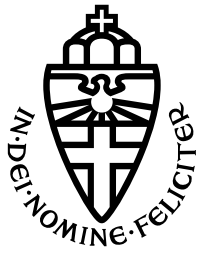
Radboud University Nijmegen



Faculty of Science

Entity Linking in Funding Domain

Thesis MSc Computing Science

Author:
Gizem Aydin

Supervisor:
Dr. Faegheh Hasibi

External Supervisor:
Dr. S. Amin Tabatabaei

Second reader:
Prof. Dr. Arjen P. de Vries

June 2021

Abstract

Automatic extraction of funding information from academic articles adds significant value to industry and research communities, such as tracking research outcome by funding organizations and aiding open access rules. An important part of funding information extraction is detecting mentions of grant numbers and funding organizations, and mapping them to their corresponding entities in a knowledge base. For this purpose, various approaches have been proposed. In this thesis, we investigate general-purpose neural architectures for Named Entity Recognition and Disambiguation, and adapt them to the problem of Entity Linking in funding domain. A neural language model (BERT) is pretrained with sentences that contain funding information and is used in the proposed neural solutions. The developed approaches are compared with strong feature-based models, showing improvement on mention detection and end-to-end Entity Linking. At the end, precision, recall and F1 scores of 72.9, 79.0 and 75.8 are reached for Entity Linking for funding organizations, and scores of 94, 96.6 and 95.2 for grant mention detection using the developed neural approaches.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

Researchers spend a significant amount of time to find the financial resources they need, and many organizations spend large amounts of capital to sponsor research. This puts funding information in an important position for research output, researchers and organizations. As a consequence, displaying funding information of academic articles brings certain benefits. For example, it enables organizations to track the outcome of the research they funded [47]. Successful research output on one topic may persuade an organization to invest more in that topic, expanding the resources available to the researchers interested in that area. Additionally, if a researcher is looking for funding for a certain research question, they can search for organizations that funded a similar topic in the past. Lastly, some funding organizations, such as NWO[1] and National Institutes of Health[2], may require the researchers to make the resulting publications publicly available. Displaying funding information may aid the compliance of such open access rules [16].

Automatic extraction of funding information from academic articles has been an interesting subject for researchers, and various approaches have been proposed for this purpose [47, 88, 16]. Annotating articles with their corresponding funding information adds significant value to the research community. It is not possible to do this annotation manually due to the immense amount of literature, making automatization a must.

Funding information extraction contains subtasks in itself. These can roughly be summarized as:

(i) selecting a piece of text that contains the funding information from the articles,
(ii) extracting the strings that refer to funding organizations and grant numbers from the selected text,
(iii) for the strings of funding organizations, determining which real-world organization is being referred to,
(iv) for the strings of grant numbers, determining which funding organization they belong to.

In this thesis, the aim is to develop a neural Entity Linker for funding domain, hence tackling subtasks (ii) and (iii).

Entity Linking (EL) is the task of annotating text with corresponding entity identifiers from a knowledge base (KB) [4]. A KB is a collection of entities representing uniquely identifiable artefacts in real world. An entity could be many things, such as a person, a place or even an artwork. EL often consists of two subtasks: Named Entity

[1] `https://www.nwo.nl/en/open-access-publishing`
[2] `https://en.wikipedia.org/wiki/NIH_Public_Access_Policy`

Recognition (NER) and Entity Disambiguation (ED). NER corresponds to detecting mentions and their respective types from the text, and ED corresponds to finding which entity in the KB a mention is referring to [4]. In this definition, mentions are strings in text indicating entities.

The state-of-the-art research in NER, ED and EL employ neural approaches [81, 94, 62]. These methods show significant improvement over rule-based systems or systems utilizing classical Machine Learning algorithms, such as DBPedia Spotlight [59] and TagMe [23]. A large number of studies on NER, ED and EL focus on performing these tasks in general-domain, often times considering Wikipedia pages as a KB [21]. There is no guarantee that these approaches will perform well in funding domain as there are a number of differences. Firstly, in order to disambiguate funding organizations, a different KB is needed as most of the small funders are not present in general-purpose KBs. Secondly, mentions of organizations that are not funders should not be extracted. Thirdly, grant numbers are mostly a combination of digits and letters, different from mentions of popular entities such as persons or places. Finally, there is limited amount of labelled data available for training and evaluation. In general-domain setting, Wikipedia could be exploited to extract millions of samples [11]. Supervised neural architectures for NER require a large amount of training data to obtain high performance [92].

1.2 Objective

The objective of this research is to adapt the current advancements on EL and its subtasks to funding information extraction. Considering the challenges funding domain adds to EL, this thesis aims to answer the following research question:

> *RQ: How can we use neural approaches for Entity Linking in funding domain, where labelled data is limited and a domain-specific knowledge base is used?*

To be able to answer the main research question, these sub-questions have to be answered first:

> *RQ1: How to detect mentions of funding organizations and grants?*
>
> *RQ2: How to generate distributed representations of entities for a domain-specific knowledge base?*
>
> *RQ3: How to perform Entity Disambiguation for funding organizations?*

The KB that is used in this thesis entails different information compared to general-purpose KBs. Each entity is represented by their name variants and country of origin. This KB does not contain a graph structure, only sparse relations exist among some entities denoting organizational hierarchies. Hence, popular methods for obtaining entity representations such as TransE [7] and Wikipedia2Vec [93] are not applicable.

1.3 Approach and Contributions

Initially, the literature on general-domain NER, ED and EL is reviewed. Among the reviewed solutions, some of them are selected for experimentation based on specific criteria. We checked whether the selected solutions perform well compared to others, and whether they can be adapted to the task at hand without losing their core properties. For example, a solution that is based on graph structures [89] is not found convenient, as that information is not available. It is also preferred that the models offer enough

flexibility so that some components can be modified to capture the important aspects of funding information extraction. In addition, it is taken into consideration that the selected solutions should not require too much computational power or too much labelled data as both resources are limited. The literature on domain-specific approaches are also reviewed to obtain some insights on the challenges other domains had faced and the respective solutions. Lastly, various techniques on domain adaptation of neural language models and different entity embeddings are investigated as they are crucial elements of possible solutions.

After careful consideration, we decided to develop two separate components for NER and ED to perform EL instead of a single end-to-end component. This decision was made to create a more flexible system and to have a system that is less computationally demanding to train. For the NER component, two systems are selected: the NER model that is proposed by Akbik et al. [1] and the BERT-based NER model. Bidirectional Encoder Representations from Transformers (BERT) [18] is a neural language model that can be pretrained with unlabelled data while utilizing both left and right context simultaneously. For ED, BLINK's architecture [90] is selected because of its flexibility and success in zero-shot setting. As BERT takes part in the majority of experiments, a BERT model is pretrained on the funding information sentences and used in both NER and ED systems.

Recently an end-to-end neural system to perform EL on questions, ELQ [55], is proposed. The architecture of ELQ encompasses the neural candidate selector of BLINK. Hence, it will be possible to extend an ED system built inspired by BLINK to perform mention detection following ELQ.

This work is done at Elsevier B.V.[3]. Elsevier uses Natural Language Processing (NLP) solutions to extract funding information from academic articles. Later on, this information is displayed in some of its products such as Scopus[4], one of the world's largest citation and abstract databases. Elsevier has a high-quality labelled dataset for this task and a domain-specific KB for funding organizations. The chosen models are trained and evaluated using the labeled dataset provided by Elsevier, and also disambiguation is performed to the provided KB. The developed models are compared with each other and feature-based models to determine the best method and to show the effectiveness of neural approaches.

The contributions of this thesis can be summarized as:

1. Adapting the BERT model to the funding information extraction domain (BERT_{SC}) and showing its effectiveness for NER.

2. Developing a neural EL solution to extract mentions of funding organizations and grant numbers simultaneously, and to link funding organizations to a KB or NIL.

3. Showing the effectiveness of a minimal linear reranker that can work with the proposed candidate selector.

We believe that this research will be inspirational for other domains to make use of the state-of-the-art neural architectures for EL and will provide important insights on possible strengths and weaknesses of utilizing such systems. Also, the developed neural components could be used as a part of a single end-to-end funding information extraction model in the future. The code for the developed models can be found on GitHub[5].

[3] `https://www.elsevier.com/`

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

[5] `https://github.com/gizemayydin/Entity-Linking-in-Funding-Domain`

1.4 Outline

The aim of Chapter 2 is to review the recent literature on NER, ED and EL, as well as various domain-specific solutions to these problems. Literature on entity embeddings and domain adaptation of neural language models are also touched upon due to their relevance in proposed solutions. Lastly, notable previous work on funding information extraction is presented. In Chapter 3, the task of EL in funding domain is formally defined, the approach to tackle this task is presented, and the models proposed are explained in detail. The dataset, experimentation, results and discussion are included in Chapter 4. Chapter 5 concludes this thesis by summarizing the findings and providing directions for future research.

Chapter 2

Related Work

There is a large amount of literature on Entity Linking (EL) and its subtasks, Named Entity Recognition (NER) and Entity Disambiguation (ED). The bigger part of this literature focuses on performing these tasks in the general-domain setting, often times considering Wikipedia pages as entities [21]. While this line of research introduces the state-of-the-art approaches, there is no guarantee that these approaches will perform well in a domain-specific setting with a custom knowledge base (KB). Hence, domain-specific literature is also investigated to get insights on adapting general-purpose methods to specific domains.

Section 2.1 reviews the literature on automatic funding information extraction from academic articles. In Sections 2.2, 2.3 and 2.4 state-of-the-art general-purpose NER, ED and EL solutions are presented. Domain-specific neural NER, ED and EL approaches are demonstrated in Section 2.5. Section 2.6 concentrates on entity representations in neural ED, mainly entity embeddings, and lastly, Section 2.7 reviews the literature on domain-adaptation of neural language models.

2.1 Funding Information Extraction

One of the most notable work on automatically extracting funding information from text is FundingFinder [47]. FundingFinder is a two-step pipeline that utilizes NLP techniques. In the first step, the paragraphs that contain funding information are determined, and in the second step, NER is performed using an ensemble of different Sequential Learning approaches. The authors also created a publicly available benchmark dataset for this task. The approach used in this thesis builds upon this work, keeping the first step intact while improving the second step, and adding the ED capability.

Before FundingFinder, not much literature existed on extracting funding information from text automatically, and the existing work mostly utilized regular expressions [47]. Recently, there have been more approaches presented to tackle this problem. In 2020, Wu et al. proposed AckExtract [88], which extracts funder organization mentions from the COVID-19 Open Research Dataset [83]. For NER, they use a pretrained neural model from the package Stanza [70], which uses Contextual String Embeddings [2]. However, their method does not include any ED, whereas in this thesis, one of the tasks is to link the funder mentions to their corresponding entities in the domain-specific KB. Another approach is proposed in 2021, GrantExtractor [16], which extracts funding information from articles in biomedical literature, in the form of grant numbers and their corresponding organizations. For extracting grant numbers, they train a BiLSTM-CRF [41] architecture. Using a multi-class classifier, they determine which organization the extracted grant number belongs to. They do not use any neural approaches for ex-

tracting organization mentions. Also, the focus of GrantExtractor is on linking grant numbers to their respective organizations, while the focus of this thesis includes extracting all funding organizations that financially supported the corresponding research, even though no grant information is acknowledged.

Alexander & de Vries [3] introduced AckNER in 2021. AckNER extracts various named entities indicating financial support from sentences containing funding information. AckNER utilizes both dependency parsing and regular expressions. The former is used for the mentions of organizations, programs, projects and funds, while the latter is used for grant numbers and contracts. The authors show that AckNER outperforms general-purpose NER methods found in popular NLP libraries such as Flair [1] and Stanza [70] by a great margin, showing the need for a domain-specific approach. The authors report an F1 score of 80%, which is 27% higher than the closest competitor, DeepPavlov [12], which utilizes a BiLSTM-CRF architecture.

2.2 Named Entity Recognition

In the past couple of years, Deep Learning has been a popular choice to tackle the NER problem, and the corresponding research has improved the state-of-the-art results [92]. In 2018, Akbik et al. proposed Contextual String Embeddings [2], which represents words using a character-level neural language model, and is able to produce different word embeddings depending on the context. By utilizing a BiLSTM-CRF architecture that takes the concatenation of Contextual String Embeddings and pretrained GloVe embeddings [67] as input, they report state-of-the-art results in both German and English NER, in the CoNLL-2003 [79] setup. In 2019, Devlin et al. introduced BERT [18] which obtained new state-of-the-art results on several tasks. In English CoNLL-2003, they obtained an F1 score of 92.8% using the cased version of BERT_{BASE} [18], performing very close to Akbik et al. [2], which obtained 93.1%. It is believed that both of these models could be suitable for extracting mentions of funding organizations and grant numbers, as they do not necessarily require extensive amount of labeled data and there is no dependency to any resource that is not available.

Some approaches for NER utilize external resources, such as a list of entity names, which may be called a dictionary or a gazetteer. This may boost the performance of the system, but may also hurt the generalization ability [92]. However, there have been various models presented [57, 87] that incorporate this information while performing comparable to Akbik et al. [2]. With this approach, both Liu et al. [57] and Wu et al. [87] aim to improve the performance on entities that do not appear in the training set or that are rare. These features may also be useful for this work, as the domain-specific KB contains various synonyms for each funding organization, and most of the entities available do not appear in the dataset (see Section 4.1.1).

Recently, Yamada et al. (2020) proposed LUKE [94], a contextualized representation for both words and entities, to be used in entity-related tasks. LUKE is based on the bidirectional transformer [82], however, it treats words and entities as independent tokens. For this purpose, the authors propose a modified attention mechanism as well as a new training methodology based on BERT's masked training. They pretrain LUKE using a large entity-annotated Wikipedia corpus. By using the proposed embeddings, they report the new state-of-the-art results for NER, improving upon Akbik et al. [2]. The entities that are included in this work differ significantly from the ones that are in Wikipedia or in any other famous general-purpose KB. Hence, we do not expect to obtain any gain from using LUKE instead of an NER component based on BERT_{BASE} or the one introduced by Akbik et al. [2].

2.3 Entity Disambiguation

One of the most influential work in ED is MentNorm [52], proposed by Le and Titov in 2018. MentNorm is a multi-relational neural model, and is based on the assumption that the relations between co-occurring entities provide important clues for disambiguation. Different from preceding work, they model these relations as latent variables, which enable to learn these in a way that is most useful for the ED task. They report a micro averaged F1 score of 93.07% on AIDA-B CoNLL YAGO dataset [39], outperforming previous research. Despite being successful, collective ED methods may suffer from high complexity [97]. In 2019, Yang et al. [97] proposed Dynamic Context Augmentation (DCA) that can perform collective ED with just a single pass over the whole document. DCA takes global entity coherence into account, but instead of processing the whole document at once, it processes each mention sequentially. The authors reason that this methodology is more intuitive as humans have the ability to make inference while reading, without having read the whole document. The information of previous linked entities are accumulated and used for disambiguating the proceeding mentions. Their best configuration 94.64% accuracy on AIDA-B CoNLL YAGO dataset, which is higher than that of Le and Titov [52]. In funding domain, the number of Organization mentions per document is much less than that of public benchmark datasets. For example, Yang et al. [97] report an average of 19.4 mentions per document for AIDA-B dataset. In the dataset for funding information extraction, this number is around 3. Also, entity co-occurrences do not provide important clues for disambiguation. That is why collective ED systems are not used for this work.

In 2018, Raiman and Raiman proposed DeepType [71] for ED, a neural network that is constrained by the predicted type information for a given entity. By using the type information, they also reduce the complexity of disambiguation from polynomial to linear. DeepType produced state-of-the-art results in three ED datasets, by obtaining scores of 92.36%, 94.88% and 90.85% on WikiDisamb30 [24], CoNLL (YAGO) [39] and TAC-KBP-2010[1] respectively. The authors also note that DeepType can reach 99.0% and 98.6% accuracy on CoNLL (YAGO) and TAC-KBP-2010, when the type information is provided by an Oracle. Based on that, they claim the ED problem can almost be solved if the type classifier is improved. However, in the case of funding domain, most of the ambiguities in mentions cannot be solved by using the type information. For example, the mention "Ministry of Health", can be resolved to different entities corresponding to ministries in different countries, however, the types of these entities would be the same.

The paper proposed by Mulang' et al. (2020) [62] slightly advances the state-of-the-art ED results for CoNLL (YAGO) dataset by obtaining a score of 94.94%. The authors introduce the idea of incorporating context derived from Knowledge Graphs (KG) to pretrained transformers with the aim of improving their performance for ED. They extract triplets from the KG, verbalize them into natural language form, and append them to the input sentence and mention before passing it through the transformer. When they replace the Wikipedia description used in the DCA-SL model [97] with the structured KG context they extracted, they obtain the above-mentioned score. In this work, there is not a KG that could be utilized in such manner. Otherwise, incorporation of the work by Mulang' et al. [62] could have added value to the current system.

Another interesting approach for ED is DEER [31], proposed by Gillick et al. in 2019. The model is essentially a dual-encoder, one to encode the mention and its context, and another to encode the entity using its description and categories. Then, the model is trained to maximize the cosine similarity between the encoded representations of the correct mention - entity pairs. The authors use hard negatives during training, and they show the positive effect of this to the performance. As the second encoder utilizes only

[1] `https://tac.nist.gov/`

entity information, entity embeddings can be precomputed beforehand, and hence only one encoder is run during inference. We believe that this work can be very influential for this thesis. They show good performance on public benchmark datasets, the inference time is fast even though it is a neural approach, and can scale up to unseen entities. The only downside is that each encoder has sub-architectures within itself. For example, the entity encoder has three different encoders for the title, the category and the first paragraph of the description. Hence, it may be hard to adapt these for the information available.

Wu et al. (2020) [90] proposed BLINK, which outperforms DeepType in TAC-KBP-2010 by obtaining a 94.5% accuracy . Their method also achieves state-of-the-art results in the zero-shot Entity Linking dataset derived from WikilinksNED [64]. To perform ED, they only use textual information, and architectures that utilize pretrained BERT transformers. They represent the mention using itself and its context, and the entity using its description. Using a biencoder [42], they encode the mention and entity representations in the same space, which they later use to extract candidates for a given mention using approximate nearest neighbor search. To train the biencoder, they make use of a hard negative mining strategy inspired by DEER. They make the final decision by passing representations of the candidate entities and mentions through a cross-encoder [42]. BLINK can perform well for unseen entities and has an architecture that is easy to adapt to the problem at hand. Hence, BLINK is found highly suitable for this work, and is investigated further (See Section 3.4).

Even though there had been many influential work on tackling ED, not many of them encompass NIL mentions, mostly excluding them from evaluation. In this work, NIL mentions mostly refer to emerging entities, and hold high importance. New organizations are being added frequently, and NIL mentions give important insights on these organizations. They are also used to refine the KB from time to time. One of the hardest cases of NIL mentions occurs when the emerging entities have ambiguous names [38]. These cases also exist in the funding domain. For example, "Ministry of Education" could refer to different organizations based on the country, and it could be that the ministry of a country is not in the KB. Then, the ED system should infer that this mention does not refer to any other ministries that are included in the KB. In addition, there are organizations that share the same acronym as a coincidence. It could be that a funder that does not exist in the KB is acknowledged by using its acronym, which matches a completely different organization that happens to be in the KB. Hoffart et al. (2014) [38] proposes a novel method to detect such emerging entities. They add an additional instance representing an emerging entity to the candidate set of each mention. They initially model this instance using the string of the mention, but then enhance the representation based on keyphrases utilizing latest news streams. This is not applicable for this research as ED is performed on academic articles. They also define confidence scores based on perturbing mentions and entity decisions around the input mention. However, in this work, the models investigated do not make use of collective disambiguation.

2.4 Entity Linking

Kolitsas et al. (2018) [50] proposed the first end-to-end neural EL system in 2018, and recorded state-of-the-art results in AIDA CoNLL dataset. By tackling NER and ED jointly, the authors aim to utilize the dependency between these two tasks. They suggest that this has several benefits, such as improved mention boundary recognition. Their method first extracts all possible mention spans from the input. Then, the model computes a score for each mention - candidate entity pair, using pretrained entity embeddings [30], context-aware mention representations, commonness and long range

attention scores. The final output for the input text is based on these scores and global entity coherence. This approach has some properties that make it not suitable for this thesis. First, the entity embeddings they use are for Wikipedia entities, thus new embeddings for this KB must be obtained beforehand somehow. Based on the properties of these new embeddings, the global coherence layer should be reconsidered. Also, the runtime could be a problem as the methodology involves extracting all possible mention spans. As the grant mentions do not go through ED, the loss needs to be adapted accordingly. Lastly, the advantage of this approach is that the mention boundary detection is improved by combining NER and ED, which is shown by the fact that there is only a small difference between the weak and strict matching evaluations. However, the authors report that most of their mentions consist of at most two words, which is not true for funding organizations that tend to have longer names. Hence, there is no guarantee that this advantage will persist in funding domain.

Another neural approach that can perform both NER and ED in an end-to-end fashion is proposed by Martins et al. in 2019 [58]. The approach makes use of Stack-LSTMs [20] to detect mentions on the fly rather than detecting them for the whole sequence at a time. For each token in the sequence, an action is predicted. This action can be: *Shift*, *Out* or *Reduce*. *Out* means that the token at hand is not part of a mention, while *Shift* means that it is. After all the tokens of the mention is marked with consecutive *Shift* operations, *Reduce* is performed, meaning that the detected mention is disambiguated. There are multiple *Reduce* actions, one for each mention type. Hence, the type of the *Reduce* action also predicts the type of the mention. The model outputs a probability distribution over the possible action space for each step, and the action with the highest probability is selected. In the disambiguation step, first, the mention is classified to determine if it is a NIL-mention or not. If it is not, a candidate entity set is selected with respect to entity embeddings obtained by Yamada et al. [95] and prior probabilities. Then, a score is assigned to each candidate entity by utilizing an affine transformation function, and the mention is linked to the entity with the highest score. The model is trained with a multi-task learning setting of three tasks: NER, NIL-mention detection and ED. For NER, the loss is calculated as the cross-entropy between the gold actions and the selected actions. For NIL detection, binary cross-entropy is utilized. Lastly, for ED, cross-entropy over the candidate entities is calculated. The authors show that performing NER and ED jointly benefits both of the tasks, by comparing the performance of a joint architecture with two different architectures. However, for the NER task, both Akbik et al. [2] and Devlin et al. [18] outperform the work of Martins et al. [58], and in terms of EL, the work fails to outperform Kolitsas et al. [50]. This work is still very inspiring, especially as they include NIL-mentions in their research.

In 2020, van Hulst et al. proposed REL [81], an EL toolkit that utilizes state-of-the-art NLP research, outperforming Kolitsas et al. [50] in terms of micro F1 score in AIDA CoNLL dataset. REL tackles the EL problem in three steps: NER, candidate selection and ED. For NER, they utilize Flair, namely, the sequence labelling architecture and Contextual String Embeddings proposed by Akbik et al. [2]. For each mention, up to 4 candidates are selected using commonness, and up to 3 candidates are selected based on the similarity between the context of the mention and entity embeddings. Entity embeddings are provided by Ganea and Hofmann [30], the same ones used in Kolitsas et al. [50]. For ED, MentNorm is used. Apart from obtaining state-of-the-art results, REL also offers a modular architecture, allowing easy replacement of components and it does not require a GPU during inference time [81]. The success of REL in terms of performance and efficiency is one of the factors why Contextual String Embeddings are in the agenda of experiments for this work. However, MentNorm is not suitable for funding domain because of its dependency on entity coherence. This work also inspired

this research in the sense that a comparable or higher performance than that of Kolitsas et al. [50] can be obtained even when NER and ED are performed separately, which is advantageous in terms of runtime and explainability.

Broscheit (2020) [10] proposed an architecture that jointly does NER and ED using BERT. In this approach, the task of EL is framed as a per-token multi-class classification problem. The model utilizes a pretrained BERT model and an output classification layer on top of it. Even though this approach is a big simplification on the EL task, it performs only a few percents off compared to Kolitsas et al. [50]. However, as each entity is cast as a class, the model cannot disambiguate unseen entities. In real-word, KBs keep growing, and hence it is important for the system to be extendable for entities that do not exist in the training set [33]. In addition, some training datasets may not cover the whole entity vocabulary, such as the one used in this work. However, this research is very inspiring in terms of showing BERT's capability to learn the task of EL. It also contributed in the decision process of experimenting with BERT as a neural language model to be used in this work.

In 2021, De Cao et al. proposed GENRE [13], a novel approach for end-to-end EL. The authors criticize previous work on various aspects, such as the need of hard negatives during training and the use of dot product to model the relevancy between an entity and a mention. GENRE addresses these shortcomings by using a sequence-to-sequence generative architecture for EL. For this purpose, the authors fine-tune a BART [54] model on the task. Mention detection and disambiguation is achieved by decoding the input sequence which is done dynamically. For each step in sequence generation, the model has various options: generate a mention span, generate a link for the mention or continue copying the input sequence. To start generating a mention, the model has to generate a mention-start-tag. Inside the mention, the model can either copy the input sequence or stop mention generation by generating a mention-end-tag. The latter puts the model in the link generation (i.e. disambiguation) state. In GENRE's architecture, each entity is associated with a unique textual identifier, such as a name. The model assigns a score to each entity based on the generative probability of the corresponding entity's name given the context. As the entity space is large, a Beam Search [77] variant (Constrained Beam Search) is used. This whole schema allows the model to generate an output sequence with detected mentions and their corresponding entities, while narrowing the possible search space. The authors report that there is no need for hard negative mining, as the loss for training is based on maximizing the likelihood of the output sequence, and can be calculated exactly [13]. Lastly, the model has a much smaller memory footprint compared to previous approaches as it does not need to store precomputed entity embeddings, and either outperforms or performs on par with the state-of-the-art for various public benchmark datasets. Even though GENRE has an outstanding performance, there are various concerns on using it for funding domain. The authors report that large generative models benefit from more data, and hence they pretrain their model with a large amount of labeled data. This is not possible for this work as there is no such dataset available. Also, the entity representation is based on having a unique string such as a name for each entity. In funding domain, organizations may have more than one official name such as acronyms or names in other languages, or there may be different organizations with the same name. To create a unique name per organization, some of these alternative names should be concatenated, and it is not clear whether this kind of representation would work with GENRE, or whether it would be tractable given that the inference is dependent of the size of entity representation. The authors report that their entity representations had 6 tokens on average for the experiments, which is not realistic for funding organizations.

Entity Linking in Queries

There is a specific line of work that focuses on EL in queries [6, 15, 35, 36] and questions [55]. Questions differ from well-formed sentences, and tend to be shorter and noisier, hence bring additional challenges [55]. In addition, the context is much more limited [46]. The input for funding information extraction is not as long as a document, and is also not as short as a query. Hence, this makes both lines of work interesting for this thesis.

Zi et al. (2020) [55] proposed ELQ, an end-to-end EL system for questions. ELQ extends BLINK's biencoder by adding the mention detection capability to the model. The resulting model outperforms the state-of-the-art in two benchmark Question Answering datasets in terms of EL capabilities. The end-to-end architecture of ELQ differs significantly from that of Kolitsas et al. [50], improving the efficiency and bringing some flexibility. Even though all possible mention spans are extracted from the text for mention detection, different from Kolitsas et al. [50], the spans are thresholded based on some probability and length. This eliminates the need to run a candidate selector to limit the mention space. In their experiments, the authors set the maximum span length to 10, which is still not optimal for organizations but is not unrealistic. Also, they jointly optimize the NER and ED losses, which may allow to extend the system for Grant mentions. Another advantage is that they do not need to train the entity encoder as they obtain it from BLINK. Even though the entity encoder of BLINK is for Wikipedia entities, it is possible to get embeddings for other KBs by adjusting the entity representation and retraining the model. Despite this system being for questions and thus being evaluated on data with a significantly different distribution, it is worth to inspect its performance for the task at hand, as the authors show optimism towards ELQ's performance on longer and structured documents.

2.5 Domain-Specific Systems

In this section, the research that aims to tackle EL, NER and ED in a domain-specific setting with neural architectures is reviewed.

For NER, there is a great amount of research in general domain, however, more research on domain-specific solutions is expected for supporting real-world applications [99]. Existing NER approaches rely on a large amount of annotated data, which may not be available for the domain-specific setting [75]. Hence, Shang et al. (2018) [75] proposed AutoNER, a neural architecture that is designed to learn from data that is created by distant supervision, without any human effort. AutoNER uses domain-specific dictionaries to automatically generate labelled data with distant supervision. The authors also introduce the "Tie or Break" tagging schema, that is based on predicting whether two adjacent tokens belong to the same mention or not. They reason that this tagging scheme is suitable to use noisy labels generated by distant supervision. They show the effectiveness of their work in multiple datasets, two of them being the BC5CDR [56] and NCBI-Disease [19] datasets from the biomedical domain, in which AutoNER achieves 84.8% and 75.52% F1 score respectively. For this work, as there is already labeled data available, it is believed that such labeled data generation should be used only if the high quality data at hand proves to be insufficient significantly.

Another domain-specific NER approach that utilizes a dictionary is proposed by Wang et al. (2019) [84], which tackles the Clinical NER problem in Chinese text. They show the effect of incorporating dictionary knowledge in the BiLSTM-CRF architecture on rare and unseen entities experimentally. Also, they suggest five different methods of using dictionaries in this context and compare the results. This work can be very influential for funding domain, as a large dictionary is available, and the proposed fea-

tures can be incorporated to any neural architecture easily, enabling to utilize both this resource and the labeled data.

There also exist research on tackling the domain-specific ED problem with neural architectures. In 2019, Mondal et al. [61] proposed a system that is based on string similarity to perform ED on disease names. The authors utilize a two-step solution. First, for each mention, they extract a set of candidate entities based on Jaccard overlap and the cosine similarity between the entity label and the mention. They use word embeddings to calculate the cosine similarity. For multi-word strings, they sum the embeddings for each word. Then, they rank the candidate entities with a Triplet Network [40], that learns to reduce the distance of the mention with the positive candidate, while increasing the distance with the negative candidate. As an input to this network, word embeddings is used again to represent the mention and the candidate entity's label. With this approach, they obtain 90% accuracy on the NCBI-Disease dataset, outperforming previous approaches. The drawback of this approach is that it does not utilize any context information, which is important for funding domain, because clues such as the country of the organization can be found in the context. Also, multi-word mentions are very common for organizations, and summing the embeddings may not be the best way to incorporate each word to the representation.

The input representation of entities vary between different approaches. In another research tackling clinical ED by Schumacher et al. (2020) [74], different from Mondal et al. [61], multi-word strings are represented by two different methodologies, Max Pooling over the word embeddings and running self-attention. The authors report that running self-attention produces better results compared to Max Pooling. The proposed methodology also addresses some issues that come with funding domain, such as the importance of the lexical similarity between the mention and an entity. Also, the architecture resembles the Biencoder of BLINK in terms of having two components to encode the mention and entity. However, in this work, ELMO [68] is used and the encoders have shared parameters. Also, the loss and negative mining strategy is different. The success of this work is inspiring in terms of showing that it is possible to use such a dual architecture for domain-specific problems.

Using BERT instead of ELMO could have some benefits for entity representation, for example, the [CLS] token can be used to represent the whole sequence naturally. For example, Sung et al. (2020) [76] represents each mention and each synonym of an entity using BERT's [CLS] token for the biomedical domain. On top of that, another representation based on TF-IDF scores is also introduced. For disambiguation, the authors define two different similarity function for each of these representations. The final similarity of a mention and an entity synonym is defined based on the weighted average of these two similarity scores. It is believed that utilizing such a sparse representation could possibly add value in the funding domain as well. However, this will not be experimented with as it is decided that the additive value of it shown by the results of the paper is not significant enough compared to the time that should be spent for the incorporation of such representation.

Architectures utilizing the type information are also present. Zhu et al. (2020) introduced LATTE [101], an architecture for ED in medical domain. The authors emphasize the importance of fine-grained types in their setting, and as this information is not available, they model the fine-grained types as latent variables. For ED, in addition to the latent fine-grained types, they use the similarity between the entity's label, and the mention and its context. To train their model, they use multi-task learning for both type classification and ED. Even though LATTE is a very inspiring research, it would not have the same affect on funding domain as the reliability to the type information for disambiguation is highly limited.

Some proposed methodologies tackle the EL problem as a whole in a domain-specific

setting using neural architectures. For biomedical domain, Zhao et al. (2019) [100] proposed a joint neural architecture for NER and ED tasks for performing EL. Their architecture utilizes explicit feedback between the two tasks in a multi-task learning setting. With this architecture, they obtain F1 scores of 87.43% and 88.23% in NER and ED tasks of the NCBI-Disease dataset respectively, and 87.62% and 89.17% in BC5CDR dataset. With these numbers, they outperform AutoNER in NER setting for both datasets, and perform comparable to Mondal et al. [61] in NCBI-Disease dataset for ED. As the task of ED is framed as classification over the controlled entity set, it is not clear whether the model would work in zero shot setting.

Biomedical domain is not the only one in which neural approaches are used for NER, ED and EL. In 2019, Espejo-Garcia et al. [22] proposed a solution to extract named entities that refer to the important parts of phytosanitary regulations, which is related to the agricultural domain. The authors experimented with eight different state-of-the-art neural architectures. For their setting, the best performing architecture was a bidirectional LSTM [32] that utilized a Softmax layer for inference and got the concatenation of pretrained Word2Vec [60] embeddings with character based word representations as input. With this architecture, they obtained an F1 Score of 88.3%. Apart from agricultural domain, Yang et al. (2020) [96] proposed Headword Oriented Entity Linking, an EL setting where the mention scopes do not need to be identified, to extract cosmetic products from blogs and to disambiguate them to a domain-specific KB. First, using word segmentation techniques, they identify the headwords of the mentions. Then, they apply classification on the mentions to decide whether the mention can be linked to a product that is in the KB. Lastly, they use a modified version of the architecture proposed by Gupta et al. [33] for ED. Another interesting study is by Kurz et al. (2020) [51], where they experiment with different BERT-based architectures to disambiguate mentions of machine parts and errors belonging to German technical service tickets.

2.6 Entity Representation

In neural ED, entity representation plays an important role. Some research frames the problem as multi-class classification and represent entities as different classes [10, 85, 100], while other research tends to use architectures that takes properties of entities as input and learns a representation internally during training for ED [90], implicitly or explicitly. Another line of research utilizes entity embeddings [81, 50, 97, 61]. In this section, the literature on entity embeddings will be reviewed.

There is a large body of literature on embedding entities and relations found in KBs. One of the most notable work is TransE [7], introduced in 2013. TransE generates embeddings for each entity and relation in the input KB. The idea behind TransE is to model relations as translations in the embedding space. For a given triplet (h, l, t) in the KB where h, l and t denote head entity, relation and tail entity respectively; TransE aims to make the embedding of t as close as possible to the sum of the embeddings of h and l, using an energy-based model. Later on, there has been models that improved upon TransE such as TransH, TransR, CTransR and TransD, each improving upon the previously proposed one respectively [44]. Another interesting work that generates entity embeddings utilizing the triplets in KBs is RDF2Vec [72]. RDF2Vec extracts graph sub-structures, and treats them as sentences to train a Word2Vec model.

Wikipedia2Vec [93] is also a famous and successful method for obtaining entity embeddings. Wikipedia2Vec pretrained embeddings are also available for direct use. Wikipedia2Vec encodes words and entities in the same space and utilizes Word2Vec. To train Wikipedia2Vec, they use three models. Word-based skip gram model puts the embeddings of words that occur in similar context close, anchor context model puts the embeddings of entities close to embeddings of words that occur near the anchor texts

of the entity, and lastly, link graph model puts the entity embeddings close based on Wikipedia's hyperlink graph.

Although proven to be very successful, it is not very likely that embeddings such as Wikipedia2Vec or TransE would perform well for funding organizations. These kind of systems rely heavily on KGs and diverse relations between entities. Besides, it would not be possible to use the already-trained ones, due to the fact that the entities at hand not being present in general-purpose KBs. In the domain-specific KB used for this work, although there exist some relations between entities, they are sparse and not informative in terms of disambiguation. Hence, it is believed that other ways of representation is needed to make sure the available information is utilized fully. Some ED architectures such as BLINK and DEER train a dual architecture that encodes entities and mentions separately. This allows learning entity embeddings that are directly relevant with the data and task at hand. Also, it is again possible to precompute these embeddings and store them for efficient inference or for using in other tasks. In this work, such architectures are favored because of their success and flexibility on obtaining entity embeddings by being able to utilizing different representations.

The dual architectures are not the only type of ED systems that allow learning entity embeddings with the task. In 2017, Gupta et al. [33] proposed a neural architecture that can generate entity embeddings by jointly encoding the information on the entity's description, the context of its mentions and its type. The architecture consists of three models that encode different information. The parameters of the models and the embeddings are jointly learned based on the sum of four different losses that ensure the entity embeddings and the encoded information is similar. The summation guarantees that entity embeddings can be generated even though some information is missing, such as the description [33]. They also proposed an ED model based on these embeddings. As mentioned in Section 2.5, a modification of this architecture is used to create entity embeddings in Yang et al. [96], showing that the model can be utilized for different types of information as well. The downside of this methodology, for example when compared to BLINK, is that new models and losses should be defined and incorporated into the architecture for using different information. In BLINK, it is possible to just modify the input representation of the entity for the same purpose.

2.7 Using Pretrained Language Models in Domain-Specific Applications

BERT, and other pretrained language models have been used extensively in various NLP applications and have obtained state-of-the-art results in benchmark tasks [63]. However, for some domains, they may be too generic and may not be able to cover specific needs [8]. Hence, it may be worthwhile to adapt the pretrained language model that will be used to the specific domain. Gururangan et al. (2020) [34] shows that a second-round of pretraining of a pretrained language model improves the performance in both low and high resource settings, using different domains and tasks. Also, Fraser et al. (2019) [26], reports that language models which are pretrained with domain-specific text perform better on the task of NER in biomedical domain. However, it should be noted that training a neural language model from scratch for a specific domain can take weeks [98].

There is emerging research on domain adaptation of pretrained language models that is not as costly as training them from scratch. Gururangan et al. (2020) [34] proposed Task-Adaptive Pretraining (TAPT), which corresponds to pretraining the neural language models with the unlabeled data of the task at hand. They report that this approach performs comparable to pretraining with larger amount of text from the same

domain. TAPT is intuitive, has a well-documented open-source implementation and does not require additional resources which makes it easy to use.

Tai et al. (2020) proposed exBERT [78], a low-cost method to perform domain adaptation and to add new domain-related words to the vocabulary of BERT while not changing its weights. The authors accomplish this by adding an extension module to the embedding layer and to each transformer layer. The authors show the effectiveness of their approach on biomedical domain. In funding domain, even though the input distribution is different, vocabulary changes as substantial as in biomedical domain is not expected. Hence, expanding the vocabulary of the original BERT is not needed.

Another methodology is introduced by Poerner et al. in 2020, and is called GreenBioBERT [69]. GreenBioBERT stands out by not requiring a GPU and hence being environment-friendly. In this approach, a Word2Vec model is trained on the domain-specific data, and the embedding vectors of the pretrained language model is aligned based on that. The authors evaluate their model on Biomedical NER and compare it to BioBERT [53]. Even though BioBERT outperforms GreenBioBERT, it shows that it is still possible to perform domain adaptation in a low-resource setting.

Chapter 3

Approach

To tackle the Entity Linking (EL) problem in funding domain, a two-step solution is developed. The first step is Named Entity Recognition (NER) and the second step is Entity Disambiguation (ED). For both of these steps, after an extensive literature review, the state-of-the-art systems that can be adapted to the problem at hand are determined.

For the NER component, several models were implemented and tested. The first model that is tried is the sequence labelling architecture proposed by Akbik et al. [2]. The choice of experimenting with this model was due to its success in English NER. Also, this model is used by REL [81], a system that achieved state-of-the-art results on EL, and AckExtract [88], a system for extracting funder organization mentions. This model will be denoted as Flair^{NER}.

Flair^{NER} is based on Contextual String Embeddings (CSE) [2], which are obtained using the concatenation of vectors from a Left-to-Right and a Right-to-Left language model. However, Devlin et al. [18] reports that BERT is inherently more powerful than such language models as it is using MLM objective, that enables it to train a single representation for which both right and left contexts are used. Because of this claim and the recent popularity of BERT models, it is decided to experiment with BERT as well, which will be denoted as BERT^{NER}.

It is shown that domain-adaptation of BERT improves the performance on downstream tasks in different domains [34, 78, 26], we therefore pretrain BERT on funding acknowledgement sentences using Task-Adaptive Pretraining (TAPT) [34]. This model is denoted as BERT$_{SC}$. Later on, an NER model is trained using BERT$_{SC}$, BERT$_{SC}^{NER}$.

For the ED component, the biencoder of BLINK architecture proposed by Wu et al. (2020) [90] is implemented. BLINK consists of two models, a biencoder inspired by Humeau et al. [43] for candidate entity selection and a cross-encoder [43] for candidate entity reranking. As input, it utilizes mentions with their surrounding context and entity titles with their descriptions. The reason why this architecture was chosen is five-fold. First, BLINK can work with unseen entities. Second, the inference time is not long, and third, the authors demonstrate that this approach obtains state-of-the-art performance. Fourth, BLINK's architecture can be adapted to the setting of this work without losing any important properties, and fifth, it can be extended to perform end-to-end EL following the ELQ architecture [55]. In this work, for efficient inference, a feature-based model is used as a reranker instead of the cross-encoder.

Section 3.1 defines the problem this thesis is tackling. Section 3.2 focuses on the BERT architecture and the domain adaptation procedure. Section 3.3 introduces Flair^{NER}, BERT^{NER} and BERT$_{SC}^{NER}$, and Section 3.4 explains the methodology used for ED.

3.1 Formal Task Definition

Given a piece of text $d = \{x_1, ..., x_N\}$ of N tokens, the task of NER is to identify a set of spans $M = \{m_i \mid m_i = \{x_s, .., x_e\}, s \geq 1 \wedge s \leq e \wedge N \geq e\}$, where each span corresponds to a mention. Usually, detecting the types of the extracted mentions is also part of the NER task. The available mention types differ among systems based on their objectives. In this thesis, each mention m_i corresponds to either a *Funding Organization* (ORG) or a *Grant Number* (GRT). Hence, $type(m_i) \in \{ORG, GRT\}$. Throughout this research, it is assumed that the mentions are not nested and are not overlapping.

Given the set of mentions M, the aim of ED is to link each mention M to its corresponding entity $e_i \in \mathcal{E}$ in a KB or to NIL. A knowledge base is a collection of entities, and may contain information on the entities and relations between them. The contents and specifications of the KB used is task-dependent. The task of ED is not trivial as many different mentions may be used to refer to the same entity, or the same mention may be used to refer to different entities. When the correct link of a mention m_i is entity e_i, this will be denoted as $link(m_i) = e_i$. Another thing to note is that some mentions may not correspond to any entity in the target knowledge base. These mentions are usually referred to as *NIL Mentions*. To denote these cases, a NIL entity $\emptyset \in \mathcal{E}$ will be used. In this thesis, a KB of funding organizations is used, hence, only the mentions with type ORG will be considered for ED.

Sometimes, the set of entities $\mathcal{E}$ may be extremely large. In that case, a Candidate Selector (CS) may be used to limit the search space. The aim of the CS is to extract a set of candidate entities $C_i = \{e_1, ...e_K\} \subset \mathcal{E}$ for a given mention m_i. Usually the size of C_i is much smaller than that of $\mathcal{E}$. This enables using more complex algorithms for ED as it reduces the number of entities to consider.

Lastly the task of EL, which is the aim of this thesis, is to extract a set of mention-entity pairs from an input text d. In this thesis, we detect mentions of ORG and GRT types, and only link ORG mentions to entities in KB. Formally, we extract the set $T = E_{ORG} \cup M_{GRT}$ for each input d, where,

$$E_{ORG} = \{(m_i, e_i) \mid m_i \in M \wedge e_i \in \mathcal{E} \wedge link(m_i) = e_i \wedge type(m_i) = ORG\} \tag{3.1}$$

and

$$M_{GRT} = \{m_i \mid m_i \in M \wedge type(m_i) = GRT\}. \tag{3.2}$$

3.2 Background

3.2.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a language model introduced by Devlin et al. in 2019, that obtained new state-of-the-art results on many different NLP tasks with large gains in performance [18]. BERT essentially consists of bidirectional Transformer [82] blocks stacked together. It can use a single sentence or multiple sentences together as input. To represent the input sequence, first, tokenization is performed using WordPieces [91]. Each token is then represented with the sum of three different embeddings: WordPiece embeddings, position embeddings and segment embeddings, the last one indicating which sentence a token belongs to. After that, special tokens are added to the sequence such that each input sequence starts with a [CLS] token, and ends with a [SEP] token. The latter is also used to separate sentences when the input consists of two sentences.

The authors of BERT criticize previous neural language models on the fact that they are learning unidirectional representations. They argue that this can deteriorate the performance for token-level tasks where information from both directions are very

important. By using Masked Language Modeling (MLM) task, they manage to train a representation utilizing both left-to-right and right-to-left directions simultaneously. This task refers to masking words randomly and predicting the masked words only using the context. BERT also utilizes the Next Sentence Prediction (NSP) task, in order to learn the relationship between two input sentences. For this task, the final hidden vector corresponding to the [CLS] token is used to distinguish whether the two input sentences are actually adjacent or not.

BERT is pretrained with BookCorpus [102] and English Wikipedia. The authors introduce two different architectures, BERT_{BASE} and BERT_{LARGE}, which have 110M and 340M parameters respectively. In this study, BERT_{BASE} is used due to computational limitations. For each architecture, there are also two different versions based on case-sensitivity. The authors obtain state-of-the-art results in various NLP problems by adding minimal task-specific layers and fine-tuning all the parameters end-to-end.

3.2.2 Domain Adaptation of BERT

Previous work suggests that pretraining a BERT model with in-domain data improves the overall performance of the model for the task at hand [34, 78, 26]. For funding data extraction, the input sentences are the ones where authors acknowledge the financial support they had received. Hence, it is trivial that these sentences differ significantly from the data that BERT was pretrained on, i.e. books and Wikipedia. For this purpose, a domain-relevant BERT, denoted by BERT_{SC}, is trained. The choice of terminology is attributed to the fact that the training data consists of a subset of articles that can be found in Scopus[1], one of the largest database for peer-reviewed literature. For each article, Scopus displays the corresponding funding text, using artificial intelligence solutions developed by Elsevier. These texts are used as training data for BERT_{SC}.

To pretrain BERT_{SC}, Task-Adaptive Pretraining (TAPT) schema proposed by Gururangan et al. [34] is used. The idea behind TAPT is to pretrain a BERT model, which was pretrained on a generic dataset, using unlabelled data from the specified task with MLM objective. The authors compare this approach with Domain-Adaptive Pretraining (DAPT), which they define as pretraining a BERT model from scratch using documents from a specific domain. DAPT is much more expensive in terms of both data and computational power compared to TAPT, and yet the authors show that TAPT performs comparable to DAPT. Although there are other works on adapting BERT to a specific domain in an inexpensive way [78, 69], TAPT was chosen due to its easy-to-use, open-source implementation[2].

3.3 Named Entity Recognition

To train Flair^{NER}, BERT^{NER} and BERT_{SC}^{NER}, the NER problem is cast as a token classification task using the IOB tagging schema. In this schema, the initial tokens of the mentions are labelled with a "B" ("Beginning") and the remaining tokens are labelled with an "I" ("Inside"). The tokens that are not a part of any mention are labelled with "O" ("Outside"). In NER, there may be different types of mentions. In that case, the type information is appended after the "B" and "I" labels. Since there are two types in this work, *Organization* and *Grant*, a total of 5 tags are used: $\{\text{B-ORG}, \text{I-ORG}, \text{B-GRT}, \text{I-GRT}, \text{O}\}$.

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

[2] `https://github.com/allenai/dont-stop-pretraining`

Flair^{NER}

The sequence labelling architecture proposed by Akbik et al., denoted by Flair^{NER} in this work, utilizes a BiLSTM-CRF [41] which can be trained with the labelled task dataset. The novelty of their approach comes from the way that the input is represented. In this paper, the authors propose Contextualized String Embeddings (CSE). CSE represent a word based on its characters and are contextualized, meaning that the representation of a word changes depending on its context. And as the words are represented using characters, the vocabulary size is smaller compared to word-level language models. CSE are formed by concatenating a forward and a backward language model. Both language models are trained to predict the next character given the previous characters using an LSTM [37]. For an input $I = \{c_1, ..., c_k\}$ of k characters, the contextual string embedding CSE_w of a word $w = \{c_s, ..., c_e\}$ can be defined as:

$$CSE_w = [LSTM^{forward}_{Flair}(c_e|c_0, ..., c_{e-1}); LSTM^{backward}_{Flair}(c_s|c_k, ..., c_{s-1})] \tag{3.3}$$

where $LSTM^{forward}_{Flair}$ and $LSTM^{backward}_{Flair}$ denote the forward and backward language models respectively. Apart from CSE, the sequence labeling architecture uses pretrained GloVe [67] word embeddings as well. For each word w, these embeddings are concatenated with CSE_w resulting in the representation v_w:

$$v_w = [CSE_w; GloVe_w] \tag{3.4}$$

where $GloVe_w$ denotes the corresponding GloVe embeddings. The authors show that this representation achieves a higher performance compared to only using CSE, and hypothesize that GloVe embeddings may be capturing some semantic information that can complement CSE. However, the authors do not report significant performance gains when using additional task-specific character features, concluding that this information is captured by CSE. The resulting input representation is used by the BiLSTM-CRF model. Given an input string $d = \{x_1, ..., x_N\}$ of N words, let r_i be the BiLSTM output for word i, such that:

$$r_i = BiLSTM_{Flair}(v_{x_1}, ..., v_{x_N})[i]. \tag{3.5}$$

Since there are 5 labels in this task, $r_i \in \mathbb{R}^5$, modelling the probability distribution of each label. However, in NER, the labels of each word/token are not independent. For example, following the IOB schema, there cannot be an I label following an O label. Hence, instead of assigning a label to each token based on r_i, the probability of the labels of the whole sequence is calculated. For this purpose, a CRF layer is utilized. Then, the probability of each possible label sequence Y_{Flair} is calculated as:

$$P(\mathrm{Y}_{Flair}) = \prod_{i=1}^{N} \exp(\mathbf{W}_{(y_{i-1} \to y_i)} r_i + \mathbf{b}_{(y_{i-1} \to y_i)}) \tag{3.6}$$

where the matrices $\mathbf{W}$ and $\mathbf{b}$ store the weights and biases of each label transition. The label sequence with maximum probability is chosen as the final prediction. $LSTM^{forward}_{Flair}$ and $LSTM^{backward}_{Flair}$ are trained separately, and their parameters are frozen during the training of the other components added for NER. For the remaining parameters, the loss is set to the score difference of the gold label sequence and predicted label sequence.

BERT^{NER} and BERT$^{NER}_{SC}$

BERT^{NER} is the same NER architecture proposed by Devlin et al. [18]. Let d_t be the WordPiece-tokenized version of the input string $d = \{x_1, ..., x_N\}$ of N words, such that:

$$d_t = \{x_{11}, ..., x_{1l_1}, ..., x_{N1}, ..., x_{Nl_N}\} \tag{3.7}$$

where l_i corresponds to number of WordPiece tokens for word i. After appending the special tokens [CLS] and [SEP], d_t is passed through a case-sensitive BERT model (BERT_{BASE} in this study) to get the hidden state vectors of the last Transformer block for each WordPiece token. The words are represented by their first WordPiece token's vector. A linear layer with weights $\mathbf{W}_{BERT} \in \mathbb{R}^{768\times5}$ and bias $\mathbf{b}_{BERT} \in \mathbb{R}^{1\times5}$ is applied to to get the scores for each label and for each word $\mathbf{Y}_{BERT} \in \mathbb{R}^{N\times5}$, such that:

$$\mathbf{Y}_{BERT} = \mathbf{W}_{BERT}\ \text{BERT}_{BASE}(d_t)[x_{11}, ..., x_{i1}, ..., x_{N1}] + \mathbf{b}_{BERT}. \tag{3.8}$$

Lastly, the label with the highest score is selected for each word, resulting in the predicted label sequence $\hat{\mathbf{Y}}_{BERT} \in \mathbb{R}^N$, such that:

$$\hat{\mathbf{Y}}_{BERT}[i] = \underset{1\leq j\leq 5}{argmax}\ \mathbf{Y}_{BERT}[i]\ ,\ \ 1 \leq i \leq N. \tag{3.9}$$

Following Devlin et al. [18], the whole architecture is fine-tuned end-to-end. Cross-entropy loss over the NER labels is used for training. BERT_{SC}^{NER} has the same architecture and training strategy with BERT^{NER}, the only difference is that BERT_{BASE} is changed with BERT_{SC} in Equation 3.8.

3.4 Entity Disambiguation

BLINK [90] utilizes a candidate selector (CS) and a reranker to tackle the ED problem. The aim of candidate selection is to reduce the number of possible entities to consider for a given mention in a computationally cheap manner. This enables the usage of a more expensive but better algorithm on the reduced entity space to select the best entity, or to select no entity at all. The CS extracts a candidate set C_i from the entity set $\mathcal{E} - \emptyset$ for each mention m_i , such that:

$$C_i = \{e_1, ...e_K\} \tag{3.10}$$

where K is the number of candidates. When $K = 1$, the system can work as a single ED component. In this work, only the biencoder of BLINK is implemented as fast inference is a must. A linear reranker model is trained instead of the cross-encoder. This section details the architectures used in BLINK and present the modifications made to adapt them to the task at hand.

Biencoder in BLINK: BI_{BL}

The biencoder architecture used in BLINK is shown in Figure 3.1 (left). The architecture consists of two BERT models, $M_{Mention}$ and M_{Entity}, first for encoding the mention representation $R_{Mention}$ and the second for encoding the candidate entity representation R_{Entity}. $M_{Mention}$ takes the tokenized version of $R_{Mention}$ as input, and outputs BERT's final hidden vectors for each token. M_{Entity} does the same for R_{Entity}. The mention representation used in this work is identical with that of BLINK:

$$R_{Mention} = \text{[CLS] left context } [M_s] \text{ mention } [M_e] \text{ right context [SEP]} \tag{3.11}$$

where $[M_s]$ and $[M_e]$ are two WordPiece tokens selected among the unused tokens of BERT. The aim of these tokens is to distinguish the mention from its context. The candidate representation is different from BLINK which uses the title and entity description.

In this work, the entity descriptions are not available. Each entity is associated with various labels and the country of origin. Both the labels and the country are crucial for disambiguation of funding organizations, and the latter helps tremendously with ambiguous organization labels. Hence, the candidate entities are represented as:

$$R_{Entity} = \text{[CLS] label}_1 \ [E_l] \ ... \ [E_l] \ \text{label}_{N_{label}} \ [E_c] \ \text{name}_{Country} \ \text{[SEP]} \tag{3.12}$$

where N_{label} denotes the number of available labels per funding organization, and $\text{name}_{Country}$ is the name of the country as stated in the GeoNames[3] database. $[E_l]$ and $[E_c]$ are special tokens similar to $[M_s]$ and $[M_e]$, and their aim is to separate different labels and the country information. The representations are passed through the BERT models, and the hidden state vectors of the last Transformer layer corresponding to the [CLS] token are extracted to obtain the representations r_{Entity} and $r_{Mention}$, such that:

$$r_{Entity} = M_{Entity}(R_{Entity}) \ \text{[CLS]} \tag{3.13}$$

and

$$r_{Mention} = M_{Mention}(R_{Mention}) \ \text{[CLS]}. \tag{3.14}$$

Then, the score of the mention and the candidate is defined to be the dot product of r_{Entity} and $r_{Mention}$:

$$Score(Mention, \ Entity) = r_{Mention} \cdot r_{Entity}. \tag{3.15}$$

The candidate entity set C_i of a mention m_i is set to the top K entities for which the score is highest, such that:

$$C_i = \{e_1, ..., e_K | \ \nexists \hat{e} \ Score(m_i, \hat{e}) > \min_{e \in C_i}(Score(m_i, e)); \hat{e} \notin C_i; \ \hat{e} \in \mathcal{E} - \emptyset; \ C_i \subset \mathcal{E} - \emptyset\}. \tag{3.16}$$

In this work, different from BLINK, both $M_{Mention}$ and M_{Entity} are initialized with BERT_{SC}. To find the top K entities for a mention, BLINK utilizes FAISS [45], which performs approximate nearest neighbor search. Since the entity space is much smaller in this work compared to BLINK, with 26k entities to 5.9M entities, an exact nearest neighbor search is done instead of using an approximation.

To train the biencoder, same strategy proposed in BLINK is used. For each mention, the loss is computed as:

$$L(m_i, e_i) = -Score(m_i, e_i) + \log \sum_{\hat{e} \in IE} \exp(Score(m_i, \hat{e})) \tag{3.17}$$

where e_i is the correct entity for mention m_i, $link(m_i) = e_i$. IE stands for incorrect entities, hence, the second term in the loss function corresponds to the score between the mention and incorrect entities. The set IE is selected as in-batch negatives, i.e. the correct entities of other mentions in the same batch. In addition to these in-batch negatives, for each mention, top Neg_H highest scoring entities are extracted and added to IE. These entities are referred to as hard negative examples. Neg_H is the number of hard negatives and is a hyperparameter. In this work, when the loss is calculated for NIL mentions, the first term of the loss equation is set to 0:

$$L(m_i, e_i) = \log \sum_{\hat{e} \in IE} \exp(Score(m_i, \hat{e})), \text{ if } e_i = \emptyset. \tag{3.18}$$

[3] `https://www.geonames.org/`

This whole architecture will be referred to as BI_{BL}, where BI and BL stand for biencoder and BLINK respectively. BI_{BL} has some downsides for this problem setting. Even though they are not included in BLINK's research [90], NIL mentions are very important for this work. These mentions cover between 15-20% of the dataset at hand (see Section 4.1.1) and give important insights on funding organizations that are not yet included in the KB. Having to handle NIL mentions introduces additional challenges on using BLINK's architecture for this problem. For example, the loss used cannot handle NIL mentions naturally. Hence, a threshold is applied on the score of the returned entity to detect the NIL mentions. As dot product is unbound, every score is scaled between 0 and 1 using Min-Max Scaling with the minimum and maximum score values observed in the training set. The optimum threshold is selected on the training set using Grid Search over the interval $[0, 1]$ with a step size of 0.001.

Due to the issues described above, a modified version of BI_{BL} is proposed, which is named BI_{AD}, where AD stands for adapted.

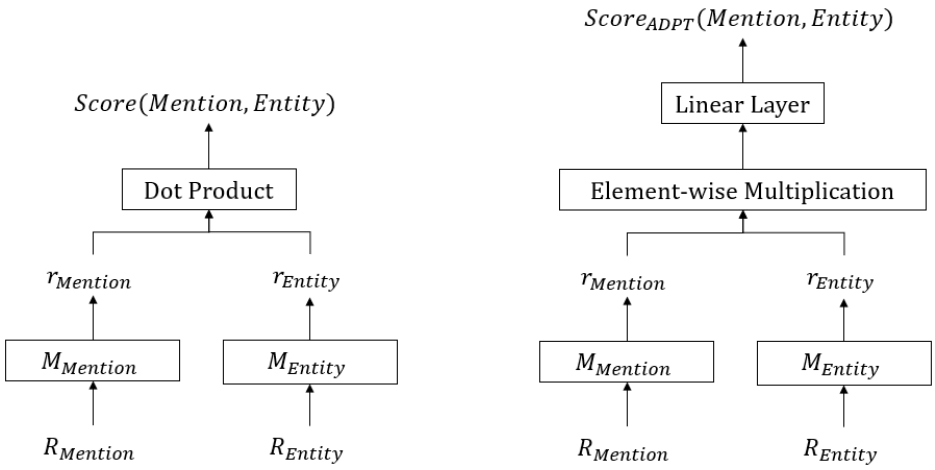


Figure 3.1: BLINK Biencoder architecture BI_{BL} (left) and the adapted Biencoder architecture BI_{AD} (right)

BI_{AD}

Different from BI_{BL}, BI_{AD} uses a binary classification setting. For this purpose, a training set is constructed. Every mention and the corresponding correct entity are added with a positive label, and the incorrect entities for a mention are added with a negative label. Hence, the dataset S has the form:

$$S = \{(m_k, e_k, l_k) \mid l_k \in \{0, 1\} \wedge l_k = 1 \iff link(m_k) = e_k\}. \tag{3.19}$$

where m_k is a mention, $e_k \in \mathcal{E} - \emptyset$ is an entity and l_k is the binary classification label of that sample. To get the class probabilities for each mention and entity pair, an additional linear layer is used. Then, the score of a pair is defined to be the probability of the positive class:

$$Score_{AD}(Mention,\ Entity) = \mathbf{W}_{AD}\ (r_{Mention} \odot r_{Entity})[positive] \tag{3.20}$$

where $\odot$ refers to element-wise multiplication of the vectors, and $\mathbf{W}_{AD} \in \mathcal{R}^{768x2}$ is a matrix denoting the weights of the additional linear layer[4]. The binary cross-entropy loss L_{AD} is used to train this model:

[4]the size of the final hidden vectors of BERT_{SC} is 768 as it is trained from BERT_{BASE}

$$L_{AD}(m_i, e_i, l_i) = -l_i \log(Score_{AD}(m_i, e_i)) - (1 - l_i) \log(1 - Score_{AD}(m_i, e_i)). \quad (3.21)$$

To get incorrect (negative) entities for each mention, a different strategy is used. For the initial round of training, Neg_R entities are sampled from the entity set for each mention randomly. Here, Neg_R is a hyperparameter. As Neg_R increases, the size of the dataset increases and hence the training becomes longer. Even though in-batch negatives are much faster to use, the advantage of this strategy is the possibility to show the model some entities that do not have any correct mention in the KB. Since the entity distribution of the dataset used is highly skewed (see Figure 4.2), it is inevitable that the same entities are used as the random negatives for most of the mentions when in-batch negatives are used.

As in BLINK, hard negatives are defined as the top Neg_H entities predicted for a given mention. Different from BLINK, following DEER [31], an entity is considered to be a hard negative only if it is ranked above the correct entity. Hence, some mentions may have less hard negative samples compared to others. NIL mentions always have Neg_H number of hard negative samples. One may argue that the entities that have a score lower than 0.5 should not be considered as a hard negative since they are predicted as the negative class, empirical results suggested that adding such entities helped. Another advantage of this setting is, since hard negatives are not added to the random negatives for loss calculation, it is possible to increase Neg_H without the need for additional memory.

The training is done in rounds, similar to BLINK (and BI_{BL}) and DEER. In each round, a new set of hard negatives and random negatives are sampled, hence the training set is updated. However, hard negatives are added starting from the second round, as initially some of the model weights are initialized randomly, such as $\mathbf{W}_{AD}$. In this work, one round may correspond to one or more epochs (see Section 4.1.3)

The number of random negatives in each round, Neg_R is also a hyperparameter. In the first round of training, Neg_R is set to 3 as the training time increases proportionally with Neg_R. In the following rounds, Neg_R is selected based on the number of hard negative samples. After extracting all the hard negatives for the training set, the total number of such samples are calculated, which can be denoted as Neg_H^{Sum}. Then for each mention, $Neg_R = \lfloor Neg_H^{Sum}/(\text{Number of Mentions}) \rceil$ entities are sampled randomly and added to the dataset. This way, it is aimed to have a similar proportion between random and hard negatives in the training set. The aim with this was to somehow mimic the strategy of DEER, which uses a multi-task learning setup with equal weights to incorporate both hard and random negatives.

One interesting property of the setup of BI_{AD} is that for the mentions with more hard negatives, i.e. mentions that are ranked lower by the model, the training set has more hard negatives compared to random negatives. We hypothesize that the need for hard negatives are higher for these kind of mentions, as it seems that the random negatives were not enough to teach the model to make the correct decision in the first round.

In the setup for BI_{AD}, it is intuitive to add NIL mentions to training. The only difference between a NIL mention and a non-NIL mention is that the former does not have any instance in the training set with a positive label. Also, when BI_{AD} is used as an ED system by itself, it is straightforward to detect NIL mentions. Since this is a binary classification task, if the highest scoring entity of a mention has a score lower than 0.5, it should be a NIL mention, as there are no mention-entity pairs including that mention such that the label is positive.

The architecture of BI_{AD} is shown in Figure 3.1 (right).

Candidate Reranking

Instead of using the computationally expensive cross-encoder, feature-based models (logistic regression and a Gradient Boosting Machine (GBM) [27]) are used as a reranker. After selecting the best biencoder, the number of candidates K is determined by investigating the change of recall with respect to the rank. After K is selected, a training dataset is created by extracting the top K entities for each mention. For the instances where the entity is the correct link for the mention, the class label is set to positive, and for the other instances, it is set to negative.

For the reranker, we make use of 5 features, chosen from the top important features of an existing ED model at Elsevier: score assigned by the candidate selector, maximum sorted Levenshtein similarity between the mention and the labels of the candidate entity, popularity of the candidate entity, link probability of the mention, and commonness. To calculate Levenshtein similarity, the normalized Levenshtein distance is subtracted from 1. For calculating the maximum sorted Levenshtein similarity for a given mention and entity, first the mention and the labels of the entity are tokenized. Then, the tokens of the mention and each label are sorted in alphabetic order. Levenshtein similarity is calculated between the mention for each entity label. The value of the feature is set to the maximum of these similarities. The sorted Levenshtein similarity is calculated using the Fuzzywuzzy library[5].

The popularity of an entity is defined as the number of times it appears as a link divided by the number of all links. Similarly, the link probability [4] of a mention is defined as the number of times the mention is linked to any entity in the KB divided by the number of times it appears in the dataset used for estimation. Commonness measures the maximum likelihood probability of an entity being the link to the given mention [4]. For a mention-entity pair, it is calculated as number of times that the mention was linked to the given entity, divided by the number of times that the mention appears as a link [4]. For popularity, link probability and commonness, the statistics used by the feature-based ED model is utilized. It is made sure that no document that is contained in the evaluation dataset (Dev, Validation and Test splits, see Section 4.1.1) is included in the estimation of these statistics.

First, a logistic regression model is trained as a reranker. This model will be denoted by LM which stands for *Linear Model*. A logistic regression model can be defined as:

$$y_{LM}(\phi_{LM}) = \sigma(\boldsymbol{w}_{LM}^{\mathrm{T}}\phi_{LM}) \tag{3.22}$$

where σ, y_{LM}, $\boldsymbol{w}_{LM}$ and ϕ_{LM} denote the sigmoid function, the probability of positive class, weights and the feature vector respectively [5]. One of the feature values in ϕ_{LM} is set to 1 to include bias. Hence, the length of ϕ_{LM} is 6, considering 5 features are used. Weights of the model are learned during training, which are optimized by SAGA [17]. Binary cross-entropy is used as the loss function, with both L_1 and L_2 penalty. The loss function for LM, Loss_{LM}, can be defined as:

$$\mathrm{Loss}_{LM}(a) = -t_a \log(y_{LM}(\phi_{LM}(a))) - (1 - y_a) \log(y_{LM}(\phi_{LM}(a))) + \mathrm{R}_{LM} \tag{3.23}$$

for a single training sample a. Here, $\phi_{LM}(a)$ and t_a denote the feature vector and the label for a sample a consecutively. R_{LM} is the regularization term and corresponds to Elastic Net regularization [103] as both L_1 and L_2 penalties are utilized.

In the training data, the negative class is prominent as there is only a maximum of one positive entity per mention. To address this, class weights are used during the binary cross-entropy loss calculation for LM. Each class is assigned a weight that is inversely proportional to the number of samples of that class in the training data normalized by

[5] `https://github.com/seatgeek/fuzzywuzzy`

the total number of samples. Two hyperparameters of LM is tuned by grid search: the regularization strength and the ratio between L_1 and L_2 penalties. For the former, the search interval is set to [0,1] with a step size of 0.1, and for the latter, it is set to [1,10] with a step size of 1. Feature selection is also performed on all possible feature sets, excluding the ones that do not contain the score of the candidate selector as feature. Then, the best hyperparameter configuration and feature set is selected.

A GBM model is also trained with the same features that LM is using. This model will be referred to as GBM_{F5} as it uses 5 features.

Boosting refers to combining multiple weak (base) classifiers in an ensemble setting to produce a classifier that performs significantly better compared to these weak classifiers [5]. The base classifiers are trained sequentially, giving a higher weight to the training samples that the previously trained classifiers performed poorly on [5]. During inference, a weighted voting schema is used, where the weights are related to the performance of the corresponding base classifiers [5]. Decision trees [9] are used as the base classifiers [27], and are learned by utilizing the negative gradients [48]. For the mathematical details, we refer the reader to Friedman [27].

During inference, the probability of the positive class is extracted for each mention-candidate entity pair. The mention is linked to the candidate entity with the highest score, if the score is greater than or equal to 0.5. Otherwise, the mention is linked to NIL. For experimental purposes, another NIL-mention detection threshold is selected within the interval [0.5,1] with using grid search with a step size of 0.001. For GBM_{F5}, this interval had to be extended to [0,1] as the model performed poorly with high thresholds.

Chapter 4

Evaluation

In this work, different experiments were conducted to investigate the optimal Entity Linking (EL) approach for funding information extraction. First, BERT_{SC}, a BERT model that is adapted to funding text, is pretrained using the Task-Adaptive Pretraining (TAPT) strategy proposed by Gururangan et al. [34]. Then, different state-of-the-art Named Entity Recognition (NER) components are compared and a neural Entity Disambiguation (ED) model is developed. Lastly, the end-to-end EL performance of these approaches are investigated.

In Section 4.1, the experimental setup is presented. The dataset used and the evaluation metrics are detailed in Section 4.1.1 and 4.1.2 respectively. The training, hyperparameters and model selection is shown in Section 4.1.3. Lastly, the result and the analysis are presented in Sections 4.2 and 4.3.

4.1 Experimental Setup

4.1.1 Data

The dataset for funding data extraction and the knowledge base (KB) for ED used in this research are provided by Elsevier B.V.. The dataset consists of a set of labelled articles annotated by humans. To create this dataset, each article is annotated by three people. First, two annotators extracted the funding information from the articles independently. Then, a third annotator harmonized the decisions of the previous two annotators, resolving the conflicts if necessary.

For developing models and evaluating various approaches, the dataset is divided into four subsets: Training, Dev, Validation and Test. The Training split is used to train the models. Dev split is used to monitor the progress of training, while Validation split is used to select the best approach for each task. The intermediate error analyses are done on the Dev split. Test is used to evaluate only the feature-based models and the selected approach for each task. The splits are arranged such that there is no overlap in terms of articles. Table 4.1 shows the number of annotated articles contained in each split.

For the ED and EL task, the KB provided by Elsevier is used. This KB contains 26,892 entities of funding organizations with information such as the country of origin, type of organization and different names that the organization can be referred to with. There are also sparse amount of relations between organizations to show affiliations and hierarchies. One interesting property of the KB is that most of the entities do not exist in general-purpose knowledge repositories. Hence, it is not trivial to obtain more information from other sources.

Dataset Split	Number of Articles
Training	37,484
Dev	1,000
Validation	4,000
Test	19,920

Table 4.1: Number of articles annotated with funding information in each dataset split

Sometimes, organizations may change their names, or may be merged with other organizations. Hence, it is possible that one funding organization is referenced by multiple entities in the KB, which is not desirable. To prevent this, entities are grouped together based on the relations indicating such cases. This operation resulted in 25,859 entity groups. It is assumed that the entities in each group refer to the same organization, and hence can be used interchangeably. Another option could be to only keep the newest versions of the entities. However, this may cause problems with disambiguating older publications, where the authors may have used an older name variant to refer to the same organization.

The data to train and evaluate the approaches for different tasks are derived from this main dataset. However, as each task has a different nature, some preprocessing and filtering is applied when necessary.

Task-Adaptive Pretraining (TAPT)

For EL in funding domain, the input text is the sentences where the authors acknowledge the funding support they had received for their research. As TAPT can be done with unlabelled data, 13 million such sentences are extracted from Scopus, where they are displayed for each article. Using the identifiers of the articles, the sentences belonging to the articles in Dev, Validation and Test splits are removed.

Named Entity Recognition (NER)

As mentioned in Section 3.3, IOB tagging is used to train and evaluate the NER models. In the dataset used for this work, the annotations are not done in terms of tokens, but in terms of character spans of the input text. That is, each gold mention is provided using their character offsets with respect to the article text. Hence, first the input text is tokenized and the labels are assigned to tokens based on some predefined rules to tackle some edge cases that mostly correspond to annotation errors. These rules are extracted based on empirical results to maximize the correctness of the annotations. The experiments were done on a portion of the training set, and all the edge cases found were present for less than 0.5% of the investigated dataset. In Appendix B the details of the labeling step can be found.

Dataset Split	#Articles	#Sentences	#ORG Mentions	#GRT Mentions
Training	22,720	26,132	67,671	45,263
Dev	1,000	1,284	4,333	2,770
Validation	4,000	5,012	16,355	10,112
Test	13,851	15,590	37,495	25,349

Table 4.2: Dataset splits and statistics for NER. For each split; number of articles with at least one funding sentence, number of sentences and number of mentions are shown.

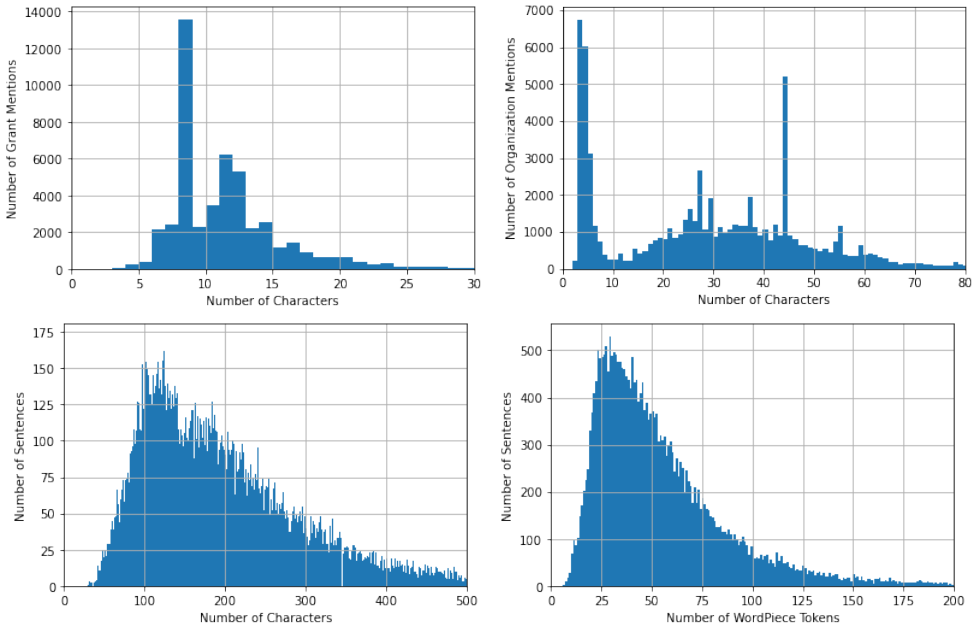


Figure 4.1: Distribution of length of grant mentions (top-left), organization mentions (top-right), sentences (bottom-left) in terms of number of characters and distribution of number of tokens per sentence (bottom-right). Plots are cut over the x-axis, and the maximum x values are 75, 223, 9175 and 2760 respectively.

The NER component should extract the mentions of organizations only if they funded the corresponded research. For this purpose, the classifier developed by Elsevier is used as a preprocessing step. This classifier identifies whether a sentence contains funding information or not. As the NER component will directly work with this classifier, as a preprocessing step, the dataset splits are reduced to the sentences that are identified as positive by this classifier. The second column of Table 4.2 shows the number of articles with at least one sentence with funding information for each dataset split, and the third column shows the total number of such sentences. The number of organization and grant mentions on each split can be found in the fourth and fifth columns respectively. In Figure 4.1, the distribution of number of characters for the sentences and mentions contained in the Training and Dev splits are presented.

Entity Disambiguation (ED)

Table 4.3 displays the statistics for each dataset split. The second column of this table shows number of articles with at least one organization mention. It also can be seen that not all organization mentions have a corresponding entity in the KB. These mentions will be referred to as NIL mentions. A mention being NIL means that a funding organization is extracted by the annotators, however, as this organization was not yet included in the KB, it was not linked. Because of this, all NIL mentions for this task can be classified as emerging entities (EE). These are very important for this work, as the current KB is being updated regularly consulting to the detected EEs.

Another interesting property of this dataset is that only the entities belonging to a small part of the KB appear as a link, as can be seen from Table 4.4. For example, only 26.9% of the entities in the KB appear as a link in the Training split. In addition, the Dev, Validation and Test splits contain links to entities that do not appear in the

Dataset Split	#Articles	#ORG Mentions	#Links	NIL Mentions
Training	29,118	95,761	77,972	18.58%
Dev	991	5,618	4,749	15.47%
Validation	3,943	19,765	16,689	15.56%
Test	17,333	52,378	42,514	18.83%

Table 4.3: Dataset splits and statistics for ED. For each split, number of articles with at least one organization mention, number of organization mentions, number of mentions that are linked to an entity and the percentage of NIL mentions are shown.

Dataset Split	# Unique Entities	Overlap with Training	KB Coverage
Training	7,234	-	26.9%
Dev	1,222	88.63%	4.54%
Validation	2,658	81.6%	9.88%
Test	5,590	71.91%	20.79%

Table 4.4: Dataset splits and number of unique entities in each split. Third column indicates the percentage of unique entities that are also present in the Training split for Dev, Validation and Test splits. The last column shows the percentage of KB covered by each split.

Training split. However, when Table 4.5 is investigated, it is possible to see that such instances are long-tail entities. For example, even though 28.1% of the unique entities in the Test split do not appear in the Training split; when the overall number of links are checked, only 5.14% of the links are to these entities. Nevertheless, it is important that the ED system can link mentions of unseen entities correctly as well.

Figure 4.2 shows the number of occurrences of the top 25 most frequent entities on Training and Dev splits. It can be seen that the distribution is highly skewed, even with the most common entities.

Entity Linking (EL)

In this work, the task of EL is tackled in two-steps, NER and ED. Hence, the data described is used to evaluate the end-to-end performance of NER and ED steps together.

As the NER is also evaluated here implicitly, the Sentence classifier is used again to limit the dataset to the sentences detected by this classifier, as done for the NER task dataset. Also, all the NIL mentions are classified as emerging entities, due to the same reason reported for the ED task dataset. In fact, the dataset for EL is a subset of that of ED, limited by the sentence classifier. Tables 4.6, 4.7 and 4.8 show statistics of the dataset splits such as number of mentions, links and percentage of NIL mentions.

Dataset Split	# Links	Links not in Training
Dev	4,749	3.39%
Validation	16,689	3.52%
Test	42,514	5.14%

Table 4.5: Dataset splits and number of links. The third column shows the percentage of links for which the target entity do not exist as a link in the Training split.

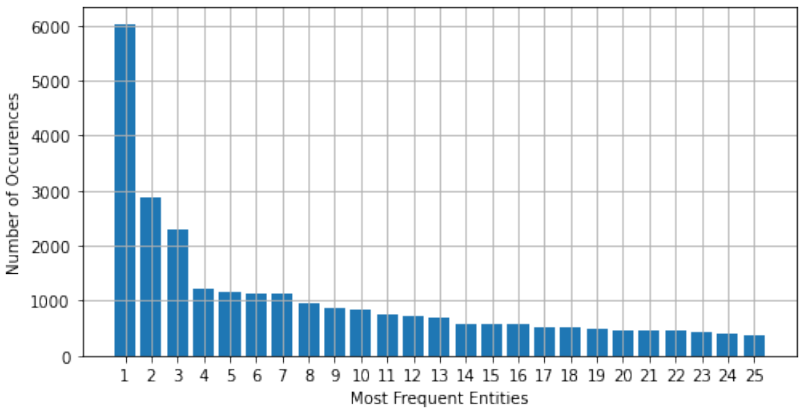


Figure 4.2: Number of occurrences of top 25 most frequent entities. Statistics obtained using Training and Dev splits.

Dataset Split	#Articles	#ORG Mentions	#Links	NIL Mentions
Validation	4,000	16,276	13,958	14.24%
Test	13,851	37,340	31,153	16.57%

Table 4.6: Dataset splits and statistics for EL. For each split, number of articles with at least one organization mention, number of organization mentions, number of mentions that are linked to an entity and the percentage of NIL mentions are shown.

Dataset Split	# Unique Entities	Overlap with Training	KB Coverage
Validation	2,302	83.36%	8.57%
Test	4,350	76%	16.18%

Table 4.7: Dataset splits and number of unique entities in each split. Third column indicates the percentage of unique entities that are also present in the Training split. The entities in the Training split is determined using the dataset for the ED task. The last column shows the percentage of KB covered by each split.

Dataset Split	# Links	Links not in Training
Validation	13,958	3.19%
Test	31,153	4.28%

Table 4.8: Dataset splits and number of links. The third column shows the percentage of links for which the target entity do not exist as a link in the Training split. The entities in the Training split is determined using the dataset for the ED task.

4.1.2 Evaluation Metrics

Each different problem tackled in this work is evaluated with a suitable metric selected from the literature.

Task-Adaptive Pretraining (TAPT)

To evaluate BERT$_{SC}$ and monitor its progress, Perplexity is used. This metric corresponds to the inverse probability of the dataset based on the model [29], and it it is the

most popular metric to evaluate language models [29].

Named Entity Recognition (NER)

To evaluate the NER task, precision recall and F1 scores are used for each entity type, Organization and Grant. These metrics are defined in terms of True Positives (TP), False Positives (FP) and False Negatives (FN). Precision is defined as the fraction of TPs among all mentions extracted by the system, and recall is defined as the fraction TPs among all ground truth mentions. F1 score is the harmonic mean of precision and recall metrics. The formulas of these metrics are shown in Equations 4.1,4.2 and 4.3. A mention is considered to be a TP if and only if both the extracted span and type information is correct. A FP corresponds to a mention that is extracted by the system wrongly, and a FN corresponds to a mention that is not extracted by the system while being present in the ground truth. This scheme is chosen as it is inline with evaluation of the CoNLL-2003 NER task [79].

$$\text{Precision} = \frac{\#TP}{\#TP + \#FP} \tag{4.1}$$

$$\text{Recall} = \frac{\#TP}{\#TP + \#FN} \tag{4.2}$$

$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.3}$$

Entity Linking (EL)

To evaluate the EL task, a strategy inspired by GERBIL [80] is used. GERBIL is a framework for evaluating various entity-related tasks, such as NER, ED and EL. The framework supports many popular databases, hence, systems evaluating on such datasets can use it when they provide an API support. However, since this work is using a domain-specific and private dataset, it is not possible to make use of it directly. Hence, the metrics are reimplemented consulting the paper [80] and the GitHub repository[1].

GERBIL offers various settings to evaluate systems. In this work, "Normal", "EE" (Emerging Entities) and "InKB" (In Knowledge Base) settings are used for evaluation and the results for each are reported separately.

To calculate the scores in "Normal" setting, for each document, the TP, FP and FN instances are counted. First, for each gold annotation, a matching annotation is looked for in predictions. Two annotations are considered to be matching based on strict criteria. If a match is found, this counts as a TP. Gold annotations for which no match is found are counted as FN. Similarly, the predictions that were not marked as a match for any gold annotation are counted as FP. If a prediction is marked as a match for a gold annotation, it could not be matched again. And, a gold annotation could not be matched with more than one prediction. After obtaining the TP, FP and FN counts; precision, recall and F1 score for that document are calculated. The precision, recall and F1 score for all the documents are averaged to produce macro averaged results. To obtain micro averaged results; the TP, FP and FN counts of all documents are summed together before calculating precision, recall and F1 score.

GERBIL distinguishes NIL mentions to two, emerging entities and ones where the system cannot produce a link. In this work, for the ED task, it is known that all NIL mentions are emerging entities, and the systems developed are not able to make such distinction between NIL mentions. Hence, for the EL task, the mentions extracted by the NER components are also assumed to be emerging entities, even though this may

[1] `https://github.com/dice-group/gerbil`

not always be the case. Based on this assumption, to get the scores for the "EE" setting, the TP counts are discarded when the annotations both contained an entity that is in the KB. The gold annotations that were not matched with any prediction did not count as FN if the entity was in the KB. Similarly, the predictions that were not matched did not count as FP if the entity was in the KB. For the "InKB" setting, the TP counts are discarded when the entities were both NIL. The gold annotations that were not matched with any prediction did not count as FN if the entity was NIL. And, the predictions that were not matched did not count as FP if the entity was NIL.

Algorithm 1 shows the pseudocode for calculating TP, FP and FN for a document in "Normal", "InKB" and "EE" settings.

Algorithm 1: Calculate TP, FP, FN per document for EL evaluation.

```
// Initialize the counts.
TP = 0, FP = 0, FN = 0;
// N gold annotations, M predicted annotations.  Each annotation
   consists of mention start index, mention length, and predicted
   entity (or NIL).
g = [g_0,..., g_N];
p = [p_0,..., p_M];
// Evaluation mode ''Normal'', ''InKB'' or ''EE''
mode = "Normal";
for gold in g do
    found = False ;
    for pred in p do
        if gold == pred then
            if mode=="Normal" then
                TP += 1;
            else if mode=="InKB" and Entity not NIL. then
                TP += 1;
            else if mode=="EE" and Entity is NIL then
                TP += 1;
            p = p - pred;
            found = True;
            break;
    if not found then
        if mode=="Normal" then
            FN += 1;
        else if mode=="InKB" and Entity of gold is not NIL then
            FN += 1;
        else if mode=="EE" and Entity of gold is NIL then
            FN += 1;
if mode=="InKB" then
    remove predictions from p where entity is NIL;
else if mode=="EE" then
    remove predictions from p where entity is not NIL;
FP = length(p)
```

Entity Disambiguation (ED)

The definitions provided by Hoffart et al. [38] are used to calculate Precision, Recall and F1 scores for "InKB" and "EE" settings. However, micro averaged scores are reported

instead of macro. Following Hoffart et al. [38], micro and macro averaged accuracy is also reported. These two scores will be referred to as the "Normal" setting, as they take all predictions and ground truth annotations into account. Equations (4.4-10) define the evaluation metrics.

$$\text{Accuracy} = \frac{\#\text{ Prediction} = \text{Ground Truth}}{\#\text{ All Instances}} \tag{4.4}$$

$$\text{“InKB” Precision} = \frac{\#\text{ (Prediction} = \text{Ground Truth)} \wedge \text{(Entity in KB)}}{\#\text{ Predictions where link is to an entity in KB}} \tag{4.5}$$

$$\text{“InKB” Recall} = \frac{\#\text{ (Prediction} = \text{Ground Truth)} \wedge \text{(Entity in KB)}}{\#\text{ Ground truth ann. where link is to an entity in KB}} \tag{4.6}$$

$$\text{“InKB” F1 Score} = \frac{2 \cdot (\text{“InKB” Precision}) \cdot (\text{“InKB” Recall})}{(\text{“InKB” Precision}) + (\text{“InKB” Recall})} \tag{4.7}$$

$$\text{“EE” Precision} = \frac{\#\text{ (Prediction} = \text{Ground Truth)} \wedge \text{(Link is NIL)}}{\#\text{ Predictions where link is to NIL}} \tag{4.8}$$

$$\text{“EE” Recall} = \frac{\#\text{ (Prediction} = \text{Ground Truth)} \wedge \text{(Link is NIL)}}{\#\text{ Ground truth ann. where link is to NIL}} \tag{4.9}$$

$$\text{“EE” F1} = \frac{2 \cdot (\text{“EE” Precision}) \cdot (\text{“EE” Recall})}{(\text{“EE” Precision}) + (\text{“EE” Recall})} \tag{4.10}$$

Accuracy measures the fraction of instances (mentions) for which the predicted link (an entity from the KB or NIL) is the same with that of ground truth annotation. "InKB" precision measures the fraction of correct links, among the predicted links that point to an entity in the KB. Contrary, "EE" precision measures the fraction of correct links, among the predicted links that point to NIL. "InKB" recall measures the fraction of ground truth annotations captured by the model where the mentions are linked to an entity in the KB. "EE" recall measures the fraction of ground truth annotations captured by the model where the mentions are linked to NIL.

4.1.3 Training, Hyperparameters and Implementation

To conduct the experiments, the models introduced in Chapter 3 are trained and implemented. For training, the Training split prepared for that specific task is used for each model. The details for all the models are explained below.

$\textbf{BERT}_{SC}$

The weights of BERT_{SC} are first initialized with the case-preserving version of BERT_{BASE}. Following TAPT, the model is trained end-to-end with the sentences extracted from Scopus, using Masked Language Modeling (MLM). The choice of a case-preserving model is due to the fact that case information can provide important information to the NER task, for example, it is common in English to capitalize organization names. The implementation is based on the GitHub repository of the paper where TAPT was introduced[2] [34]. Throughout training, the progress is monitored on Dev split. The hyperparameters recommended by Gururangan et al. [34] is used as much as possible. Number of epochs are reduced from the recommended number (100) to 2 as the training set is rather large. It is thought that 1 epoch would be sufficient, and a second epoch would be beneficial to see whether Dev scores would improve with more epochs or not. The batch size is set

[2] `https://github.com/allenai/dont-stop-pretraining`

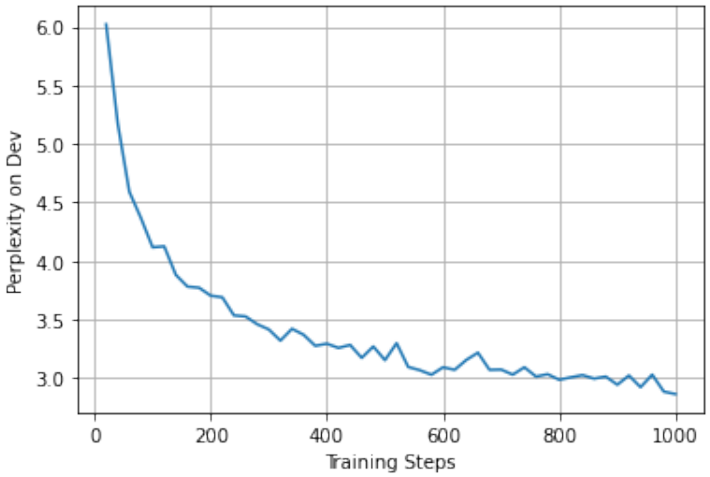


Figure 4.3: Perplexity on Dev during training

to 4 due to memory requirements, but using gradient accumulation, an effective batch size of 2048 is maintained as recommended.

However, the initial setup to train BERT$_{SC}$ could not be performed. According to the initial setup, TAPT was going to be performed for around 13,500 steps in 2 epochs. However, after 2 days of training it was observed that only 1000 steps were finished, and hence, only around 2 million training examples were utilized. Due to the time constraints, it was determined to stop the training at that point.

After 1000 steps, a perplexity of 2.86 is achieved on the Dev set. Figure 4.3 shows the change of perplexity on Dev, recorded every 20 training steps. Based on this plot, we believe that it is possible to obtain a model with higher performance when the training with the full dataset is completed. However, this is left to future work.

The training of BERT$_{SC}$ is done on an NVIDIA Tesla K80 GPU with 12 GB of memory. More details on the hyperparameter configuration can be found in Appendix C.1.

Flair^{NER}

The implementation of Flair^{NER} is done using the Flair library [1]. In this library, both $LSTM^{forward}_{Flair}$ and $LSTM^{backward}_{Flair}$ trained on the 1 billion word corpus [14] are available. Using these and pretrained GloVe embeddings, the BiLSTM-CRF model is trained on the Training split and the progress is monitored on Dev split using the training interface of Flair. Mainly, the hyperparameters and training strategy reported by Akbik et al. [2] is followed, while changing minor things to address the computational limitations.

A batch size of 8 is used, and the model started training with a learning rate of 0.1. When for 2 epochs, no improvement on the Dev split was observed, the learning rate was halved. The training is stopped when no improvement was made in a certain learning rate, amounting to 37 epochs in total. The performance on Dev was measured as micro averaged F1 score of the output tags. The models were saved at the end of each epoch, and the model that performed best on Dev is selected. It was observed that the model at the end of 33 epochs was the best one. The losses, scores and learning rates per epoch are presented in Appendix C.2. One epoch of training Flair^{NER} took approximately 50 minutes on an NVIDIA Quadro T1000 GPU with 4GB memory.

BERT^{NER}

All parameters of BERT^{NER} are fine-tuned end-to-end with different hyperparameter settings. The progress of training is monitored on Dev across epochs, with respect to the same evaluation metric of the NER task.

Devlin et al. [18] suggested different hyperparameter settings for fine-tuning BERT: a batch size of 16 or 32, learning rate of 5×10^{-5}, 3×10^{-5} or 2×10^{-5}; and training for 2, 3 or 4 epochs. Due to memory requirements, the batch size is set to 8. Then, the model is trained for 10 epochs with a learning rate of 2×10^{-5}, while saving the model at the end of each epoch. On top of that, 3 more models are trained using a linear learning rate scheduler for 2, 3 and 4 epochs respectively. The number of warm-up steps is set to 50. For implementation, the library Transformers [86] by Hugging Face[3] is used.

After the experimentation with different hyperparameter settings, it was decided that training for 3 epochs with a linear learning rate scheduler produced the best results. Appendix C.3 shows detailed results on this experimentation.

BERT^{NER} can process a maximum of 512 tokens per input, similar to BERT$_{BASE}$. Hence, the sentences having more tokens are split into smaller chunks, such that there is no overlap between chunks and no word is scattered across chunks. Later on, the predictions are merged as a postprocessing step. The training is done using an NVIDIA Tesla K80 GPU with 12 GB of memory. On this device, one epoch took approximately 1 hour.

BERT$^{NER}_{SC}$

BERT$^{NER}_{SC}$ is trained with the exact same hyperparameter settings that produced the best results for BERT^{NER}, i.e. for 3 epochs with a linear learning rate scheduler.

One aim of this study is showing the effect of domain adaptation, which is achieved by comparing BERT^{NER} and BERT$^{NER}_{SC}$. To be able to show that the improvement gained by pretraining is not random, a second BERT$^{NER}_{SC}$ is trained with the second-best hyperparameter setting, for 2 epochs with a linear learning rate scheduler. However, the aim of this second BERT$^{NER}_{SC}$ is just for observing the effect of domain adaptation, not for measuring the success of any other task. Hence, when BERT$^{NER}_{SC}$ is referred to in any other setting, it is the one trained for 3 epochs.

BERT$^{NER}_{SC}$ can also process a maximum of 512 tokens, and is also trained on the same GPU with BERT^{NER}. One epoch of training took approximately 1 hour.

BI$_{BL}$

The implementation of BI$_{BL}$ is inspired by BLINK's GitHub repository[4]. Mainly, the code for data preprocessing is obtained from this repository. The models and training are implemented using PyTorch [65] and Transformers library by Hugging Face.

As there was no computational power to be able to do an extensive hyperparameter search, the values reported by Wu et al. [90] is followed as much as possible. In BLINK, the maximum number of tokens for the mention representation ($R_{Mention}$) are either 32 or 128, depending on the dataset. This number is set to 64 in this work. Originally it was planned to set it to 32 to address the memory limitations, however, it was observed that there were mentions longer than 32 tokens themselves.

For candidate representation (R_{Entity}), BLINK uses 128 tokens. However, in this setting, there are candidates that have longer representations. To be more specific, there are 96 entities with representation longer than 128 tokens, 36 entities longer than 256 tokens, and 23 entities longer than 256. Hence, for the 96 entities that had longer

[3] `https://huggingface.co/`

[4] `https://github.com/facebookresearch/BLINK/tree/master/blink`

representations, a label is removed iteratively until the representation was equal to or shorter than 128 tokens. The label that had the highest sorted Levenshtein distance with any other label is removed in each iteration, as it was thought that this label would be the least informative.

BLINK uses a batch size of 128, and adds 10 hard negatives (Neg_H=10) among the in-batch negatives. Hence, for a mention, they make use of a maximum of 127 random negatives, and a maximum of 137 negatives in total ($max(|IE|) = 127 + 10 = 137$). As the training is done on a GPU with 12 GB memory, this batch size could not be maintained in terms of random negatives. Hence, the hyperparameters are scaled down proportionally. It was observed that a maximum batch size of 16 was possible. Proportionally, Neg_H is set to 1, making $max(|IE|) = 15 + 1 = 16$.

Even though a batch size of 128 cannot be maintained in terms of in-batch random negatives, it is possible to maintain this number in terms of gradient updating using gradient accumulation. Initially, the batch size was set to be inline with BLINK. Similarly, the learning rate was set to 10^{-5}. However, no learning was observed with this setting. We hypothesize that it is because of the size of the dataset. At the end, the batch size is set to 64 using gradient accumulation and the learning rate is set to 2×10^{-5}, which is also the minimum learning rate recommended by Devlin et al. [18].

The models in BLINK are trained for either 4 or 5 epochs. It is not clear whether the hard negative sampling is done for each epoch or not. However, as they report that they are following the strategy of DEER, it will be assumed that one epoch corresponds to one round for this case. Based on this, initially, it was thought to have 4 rounds of training, each consisting of 1 epoch. No hard negatives are added in the first round, as some weights of the model are initialized randomly anyway. To observe the course of training and the change in performance after each round, Dev is used.

Table 4.9 shows the performance of BI_{BL} on Training and Dev sets after the first two rounds. It can be seen that after the second round, the performance drops further for the "InKB" setting but improves for the "EE" setting, resulting in an overall performance decline as shown with accuracy. We hypothesize that the second round of training resulted in better separation of the scores of NIL and not-NIL mentions, however, did not improve the model in terms of finding the correct entity.

To see the separation between NIL mentions and non-NIL mentions, it is possible to model the distribution of scores. For this purpose, two normal distributions are fit on the normalized scores (see Section 3.4 for details of the normalization) of the highest ranked entity for each mention, one for NIL mentions and one for others. The parameters of the distribution are calculated using Maximum Likelihood and Dev dataset, and are shown in Table 4.10. Figure 4.4 shows the histogram of scores for each distribution. When the Bhattacharyya distance [28] is checked, it is possible to see that the distribution of NIL mention scores are more different than the other mentions in Round 2, compared to Round 1. The choice of this metric is due to the fact that it is reported as a convenient metric to measure the class separability of normal distributions [28].

Based on the results, the training of BI_{BL} is stopped after two rounds, and was not used any further. The training of the first round took 1.3 and the second round took 1.7 hours in the NVIDIA Tesla K80 GPU with 12 GB of memory.

BI_{AD}

To train BI_{AD}, the same batch size and learning rate with that of BI_{BL} is used. The progress of the training is also monitored using the Dev split.

The model is trained for 4 rounds, each consisting of one epoch. The initial round of training is done just with random negatives, and Neg_R is set to 3. This hyperparameter is not tuned as there is not much performance improvement expected from the initial

Round	Dataset	Micro Averaged Accuracy	Macro Averaged Accuracy
1	Training	61.88	62.9
2	Training	59.2	59.21
1	Dev	60.16	64.04
2	Dev	55.06	59.13

		EE Setting		
Round	Dataset	Precision	Recall	F1 Score
1	Training	60.32	60.64	60.48
2	Training	64.42	85.03	73.3
1	Dev	50.4	57.31	53.63
2	Dev	54.1	72.96	62.13

		InKB Setting		
Round	Dataset	Precision	Recall	F1 Score
1	Training	62.24	62.16	62.2
2	Training	57.51	53.31	55.33
1	Dev	62.25	60.69	61.46
2	Dev	55.31	51.78	53.49

Table 4.9: Scores of BI_{BL} for the first two rounds. A threshold of 0.324 is used for the first and 0.321 for the second round.

	NIL Mentions		Other Mentions		Bhattacharyya
Round	Mean	Std. Dev.	Mean	Std. Dev.	Distance
1	0.311	0.109	0.465	0.11	0.247
2	0.263	0.099	0.534	0.17	0.541

Table 4.10: Parameters of the distributions for scores and the Bhattacharyya distance between the two distributions for each round.

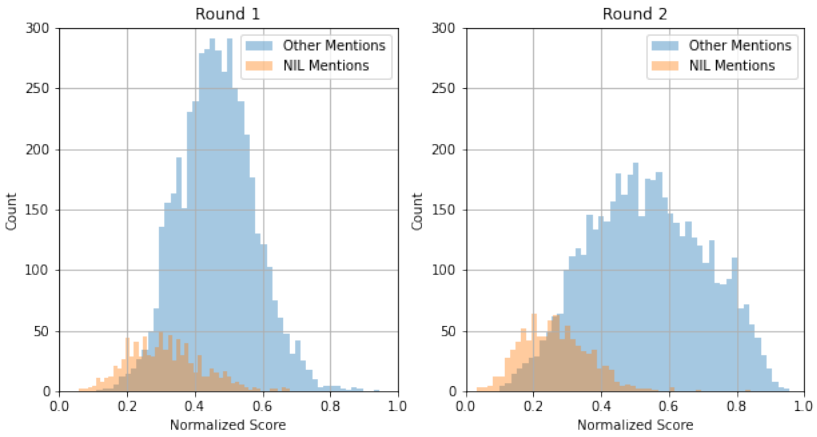


Figure 4.4: Distribution of scores for BI_{BL}, first two rounds, on Dev.

Dataset	Round	None	1	2	3	4	5	6	7	8	9	10+
Training	1	51,482	12,711	4,470	1,873	1,244	1,034	691	517	450	356	3,144
Training	2	70,566	3,024	916	516	327	194	147	135	133	104	1,910
Training	3	71,547	3,329	899	402	272	187	153	102	88	85	908
Training	4	73,273	2,273	821	313	200	117	85	57	62	53	718
Dev	1	3,063	771	283	122	63	63	49	36	17	30	252
Dev	2	4,204	173	58	36	24	6	12	12	12	5	207
Dev	3	4,222	212	74	33	19	10	14	12	9	5	139
Dev	4	4,291	148	64	24	13	12	13	5	9	10	160

Dataset	Round	None	1	2	3	4	5	6	7	8	9	10+
Training	1	59,711	9,356	2,787	1,559	907	794	436	326	263	205	1,628
Training	2	73,259	2,562	579	374	180	115	100	66	48	42	647
Training	3	74,484	2,141	484	252	152	79	68	45	33	17	217
Training	4	75,403	1,875	295	107	60	38	31	17	9	14	123
Dev	1	3,574	556	161	99	49	57	20	25	15	14	179
Dev	2	4,288	183	43	31	12	12	11	10	5	3	151
Dev	3	4,355	159	51	32	14	20	12	8	4	6	88
Dev	4	4,354	178	37	13	7	11	11	8	6	3	121

Table 4.11: Number of not-NIL mentions with {None, 1,2,...,9 ,10 or more} hard negatives in each round for BI_{AD}. Upper table is with the **first** set of hyperparameters, and lower table is with the **second** set of hyperparameters.

round. It could be that a higher Neg_R leads to a better result, however, the training time would also increase accordingly. In the following rounds, Neg_H is set to 10, as BLINK also uses 10 hard negatives. Based on the formula introduced in Section 3.4, Neg_R is set to 3, 2 and 2 for the second, third and fourth rounds respectively. Also, in the initial round, class weights are used for loss calculation to prevent the model to predict everything as the negative class. According to the random negative sampling strategy, a weight of 0.25 is given to the negative class and 0.75 is given to the positive class. However, class weights are not used in the following rounds, even though the class imbalance continues. During training, it was seen that not using class weights was slightly better for the performance. This could be because in these rounds the negative samples are selected with hard negative mining, and hence are very informative.

Table 4.11 (upper) shows the number of hard negatives for Training and Dev splits after each round. As can be seen, the mentions with no hard negatives increase significantly after the second round, and this increase becomes much smaller in the following rounds. In this setting, the training took around 5, 8.5, 6.5 and 6.5 hours for the first, second, third and fourth rounds respectively; using the NVIDIA Tesla K80 GPU with 12 GB of memory.

As a second set of hyperparameters, it was tried to train each round for 2 epochs instead of one. Also, the maximum tokens for R_{Entity} is set to 256. It was thought that a longer representation would be better in the event that new entities would have many labels. The number was not increased to 512, in case the Cross-Encoder of BLINK [90] would be implemented later on. A separate experiment to see the effect of this increase to the performance was not conducted as the impacted entities cover 2.42 % of Training, 2.3 % of Dev, 2.41 % of Validation and 2.66 % of Test split for the ED task. There were 1037 samples in Validation split for which BI_{AD} made a mistake when trained with the first hyperparameter configuration, but got the correct answer when trained with the second hyperparameter configuration. Among these samples, in only 10 of them,

Setting	Threshold	Micro Averaged Accuracy	Macro Averaged Accuracy
1	0.5	83.85	87.3
2	0.5	86.95	89.9
1	0.732	86.61	89.17
2	0.728	88.44	90.75

		EE Setting		
Setting	Threshold	Precision	Recall	F1 Score
1	0.5	81.06	47.85	60.18
2	0.5	84.22	58.97	69.37
1	0.732	70.38	76.24	73.19
2	0.728	75.75	76.27	76.01

		InKB Setting		
Setting	Threshold	Precision	Recall	F1 Score
1	0.5	84.13	90.48	87.19
2	0.5	87.29	92.11	89.64
1	0.732	89.9	88.53	89.21
2	0.728	90.8	90.68	90.74

Table 4.12: Comparing the two hyperparameter settings on Validation split.

the correct entity was affected by this change. Lastly, as the entity embeddings are precomputed before inference, this increase does not have any impact on the efficiency.

Apart from number of epochs per round and the maximum tokens for R_{Entity}, the other hyperparameters were not changed. During training, following the formula, Neg_R is set to 2 for second, third and fourth rounds. In this setting, the training took around 11, 15, 13.5 and 14 hours for the first, second, third and fourth rounds respectively; using the NVIDIA Tesla K80 GPU with 12 GB of memory.

Table 4.11 (lower) shows the number of hard negatives for the second configuration and Table 4.12 shows the results on Validation for both hyperparameter settings. Even though a score of 0.5 is a natural threshold for NIL mention detection as the problem is cast as binary classification, a threshold that maximizes the micro averaged accuracy is selected on the Training split using grid search in the interval [0.5,1] with a step size of 0.001. The thresholds 0.732 and 0.728 are selected for first and second hyperparameter settings respectively. Table 4.12 compares the results for both the natural and selected thresholds.

It is possible to see that there are more mentions with no hard negatives in general for the second set of hyperparameters. Also, all the scores are higher on Validation when the second hyperparameters are used. That is why, it was decided to use the model that is trained for 2 epochs per round and that has a maximum R_{Entity} of 256.

An interesting observation is, a threshold selection is beneficial for NIL mention detection, which is shown by the "EE" evaluation setting. This could indicate that the models do not have a full capability of detecting NIL mentions themselves.

Appendix C.4 presents more results on Training and Dev splits for each round of training.

K	Coverage	Increase
1	91.68%	-
2	95.43%	+3.75 %
3	96.21%	+0.78 %
4	96.48%	+0.27 %
5	96.63%	+0.15 %
6	96.86%	+0.23 %
7	97.09%	+0.23 %
8	97.26%	+0.17 %
9	97.39%	+0.13 %
10	97.45%	+0.06 %
11	97.52%	+0.06 %
12	97.66%	+0.15 %
13	97.73%	+0.06 %
14	97.81%	+0.08 %
15	97.83%	+0.02 %
16	97.85%	+0.02 %
17	97.85%	+0.0 %
18	97.87%	+0.02 %
19	97.89%	+0.02 %
20	97.98%	+0.08 %

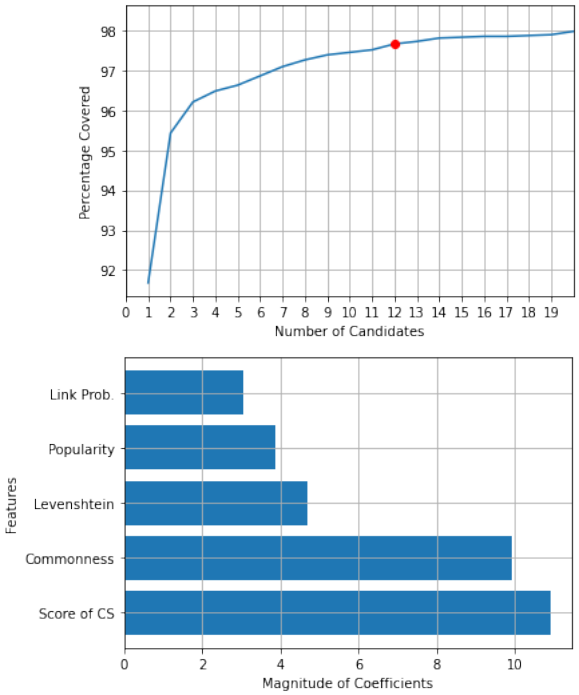


Table 4.13: Left: Coverage estimation for each rank on Dev split. Top Right: Coverage plot. Bottom Right: The magnitude of coefficients for LM. Score of CS refers to the score obtained from BI_{AD}.

Reranker

Before training the rerankers, the number of candidates are selected using Dev based on the best BI_{AD}. First, top 20 predicted entities are extracted for each non-NIL mention in the dataset. Then, the coverage of each rank is calculated as the number of mentions for which the correct entity is extracted within that rank divided by the number of non-NIL mentions. Table 4.13 shows the coverage of each rank and how much the coverage increase between ranks. The figure next to the table plots the coverage over ranks. The red dot indicates the selected K value, $K = 12$. This value is selected as the coverage does not increase for more than 0.1% in the following ranks.

For LM, the best hyperparameter configuration is chosen based on micro averaged accuracy when the NIL mention detection threshold is 0.5. The best feature set included all features, the L_1 penalty ratio is set to 0.5 and the regularization strength is set to 10. Figure in Table 4.13 shows the magnitude of coefficients for LM. A NIL mention detection threshold is selected on the interval [0.5,1] using grid search with a step size of 0.001 that maximizes the micro averaged accuracy on Training split. The best threshold remained to be 0.5. It is hypothesized that a lower threshold could perform even better, but is not tried as a score lower than 0.5 would indicate an entity to be classified as negative by the classifier.

LM is implemented with Scikit-Learn Python library [66]. The hyperparameters of GBM_{F5} can be found in Appendix A.

System	Organization			Grant		
	Precision	Recall	F1	Precision	Recall	F1
Stanford NER	73.7	75.1	74.39	94.11	94.77	94.44
BERT^{NER}	79.18	**86.03**	**82.46**	94.71	**97.39**	96.03
BERT$^{NER}_{SC}$	**80.28**	**86.54**	**83.29**	**94.9**	**97.63**	**96.24**
Flair^{NER}	**85.83**	78.02	81.74	**97.56**	95.24	**96.39**

Table 4.14: NER Results on Validation split

System	Organization			Grant		
	Precision	Recall	F1	Precision	Recall	F1
BERT^{NER} (2 epochs)	78.44	85.8	81.96	94.59	97.43	95.99
BERT$^{NER}_{SC}$ (2 epochs)	**79.54**	**86.56**	**82.9**	**94.73**	**97.55**	**96.12**

Table 4.15: Comparison of BERT^{NER} and BERT$^{NER}_{SC}$ with the second-best hyperparameter setting, i.e. 2 epochs with a learning rate scheduler.

4.2 Results

The results for NER, ED and EL are presented in this section. Different approaches are compared on the Validation splits, and the best performing approach is compared with feature-based models of that task on the Test split.

Named Entity Recognition (NER) and Task-Adaptive Pretraining (TAPT)

To compare the developed NER models Stanford NER [25] is used as a feature-based model. Table 4.14 compares the precision, recall and F1 scores for Organization and Grant mentions on Validation dataset. It can be seen that all models perform well on extracting grant mentions, while Flair^{NER} obtains the highest F1 score with a small difference. In contrast, using neural language models improve the performance on Organization mentions with a large margin, resulting in a minimum absolute increase of 7.4% in terms of F1.

BERT^{NER} slightly outperforms Flair^{NER} in terms of Organization F1 score by an increase of 0.7%. However, the main difference is the precision and recall values. Flair^{NER} achieves a precision that is 6.6% higher than that of BERT^{NER}, while BERT^{NER} achieves a recall that is 8% higher. In funding data extraction, for NER component, a higher recall is preferred in this study as if a mention is missed completely, there is nothing that can be done about it.

BERT$^{NER}_{SC}$ improves upon BERT^{NER} further, showing the importance of domain adaptation. When trained with the exact same setup, it is possible to see an improvement of 1% in precision and 0.5% in recall. To show that this was not a coincidence, Table 4.15 shows the performance of BERT^{NER} and BERT$^{NER}_{SC}$ on the Validation split, when trained for 2 epochs with a learning rate scheduler. It can be seen that domain adaptation improves the performance on this hyperparameter setting as well.

At the end, it is decided to use BERT$^{NER}_{SC}$ as the NER component to extract mentions of funding bodies, answering the first subquestion. Table 4.16 compares the results of BERT$^{NER}_{SC}$ and Stanford NER on the Test split. It is possible to see that the performance gain persists on the Test split as well. Hence, it is concluded that this work improved upon Stanford NER model by 2.9% gain in precision, 12.4% gain in recall and 7.6% gain in F1 score.

System	Organization			Grant		
	Precision	Recall	F1	Precision	Recall	F1
Stanford NER	76.17	72.87	74.48	93.37	93.24	93.31
BERT_{SC}^{NER}	**79.08**	**85.31**	**82.08**	**93.95**	**96.54**	**95.23**

Table 4.16: NER Results on Test split

Entity Disambiguation (ED)

A feature-based ED model is used for evaluation to compare with the neural approaches. This model uses BM25 [73] scores for selecting a candidate set and a GBM with 26 features for reranking. The candidate selectors extracts a maximum of 20 entities and has a coverage of 94%. This model is referred to as GBM_{F26}.

Table 4.17 shows the performance of the ED systems on Validation. BI_{AD} is the one that is trained with the best hyperparameter setting. The performance of BI_{AD} with a threshold of 0.728 is reported. For comparison, apart from GBM_{F26}, the performance of a system that is solely based on commonness is reported. For this system, the commonness values are obtained using the Training split, and the mention is linked to the entity with which it has the highest commonness value. If a mention did not appear as a link in the Training split (and hence no commonness value is recorded), it is linked to NIL. The results show that the feature-based model, GBM_{F26}, outperforms BI_{AD} significantly. Also, even though it is highly simple, the Commonness baseline has surprisingly good results. It can be seen that an additional linear reranker (BI_{AD} + LM) improves the performance of BI_{AD} by 2%, and performs only 0.5% worse than the feature-based model in terms of micro averaged accuracy. Lastly, a GBM reranker (BI_{AD} + GBM_{F5}) improves the performance further and outperforms GBM_{F26}. Hence, it was decided that the best ED model that is developed is GBM_{F5} that utilizes BI_{AD} as a candidate selector.

Table 4.18 compares the performance of the best performing model and GBM_{F26} on Test split. It can be seen that BI_{AD} + GBM_{F5} outperforms GBM_{F26} on this dataset as well. However, the performance difference is larger in this setting, BI_{AD} + GBM_{F5} performing 0.4% higher in terms of micro averaged accuracy.

We believe that these results are answering the sub-questions RQ2 and RQ3. For RQ2, we show that it is possible to represent the domain-specific entities concatenating their labels and country information. When these representations are passed through a BERT fine-tuned for the task, and the last hidden state vectors corresponding to the [CLS] token are obtained as the entity embeddings, it is possible to perform successful disambiguation. As for RQ3, we show that disambiguation can be performed by using BI_{AD} as the candidate selector and GBM_{F5} as the reranker.

Entity Linking (EL)

Table 4.20 shows the end-to-end EL performance of different NER and ED systems combined. It can be seen that, in all settings, the performance is lower when Stanford NER is used. It was on shown on Table 4.14 that BERT_{SC}^{NER} improves upon Stanford NER by 6.6% ,11.44% and 8.9% in terms of precision, recall and F1 score of Organization mentions, on Validation split. Table 4.20 shows that the EL performance of ED models have a similar performance increase in the "Normal" setting when BERT_{SC}^{NER} is used instead of feature-based NER.

It can also be seen that the "InKB" performance is significantly higher than the performance on EEs. Hence, it can be hypothesized that improving NIL mention detection is something that should be worked on. In addition, in the "InKB" setting, the recall

System	Micro Averaged Accuracy	Macro Averaged Accuracy
BI_{AD}	88.44	90.75
BI_{AD}+LM	90.48	92.22
BI_{AD}+GBM_{F5}	**91.15**	**92.76**
GBM_{F26}	**91.02**	**92.84**
Commonness	83.8	85.81

	EE Setting		
System	Precision	Recall	F1 Score
BI_{AD}	75.75	76.27	76.01
BI_{AD}+LM	75.68	80.85	78.18
BI_{AD}+GBM_{F5}	**77.44**	**81.14**	**79.25**
Commonness	53.55	**88.2**	66.64
GBM_{F26}	**79.11**	78.67	**78.89**

	InKB Setting		
System	Precision	Recall	F1 Score
BI_{AD}	90.8	90.68	90.74
BI_{AD}+LM	93.43	92.26	92.84
BI_{AD}+GBM_{F5}	**93.82**	**92.99**	**93.4**
Commonness	**94.22**	82.99	88.25
GBM_{F26}	93.2	**93.29**	**93.25**

Table 4.17: Performance comparison of different ED models on the Validation split. Micro averaged results are reported for "InKB" and "EE".

System	Micro Averaged Accuracy	Macro Averaged Accuracy
BI_{AD}+GBM_{F5}	**90.66**	**91.11**
GBM_{F26}	90.26	90.84

	EE Setting		
System	Precision	Recall	F1 Score
BI_{AD}+GBM_{F5}	79.26	**85.45**	**82.24**
GBM_{F26}	**80.03**	81.49	80.75

	InKB Setting		
System	Precision	Recall	F1 Score
BI_{AD}+GBM_{F5}	**93.56**	91.86	**92.7**
GBM_{F26}	92.69	**92.3**	92.5

Table 4.18: Performance comparison of the best ED model developed and the feature-based model on Test split. Micro averaged results are reported for "InKB" and "EE"..

("Normal" Setting)		Micro Averaged			Macro Averaged		
NER	ED	Precision	Recall	F1	Precision	Recall	F1
Stanford NER	GBM_{F26}	69.83	67.09	68.43	71.62	69.24	69.34
$BERT_{SC}^{NER}$	BI_{AD} + GBM_{F5}	**72.89**	**78.97**	**75.81**	**76.12**	**78.73**	**76.59**

("EE" Setting)		Micro Averaged			Macro Averaged		
NER	ED	Precision	Recall	F1	Precision	Recall	F1
Stanford NER	GBM_{F26}	45.93	41.02	43.34	71.77	71.07	71.01
$BERT_{SC}^{NER}$	BI_{AD} + GBM_{F5}	**50.71**	**55.1**	**52.82**	**73.85**	**74.29**	**73.68**

("InKB" Setting)		Micro Averaged			Macro Averaged		
NER	ED	Precision	Recall	F1	Precision	Recall	F1
Stanford NER	GBM_{F26}	83.15	72.26	77.33	78.05	73.69	74.73
$BERT_{SC}^{NER}$	BI_{AD} + GBM_{F5}	**86.62**	**83.72**	**85.14**	**83.07**	**82.25**	**81.86**

Table 4.19: Comparison of EL performance of the best NER and ED models and the feature-based models on the Test split.

values are lower than precision, but this does not hold for the "Normal" setting. This may be caused by the sub-optimal NIL mention detection threshold as well, failing to generate the link to the correct entity when the assigned score is lower than the linear threshold.

Table 4.19 shows that the best model pairing improves upon the setting utilizing only feature-based models largely. The developed approach improves micro averaged precision, recall and F1 score by 3.1%, 11.9% and 7.4% respectively. These results answer the main research question, showing that it is possible to perform EL using neural approaches in funding domain.

("Normal" Setting)		Micro Averaged			Macro Averaged		
NER	ED	Precision	Recall	F1	Precision	Recall	F1
Stanford NER	Commonness	63.41	64.96	64.18	72.45	70.14	70.31
Stanford NER	GBM_{F26}	67.92	69.57	68.74	76.74	74.28	74.44
Stanford NER	BI_{AD}	66.58	68.2	67.38	75.73	73.3	73.46
Stanford NER	BI_{AD} + LM	67.72	69.37	68.54	76.63	74.17	74.34
$BERT_{SC}^{NER}$	Commonness	69.48	75.26	72.25	75.82	78.3	76.38
$BERT_{SC}^{NER}$	GBM_{F26}	**74.19**	**80.36**	**77.15**	**80.17**	**82.88**	**80.79**
$BERT_{SC}^{NER}$	BI_{AD}	72.49	78.52	75.38	78.91	81.5	79.5
$BERT_{SC}^{NER}$	BI_{AD} + LM	**73.94**	**80.09**	**76.89**	**79.99**	**82.64**	**80.6**

("EE" Setting)		Micro Averaged			Macro Averaged		
NER	ED	Precision	Recall	F1	Precision	Recall	F1
Stanford NER	Commonness	38.45	45.82	41.81	57.82	58.88	57.78
Stanford NER	GBM_{F26}	51.52	41.54	46	71.29	70.34	70.32
Stanford NER	BI_{AD}	50.36	39.47	44.26	70.87	70.19	70.09
Stanford NER	BI_{AD} + LM	48.82	41.89	45.09	69.83	69.35	69.15
$BERT_{SC}^{NER}$	Commonness	45.67	58.5	51.3	62.43	64.48	62.84
$BERT_{SC}^{NER}$	GBM_{F26}	**56.07**	**53.8**	**54.91**	73.38	73.53	72.97
$BERT_{SC}^{NER}$	BI_{AD}	**56.6**	49.78	52.97	**74.39**	**74.26**	**73.85**
$BERT_{SC}^{NER}$	BI_{AD} + LM	55.22	**53.41**	**54.3**	**73.48**	**73.84**	**73.16**

("InKB" Setting)		Micro Averaged			Macro Averaged		
NER	ED	Precision	Recall	F1	Precision	Recall	F1
Stanford NER	Commonness	89.87	68.14	77.51	85.1	72.47	76.41
Stanford NER	GBM_{F26}	87.16	74.23	80.17	83.38	77.86	79.3
Stanford NER	BI_{AD}	86.36	72.98	79.11	82.5	76.88	78.37
Stanford NER	BI_{AD} + LM	87.59	73.94	80.19	83.75	77.56	79.3
$BERT_{SC}^{NER}$	Commonness	90.62	78.04	83.86	87.69	80.26	82.5
$BERT_{SC}^{NER}$	GBM_{F26}	**89.31**	**84.77**	**86.98**	**86.99**	**86.06**	**85.72**
$BERT_{SC}^{NER}$	BI_{AD}	88.24	83.29	85.7	85.52	84.79	84.37
$BERT_{SC}^{NER}$	BI_{AD} + LM	**89.48**	**84.53**	**86.93**	**86.9**	**85.67**	**85.51**

Table 4.20: EL performance of different NER-ED model pairs on the Validation split.

Efficiency

The runtime of the best developed NER and ED component, $BERT_{SC}^{NER}$ and BI_{AD} + GBM_{F5}, is measured on a random sample of 100 sentences from the Dev split. First, $BERT_{SC}^{NER}$ is executed on the input to detect the mentions. Then, BI_{AD} is executed on the detected Organization mentions (306 in total) to retrieve 12 candidates for each. On these 12 candidates, GBM_{F5} is ran to select the highest scoring entity, and NIL mention detection is performed. Table 4.21 shows some statistics of this subsample.

The experiment is repeated 10 times on a laptop that has an Intel Xeon E-2276M (2.80GHz, 32GB RAM) CPU and an NVIDIA Quadro T1000 GPU with 4GB memory. The average and standard deviation runtime calculated from this experiment are shown in Table 4.22. It is possible to see that, $BERT_{SC}^{NER}$ can process 39 sentences and BI_{AD} + GBM_{F5} can link 33 mentions per second with this hardware. When BI_{AD} is ran without GBM_{F5}, 35 mentions can be linked per second. Hence, the usage of GBM_{F5} does not have a large impact on the efficiency.

For comparison, the efficiency results of GBM_{F26} is also reported on the same setting. It can be seen that the performance is much lower, and only 3 mentions can be linked

per second. This inefficiency is mostly attributed to the calculation of the hand-crafted features, as the inference time of GBM is quite low.

It should also be noted that the implementations of these models may not be optimal.

	Mean	Std.	Min.	Max.
Sentence Length (in characters)	229	126	65	919
Sentence Length (in WordPiece tokens)	63	37	19	257
Number of ORG mentions per sentence	3	2	0	10

Table 4.21: Statistics of the subsample for the runtime experiment.

System	With GPU	Without GPU
BERT_{SC}^{NER}	2.55 ± 0.14	16.61 ± 1.58
BI_{AD}	$8.83\pm$ 0.4	22.78 ± 1.56
BI_{AD} + GBM_{F5}	9.26 ± 0.47	23.07 ± 0.78
GBM_{F26}	-	99.2 ± 0.45

Table 4.22: Mean and standard deviation runtime (in seconds) for 100 sentences and 306 ORG mentions, measured with and without GPU.

4.3 Analysis

The results obtained after the experiments give important insights on using the systems that work well in general domain for funding domain.

Task-Adaptive Pretraining

It is possible to see that the TAPT schema was successful in terms of domain adaptation for this work. At the end of TAPT, a perplexity of 2.86 is achieved on Dev, which is lower than that of BERT_{BASE} reported on its held-out set [18]. Also, based on the perplexity plot on Dev shown in Figure 4.3, we hypothesize that it is possible to see even further improvement when the training proceeds.

To show the effectiveness of domain adaptation, two sets of BERT_{SC}^{NER} models were trained, one with the best and one with the second-best hyperparameter combination observed on BERT^{NER}. For the best hyperparameter setting, Table 4.14 shows that BERT_{SC}^{NER} improves upon BERT^{NER} with respect to all evaluation metrics. The biggest increase comes from the precision of ORG mentions, with an improvement of 1.1%. Even though there is normally a tradeoff between precision and recall, it can be seen that the recall of Organization mentions also increase by 0.5%, showing a nice improvement. The results with the second-best hyperparameter setting also show improvement when BERT_{SC}^{NER} is used. In this setting, the precision of Organization mentions is improved by 1.1% and the recall by 0.8%, showing a similar trend with the best hyperparameter setting. Also, we believe that this improvement is not just because of a longer training. In Appendix C.3, it is shown that no significant performance increase is observed when BERT^{NER} is trained for more epochs. In the light of these observations, we conclude that domain adaptation was beneficial for this work.

Lastly, some sentences were randomly sampled from Validation, where BERT_{SC}^{NER} performed better than BERT^{NER} and were investigated manually to see whether there is a trend. In some cases, it was observed that BERT_{SC}^{NER} was better at determining mention boundaries when there were adjacent organizations or organizations with a

name that is fairly long. However, no error pattern was extracted confidently.

"... Royal Marsden NIHR Biomedical Research Centre for Cancer ..."
BERT^{NER}: "Royal Marsden" and "NIHR Biomedical Research Centre for Cancer"
"... National Institute of Diabetes and Digestive and Kidney Disease$_1$ Diabetes Research Center (DRC)$_2$ ..."
BERT^{NER}: "National Institute of Diabetes and Digestive and Kidney Disease Diabetes Research Center"

Figure 4.5: Some instances where BERT^{NER} makes an error while BERT$^{NER}_{SC}$ does not. Underlined strings are the gold annotations, and adjacent gold mentions are enumerated.

NER for Extracting Grant and Funder Mentions

For extracting the mentions of funding organizations and their respective grant numbers, Flair^{NER}, BERT^{NER} and BERT$^{NER}_{SC}$ are trained and compared. First thing that can be noticed from Table 4.14 is that the feature-based NER already has a high performance for Grant mentions. Hence, for those it may be unnecessary to switch from a linear model to a neural model. However, grant mentions are an important part of funding information extraction, and it is better to have a single model that can handle both types of mentions. Also, some performance improvement still takes place. When a neural model is used, a minimum gain of 1.6% is observed in terms of F1 score for Grant mentions.

The performance on Grant mentions for BERT^{NER} shows the ability of BERT to understand patterns even though they are not necessarily included in its WordPiece vocabulary as a single word or a series of single characters. Grant numbers are usually combinations of letters, numbers and symbols such as "-", and hence are separated in a fine-grained way by the WordPiece tokenizer. Still, BERT is able to infer that other combinations of letters and numbers are also probably grant mentions, even though that specific combination was not encountered by the model before.

Another interesting trend that can be seen from Table 4.14 is that while BERT^{NER} and Flair^{NER} do not have an immense difference in terms of F1 Score for Organization mentions, BERT$_{NER}$ has a much stronger recall and Flair^{NER} has a much stronger precision. In this work, recall is favored over precision, as there is no possibility to recover the undetected mentions in later stages, while it is possible to apply some rule-based postprocessing to discard highly improbable mentions. That being said, precision still plays an important role. It could be argued that the incorrect mentions may be removed when the ED system cannot find a link for them. However, as emerging entities are highly valued in this work due to the fact that new organizations are formed every day, the mentions without a link are still displayed and are perhaps even considered as a candidate entity to be added to the KB. It should also be noted that the training time for Flair^{NER} was much longer than that of BERT$^{NER}_{SC}$ and BERT^{NER}, as it needed more epochs to achieve a comparable performance. The model that performs best in terms of Organization mention recall and F1 score is BERT$^{NER}_{SC}$. Even though it has a lower precision than Flair^{NER}, it still improves the precision of the baseline model by 6.6%.

The results on Table 4.16 present that the developed neural NER, BERT$^{NER}_{SC}$, improves upon the feature-based NER significantly on the Test set as well. In terms of Organization mentions, gains of 2.9%, 12.4% and 7.6% for precision, recall and F1 score

respectively are reported. For Grant mentions, gains of 0.6%, 3.3% and 1.9% for precision, recall and F1 score are observed. Even though the performance on grant numbers was very high already, it is very interesting to see that the recall was improved immensely. These results show the success of BERT_{SC}^{NER} on the task of NER of funding bodies. On the other hand, the downside of using a neural model is that the inference time is significantly higher when a GPU is not used, as shown in Table 4.22. However, on the positive side, even a small GPU having 4GB memory can speedup the execution massively.

An error analysis is done on a small portion of sentences that contain annotations where BERT_{SC}^{NER} made a mistake. This analysis gave important insights on the strengths and weaknesses of the model. Sometimes, mention boundaries may be ambiguous, and the gold annotations are not necessarily consistent. For some mentions, the country name in the immediate context is also included in the gold annotation. And, there is no clear scheme on when it should be included. It seems that this random behavior is also present in the annotations made by BERT_{SC}^{NER}. These instances lower the performance estimates without a solid ground.

"... Ministry of Health and Welfare, Korea ..."
BERT$_{SC}^{NER}$: "Ministry of Health and Welfare"

"... Spanish Ministerio de Ciencia y Tecnología ..."
BERT$_{SC}^{NER}$: "Spanish Ministerio de Ciencia y Tecnología"

Figure 4.6: Some instances where the gold annotations are not consistent and BERT_{SC}^{NER} makes a mistake.

In addition, foreign text is not always handled properly. Trying a multilingual model may be a good experiment for the future. Also, it is possible to see that there are still errors when detecting the mention boundaries for adjacent Organization mentions.

"... Secretaria de Ciencia y Tecnología de la Universidad Nacional de Córdoba ..."
BERT$_{SC}^{NER}$: "Secretaria" and "Universidad Nacional de Córdoba"

Figure 4.7: BERT_{SC}^{NER} makes a mistake on a mention with foreign name.

"... NIHR Imperial Biomedical Research Centre ..."
BERT$_{SC}^{NER}$: "NIHR" and "Imperial Biomedical Research Centre"

Figure 4.8: BERT_{SC}^{NER} makes a mistake on mention boundaries

Currently, BERT_{SC}^{NER} utilizes only a linear layer for token classification. However, it is believed that using a CRF layer may reduce the precision errors. Some extracted mentions were observed to be given an "I" label after an "O" label, which is illegal in IOB-Tagging scheme. Most of these mentions are words that are commonly found in funding organization mentions but do not refer to a unique entity by themselves. Some examples are "Infections", "Resistance", "England", "System", "Hospital" and "Fund". A CRF layer may help setting an "O" label for such mentions.

ED for Funder Organizations

First, BI_{BL} is trained and evaluated. It can be seen from Table 4.9 that the performance does not improve with the given hard negative training. One of the reasons could be

that in-batch random negatives are not be utilized properly. In Figure 4.2, it can be seen that the entity distribution is highly skewed. This would mean that the same entities would appear as random negatives frequently for many mentions. In addition, as the available GPU memory is small, after scaling down the hyperparameters there was only one hard negative added per mention. This may be too little to utilize hard negatives. When it was switched from BI_{BL} to BI_{AD}, large performance gains were observed when hard negatives were included in the training. The only advantage of BI_{BL} over BI_{AD} is that the former has a much lower training time.

Table 4.17 compares the performance of BI_{AD} with a baseline system that only uses Commonness and a feature-based model, GBM_{F26}. Surprisingly, the system that is based solely on commonness performs well. When micro averaged accuracy is checked, BI_{AD} and GBM_{F26} improve upon commonness by 4.5% and 7% respectively. Considering that both BI_{AD} and GBM_{F26} are highly complex models, it would be expected to gain more improvement. The reason for this may be that most of the mentions are actually easy to disambiguate, and that the more ambiguous cases are the minority. This should come as no surprise when the entity distribution shown in Figure 4.2 is checked. With a quick math, it can be seen that the accuracy would be around 7% if all the mentions were assigned the most frequent entity. If NIL mentions were excluded, this would increase up to 8.8%. It can also be seen that GBM_{F26} performs around 2.5% higher than the best BI_{AD} in terms of micro averaged accuracy. A similar performance difference is observed in other settings as well, suggesting that this system performs better overall, both NIL mention detection and InKB entity disambiguation.

It is also shown that selecting a threshold for NIL mention detection improves the performance of BI_{AD} (see Table 4.12). With a threshold of 0.728, the overall micro averaged accuracy increases by 1.5%, and the scores for the "EE" setting improves rapidly, showing that the new threshold is better at detecting emerging entities. However, the recall of "InKB" evaluation dropped by 1.4%. This is because with the new threshold, even if BI_{AD} manages to find the correct entity, the mention will not be linked when the score is between 0.5 and 0.728. This decrease shows that a linear threshold may not be the optimal solution, and maybe a classifier or a non-linear threshold may help increasing the quality of NIL mention detection. Another interesting observation is that there is a big difference between the performance on Training and Dev datasets, shown in Appendix C.4. After 4 rounds of training is completed, the micro averaged accuracy on Training set is around 5.5% higher than that of Dev with both the default and optimized threshold. This difference is around 6.5% when the F1 score of "InKB" evaluation is checked, and is around 20% for F1 score in "EE" setting. Hence, we believe that neither the NIL mention detection learned by the model nor by the optimized threshold generalizes well outside the Training set itself. Improving this could be one of the keys to improve the performance overall.

The usage of LM improves the performance significantly as shown in Table 4.17, despite it being a linear model utilizing trivial ED features. Hence, at least for this problem, it could be that an expensive cross-encoder is not needed. Apart from efficiency, one advantage of LM over cross-encoder is that it utilizes information that was not used in the BI_{AD}, such as lexical similarity and dataset statistics. On the other hand, the cross-encoder utilizes BERT's capability of determining the strength of the relation of two sentences, different from both BI_{AD} and LM. It could be the fact that the information provided by LM is more informative for this task. Lastly, it can be seen from Table 4.13 that BI_{AD} has a very high coverage on the first rank. Thus, the improvement that can be obtained from any reranker is limited, and is questionable whether it is worth to use an expensive model for such improvement.

Another interesting question to answer is what value using LM adds to BI_{AD}. It is observed that there are 371 instances on Validation split where BI_{AD} ranks the correct

entity on top, but does not perform the linking as the score is lower than 0.728. LM solves 85 of these cases, which amounts to 22.9% of them. On the other hand, for 826 cases, BI_{AD} ranks the correct entity between the second place and the twelfth place. LM solves 376 of these cases, which amounts to 45.5%. Thus, it can be said that LM helps with both these cases. When the features of LM are investigated, it can be seen from the figure in Table 4.13 that both commonness and the score of BI_{AD} have very high coefficients, followed by the lexical similarity which is has less than half the magnitude. As LM is linear, this trend can be directly seen when the samples are manually investigated. The instances for which LM manages to fix the errors are the ones where BI_{AD} assigns high scores to the correct entities, and possibly a high commonness value and lexical similarity is also observed. The samples where the score assigned by BI_{AD} to the correct entity is lower than 0.5, the mistakes are a lot harder to fix and require very high commonness values. Hence, LM helps for the more obvious errors, and when the score assigned by BI_{AD} is not too low. Among the 371 and 826 error cases mentioned, GBM_{F5} manages to solve 163 and 397 of them respectively, which amounts to 43.9% and 48%. Hence, it can be seen that most of the improvement over LM comes from a better NIL mention threshold.

When GBM_{F5} is used instead of LM, the accuracy is improved by 0.6%. The biggest improvement came from the "EE" setting where F1 score increased by 1.1%. This shows that a linear model was not able to fully utilize the features in an effective way and a more complex model was necessary. An interesting experiment could be to change GBM_{F5} by a neural network to see if the observed improvement can be sustained.

De Cao et al. [13] report that BLINK almost always makes an accurate prediction when the mention is matching exactly to the entity name. However, this is not always the case for BI_{AD}. It is believed that there may be two possible reasons. First is that the entity representation contains the concatenation of all possible labels, so it could be the case that the significance of having an exact match with a label is not understood by the model. However, BLINK's representation also does not solely consist of the title of the entity as the description is also used. Hence, if this was the case, we believe that a similar problem could occur with BLINK as well. Second reason could be that the boundaries of the mention cannot be separated well from the context due to the fact that the representation of the separator tokens ($[M_s]$ and $[M_e]$) are not learned well with the current amount of training instances. Still, no concrete proof is obtained to explain this trend.

Some samples from Validation were manually investigated such that BI_{AD} + LM was correct and GBM_{F26} was wrong, or vice versa, to better understand the strengths and weaknesses of the developed approach. It was interesting to see that some very easy cases are missed completely by BI_{AD} + LM, and some hard cases were disambiguated correctly. Some examples of these instances are shown below. The gold mention is underlined.

".... National Institutes of Natural Sciences ..."
Correct Entity's Labels: "National Institutes of Natural Sciences" [a]

'... BBDC ..."
Correct Entity's Labels: "Banting and Best Diabetes Centre, University of Toronto", "BBDC" [b]

[a] `https://www.nins.jp/en/`
[b] `https://bbdc.org/`

Figure 4.9: Easy cases where BI_{AD} and LM failed whereas GBM_{F26} found the correct entity

"... MSMT CR ..."
Correct Entity's Labels: "Ministerstvo Školství, Mládeže a Tělovýchovy", "MŠMT" [a]

"... US DOE BES MSE ..." **Correct Entity's Labels:** "Division of Materials Sciences and Engineering" [b]

[a] https://www.msmt.cz/
[b] https://www.energy.gov/

Figure 4.10: Hard cases where BI_{AD} and LM found the correct entity whereas GBM_{F26} failed

Lastly, a similar error analysis is done to compare BI_{AD} + GBM_{F5} and GBM_{F26}. First, 10 instances out of 1074 were randomly sampled from Validation where both models made an error. In 6 of these samples, the errors were due to incorrect labels in the gold annotation or duplicate entities in the KB. In 3 instances, there was not enough information to decide whether the gold annotation or the models' decision was correct. Only for one instance, the models made an error as they could not assign a score to the correct entity high enough to link the mention to it. Then, 20 instances out of 698 were sampled such that GBM_{F26} was wrong and BI_{AD} + GBM_{F5} was correct. Interestingly, some systematic error patterns were observed. For example, if there is a typo in the name regarding whitespaces, GBM_{F26} cannot find the correct entity as it is not included in the candidate set due to BM25 working on word level, such as not being able to decide "BradleyUniversity" is Bradley University[5]. Four such instances were observed out of 20 samples. In 4 instances, the wrong entity was selected as GBM_{F26} does not use context information. For example, the mention "Ministry of Education" was linked to the one in South Korea, even though it was clear from the context that it was the Taiwanese ministry. Also, there were 2 instances where the mention was an acronym, and GBM_{F26} chose another organization with the same acronym whereas the correct one could be found when the context is utilized. However, when 20 instances out of 676 were sampled where BI_{AD} + GBM_{F5} was wrong and GBM_{F26} was correct, no pattern is observed, and only 5 instances were errors that are not related to NIL mentions or incorrect annotations. This could be attributed to the neural model not being as explainable. Both these 20-sample sets contained instances where one model was better than the other in terms of NIL mention detection. However, no pattern was observed for this trend either. It could be because none of the models excel at this task.

There were some concerns on whether a neural approach would work in funding domain with the dataset and KB at hand. Most of the neural approaches either perform neural ED by using a classifier over the whole entity space [10], or they utilize entity embeddings [90, 97, 31]. The former was not possible due to the Training split not covering the whole entity space. For the latter, there are many successful methodologies, such as TransE or Wikipedia2Vec. However, these are not suitable for this task as most of the entities in this KB do not exist in general-purpose KBs and there is no informative graph structure. The biencoder used in the BLINK architecture enabled obtaining entity embeddings with the information available. Even though the neural ED model did not outperform the feature-based model, this work showed that it was indeed possible to adapt entity embeddings and neural approaches for the problem of funding organization disambiguation. Table 4.18 shows that there is only a slight performance difference between the feature-based model and the neural model, despite the fact that the former requires significant effort and domain knowledge for feature generation. We believe that it is also a success of BLINK's system that it can perform on par with this model,

[5] https://www.bradley.edu/

given the low amount of training data and the fact that a completely different KB is being used. However, this work also revealed that there are some information that the biencoder could not capture, such as lexical similarity and dataset statistics that needed to be incorporated externally by using LM, or even better, GBM_{F5}.

Overall EL Performance

The overall EL performance of different NER and ED models combined are presented in Table 4.20. These results show the performance of the respective components in terms of extracting funding organization information.

First thing to note is that, as expected, the performance increases for all cases when BERT_{SC}^{NER} is used instead of Stanford NER. As mentioned, in Validation split, BERT_{SC}^{NER} improves the recall for 10.9%. It is possible to see a similar recall improvement for all ED systems when BERT_{SC}^{NER} is used as the NER component. This suggests that improving BERT_{SC}^{NER} could possibly push the performance further.

As for precision, the 6.5% improvement gained by BERT_{SC}^{NER} can be seen in the "Normal" and "EE" EL evaluation settings. However, in "InKB" setting, the increase in precision is lower than expected. There may be a few reasons on why this is the case. The increase in precision could be attributed to less false positive and/or more true positive mentions extracted by the NER. It is possible that the new true positive or the removed false positive mentions are referring to emerging entities.

On "Normal" setting, Validation results show only a minor difference of micro averaged F1 score when LM or the feature-based model is used for disambiguation. For the "InKB" setting, the difference is almost non-existent. The reason for it could be that the instances where the feature-based model perform better were not extracted correctly by the NER components, and most of the performance difference comes from the power of detecting emerging entities.

In all settings, the performance on emerging entities are quite low. Even the best model combination has a micro averaged F1 score of 52.82 on Test, which is immensely lower than that of the "InKB" setting, 85.14. This further supports the fact that another methodology should be used for NIL mention detection.

When Test results on Table 4.19 are checked, it can be seen that the neural approach improves the setting where feature-based models are used by a great margin on all evaluation settings, thanks to the success of BERT_{SC}^{NER}.

Some of the samples where the developed method makes mistakes were investigated manually. One of the interesting findings was that as BI_{AD} uses context, it sometimes helps with some errors regarding the mention spans. An example is shown below.

> "... South African National Research Foundation ..."
> **Correct Mention:** "South African National Research Foundation"
> **Extracted Mentions:** "South" and "National Research Foundation"
> **Linked to:** First to NIL, second to correct entity[a]
>
> [a] `https://www.nrf.ac.za/`

Figure 4.11: BI_{AD} finds correct entity even though mention span is not extracted correctly.

It is not clear from the gold annotations whether the funding programs should also be extracted, similar to funding organizations. The KB also contains some entities for specific funding programs, but there is no distinction made regarding whether an entity is a funding organization or a funding program. This distinction would be highly valuable. Many ED research, including DeepType [71], reports that type information is

the key to link ambiguous mentions, and also mention type classification is deemed highly important for NER systems. Also, the trend of annotating some and not annotating other funding programs deteriorates the quality of the training data as it introduces systematic errors, confusing the modes. Below, an example of such case is shown.

"... Support Program for the Top Young Talents of Hebei Province ..."
Example Program: "Support Program for the Top Young Talents of Hebei Province"
=> This program is not annotated as a mention in the training set.

"... Air Force Office of Scientific Research Young Investigator Program Award ..."
Gold Mention: "Air Force Office of Scientific Research Young Investigator Program Award"
Extracted Mention: "Air Force Office of Scientific Research"
Linked to: The organization offering the grant program[a]

[a] `https://www.afrl.af.mil/AFOSR/`

Figure 4.12: Example of an inconsistent annotation

Chapter 5

Conclusion

In this thesis, state-of-the-art neural Named Entity Recognition (NER) and Entity Disambiguation (ED) approaches are investigated and are adapted to funding domain to tackle the Entity Linking (EL) problem in funding information extraction. The aim was to investigate whether these general-purpose neural approaches would be suitable for a domain-specific application where a knowledge base (KB) other than Wikipedia is used and labelled data is limited.

For the NER task, the NER architectures proposed by Akbik et al. [2] and Devlin et al. [18], Flair^{NER} and BERT^{NER}, are trained and compared. Both models performed extremely well for Grant mentions, and had a similar F1 Score for Organization mentions. However, it is noticed that the recall of BERT^{NER} was significantly higher than that of Flair^{NER}, and Flair^{NER} had a much higher precision. Also, the training for Flair^{NER} took much longer as it needed a lot more epochs to achieve this performance. At the end, it is decided to continue with BERT^{NER} instead of Flair^{NER} as recall is preferred over precision. We believe that this trend may persist on other datasets as well, and we would suggest using Flair^{NER} for tasks that require high precision, and BERT^{NER} for the ones requiring high recall.

A new BERT model, BERT$_{SC}$, is developed by pretraining BERT$_{BASE}$ with sentences where funding information is acknowledged. For this purpose, the Task-Adaptive Pretraining (TAPT) strategy proposed by Gururangan et al. [34] is followed. Later on, the effectiveness of domain adaptation is shown on the NER task, where BERT$_{SC}^{NER}$ outperformed BERT^{NER} in two different hyperparameter settings. Based on these results, it is decided to use BERT$_{SC}^{NER}$ as the neural NER component for this thesis. BERT$_{SC}^{NER}$ is then compared with Stanford NER, and large gains of performance is observed. BERT$_{SC}^{NER}$ outperformed Stanford NER component by 2.9, 12.4 and 7.6% in precision, recall and F1 Score for Organization mentions, and 0.6%, 3.3% and 1.9% for Grant mentions. It is believed that using a complex model such as BERT$_{SC}^{NER}$ is not necessary to detect grant mentions, but as they are crucial parts of funding information, it is better to have a single model that can extract both types of mentions. In the beginning of the thesis, there was skepticism towards using BERT-based models for extracting Grant mentions. BERT makes use of WordPiece embeddings, and hence the grant numbers which are essentially a combination of letters and digits are broken down to characters or mostly groups of characters. However, it was observed that BERT is able to recognize this patterns as Grant mention successfully. BERT$_{SC}^{NER}$ obtained F1 scores of 82.08 and 95.23 for Organization and Grant mentions respectively on the Test split.

To tackle ED of funding organization, BLINK's architecture is selected for experimentation. BLINK offers scalable inference time, zero-shot linking capabilities and a

flexible architecture that can be modified easily. Also, the model is shown to be successful in different public benchmark datasets. BLINK uses an entity's title and description for representation. As this information is not available in this work, possible names of the organizations and country of origin are used as representation. Also, the BERT models of BLINK are replaced with BERT_{SC} due to its success in the NER task. Initially the candidate selector of BLINK is trained and the hyperparameters are scaled down to fit the computational power at hand. However, the training was not successful. Hence, a modified version of BLINK's candidate selector, BI_{AD} is proposed. BI_{AD} utilizes a binary classification setting, and hence can be trained without the need of large amounts of memory. Another advantage of BI_{AD} is that it can handle NIL mentions naturally, which is not included in the research of BLINK.

The performance of BI_{AD} as an ED system is compared with GBM_{F26}. It is observed that BI_{AD} performed 2.6% lower than GBM_{F26} in terms of micro averaged accuracy. An error analysis showed that even though BI_{AD} can handle hard samples, it did not match the performance of GBM_{F26} when the samples were very easy, such as the cases where the mention was matching exactly with the entity's label. As the inference time is important for this thesis, instead of the cross-encoder reranker of BLINK, a linear reranker, LM is implemented. LM is a logistic regression model that works on top of 12 candidate entities extracted by BI_{AD}, and utilizes classical ED features such as commonness, lexical similarity and link probability. LM improved the performance of the developed system as it aided fixing the most obvious errors.

As the error analysis revealed that LM was not capable of fixing some mistakes of BI_{AD} due to being linear, a nonlinear reranker with the same features, GBM_{F5} is trained. This model outperformed both LM and GBM_{F26} on Validation split for ED. This showed that a nonlinear model was needed to utilize the additional features properly, and that the neural model did lack competency when it comes to lexical similarity and statistical features as the performance increased rapidly when these were incorporated into the architecture with a reranker. On Test split, BI_{AD} + GBM_{F5} outperformed GBM_{F26} by 0.4% in terms of micro averaged accuracy.

Lastly, the EL capabilities of the developed approaches are compared. It was seen that using BERT_{SC}^{NER} instead of Stanford NER improved the performance for this task regardless of the ED model used. The EL system utilizing BERT_{SC}^{NER} and BI_{AD} + GBM_{F5} outperformed the feature-based alternatives, obtaining Precision, Recall and F1 scores of 72.9, 79.0 and 75.8 respectively.

One interesting observation is that none of the ED models that were experimented with are able to perform NIL mention detection successfully. It can be seen from ED evaluation that there is 10.5% and 11.8% difference between the F1 scores of "EE" setting and "InKB" setting for BI_{AD} + GBM_{F5} and GBM_{F26} respectively. This difference increases to 32.3% and 34% for EL evaluation. This could suggest that both BERT_{SC}^{NER} and Stanford NER model also performs worse for emerging entities compared to other instances.

In conclusion, it can be seen that it is possible to utilize the latest neural approaches for performing Entity Linking in funding domain. The developed approach that uses BERT_{SC}^{NER} for NER, BI_{AD} for candidate selection and GBM_{F5} for reranking obtained F1 scores of 75.8, 52.8 and 85.1 in "Normal", "EE" and "InKB" settings respectively, outperforming the feature-based EL system by 7.4%, 9.5% and 7.8% in terms of F1 score for the corresponding settings. In terms of Grant mention extraction, BERT_{SC}^{NER} outperformed Stanford NER by 1.9% in terms of F1 score.

5.1 Future Work

Both the results of evaluation and the last manual investigation done on the predictions of the developed models revealed important points for improvement. First of all, it was noticed that detecting emerging entities was the weakness of the models. NIL mention detection is an important part of EL, and it is especially important for the task at hand, as they are observed frequently and are later used to enhance the KB. We believe that a separate system or component to detect NIL mentions would increase the overall performance.

Another weakness of the approach was observed with the NER component. Sometimes, stop words and other tokens that occur frequently in Organization mentions were extracted as singleton mentions due to having a high probability for an "I-ORG" tag. It was also seen that the model could perform poorly when mentions of two different organizations occur side by side. It is believed that utilizing a CRF layer on BERT_{SC}^{NER} could help with these issues tremendously, especially the former. Adding dictionary features extracted from the KB could also help with detecting better mention boundaries as it could push the model to assign higher probability to the patterns observed in KB. For this purpose, the features defined by Wang et al. [84] could be used. However, it should be noted that these kind of features could deteriorate the performance of detecting mentions of emerging entities. Performing end-to-end EL could also improve the performance in terms of mention boundary errors. For this purpose, BI_{AD} could be extended for mention detection following the ELQ model. However, the efficiency could decrease as ELQ requires assigning probabilities to each possible mention span. However, we believe that performing end-to-end EL could increase the performance in general, as it did for ELQ.

If BI_{AD} is improved in the future, GBM_{F5} could be removed. To improve BI_{AD}, an experiment could be to train it with more labeled data. Automatic labeled data generation techniques for EL could be investigated for this purpose. Trying different representations for candidate entities could also be a good experiment.

There were also some inconsistencies observed in the dataset used. The performance can be improved for all models if these inconsistencies are resolved. For example, it should be decided if the country of an organization should be extracted with the rest of the mention or not. Or, whether funding programs are going to be extracted or not. Separating funding programs and organizations in the KB would add value to the performance and to the dataset at hand.

Lastly, completing the training of BERT_{SC} is also left for future work. We hope that the version that is trained for longer could improve the performance of the tasks further, and also could be used for other funding related tasks in the future.

Bibliography

[1] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. Flair: An easy-to-use framework for state-of-the-art nlp. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54-59, 2019.

[2] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th international conference on computational linguistics*, pages 1638-1649, 2018.

[3] Daria Alexander and Arjen P. de Vries. "this research is funded by..": Named entity recognition of financial information in research papers. In *Proceedings of the 11th International Workshop on Bibliometric-enhanced Information Retrieval co-located with 43rd European Conference on Information Retrieval (ECIR 2021)*, volume 2847 of *CEUR Workshop Proceedings*, pages 102 - 110. CEUR-WS.org, 2021.

[4] Krisztian Balog. *Entity-oriented search*. Springer Nature, 2018.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.

[6] Roi Blanco, Giuseppe Ottaviano, and Edgar Meij. Fast and space-efficient entity linking for queries. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 179-188, New York, NY, USA, 2015. Association for Computing Machinery.

[7] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 2787-2795, Red Hook, NY, USA, 2013. Curran Associates Inc.

[8] Peter Bourgonje, Anna Breit, Maria Khvalchik, Victor Mireles, Julian Moreno-Schneider, Artem Revenko, and Georg Rehm. Automatic induction of named entity classes from legal text corpora. In *International Workshop on Artificial Intelligence for Legal Documents (AI4LEGAL2020)*, volume 2722, pages 1-11, November 2020.

[9] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Statistics/Probability Series. Wadsworth Publishing Company, Belmont, California, U.S.A., 1984.

[10] Samuel Broscheit. Investigating entity knowledge in BERT with simple neural end-to-end entity linking. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 677-685, Hong Kong, China, November 2019. Association for Computational Linguistics.

[11] Razvan Bunescu and Marius Paşca. Using encyclopedic knowledge for named entity disambiguation. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy, April 2006. Association for Computational Linguistics.

[12] Mikhail Burtsev, Alexander Seliverstov, Rafael Airapetyan, Mikhail Arkhipov, Dilyara Baymurzina, Nickolay Bushkov, Olga Gureenkova, Taras Khakhulin, Yuri Kuratov, Denis Kuznetsov, Alexey Litinsky, Varvara Logacheva, Alexey Lymar, Valentin Malykh, Maxim Petrov, Vadim Polulyakh, Leonid Pugachev, Alexey Sorokin, Maria Vikhreva, and Marat Zaynutdinov. DeepPavlov: Open-source library for dialogue systems. In *Proceedings of ACL 2018, System Demonstrations*, pages 122-127, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[13] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. In *International Conference on Learning Representations*, 2021.

[14] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

[15] Marco Cornolti, Paolo Ferragina, Massimiliano Ciaramita, Stefan Rüd, and Hinrich Schütze. Smaph: A piggyback approach for entity-linking in web queries. *ACM Trans. Inf. Syst.*, 37(1), December 2018.

[16] S. Dai, Y. Ding, Z. Zhang, W. Zuo, X. Huang, and S. Zhu. Grantextractor: Accurate grant support information extraction from biomedical fulltext based on bi-lstm-crf. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(1):205–215, 2021.

[17] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, pages 1646-1654, Cambridge, MA, USA, 2014. MIT Press.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171-4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[19] Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1-10, 2014.

[20] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*

(Volume 1: Long Papers), pages 334-343, Beijing, China, July 2015. Association for Computational Linguistics.

[21] Jacob Eisenstein. *Natural Language Processing*. GitHub, 2018.

[22] Borja Espejo-Garcia, Francisco J. Lopez-Pellicer, Javier Lacasta, Ramón Piedrafita Moreno, and F. Javier Zarazaga-Soria. End-to-end sequence labeling via deep learning for automatic extraction of agricultural regulations. *Computers and Electronics in Agriculture*, 162:106–111, 2019.

[23] Paolo Ferragina and Ugo Scaiella. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). CIKM '10, pages 1625-1628, New York, NY, USA, 2010. Association for Computing Machinery.

[24] Paolo Ferragina and Ugo Scaiella. Fast and accurate annotation of short texts with wikipedia pages. *IEEE Softw.*, 29(1):70-75, January 2012.

[25] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363-370, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[26] Kathleen C. Fraser, Isar Nejadgholi, Berry de Bruijn, Muqun Li, Astha LaPlante, and Khaldoun Zine El Abidine. Extracting UMLS concepts from medical text using general and domain-specific deep learning models. *CoRR*, abs/1910.01274, 2019.

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

[28] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[29] Pablo Gamallo, Jose Ramom Pichel, and Iñaki Alegria. A perplexity-based method for similar languages discrimination. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 109-114, Valencia, Spain, April 2017. Association for Computational Linguistics.

[30] Octavian-Eugen Ganea and Thomas Hofmann. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2619-2629, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[31] Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. Learning dense representations for entity retrieval. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 528-537, Hong Kong, China, November 2019. Association for Computational Linguistics.

[32] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645-6649. Ieee, 2013.

[33] Nitish Gupta, Sameer Singh, and Dan Roth. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2681-2690, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[34] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of ACL*, 2020.

[35] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. Entity linking in queries: Tasks and evaluation. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, ICTIR '15, pages 171-180, New York, NY, USA, 2015. Association for Computing Machinery.

[36] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. Entity linking in queries: Efficiency vs. effectiveness. In Joemon M Jose, Claudia Hauff, Ismail Sengor Altıngovde, Dawei Song, Dyaa Albakour, Stuart Watt, and John Tait, editors, *Advances in Information Retrieval*, pages 40-53, Cham, 2017. Springer International Publishing.

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

[38] Johannes Hoffart, Yasemin Altun, and Gerhard Weikum. Discovering emerging entities with ambiguous names. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 385-396, New York, NY, USA, 2014. Association for Computing Machinery.

[39] 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 2011 Conference on Empirical Methods in Natural Language Processing*, pages 782-792, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.

[40] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In Aasa Feragen, Marcello Pelillo, and Marco Loog, editors, *Similarity-Based Pattern Recognition*, pages 84-92, Cham, 2015. Springer International Publishing.

[41] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[42] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv preprint arXiv:1905.01969*, 2019.

[43] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *International Conference on Learning Representations*, 2020.

[44] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 687-696, Beijing, China, July 2015. Association for Computational Linguistics.

[45] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, pages 1–1, 2019.

[46] Hideaki Joko, Faegheh Hasibi, Krisztian Balog, and Arjen P. de Vries. Conversational entity linking: Problem definition and datasets. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21)*. ACM, July 2021.

[47] Subhradeep Kayal, Zubair Afzal, George Tsatsaronis, Marius Doornenbal, Sophia Katrenko, and Michelle Gregory. A framework to automatically extract funding information from text. In Giuseppe Nicosia, Panos Pardalos, Giovanni Giuffrida, Renato Umeton, and Vincenzo Sciacca, editors, *Machine Learning, Optimization, and Data Science*, pages 317-328, Cham, 2019. Springer International Publishing.

[48] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 3149-3157, Red Hook, NY, USA, 2017. Curran Associates Inc.

[49] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[50] Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. End-to-end neural entity linking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 519-529, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[51] N. Kurz, F. Hamann, and A. Ulges. Neural entity linking on technical service tickets. In *2020 7th Swiss Conference on Data Science (SDS)*, pages 35–40, 2020.

[52] Phong Le and Ivan Titov. Improving entity linking by modeling latent relations between mentions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1595-1604, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[53] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 09 2019.

[54] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871-7880, Online, July 2020. Association for Computational Linguistics.

[55] Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. Efficient one-pass end-to-end entity linking for questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6433-6441, Online, November 2020. Association for Computational Linguistics.

[56] Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wiegers, and Zhiyong Lu. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database*, 2016, 2016.

[57] Tianyu Liu, Jin-Ge Yao, and Chin-Yew Lin. Towards improving neural named entity recognition with gazetteers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5301-5307, Florence, Italy, July 2019. Association for Computational Linguistics.

[58] Pedro Henrique Martins, Zita Marinho, and André F. T. Martins. Joint learning of named entity recognition and entity linking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 190-196, Florence, Italy, July 2019. Association for Computational Linguistics.

[59] Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, pages 1-8, New York, NY, USA, 2011. Association for Computing Machinery.

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

[61] Ishani Mondal, Sukannya Purkayastha, Sudeshna Sarkar, Pawan Goyal, Jitesh Pillai, Amitava Bhattacharyya, and Mahanandeeshwar Gattu. Medical entity linking using triplet network. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 95-100, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics.

[62] Isaiah Onando Mulang', Kuldeep Singh, Chaitali Prabhu, Abhishek Nadgeri, Johannes Hoffart, and Jens Lehmann. Evaluating the impact of knowledge graph context on entity disambiguation models. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, pages 2157-2160, New York, NY, USA, 2020. Association for Computing Machinery.

[63] Keval Nagda, Anirudh Mukherjee, Milind Shah, Pratik Mulchandani, and Lakshmi Kurup. Ascent of pre-trained state-of-the-art language models. In Hari Vasudevan, Antonis Michalas, Narendra Shekokar, and Meera Narvekar, editors, *Advanced Computing Technologies and Applications*, pages 269-280, Singapore, 2020. Springer Singapore.

[64] Yasumasa Onoe and Greg Durrett. Fine-grained entity typing for domain independent entity linking. In *AAAI*, 2020.

[65] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024-8035. Curran Associates, Inc., 2019.

[66] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825-2830, 2011.

[67] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532-1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[68] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227-2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[69] Nina Poerner, Ulli Waltinger, and Hinrich Schütze. Inexpensive domain adaptation of pretrained language models: Case studies on biomedical NER and covid-19 QA. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1482-1490, Online, November 2020. Association for Computational Linguistics.

[70] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020.

[71] Jonathan Raiman and Olivier Raiman. Deeptype: multilingual entity linking by neural type system evolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[72] Petar Ristoski and Heiko Paulheim. Rdf2vec: Rdf graph embeddings for data mining. In Paul Groth, Elena Simperl, Alasdair Gray, Marta Sabou, Markus Krötzsch, Freddy Lecue, Fabian Flöck, and Yolanda Gil, editors, *The Semantic Web ISWC 2016*, pages 498-514, Cham, 2016. Springer International Publishing.

[73] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. Okapi at trec-3. *Nist Special Publication Sp*, 109:109, 1995.

[74] Elliot Schumacher, Andriy Mulyar, and Mark Dredze. Clinical concept linking with contextualized neural representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8585-8592, Online, July 2020. Association for Computational Linguistics.

[75] Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. Learning named entity tagger using domain-specific dictionary. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2054-2064, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[76] Mujeen Sung, Hwisang Jeon, Jinhyuk Lee, and Jaewoo Kang. Biomedical entity representations with synonym marginalization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3641-3650, Online, July 2020. Association for Computational Linguistics.

[77] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104-3112, Cambridge, MA, USA, 2014. MIT Press.

[78] Wen Tai, H. T. Kung, Xin Dong, Marcus Comiter, and Chang-Fu Kuo. exBERT: Extending pre-trained models with domain-specific vocabulary under constrained training resources. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1433-1439, Online, November 2020. Association for Computational Linguistics.

[79] 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*, pages 142-147, 2003.

[80] Ricardo Usbeck, Michael Röder, Michael Hoffmann, Felix Conrads, Jonathan Huthmann, Axel-Cyrille Ngonga Ngomo, Christian Demmler, and Christina Unger. Benchmarking question answering systems. *Semantic Web*, 10(2):293-304, 2019.

[81] Johannes M. van Hulst, Faegheh Hasibi, Koen Dercksen, Krisztian Balog, and Arjen P. de Vries. Rel: An entity linker standing on the shoulders of giants. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20. ACM, 2020.

[82] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[83] Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Darrin Eide, Kathryn Funk, Rodney Kinney, Ziyang Liu, William Merrill, et al. Cord-19: The covid-19 open research dataset. *ArXiv*, 2020.

[84] Qi Wang, Yangming Zhou, Tong Ruan, Daqi Gao, Yuhang Xia, and Ping He. Incorporating dictionaries into deep neural networks for the chinese clinical named entity recognition. *Journal of Biomedical Informatics*, 92:103133, 2019.

[85] Maciej Wiatrak and Juha Iso-Sipila. Simple hierarchical multi-task neural end-to-end entity linking for biomedical text. In *Proceedings of the 11th International Workshop on Health Text Mining and Information Analysis*, pages 12-17, Online, November 2020. Association for Computational Linguistics.

[86] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38-45, Online, October 2020. Association for Computational Linguistics.

[87] Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. Named entity recognition with context-aware dictionary knowledge. In *Proceedings of the 19th Chinese National Conference on Computational Linguistics*, pages 915-926, Haikou, China, October 2020. Chinese Information Processing Society of China.

[88] Jian Wu, Pei Wang, Xin Wei, Sarah Rajtmajer, C. Lee Giles, and Christopher Griffin. Acknowledgement entity recognition in CORD-19 papers. In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 10-19, Online, November 2020. Association for Computational Linguistics.

[89] Junshuang Wu, Richong Zhang, Yongyi Mao, Hongyu Guo, Masoumeh Soflaei, and Jinpeng Huai. Dynamic graph convolutional networks for entity linking. In *Proceedings of The Web Conference 2020*, WWW '20, pages 1149-1159, New York, NY, USA, 2020. Association for Computing Machinery.

[90] Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397-6407, Online, November 2020. Association for Computational Linguistics.

[91] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[92] Vikas Yadav and Steven Bethard. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145-2158, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[93] Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, and Yuji Matsumoto. Wikipedia2Vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 23-30, Online, October 2020. Association for Computational Linguistics.

[94] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442-6454, Online, November 2020. Association for Computational Linguistics.

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

[96] Mu Yang, Chi-Yen Chen, Yi-Hui Lee, Qian-hui Zeng, Wei-Yun Ma, Chen-Yang Shih, and Wei-Jhih Chen. Headword-oriented entity linking: A special entity linking task with dataset and baseline. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1910-1917, Marseille, France, May 2020. European Language Resources Association.

[97] Xiyuan Yang, Xiaotao Gu, Sheng Lin, Siliang Tang, Yueting Zhuang, Fei Wu, Zhigang Chen, Guoping Hu, and Xiang Ren. Learning dynamic context augmentation for global entity linking. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 271-281, Hong Kong, China, November 2019. Association for Computational Linguistics.

[98] Qingkai Zeng, Wenhao Yu, Mengxia Yu, Tianwen Jiang, Tim Weninger, and Meng Jiang. Tri-train: Automatic pre-fine tuning between pre-training and fine-tuning for SciNER. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4778-4787, Online, November 2020. Association for Computational Linguistics.

[99] Hongzhi Zhang, Weili Zhang, Tinglei Huang, Xiao Liang, and Kun Fu. A two-stage joint model for domain-specific entity detection and linking leveraging an unlabeled corpus. *Information*, 8(2), 2017.

[100] Sendong Zhao, Ting Liu, Sicheng Zhao, and Fei Wang. A neural multi-task learning framework to jointly model medical named entity recognition and normalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 817-824, 2019.

[101] Ming Zhu, Busra Celikkaya, Parminder Bhatia, and Chandan K. Reddy. Latte: Latent type modeling for biomedical entity linking. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9757–9764, Apr. 2020.

[102] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27, 2015.

[103] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301-320, 2005.

Appendix A

Hyperparameters for GBM_{F5}

GBM_{F5} is trained using the LightGBM [48] Python library. Table A.1 shows the hyperparameters for this model.

Maximum Number of Bins	63
Learning Rate	0.1
Minimum Number of Samples per Leaf	100
Bagging Frequency	1
Bagging Fraction	0.9
L1 Regularization Strength	1
L2 Regularization Strength	1
Minimum Gain to Split	0.1
Objective	Binary Classification
Metric	Binary Cross-Entropy
NIL Detection Threshold	0.042

Table A.1: Hyperparameters for GBM_{F5}

Appendix B

NER Data Preprocessing

Below, you may find the steps to assign labels to tokens using the gold annotations in sequential order.

1. Label tokens of ORG mentions. Sometimes, annotators tend to extract mentions not as a continuous span, but rather a list of individual words. If there more than two characters in-between, take the first continuous set of words. The decision of not taking the mention from the first annotated word until the last is based on the cases where there are too many characters or grant mentions in-between these words. It was observed that the first span mostly contained the important words to be able to identify the organization. Example annotations where underlined text corresponds to a single mention based on the gold annotation:

   (a) National Instituteo f Child Healtha ndH umanD evelopmen t
   (b) the Technological Innovation and Demonstration of Social Undertakings Project fund (HS2014003) of Nantong, Jiangsu, China;

2. Remove duplicate ORG mentions based on their position on text. If there are two mentions with same text in different parts of the input, both are kept.

3. Remove ORG mentions that are too long. Very rarely, the annotators extracted too large of a span as a mention, sometimes even the whole article. ORG mentions longer than 200 characters are discarded.

4. If there are overlapping ORG mentions, keep only the one with the largest span. Example overlapping gold annotations:

   (a) "National grant no. Science NSC Council"
   (b) "NSC"

5. Label tokens of GRT mentions. Follow the same rule as the first step for mentions that are not continuous spans.

6. Remove duplicate GRT mentions similar to the second step.

7. Discard the grant mentions that are longer than 100 characters.

8. Resolve overlapping GRT mentions similar to the fourth step.

9. Resolve overlapping ORG and GRT mentions. Keep the label of the ORG mention, if there are tokens left on the right-hand-size, label them as GRT.

Text: “supported by the European Community, FP6 036097-2”
(a) ORG Mention: “European Community, FP6”
(b) GRT Mention: “FP6 036097-2”
(c) Span that is labelled as ORG: “European Community, FP6”
(d) Span that is labelled as GRT: “036097-2”

As the candidate models for NER were BERT-based [18] and Flair-based [2] models, the tokenizers these models use were tried for the tokenization of the input text before assigning the NER labels. After empirical analysis, it was decided the use the tokenizer of the case-sensitive BERT_{BASE} model [18], as it was splitting the text to smaller pieces, which was crucial to minimize labelling errors. One drawback of this tokenizer is that it being a word-piece tokenizer. Hence, it also splits some words into smaller pieces based on the vocabulary of the model. As a post-processing step, these WordPieces are merged back together. The choice of using the same tokenizer through all NER models is to eliminate any effect that can be caused by using different tokenizers during comparison.

Appendix C

Training Details

C.1 BERT$_{SC}$

The table below shows the hyperparameters for Task-Adaptive Pretraining [34].

Number of Epochs:	2
Batch Size:	4
Effective Batch Size:	2048
Maximum Learning Rate:	0.0005
MLM Probability	0.15
Max. Gradient Norm	1
Optimizer	Adam [49]
Learning Rate Scheduler	Linear
Warmup Steps	0
Weight Decay	0
Adam Epsilon	10^{-8}

Table C.1: Hyperparameters for Task-Adaptive Pretraining

C.2 Flair^{NER}

Figure C.1 shows the Training and Dev losses, learning rate and Dev scores over epochs.

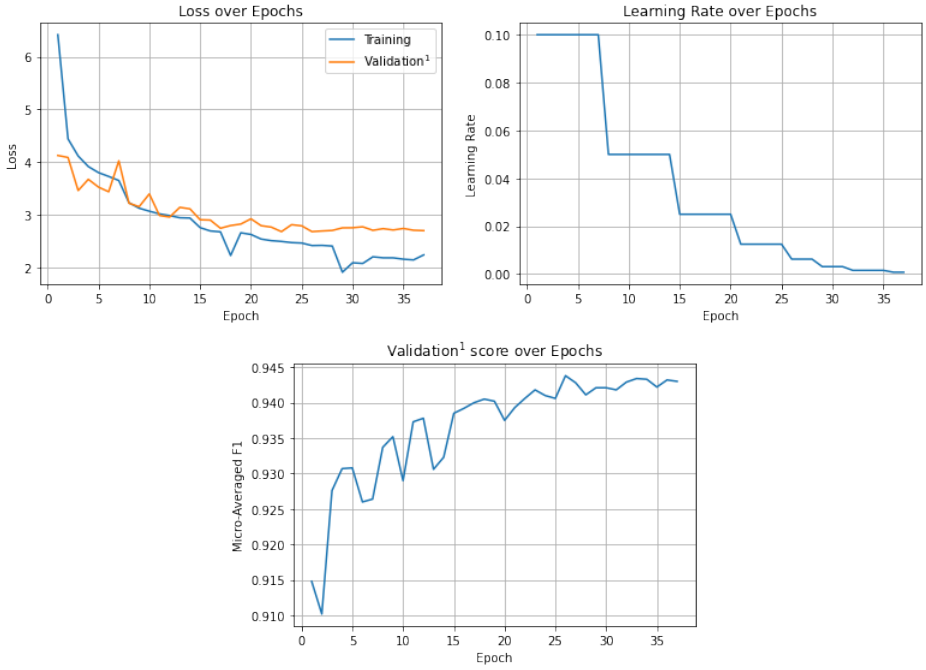


Figure C.1: Losses (up-left), learning rate (up-right) and Dev scores (bottom) per epoch

C.3 BERT^{NER}

The tables below show the results of the models for each hyperparameter configuration on Validation dataset. Table C.2 displays the results where the model was trained for 10 epochs and saved at the end of each. The Validation results are obtained on specific epochs: 2, 3, 4, 6 and 10. 2, 3 and 4 are included as they were among the recommended hyperparameters. Epoch 6 and 10 are included as the former resulted in the highest Dev ORG-F1 score while the latter was the last epoch. Table C.3 shows the results on Validation dataset for the setting where a linear learning rate scheduler is used.

	Organization			Grant		
Epoch	Precision	Recall	F1	Precision	Recall	F1
2	80.38	84.14	82.22	94.18	96.95	95.55
3	78.76	84.9	81.72	94.51	96.78	95.63
4	79.18	84.18	81.6	94.3	96.89	95.58
6	79.88	84.71	82.23	94.73	96.53	95.62
10	81.04	80.24	80.64	94.62	96.38	95.49

Table C.2: BERT^{NER} results on Validation. Trained for 10 epochs without a scheduler, the model is saved after every epoch.

	Organization			Grant		
Epoch	Precision	Recall	F1	Precision	Recall	F1
2	78.44	85.8	81.96	94.59	97.43	95.99
3	79.18	86.03	82.46	94.71	97.39	96.03
4	79.41	85.88	82.52	94.95	97.38	96.15

Table C.3: BERT^{NER} results on Validation. Trained for 2,3 and 4 epochs respectively with a scheduler, the model is saved after the training is done.

As the scores for Grant mentions are high in each setup, the model is chosen based on the scores on Organization mentions. Recall is favored over precision and based on this intuition, the model that is trained for 3 epochs with a learning rate scheduler is selected. This model has the highest recall for Organization mentions and the second highest F1 score.

C.4 BI$_{AD}$

Table C.4 the scores of the models after each round on Training and Dev splits for BI$_{AD}$ with the second hyperparameter setting, the one where one round corresponds to two epochs and maximum number of tokens is set to 256 for the candidate representation.

Dataset	Round	Threshold	Micro Averaged Accuracy	Macro Averaged Accuracy
Training	1	0.5	62.36	64.44
Training	2	0.5	87.06	88.33
Training	3	0.5	90.3	91.09
Training	4	0.5	94.94	95.61
Training	1	0.972	70.53	71.4
Training	2	0.75	91.66	92.29
Training	3	0.759	93.62	94.12
Training	4	0.728	96.16	96.67
Dev	1	0.5	63.62	70.03
Dev	2	0.5	82.56	87.07
Dev	3	0.5	84.16	88.21
Dev	4	0.5	86.26	89
Dev	1	0.972	70.24	75.05
Dev	2	0.75	86.38	89.45
Dev	3	0.759	86.74	89.94
Dev	4	0.728	87.74	90.14

			EE Setting		
Dataset	Round	Threshold	Precision	Recall	F1 Score
Training	1	0.5	100	0.02	0.04
Training	2	0.5	96.68	57.56	72.16
Training	3	0.5	99.13	67.77	80.5
Training	4	0.5	99.16	87.66	93.06
Training	1	0.972	72.15	57.81	64.19
Training	2	0.75	86.11	89.88	87.96
Training	3	0.759	91.59	90.96	91.27
Training	4	0.728	95.85	96.45	96.15
Dev	1	0.5	0	0	0
Dev	2	0.5	83.75	42.69	56.55
Dev	3	0.5	87.22	44.76	59.16
Dev	4	0.5	83.84	60.3	70.15
Dev	1	0.972	66.03	55.7	60.43
Dev	2	0.75	69.64	79.98	74.45
Dev	3	0.759	72.9	73.99	73.44
Dev	4	0.728	74.81	77.91	76.33

			InKB Setting		
Dataset	Round	Threshold	Precision	Recall	F1 Score
Training	1	0.5	62.36	76.58	68.74
Training	2	0.5	85.86	93.79	89.65
Training	3	0.5	89.02	95.44	92.12
Training	4	0.5	94.11	96.6	95.34
Training	1	0.972	70.25	73.44	71.81
Training	2	0.75	93	92.07	92.53
Training	3	0.759	94.08	94.23	94.16
Training	4	0.728	96.23	96.09	96.16
Dev	1	0.5	63.62	75.26	68.95
Dev	2	0.5	82.45	89.85	85.99
Dev	3	0.5	83.89	91.37	87.47
Dev	4	0.5	86.56	91.01	88.73
Dev	1	0.972	70.87	72.9	71.87
Dev	2	0.75	90	87.56	88.76
Dev	3	0.759	89.32	89.07	89.2
Dev	4	0.728	90.22	89.53	89.87

Table C.4: Intermediate Results for BI_{AD}