

As was stated in section 3.1, we use label-lookup for candidate generation part of EL pipeline. Label-lookup is inspired by CrossWiki Search [3, 59] which uses a dictionary[1] where each entry consists of a string s, Wikipedia URL u and a probability $P(U = u|S = s)$ of the URL u given the string s. Based on this data, label-lookup generates entity candidates given a string, and conditional probabilities are used for scoring. The second idea used in

[1] <http://nlp.stanford.edu/data/crosswikis-data.tar.bz2/dictionary.bz2>

label-lookup is FuzzySearch [70] which aims to find the correct entity even if the label is not spelt properly.

Label-lookup is utilized as a standard component of Alquist and YodaQA systems. Since one of the objectives of our work is to create EL system that could be possibly used in these systems, we found it reasonable to use a component of systems that we would like to extend. Additionally, by using label-lookup, our EL system can be used on general datasets while by using existing candidate datasets for AIDA CoNNL-YAGO would restrict us only to the particular dataset. These are the main reasons why we opted for label-lookup instead of using existing candidate datasets, even though the results on the AIDA CoNLL-YAGO dataset could be possibly better with candidates created specifically for this dataset.

5.3 Named Entity Disambiguation

In the last part of our EL pipeline, we focused on NED. We connected all previous parts of the system in order to get input for NED. Afterwards, we used Freebase and Microsoft Concept Graph to generate a clarifying question to disambiguate the entities.

Firstly we selected sentences from test set which contained at least one entity with Freebase mid. Out of 3453 sentences from the test set of AIDA CoNLL-YAGO dataset, 2464 passed through this filter. Since our approach is based on user interaction, we had to select a reasonable number of sentences to create sample dataset for manual evaluation of clarifying questions. We sampled 100 sentences from 2464 possible test sentences and proceeded with our task.

We passed selected sentences to the NER model which returned tagged sentences in one output file. Given pairs of untagged and tagged sentences, we started generating clarifying questions for each pair. Untagged input sentence was passed to label-lookup which returned possible entity candidates matched to a certain substring from given sentence. Afterwards, we used tagged output sentence to filter which candidates match with tagged entity mentions.

5.3.1 Freebase approach

Given a set of entity candidates for each tagged entity mention, we used Freebase to find the typical features of each entity candidate. Based on this typical features, we created a question that should theoretically help with disambiguation. In the end, we decided to use *notable_for* property, which is single-valued and therefore uniquely describes what entity is generally known for. Furthermore, we also used *rdf_label* property. The reason was that over 72 million triples have this property and it provides us with a human-readable label of our entity candidate. The whole process of generating one clarifying question is shown in figure 5.1.

For a simple estimation of the performance, we restricted our performance estimation only on words that should be labelled as entity mentions. Out of such 247 entity mentions 147 were labelled by the correct entity. Therefore we estimated our accuracy as the fraction of correctly predicted entities divided by the number of entities. This gave us the estimated accuracy of 59.51%.

```
untagged input: "ANAHEIM AT PITTSBURGH"
tagged output: "ANAHEIM__B-ORG AT__O PITTSBURGH__B-LOC"
db_ids of filtered entity candidates: ['Anaheim,_California', 'AnaheimDucks']
corresponding Freebase mappings:['m.0k9p4', 'm.0jnpc']
ground truth from AIDA CoNLL-YAGO:['m.0jnpc']

generated question where suggestions contains \textit{rdf\_label}
and value of \textit{notable\_for} property given the entity candidate
Do you mean "ANAHEIM" like:
1. Anaheim notable for City/Town/Village?
2. Anaheim Ducks notable for Professional Sports Team?
```

Figure 5.1: Full process of question generation

Furthermore, we analysed our approach on the three possible outcomes. The first outcome is that the correct entity was among the entity candidates returned by label lookup and the clarifying question helped to disambiguate correctly. This is the case of the question in example 5.1. In this case, we observed that Freebase approach generated well-posed question as both *rdf_ label* and *notable_for* property directs us to the correct ground truth.

The second outcome is that the correct entity is among the entity candidates returned by label-lookup, but the question could not distinguish entity candidates well enough. This is the case shown in figure 5.2.

```
input sentence: The Wallabies ran in five tries with
 Campese , who has retired from test rugby after...
 ['David_Campese', 'Terry_Campese']
['m.05djx3', 'm.07_jlc']
ground truth from AIDA CoNLL-YAGO:['m.05djx3']
Do you mean "Campese" like:
1. David Campese notable for Athlete
2. Terry Campese notable for Rugby Player
```

Figure 5.2: Freebase approach on the second type of outcome

On the contrary with the previous outcome, in this case, Freebase approach was not sufficient. The first name which was not present in the original sentence was the only difference in *rdf_ label.* More importantly, *notable_for* values athlete and rugby player told us that the person was known for a sport which was not discriminative enough.

The last possible outcome is when the correct entity is not in the candidate entities at all. This can be seen in the first example where there was no candidate for recognized mention Pittsburgh. However this case cannot be solved in NED part but rather in the previous candidate generation part, so we disregard it in the evaluation from the point of NED approach.

5.3.2 Microsoft Concept Graph Approach

Instead of using Freebase for finding the typical property of entity candidate we tried to use Microsoft Concept Graph [68, 10] for question generation while we still keep freebase *rdf_label*. The main purpose was to find out whether MCG can return better representative properties then freebase. We took into account only the first two possible outcomes which can be solved by NED part and compared the Microsoft Concept Graph based question and freebase based question.

In the first case, given the same example as in Freebase approach, the second approach yielded the following question.

```
 Do you mean "ANAHEIM" like:
1. "Anaheim, California" with concept "-"
2. "Anaheim Ducks" with concept "team"
```

Figure 5.3: Microsoft Concept Graph approach on the first type of outcome

We observed that MCG did not return any concept for Anaheim. This was not the only case when MCG did not return concept given candidate. In fact, most of the candidates were without the concepts, which is mainly because MCG matches concept based on our entered entity label while freebase approach matches the property by unique mid. Furthermore, in the case where the concept was returned, it generally carrying the same information as freebase concept which can be seen in this particular case too. California as a location while returning team concept for Anaheim Ducks

The second outcome taken in the account was when the correct entity was among the candidates, and we were not able to disambiguate. If we again compared the question generated by Freebase approach 5.2 and the question generated by Microsoft Concept Graph 5.4, we saw that the returned concepts were less descriptive than the Freebase *notable_for* property.

```
input sentence: The Wallabies ran in five tries with
 Campese , who has retired from test rugby after...

Do you mean "Campese" like:
1. "David Campese" with the concept "great"
2. "Terry Campese" with the concept "player"
```

Figure 5.4: Microsoft Concept Graph approach on the second type of outcome

Based on our experiments with two NED approaches, we consider the Freebase approach as a preferable choice. The reason is that the Freebase approach performed relatively well and was able to generate well-posed questions to disambiguate entity candidates. Moreover, the Microsoft Concept Graph approach did not help with disambiguation of cases that the Freebase approach could not solve, which means that the second approach did not have any additional value. We argue that this was caused by the fact that Microsoft Concept Graph returned concept based on the textual match with entered candidate string while Freebase was matching properties based on unique mid.

Chapter 6

Conclusion

The aim of our project was to create entity disambiguation system based on dialogue. This objective was motivated by the small existing number of entity disambiguation systems based on dialogue and simultaneously by the growing number of devices that could be used for interaction with a user.

To gain a better understanding of a complex task such as entity linking, we firstly researched current state-of-the-art methods. Afterwards, we investigated existing knowledge bases, the ways of querying them and we also examined standard datasets for named entity recognition and disambiguation. Furthermore, we analyzed fundamental theoretical concepts needed for the implementation of our approach.

Based on the previous research, we proceeded with the implementation of subtasks in entity linking namely, named entity recognition, candidate generation and named entity disambiguation.

In named entity recognition, we followed the current trend of using neural networks for automatic feature extraction. Even though that we experimented with techniques such as embeddings, CNNs and BLSTMs which are used in the current state-of-the-art approaches, we were not able to reach similar performance. We assume that the decision to train one sentence at the time, which slowed convergence, together with limited computational resources is possibly one of the reasons behind the difference in performance.

In the end, we decided to use existing neural model for entity recognition. This model reached the F1 score of $90,94\%$ while our best model reached $68,97\%$. However, despite such difference in performance, we were still able to observe the effect of individual changes in model architecture while the most significant boost in performance was seen after adding pre-trained embeddings and bidirectional layer.

After using existing label-lookup for candidate generation instead of opting for existing entity candidate datasets, we finally moved to the disambiguation part. In this part, we generated clarifying questions given a set of candidate entities. We used two approaches, the first one based on Freebase and the second one using Microsoft Concept Graph. By combining all previous subtasks, we achieved the main objective of our work, which was to implement entity linking and disambiguation system based on dialogue.

During the evaluation of generated questions, we noticed that one of the limitations of our approach is disambiguating model with relatively similar concepts as both Freebase approach and Microsoft Concept Graph approach struggled with finding discriminative properties. Moreover, we consider the necessity of manually checking the generated questions as another limiting factor of the possible training.

If we compare our estimated accuracy of 59.51% with the state-of-the-art accuracies which exceed 90% we see that the existing approaches outperform our dialogue-based approach. On the other hand, we argue that there is a lot of space for possible improvements of dialogue based linking systems as they are not thoroughly explored.

6.1 Future Work

In our particular case, the possible improvements can be aimed at each subtask of the entity linking pipeline. Based on the results obtained during the evaluation of our system, we would like to target the candidate generation step since we think that it would make the most notable change in overall performance of our approach. Another possible future work is creating voice interface instead of the current textual one. Last but not least future work is the integration of our entity disambiguation system into the existing Alquist chatbot.

Bibliography

[1] Amazon. wikipedia.org traffic statistics, Apr 2018. [Online; accessed 20-April-2018].

[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[3] Hannah Bast and Elmar Haussmann. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1431–1440. ACM, 2015.

[4] Petr Baudiš. Yodaqa: a modular question answering system pipeline. In *POSTER 2015-19th International Student Conference on Electrical Engineering*, pages 1156–1165, 2015.

[5] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia-a crystallization point for the web of data. *Web Semantics: science, services and agents on the world wide web*, 7(3):154–165, 2009.

[6] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM, 2008.

[7] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[8] BruceBlaus. Biological neuron, Apr 2018. [Online; accessed 24-April-2018].

[9] Rich Caruana, Steve Lawrence, and C Lee Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*, pages 402–408, 2001.

[10] Jianpeng Cheng, Zhongyuan Wang, Ji-Rong Wen, Jun Yan, and Zheng Chen. Contextual text understanding in distributional semantic space. In *ACM International Conference on Information and Knowledge Management (CIKM)*. ACM – Association for Computing Machinery, October 2015.

[11] Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*, 2015.

[12] Chrislb. Artificial neuron, Apr 2018. [Online; accessed 24-April-2018].

[13] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[14] Anni Coden, Daniel Gruhl, Neal Lewis, and Pablo N Mendes. Did you mean a or b? supporting clarification dialog for entity disambiguation. In *SumPre-HSWI@ ESWC*, 2015.

[15] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[16] Wikipedia contributors. Wikipedia main page, Apr 2018. [Online; accessed 20-April-2018].

[17] Silviu Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.

[18] Francois Deloche. Gru unit, Apr 2018. [Online; accessed 24-April-2018].

[19] Francois Deloche. Lstm unit, Apr 2018. [Online; accessed 24-April-2018].

[20] Francois Deloche. Recurrent neural network, Apr 2018. [Online; accessed 24-April-2018].

[21] Cicero dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, 2014.

[22] Mahmoud Elbattah, Mohamed Roshdy, Mostafa Aref, and Abdel-Badeh M Salem. Exploring freebase potentials for big data challenges.

[23] Serena Yeung Fei-Fei Li, Justin Johnson. Backpropagation and neural networks, Apr 2018. [Online; accessed 24-April-2018].

[24] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[25] Xianpei Han, Le Sun, and Jun Zhao. Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 765–774. ACM, 2011.

[26] Simon Haykin and Neural Network. A comprehensive foundation. *Neural networks*, 2(2004):41, 2004.

[27] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[28] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

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

[30] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics, 2011.

[31] Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*, 2014.

[32] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[33] Graham Klyne and Jeremy J Carroll. Resource description framework (rdf): Concepts and abstract syntax. 2006.

[34] Vijay Krishnan and Vignesh Ganapathy. Named entity recognition. 2005.

[35] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in neural information processing systems*, pages 231–238, 1995.

[36] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.

[37] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[38] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*, 2016.

[39] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[40] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[41] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[42] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.

[43] Andrea Moro, Alessandro Raganato, and Roberto Navigli. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244, 2014.

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

[45] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR, abs/1211.5063*, 2012.

[46] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[47] George Pavlidis. Fully connected neural network, Apr 2018. [Online; accessed 24-April-2018].

[48] Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. From freebase to wikidata: The great migration. In *Proceedings of the 25th international conference on world wide web*, pages 1419–1428. International World Wide Web Conferences Steering Committee, 2016.

[49] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of sparql. In *International semantic web conference*, pages 30–43. Springer, 2006.

[50] Maria Pershina, Yifan He, and Ralph Grishman. Personalized page rank for named entity disambiguation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 238–243, 2015.

[51] Jan Pichl, Petr Marek, Jakub Konrád, Martin Matulík, Hoang Long Nguyen, and Jan Šedivỳ. Alquist: The alexa prize socialbot. *arXiv preprint arXiv:1804.06705*, 2018.

[52] Eric Prud, Andy Seaborne, et al. Sparql query language for rdf. 2006.

[53] Delip Rao, Paul McNamee, and Mark Dredze. Entity linking: Finding extracted entities in a knowledge base. In *Multi-source, multilingual information extraction and summarization*, pages 93–115. Springer, 2013.

[54] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[55] Dennis W Ruck, Steven K Rogers, Matthew Kabrisky, Mark E Oxley, and Bruce W Suter. The multilayer perceptron as an approximation to a bayes optimal discriminant function. *IEEE Transactions on Neural Networks*, 1(4):296–298, 1990.

[56] Cicero D Santos and Bianca Zadrozny. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826, 2014.

[57] Cicero Nogueira dos Santos and Victor Guimaraes. Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008*, 2015.

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

[59] Valentin I Spitkovsky and Angel X Chang. A cross-lingual dictionary for english wikipedia concepts. In *LREC*, pages 3168–3175, 2012.

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

[61] Andreas Stolcke. *Bayesian learning of probabilistic language models*. PhD thesis, University of California, Berkeley, 1994.

[62] Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. Modeling mention, context and entity with neural networks for entity disambiguation. In *IJCAI*, pages 1333–1339, 2015.

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

[64] Marieke Van Erp, Pablo N Mendes, Heiko Paulheim, Filip Ilievski, Julien Plu, Giuseppe Rizzo, and Jörg Waitelonis. Evaluating entity linking: An analysis of current benchmark datasets and a roadmap for doing a better job.

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

[66] Peng Wang, Jiaming Xu, Bo Xu, Chenglin Liu, Heng Zhang, Fangyuan Wang, and Hongwei Hao. Semantic clustering and convolutional neural network for short text categorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 352–357, 2015.

[67] Wikipedia contributors. Freebase, Apr 2018. [Online; accessed 20-April-2018].

[68] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q. Zhu. Probase: A probabilistic taxonomy for text understanding. May 2012.

[69] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. Joint learning of the embedding of words and entities for named entity disambiguation. *arXiv preprint arXiv:1601.01343*, 2016.

[70] Xuchen Yao. Lean question answering over freebase from scratch. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 66–70, 2015.

[71] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

[72] Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.

Appendix A

CD/DVD Contents

This chapter contains the list of files that are on the enclosed disk.

/**readme.txt** Readme file with the instructions.
/**src.zip** Archive with the source codes of EL system.
/**thesis.pdf** Thesis in PDF format.
/**thesis_src.zip** Archive with source codes the thesis.