POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

Fine-grained Named Entity Recognition using Ontology-Guided Knowledge Graphs

Politecnico di Torino

Supervisor

Prof. Andrea BOTTINO

Co-supervisor

Sujoung BAECK

Candidate

HADI NEJABAT

2022

Abstract

The field of Fine-Grained Named Entity Recognition (FG-NER) has received noticeable attention in recent years. Scientific literature, academia, and real-world analysis need fine-grained NER tools to be able to categorize and process a wide range of information and semantics. With the advent of transformer models in the field of NLP, several studies have shown a considerable rise in the performance of transformer-based NER models compared to their prior state-of-the-art methods.

In consonance with literature, many of the most performant NER models in this field are limited to coarse-grained entity labels, with fewer than 10 categories. And there is limited research work on classifying named entities into finer and more detailed subgroups. These labels are far from enough for downstream tasks like improving automated QA systems, powering recommender systems, etc. Moreover, a series of studies dedicated to this scope indicate that fine-grained NER mostly suffers from a lack of sufficient training data.

The work proposed for this thesis project is also motivated by the fact that although FG-NER still generates a lot of coverage in research, it usually lacks the flexibility to produce convincing results in newly introduced domains. Furthermore, the literature is focused on KB-matching and distant-supervised learning, which rely on the use of heavy models to partially solve problems.

We highlight Ontology Guided Named Entity Recognition model (OG-NER), a framework that is capable of improving FG-NER using its ontology guided technique. It introduces a new way to utilize the power of knowledge graphs to adequately leverage the gathering of new named entities from knowledge bases and perform semi-supervised learning to avoid the challenges mentioned, with the additional goal of training a model that can be easily re-trained and fine-tuned with limited preemptive human effort regarding training data annotation, which is an expensive task. The system is trained both on a specific domain of educational background and also on the benchmark open domain of newswire and the results are compared with two of the most cited SotA NER models.

Acknowledgements

First of all, I would like to express my profound appreciation to the Hubert Foundation for giving me the opportunity and the resources to work on this enthralling topic.

A particular thank you goes to Sujoung Baeck, whose knowledge and aid have been critical in realizing this thesis work.

I also wish to thank Professor Andrea Bottino for his enthusiasm in guiding me through this experience.

My utmost appreciation goes out to my parents. Their endless motivation and patient support have been fundamental in helping me achieve all I have. thank you for believing in me and cheering for my every success.

My sincerest gratitude goes my brother Taha for inspiring me to try and stay positive regardless of all obstacles in my path.

Special thanks goes out to my coworkers and the people I'm honored to call friends for being a vital source of support, sharing in my joy during the highs, and inspiring me through the lows. Even when I didn't have confidence in myself, you encouraged me to work toward achieving my potential.

Table of Contents

List of Tables

List of Figures

Acronyms

SotA
State of the Art

NLP
Natural Language Processing

NER
Named Entity Recognition

FG-NER
Fine-Grained Named Entity Recognition

KB
Knowledge Base

KG
Knowledge Graph

EL
Entity Linking

DA
Data Augmentation

BERT
Bidirectional Encoder Representations from Transformers

MLM
Masked Language Model

Chapter 1

Introduction

In this thesis project, we explore the use of transformer based pre-trained language model in combination with Knowledge graph techniques for developing a Fine-grained Named Entity Recognition(FG-NER) model and compare it to the state-of-the-art (SotA) in both analysis of Educational Background domain and Open domain. This project was carried out with the aid of the Hubert.ai Foundation, which proposed the idea for this thesis topic and provided the necessary facilities to carry out the research explained in this document.

1.1 Motivation

In the field of Artificial Intelligence, Chatbots have a lengthy history. Despite their popularity in recent years, chatbots still have certain fundamental flaws that limit the extent of their applications. One of their shortcomings is when processing Named Entities, which leads to their performance being dependent on understanding human in-query. The industry is looking for automated solutions for high-volume recruitment. This is in perfect alignment with Hubert's profession, Chatbot for Recruitment.
A closer inspection reveals that chatbots are equipped with capabilities like Natural Language Processing(NLP) and NLU to conduct contextual dialogues. Additionally, they are smart enough to ask your applicants specific questions and record the answers for use in later discussions. This improves interaction with your candidates and gives chatbot chats a more human feel. Entity ambiguity is one issue that QA systems may encounter. When one thing is used to refer to several meanings. It happens that most of them could benefit from domain-specific and fine-grained named entity recognition models.
The particular choice of architecture and pipeline used for this thesis project was driven by the potential implications of successful semantic enrichment of a

knowledge base applied to Data Augmenter for compensation for the lack of data in training.

1.2 The NER problem

Named Entity Recognition (NER) is the technique of locating particular word groups that have similar semantic properties. NER is an essential part of the knowledge extraction process and is crucial for many downstream applications, such as conversational models, QA systems, and intention prediction. According to recent advancements in this field, NER is struggling more with data than algorithms. Training a NER model is heavily dependent on appropriate and annotated data. *Appropriate* refers to the domain of the documents. For example, newspaper stories and the records produced by law firms are not the same as those produced by medical firms. One should match the training data to the type of document you want to analyze. *Annotated* refers to the data simply being labeled for training. These annotations must be reliable to produce a well-performing engine.
In contrast to the normal entity recognition data set used for academic evaluation, the texts seen in the real industrial scene are complicated and diverse, both in format and in terms of the standardization of the text. Another common problem is that some named entities are spanned too long, and this brings about the problem of "entity name discontinuity." This refers to the separation of parts of the entity name and being categorized independently. Considering many machine learning approaches, such as image recognition and classification, ultimately rely on data created by people; NLP tasks are more reliant on expert-annotated data.
This arises from the fact that a change in only one letter of a word can make a drastic change in its meaning and shift that word to another tag class. These sensitive situations lead to data annotation being a costly and laborious process. Semi-supervised learning methods have been introduced to ease some of the vast annotation efforts. This will introduce its own challenges, which, aside from changing the learning algorithm, also throw in the correct choice of data re-compensation methods like data augmenters, knowledge-bases, and gazetteers. More descriptively, this problem has also affected the following divisions, which we will address in our work.

1.2.1 Fine-grained NER

Scientific literature, academia, and real-world analysis need fine-grained NER to be able to categorize and process a wide range of information and semantics. In consonance with literature, many of the most performant NER models in this field are limited to coarse-grained typing, with fewer than 10 categories. Most well-known classes(tags) are: [`PER`: Person, `ORG`: Organization, `LOC`: Location, and

`MISC`: Miscellaneous]. And there is limited research work in classifying named entities into finer and more detailed subgroups(e.g., organization tags can be broken down into universities, companies, etc) to better categorize the data for the predetermined purpose. These tags are far from enough for downstream tasks like improving automated QA systems, powering recommender systems, etc.
Following our project's vision to recognize an applicant's educational background from his/her conversation with the chatbot, the pipeline falls under the "Personal Data Processing" category, which introduces additional obstacles to the lack of training data and the importance of having great prediction results. In this case, having a structured and carefully labeled dataset is of even greater concern.
Moreover, an excellent feature to have is the improvement of coverage. We want to be able to generalize the model to different contexts and domains by investing a lower amount of time in baseline data annotation and data augmentation reconfiguration.

1.2.2 Ambiguity of NER tags

Making machine-understandable natural language or Natural Language Understanding(NLU) is the primary goal of NLP. However, comprehending natural languages falls under the category of unstructured data, and that is where NER and NLU approaches are applied. One major issue in NLU is ambiguities associated with different stages of language processing, such as morphology, lexicon, parsing, and semantic processing.
Named Entity Disambiguation (NED), a.k.a. Named Entity Linking (NEL), refers to an NLP research topic that is concerned with connecting a reference inside a textual passage to its matching entity in a knowledge base, such as a node in a knowledge graph. This field has many applications with regard to having a better understanding of an entity for research and commercial objectives.
A novel and efficient way of using NED techniques is to deploy them in a distant-learning model to make use of an open-source knowledge base such as Wikidata for relation extraction. Taking into account that the knowledge base is already populated with expert-evaluated data, connection to such a node creates the disambiguation we hope to achieve.

1.3 Purpose and Goal

This thesis will examine several learning techniques, focusing on fine-grained analysis of NER by re-compensation of a lack of data in that curtain domain, in order to determine which techniques produce the most promising results in comparison to traditional DA techniques. The objectives of this project are to:

- Implement and analyze domain-specific fine-grained transformer-based models

for Named Entity Recognition based on the current state of the art.

- Propose and implement an Entity Linking pipeline that aids in semantic enrichment of our knowledge graph, which in turn provides named entities in that target domain for our data augmenter. The focus of this pipeline is on the possibility of creating a standard method that becomes the reference data lookup point for fields suffering from a lack of training data.
- Study and determine appropriate metrics to evaluate and compare the performance of the NER with state-of-the-art models.

1.4 Approach and Methodology

Named Entity Recognition is a topic that has captured the attention of many different research studies over recent years. While little research has been done focusing on fine-grained and domain-specific NER, with the rise in demand for chatbots, this area is in the spotlight of researchers and has been subject to analysis in limited papers. The reasons mentioned earlier have fostered this interest.
The main goal of this thesis work was to provide a straightforward pipeline, from the generation of a dataset to the selection of the proper architecture, while suggesting relevant and important enhancements to previously trustable state-of-the-art models.
The first step of the project concerned the creation of the dataset. There exist many different variations of corpora, with different goals for the definition of NER-tags and intended use. Nonetheless, the lack of data in this curtain domain, as well as the effort required to generate an appropriate dataset, determined the precise scope of the thesis project. Choosing an applicant's educational background pushed the decision towards a domain-specific NER task. In parallel to creating the dataset, we conducted an investigation into the most relevant state-of-the-art transformer-based models in order to establish a baseline performance of the capabilities of the most recent NER technologies. While several NER models have been tested, only the ones relevant to this thesis's contributions will be described in detail. Applying the selected BERT-based NER models and investigating their performance in different scenarios of this specific problem played a fundamental role in choosing the path for our augmenter pipeline's input collection.
The baseline study also came in handy in finding a starting point for proposing improvements and modifications, in conjunction with the hardware and time resources available to run the experiments. In this project, the majority of time is dedicated to working with entity linking, configuring our DA, and reshaping our data to be able to both reproduce and re-train the results of the most performant NER models in order to reach acceptable performance levels worth mentioning in this report.

1.5 Contributions

The main contributions of this thesis project are the following:

- Propose a pipeline that uses Ontology-guided knowledge graphs to collect expert-annotated data. While taking into account the objective of training a model that can be used to recognise detailed information about an applicant's educational background(e.g. university name, field of studies).
- Propose a BERT-based NER architecture that utilizes a knowledge graph on different scales to aim at building a model capable of handling fine-grained NER tasks. To be more precise, the goal is to produce a model that, in combination with our Data Augmenter, is capable of producing results comparable to those of current state-of-the-art models and addresses their shortcomings in ability to generalize into different domains with minimum training data and training cost.
- Propose a proof of concept of the impact of DA techniques to effectively compensate for the lack of annotated data in the mentioned domain.

1.6 Outline

The rest of the report is organized as follows:

- Chapter 2 proposes an overview of the most relevant theories and concepts that are fundamental to the understanding of the work done with this thesis project.
- Chapter 3 proposes an analysis of some of the most relevant state-of-the-art models for NER applications.
- Chapter 4 contains the details of how the data used for the project has been gathered and explains how the previously analyzed state-of-the-art models have been applied to this specific case study, along with the motivations behind the proposed architecture and a comparison of the obtained results.
- Chapter 5 concludes this thesis project with some considerations on the experiments performed and a comment on possible ideas for future work related to what has been proposed.

The results reported in this document appear as they have been obtained without any manipulation. And appropriate sources are given credit wherever possible to avoid plagiarism.

Chapter 2

Relevant Theory

In this section, we will explore the underlying theory of the main topics touched upon in this thesis work. We will start by more formally defining the problem of NER, exploring how it has been tackled via traditional mathematical approaches and how deep learning techniques have been adapted to this sort of problem.
We will next define Deep Learning, discuss its relationship to Machine Learning, and discuss why the field of artificial intelligence is currently playing an increasingly significant role in addressing challenging real-world challenges.
The contents of this chapter are mostly sourced from [1] and [2], with additions from other online sources and papers that are referenced accordingly.

2.1 Named Entity Recognition

Named-entity recognition (NER) (also known as entity identification, entity chunking, and entity extraction) is a sub-task of information extraction that seeks to locate and classify named entities in text into pre-defined categories such as the names of persons, organizations, locations, expressions of time, quantities, monetary values, percentages, etc .
NER systems have been created that use linguistic rule-based techniques as well as statistical models such as machine learning. Hand-crafted grammar-based systems typically obtain better precision, but at the cost of lower recall and months of work by experienced computational linguists . Statistical NER systems typically require a large amount of manually annotated training data. Semi-supervised approaches have been suggested to avoid part of the annotation effort [3]. Information retrieval and natural language understanding both benefit greatly from NER. NER models are categorised into the following groups:

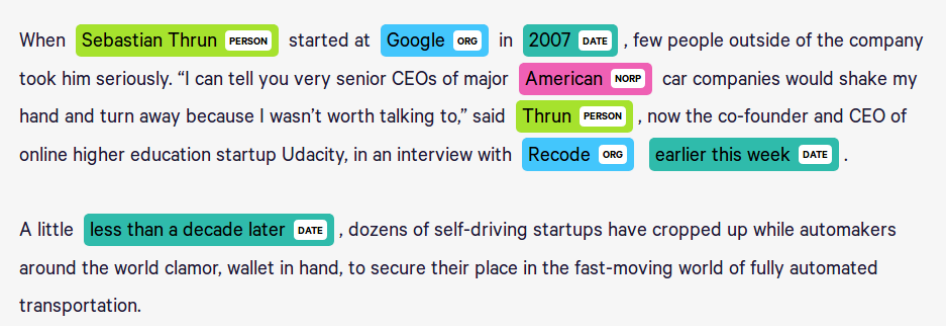

Figure 2.1: NER example visualization with SpaCy [4]

2.1.1 Non-Transformer-based techniques

CRF Model

A Conditional Random Field (CRF) is a statistical model that, because of its ability to consider neighboring samples with respect to the target sample, is usually used for pattern recognition. And in the field of NLP, this potential is used for NER by taking into account the context of the data. For example, a linear chain CRF is similar to a Hidden Markov Model in the sense that it assumes the tag for the current word is only reliant on the tag of one word behind.
If the model reaches a pair like "Thomas Edison University", it will probably categorize "Edison" as a name(`PER-tag`) because when the token that the model is traversing is "Edison" it probably has already tagged "Thomas" as `PER` one token earlier. Adding a Bidirectional Long Short-Term Memory(BiLSTM) network between the inputs (words) and the CRF is one technique to overcome this problem. The bidirectional LSTM is made up of two LSTM networks, one of which receives input going forward and the other going backward. The two networks combined result in an output context that describes the samples that surround each token individually.
A recurrent neural network (RNN) operates in exactly the same way; it has loops that enable information to endure. This fills the gap between effective language models and conventional neural networks. The research work of [ma and hovy 2016] showed that the BiLSTM-CNN-CRF model performed best in terms of F1-score in the English language with and without the use of gazetteers.

2.1.2 Transformer-based techniques

Thanks to the rise in popularity of Deep Learning, extensive research has been performed on neural network-based models applied to NER. In 2017, the work of *Vaswani et al*[6] created a contemporary change in the NLP field by introducing

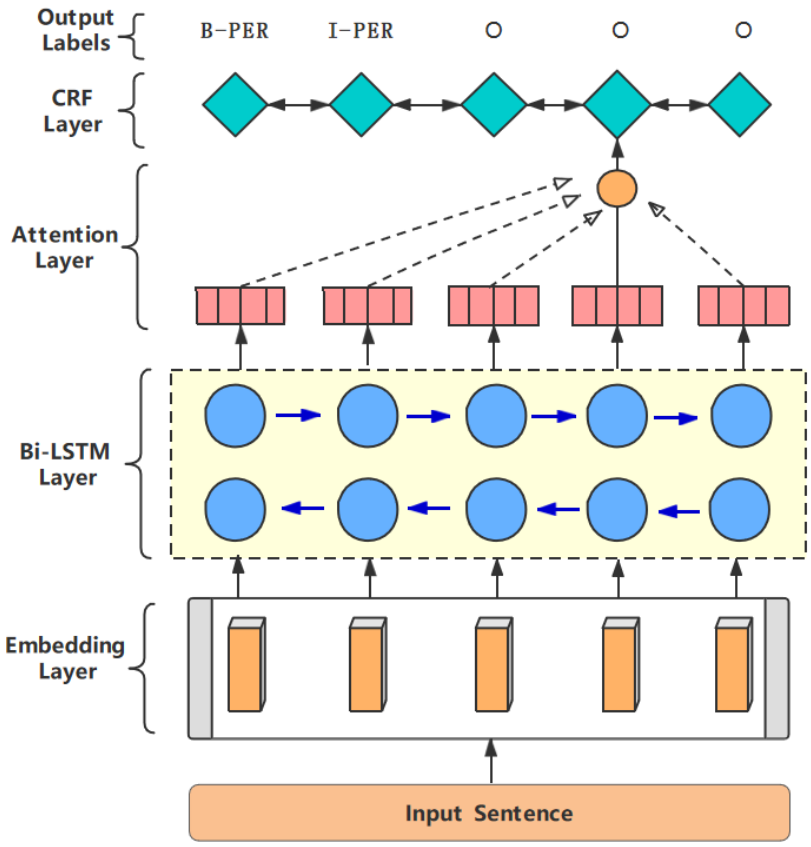

Figure 2.2: BiLSTM Network[5]

the Transformer models, which changed the perspective of many state-of-the-art language modelling techniques such as NER. Transformers allow for better parallelization than LSTMs or other recurrent neural network models since they do not need to process sentences sequentially.
As a result of this advantage, transformers have become essential for cutting-edge NLP models. Following this work, in 2018 **B**idirectional **E**ncoder **R**epresentations from **T**ransformers (BERT) model was made known [7]. BERT is a robust language modeling method that is regarded as one of the most important advances in NLP in recent times. It uses masked language models to enable pre-trained deep bidirectional representations. The input representation of a token is made up of the total of its corresponding position, segment, and token embeddings. Be aware that pre-trained language model embeddings frequently require extensive training corpora and automatically involve auxiliary embeddings (e.g., position and segment embeddings).

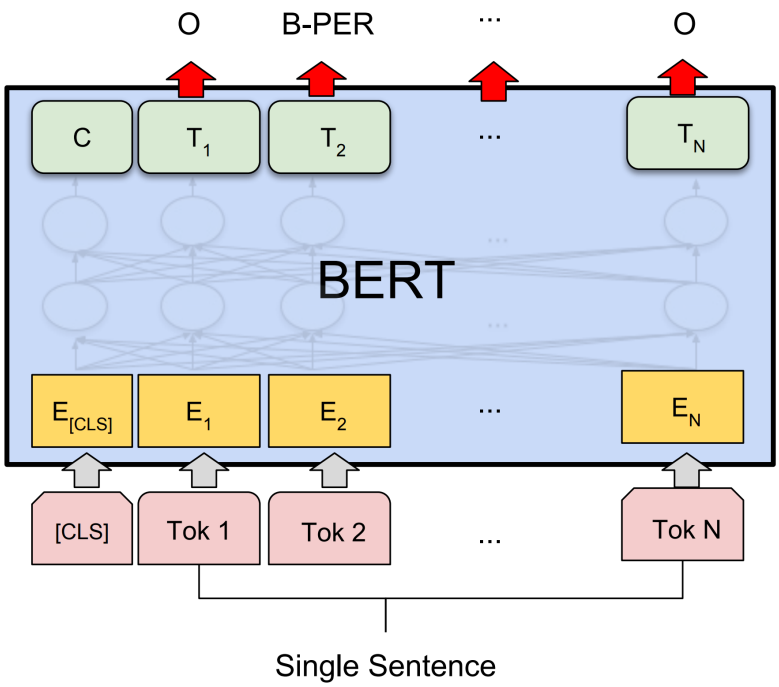

Figure 2.3: BERT NER l[7]

After the success of BERT in different areas of NLP, each downstream task has been separately fine-tuned with proper data and parameters. According to the study by [8] , overall, the transformer-based models outperform CRF and BiLSTM-CNN-CRF in a number of domains in terms of F1-score. Specifically, the results show that the BERT and RoBERTa models yield the highest and second-highest F1 scores for almost every domain. Regarding the precision, the CRF model almost consistently outperforms all of the other models. On the other hand, the transformer-based models significantly outperform the other models with regards to recall. When we apply the models to various domains, we also notice a lot of differences. Furthermore, This implies that although transformer-based models can significantly improve performance, there may not be a major difference in how well they grasp languages.
In the following sections, we will discuss machine learning, deep learning, and different learning methods. And we dive into details about the neural networks of Transformers, BERT, and NER models.

2.1.3 Evaluation metrics

The standard method for NER system evaluation is comparing system output tags to the gold standards(i.e. ones deemed suitable by several expert or inter-annotators agreements). You can use either an Exact-Match or a Relaxed-Match to quantify the comparison. In Exact-Matching, when recognizing a named entity, aside from classifying the correct tag, it is important to precisely select the boundary. This emphasizes the need to include precision and recall into our evaluations. For Relaxed-matching, calculating performance based on the proportion of entity tokens that were identified as the correct entity type, regardless of whether the boundaries of the entity were correct.

- **False Positive (FP):** entity that is returned by a NER model but does not appear in the ground truth
- **False Negative (FN):** entity that is not returned by a NER model but appears in the ground truth.
- **True Positive (TP):** entity that is returned by a NER model and also appears in the ground truth.

The percentage of your system's findings that are accurately identified is referred to as precision. Recall is the proportion of total entities that your system successfully recognized.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{2.4}$$

In addition, the macro-averaged F-score and micro-averaged F-score both consider the performance across multiple entity types. The Macro-averaged F-score independently calculates the F-score on different entity types, then takes the average of the F-scores. The micro-averaged F-score sums up the individual false negatives, false positives, and true positives across all entity types then applies them to get the statistics. The latter can be heavily affected by the quality of recognizing entities in large classes in the corpus [9].
Moreover, the weighted average F1-score is calculated by taking into account the mean of all per-class scores while also considering each class's support. The term "support" describes how many instances of the class there are in the training set.

The output average would have taken into consideration the contribution of each class as weighted by the quantity of samples belonging to that particular class if weighted averaging had been used.
In general, employing the macro average would be an excellent decision as it considers all classes equally when working with an imbalanced dataset where all classes are equally relevant, as is the situation in our project. The weighted average is preferable when an unbalanced dataset is available, yet the objective is to provide classes with more examples in the dataset with a bigger contribution. This is due to the fact that each class's contribution to the F1 average is weighted in weighted averaging according to its size.
Overall, when creating a balanced dataset and seeking a simply defined indicator of performance across all classes, choosing accuracy with micro F1 score is the best option.

These measures are effective for hyperparameter-tuning NER systems. On the other hand, it is more beneficial to analyze with metrics at a full named-entity level when employing the projected named-entities for downstream tasks. It is noteworthy to illustrate a series of occurring scenarios that go above and beyond token-level performance. Noting that excluding all other scenarios and only taking into account the first three, we will have simple and efficient evaluation method that considers false negatives, true positives, false negatives and false positives, and subsequently compute precision, recall and f1-score for each named-entity type.
This method is called Computational Natural Language Learning (CoNLL) and was introduced at CoNLL-2003 where "precision is the percentage of named entities found by the learning system that are correct. Recall is the percentage of named entities present in the corpus that are found by the system. A named entity is correct only if it is an exact match of the corresponding entity in the data file." [10].

2.2 Artificial Intelligence

The origins of the idea of Artificial Intelligence(**AI**) may be traced to ancient cultures, such as philosophers who, as early as 400 BCE, held the view that intellect could be used to choose which actions to carry out and that AI relied on data from an internal language.

Warren McCulloch and Walter Pitts created the initial work that is today widely acknowledged as AI (1943). Ed Feigenbaum and Raj Reddy (1994) created expert systems that encapsulate human knowledge and use it to solve real-world issues. Judea Pearl (2011) established principled probabilistic reasoning tools for dealing with uncertainty.

Scenario	Surface String	Gold Standard	System Prediction
I. Surface string and entity type match	in	O	O
	New	B-LOC	B-LOC
	York	I-LOC	I-LOC
	.	O	O
II. System hypothesized an entity	an	O	O
	Awful	O	B-ORG
	Headache	O	I-ORG
	in	O	O
III. System misses an entity	in	O	O
	Palo	B-LOC	O
	Alto	I-LOC	O
	,	O	O
IV. System assigns the wrong entity type	I	O	O
	live	O	O
	in	O	O
	Palo	B-LOC	B-ORG
	Alto	I-LOC	I-ORG
	,	O	O
V. System gets the boundaries of the surface string wrong	Unless	O	B-PER
	Karl	B-PER	I-PER
	Smith	I-PER	I-PER
	resigns	O	O
VI. System gets the boundaries and entity type wrong	Unless	O	B-ORG
	Karl	B-PER	I-ORG
	Smith	I-PER	I-ORG
	resigns	O	O

Table 2.1: Occuring scenarios in comparing the gold standard annotations with the output of a NER system. Stanford NER on CoNLL [10]

Yoshua Bengio, Geoffrey Hinton, and Yann LeCun (2019) introduced "deep learning" (multilayer neural networks) as the dawn of modern computing. AI has gone through cycles of success, mistaken optimism, and subsequent reductions in excitement and funding. There have also been cycles of bringing new, inventive techniques and refining the finest ones.
AI has progressed significantly since its inception, both theoretically and methodologically. Since AI's issues got more complicated, the area shifted from hand-crafted knowledge to machine learning from data. This has resulted in the enhanced capabilities of real-world systems and improved integration with other disciplines.
Access to massive amounts of data, along with cheaper and faster processing power, and successful implementations of Machine Learning (**ML**) techniques in real-world circumstances, has allowed AI to attract enormous attention from a variety of industries, propelling it to the position it now holds.

2.2.1 Machine Learning

Machine learning research arose from AI, notably from academics' desire to create a mechanism for machines to learn from data. ML techniques are now employed in a wide range of applications, including speech recognition, natural language

processing, and computer vision.
While many people consider ML to be a subset of AI, others contend that just a small portion of ML overlaps with AI and that the two are separate but related sets.

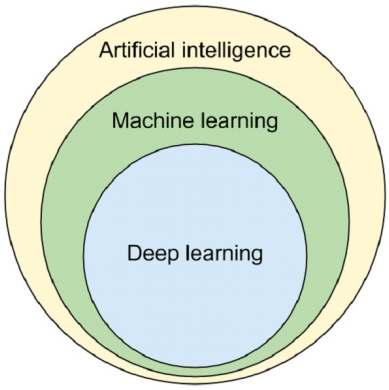

Figure 2.4: ML as a sub-field of AI [11]

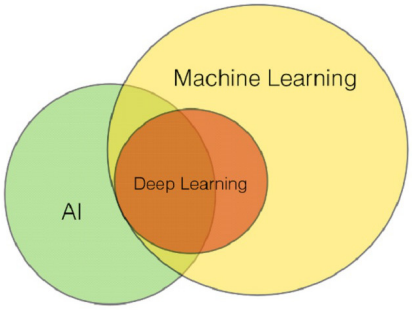

Figure 2.5: Part of ML as a sub-field of AI [11]

Some distinguish the two areas by stating that ML employs passive observations to learn, but AI involves interaction with the environment to learn.
ML approaches are classified into categories based on the type of input accessible to the system, but they all share some characteristics that may be defined by the concept of an ML pipeline. An ML pipeline is a description of the frequent phases in the ML process.

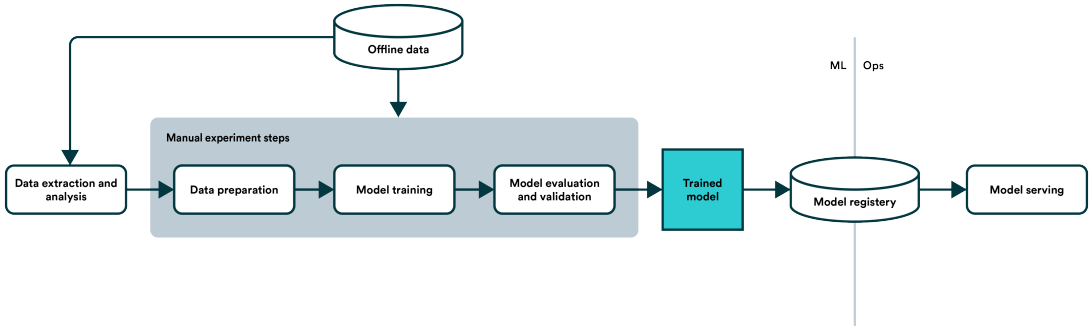

Figure 2.6: ML Pipeline steps [12]

The first phases focus around data collection, extraction, and preparation, which differs depending on the application context. In general, the collected data should be separated into three subsets.: **training set**, **validation set** and **test set**. This division is critical because it allows the subsequent processes to be performed with

as little bias as possible, especially during the validation and testing phases.
Following these processes, several models are created, trained, evaluated, and validated in order to select the one that best responds to the specific criteria.
One challenge which must be avoided in all ML models is overfitting on the training dataset; This leads to the model not being able to generalize to real-world data. So, it is subsequently assessed on the validation set. This step is frequently combined with cross-validation techniques like K-Fold or Leave One Out. The former involves splitting the dataset into K equal-sized "folds" (partitions), training the model on K-1 of them, using the remaining folds for validation, and repeating this process with the remaining folds. The latter works by saving 1 sample for validation while the rest is used for training. Eventually, final validation will be done on test set by training the selected model on the training and validation sets.
There are several performance measures available that may be used to assess a particular model's suitability, and the measure used is largely determined by the type of data, the type of model, and the type of machine learning approach used: Accuracy, F1-score, and the ROC (receiver operating characteristic) curve are some of the most frequently used metrics for supervised learning tasks like classification, while Rand Index, Mutual Information-based scores, and Silhouette are some of the most frequently used metrics for unsupervised tasks like clustering. The properties of major ML approaches will be discussed in the following sections.

Supervised Learning

This method focuses on characteristics that help distinguish between positive and bad examples. As opposed to prior hand-crafted rules, supervised learning now uses a set of training samples to infer rule-based systems or sequence labeling algorithms. Currently, this method is widely used and has numerous variations, including Conditional Random Fields, Support Vector Machines, Decision Trees, Maximum Entropy Models, and Hidden Markov Models.
In this method, having high volumes of quality annotated data is of key importance. While this method results in the most accurate classifications and all researchers try to bestow this type of learning on their models to compete in presenting the performant benchmark, the possibility of getting hold of this amount of expert-annotated data is slim in most domains. which leads to the practice of alternative learning methods.

Unsupervised Learning

Unsupervised learning is a type of task used for dimensionality reduction, anomaly detection, and categorization with unlabeled data. Unsupervised learning algorithms look for patterns in the data, as opposed to supervised learning, which compares the knowledge learned to a source of truth. This is particularly helpful

in many real-world scenarios where data labeling is frequently a time-consuming or just impractical task, and it also gives greater freedom by looking for previously undetected patterns.
These approaches usually take longer to achieve acceptable performance and frequently require a lot more training data. Unsupervised learning algorithms might also be more susceptible to inconsistencies in the data. For example, they can easily discard data that an expert will tag as essential and visa-versa. For NER tasks, this learning method is not very common, but using semantic relations present in the data is one possible approach.

Semi-supervised Learning

This method, also called "weakly supervised", is in between the two aforementioned categories. The learning process begins with a small amount of supervision, where algorithms only utilize data that is partially labeled; From these few samples, the model learns contextual clues that are then applied to the rest of the data in the next iteration. With many iterations, the model sees more and more examples to learn from. The small portion of data provided with a label, when used in conjunction with unlabeled data, allows for greatly improved learning accuracy of the models.

Reinforcement Learning

In the field of machine learning (ML), reinforcement learning focuses on how agents should behave in a given environment to maximize a potential reward. Dynamic programming approaches are used in many reinforcement learning systems. Numerous other fields, including operations research, information theory, simulation-based optimization, and statistics, are particularly interested in this area of study. The environment in machine learning is frequently depicted as a Markov decision process. In the NER tasks, as mentioned before, reinforcement learning is used in combination with CRF models both for entity recognition and for denoising data.

Distant-supervise Learning

The majority of machine learning methods need a collection of training data. Human labeling of a collection of documents is a common method for gathering training data (a.k.a. Expert Annotation). This path has the shortcomings of being time-consuming, expensive, and lacking experts for specific domains. A solution to this is distant supervision, whereby leveraging an existing database, or more specifically, a Knowledge Base(KB), we gather the required data for training with matching data with their corresponding token in the target KB. The significance of a KB is the record of relations between facts(data points). For example, Wikipedia

and YAGO2 are two major online open-source KBs that are usually easy and cheap to access. A complete description of knowledge bases is given in the next section.

The distant supervision, though it does not require large amounts of manual annotations, suffers from two major challenges. One is *incomplete annotations*, which is the result of the limited coverage of existing knowledge bases in different domains. The compromise between label accuracy and coverage leads to the necessity of strict matching rules. The second challenge is *noisy annotation*. As a result of the labeling ambiguity, the annotation is frequently noisy. Multiple entity types in the knowledge base can be linked to the same entity mention. Several studies have attempted to address the above challenges in distantly-supervised NER, which is discussed in Chapter 3.

2.3 Data Augmentation

Data Augmentation (DA) is a technique that can be used to artificially expand the size of a training set by creating modified data from an existing one or systematically generating it. It is closely related to oversampling in data analysis. DA helps to prevent overfitting or re-compensate for lack of data if the initial training dataset is too small, which is the case for NER. In short, data augmentation is good for enhancing the model's performance.
In computer vision, before training, there are no requirements and these transformations are done on the go using data generators. A batch of data is randomly transformed(augmented) as it is fed into the neural network. This is in contrast to NLP, where careful data augmentation is required prior to training due to the nature of human language not being logical. In the following section, a brief description of data augmentation approaches for NLP tasks is given according to [13]

2.3.1 DA Methods

- **Back Translation:** In this method, we translate the text data to some language and then translate it back to the original language. This can help to generate textual data with different words while preserving the context of the text data. Language translation APIs like Google Translate are used to perform the translation.

- **Easy Data Augmentation(EDA):** Easy data augmentation uses traditional and very simple data augmentation methods. EDA consists of four simple operations that do a surprisingly good job of preventing overfitting and helping train more robust models.

  - **Synonym Replacement:** Randomly choose n words from the sentence that are not stop words. Replace each of these words with one of its synonyms chosen at random.
  - **Random Insertion:** Find a random synonym of a random word in the sentence that is not a stop word. Insert that synonym into a random position in the sentence. Do this n times.
  - **Random Swap:** Randomly choose two words in the sentence and swap their positions. Do this n times.
  - **Random Deletion:** Randomly remove each word in the sentence with probability p.

- **NLP Albumentation:** This is more toward how an augmentation is applied while in transformation phase(similar to computer vision methods). It includes techniques like:

  - **Shuffle Sentences Transform:** In this transformation, if the given text sample contains multiple sentences these sentences are shuffled to create a new sample.
  - **Exclude duplicate transform:** n this transformation, if the given text sample contains multiple sentences with duplicate sentences, these duplicate sentences are removed to create a new sample.

- **Template-based Augmentation:** This is a novel technique based on synonyms replacement that utilizes an expert generated library to fill the target sequences with proper synonym permutations.

2.3.2 Gazetteers

Most recently proposed neural models for named entity recognition have been fully data-driven, with a major emphasis on eliminating the work required to gather outside resources or create custom features. Due to the models' limited ability to generalize beyond the annotated entities and their inability to access any supervision signal beyond the tiny quantity of annotated data, this could increase the likelihood of overfitting. The work of [14] demonstrate how effectively using external gazetteers could boost segmental neural NER models.
Gazetteers are collected dictionaries or lexicons consisting of long lists of entity names. Gazetteers may be used as an additional source of information to direct models toward broader coverage than just the annotated entities in NER datasets. They are frequently employed to take the form of whether the current token or current span is appearing in the gazetteer or not. There does not appear to be

any justification for a neural model not to use the commercially available gazetteers.

The most simplistic approach may be to treat each gazetteer item as a separate labeled training sentence, but given that there are frequently several gazetteer entity entries, this would induce a shift in the label distribution, and we found consistently lower performance in our initial studies. Therefore, using gazetteers in a distinct module instead of blindly using them as augmented data seems more straightforward. With this idea, recent research works are dervied to look for roust Gazetteer Generation methods and one of the most performant and validated approaches is to explore and extract entity names from open-source knowledge bases like Wikidata.
According to [15] experimental evidences two languages (English and Italian), show that extracting features from a rich model of the gazetteer and then concatenating such features with the input embeddings of a neural model is the best strategy in all experimental settings, and significantly outperform most conventional approaches.

2.4 Knowledge Graphs

In order to properly introduce Knowledge Graphs (KGs), which are the core novelty of this thesis, it is necessary to explain the role of semantic enrichment of data and how it is infused with KGs.

2.4.1 Semantic Web, RDFs, Graph Databases

The vision of the Semantic Web is to create the ability to query knowledge (a.k.a. meaningful information) on the Web in a systematically structured way. The traditional Web, which is a Web of Documents, should be transformed into a Web of Data, where any entity, such as a person or organization, and any relation between entities, such as a subset of, can be represented.

A general framework for representing related data on the web is the Resource Description Framework (RDF). Metadata is described and exchanged using RDF statements, allowing for a standardized interchange of data based on relationships. Multiple sources of data are combined using RDF. A website that showcases online catalog listings from a manufacturer and links products to product reviews on other websites and retailers selling the products is an example of this strategy. The RDF framework is the foundation of the semantic web, which organizes data based on meanings.

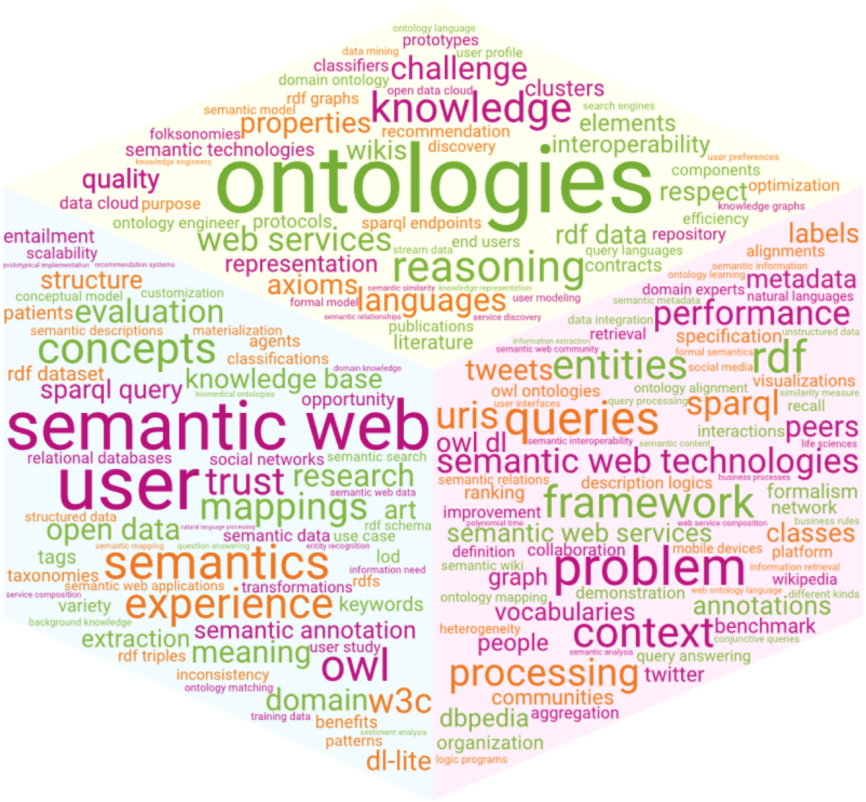

Figure 2.7: Major research topics utilizing semantic web and linked data [16]

Knowledge graphs (KGs) are a form of RDF that is currently regarded as one of the most crucial elements in the realization of the Semantic Web concept. A knowledge base (KB) is defined as the combination of an ontology and instances of the classes in the ontology, consisting primarily of facts about entities. In addition to domain-specific KGs, freely accessible KGs frequently cover generic, encyclopedic, or cross-domain information, as can be seen in DBpedia.
All these applications arise from the fact that a graph database is a form of relational data frame that incorporates nodes and edges to fully structure all the relations defined between each node type. Semantics is used in graph data models in the following ways: (i) graph nodes are interpreted as entities or values; (ii) typed relations between nodes are interpreted as facts about the involved entities; and (iii) a schema is introduced by giving instances a type and introducing relationships between classes. Therefore, the information focus enabled by the graph data model includes the schema, instances, and relationships. a visualisation of these concepts are displayed in Figure 2.8.

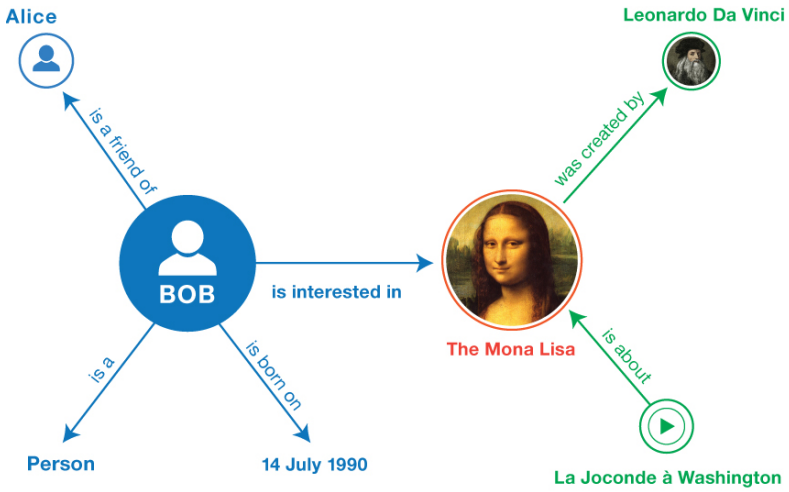

Figure 2.8: Basic KG Visualization [17]

2.4.2 Ontology

Ontology arises out of philosophy, which in philosophical studies is the study of being and existence. In information sciences and computer sciences, ontologies are frameworks for categorising, naming, and representing sets of concepts and their relations to one another. According to the definition, every academic discipline can have their own dedicated ontology to better organize data into information and knowledge. This is called a "Domain Ontology".
In knowledge graphs, ontologies are defined as a set of rules and templates applied to a set of facts to first build a knowledge graph and then provide the opportunity to enrich the proper nodes and edges with upcoming data. Ontologies are a component of the W3C standards stack for the Semantic Web and one of the fundamental elements of semantic technology. They offer users the organizational framework required to connect one piece of information to additional information on the Web of Linked Data. Ontologies facilitate database interoperability, cross-database search, and efficient knowledge management since they are used to establish common modeling representations of data from distributed and heterogeneous systems and databases.
One key benefit of ontologies is that they can be easily expanded since it is simple to add relationships and concept matching to already existing ontologies. As a result, if something goes wrong or needs to be altered, this model can change along with the expansion of data without having an effect on the processes and systems that depend on it. Additionally, ontologies offer the ability to represent all data types, including unstructured, semi-structured, or structured data, facilitating

quicker idea and text mining as well as data-driven analytics and more seamless data integration.

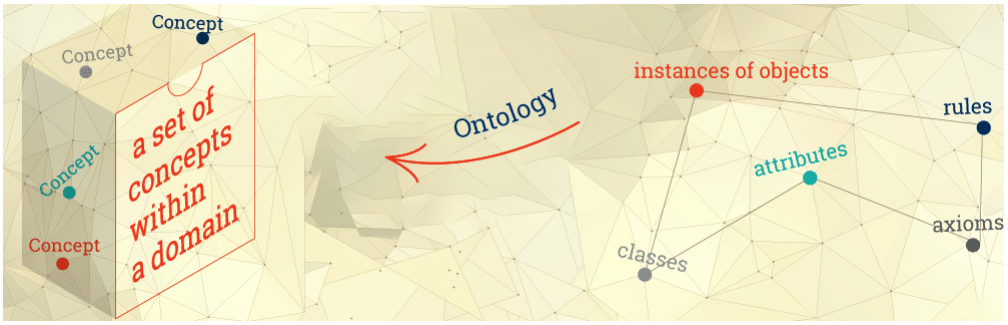

Figure 2.9: Visualization of Ontology Concept [18]

2.4.3 Knowledge Graphs

In this section, we will introduce the relevant theory behind KGs. First we define Nodes and Edges. Nodes are the main components of a graph that carry the pieces of information we refer to as "fact". The connections between these nodes are done by edges that encapsulate the relations between facts. Facts can be represented in the form of triplets in either of the two following ways:

- **HRT:** <head, relation, tail>
- **SPO:** <subject, predicate, object>

In Figure 2.8 we can see an example of a simple KG. Here the triplet `<The-Mona-Lisa, was-created-by, Leonardo-Da-Vinci>` is presented; that shows the relation connecting the two entities "The Mona Lisa" and "Leonardo Da Vinci" is "created by".

This type of representation is interestingly close to how the human mind works when categorising and memorising new information. Hence, this type of dataframe is very convenient to use and also creates a bridge between both human-understandable and machine-understandable data representations. These features drive knowledge graphs to be a reliable source of data. Some of the most famous open-source knowledge graphs are:

- **DBpedia:** is a community-based effort to extract structured content from the information present in various Wikimedia projects.
- **OpenCyc:** is one of the world's most complete general knowledge base and commonsense reasoning engines.

- **Wikidata:** is a free, collaborative, multilingual database, collecting structured data to provide support for Wikimedia projects.
- **YAGO:** is derived from Wikipedia, WordNet, and GeoNames.

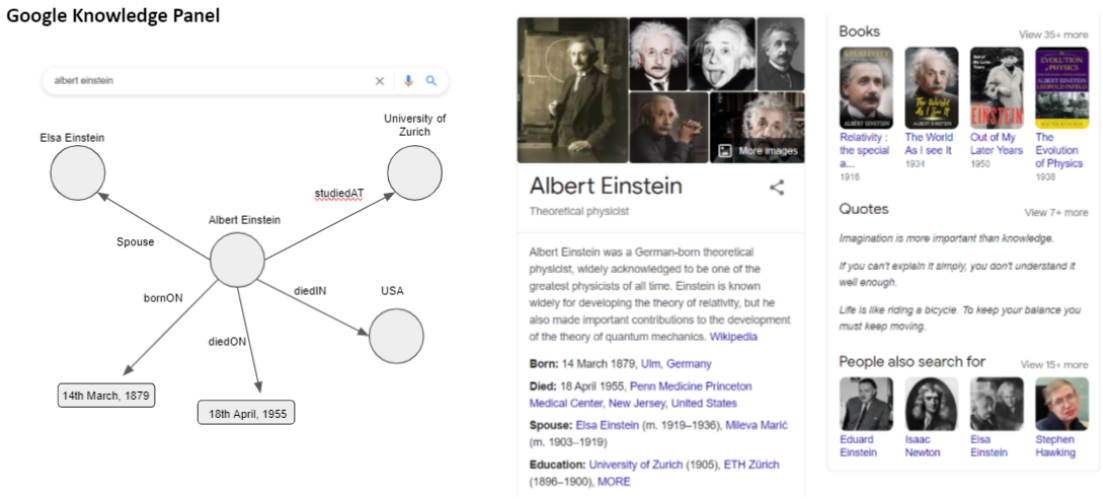

Figure 2.10: Open-source KGs example [17]

Wikidata

In this project, we will use Wikidata because of its unique capabilities for managing links and its native SPARQL query language, which uses the following for data extraction:

(i) **Properties**, which are special Wikidata entities that are used for describing relationships between entities and for assigning many types of data values to entities. (ii) **Classes** are Wikidata items that are used as the value in an `instance-of` statement, or that are subject or value in a subclass-of statement. which is found all in SQID's classes browser.(iii) **Link** is the gold relation that has been annotated(meaning the link to Qid which we know 100% is correct).

Graph Querying

Another robust advantage of KGs is the ability to query complex information. This feature, aside from being considerably faster than SQL, gives the possibility to traverse the graph and purpose any logical connections. Wikidata introduces SPARQL [19]. SPARQL is an RDF query language that is able to retrieve and manipulate data stored in RDF format.

The relation `instance-of` is represented as `P31` and the entity e.g. "house cat"

is represented as `Q146` since each entity has a unique UID. Lines 2 to 5 of the query simply state that we are looking for any entity that is an instance of a house cat, making it extremely easy to understand. Wikidata has information in many different languages, thus line 6 is required to filter outcomes for the English language. in 2.11 query are the results (entities with their UID and some basic information).

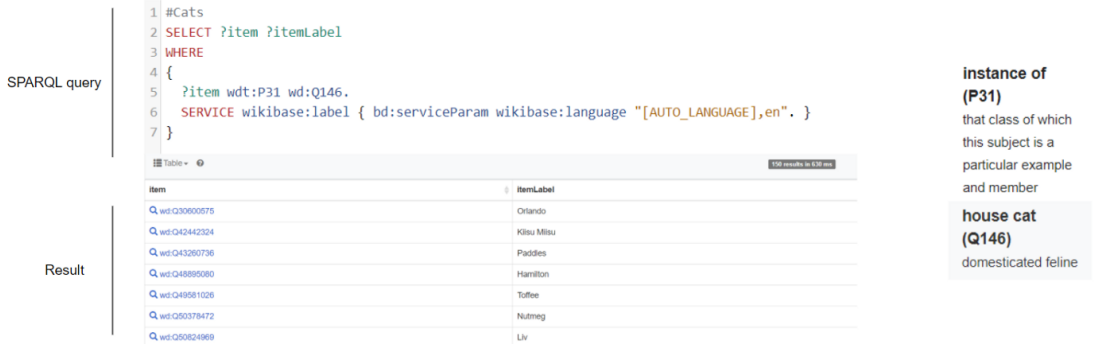

Figure 2.11: Wikidata SPARQL query example [17]

2.4.4 Entity Linking

As discussed before, any real-world object, such as persons, locations, organizations, etc., is a named entity. NER locates and groups instances of identified entities in text into predefined categories. Named Entity Linking(NEL) is the task of linking entities mentioned in text with their corresponding entities in a knowledge base [20].
The connected knowledge base depends on the domain and application, but we can use open-source knowledge bases discussed above. The challenge associated with this task is the name similarity and ambiguity problem. So, NEL is also known as Named Entity Disambiguation(NED).
Name variation means an entity can be mentioned in different ways. For example, the entity Michael Jeffrey Jordan can be referred to using numerous names, such as Michael Jordan, MJ, and Jordan. Whereas the ambiguity problem is related to the fact that a name may refer to different entities depending on the context. Here is an example of an ambiguity problem, the name Bulls can apply to more than one entity in Wikipedia, such as the NBA team Chicago Bulls, the football team Belfast Bulls, etc [21]. In general, a typical entity linking system consists of three modules, namely Candidate Entity Generation, Candidate Entity Ranking, and Unlinkable Mention Prediction [20]. A brief description of each module is given below.

- **Candidate Entity Generation:** In this module, the NEL system aims to retrieve a set of candidate entities by filtering out the irrelevant entities in the knowledge base. The retrieved set contains possible entities that may refer to an entity mention.
- **Candidate Entity Ranking:** Different kinds of evidence are leveraged to rank the candidate entities to find the most likely entity for the mention.
- **Unlinkable Mention Prediction:** This module will validate whether the top-ranked entity identified in the previous module is the target entity for the given mention. If not, then it will return NIL for the mention. Basically, this module is to deal with unlinkable mentions.

Besides information extraction, NEL can be used for a variety of purposes. It is utilized in a variety of systems, including recommender systems, intelligent tagging, question answering systems, and information retrieval. The application we are interested in is the process of connecting entities to Wikipedia, which is known as **Wikification**.

2.5 Neural Networks

Artificial neural networks (ANNs), also referred to as neural networks (NNs), are advanced computing models based on the perceptron's original concept. ANNs are composed of a number of connected units, known as nodes or artificial neurons, that simulate the function of neurons in a biological brain. Feed-forward Neural Networks (FFNNs), in which connections and neurons effectively form a directed acyclic graph and nodes of a layer are connected only to nodes of the next layer, and recurrent neural networks (RNNs), in which connections between nodes of the same or previous layers are allowed, creating loops that can feed information back into the network, are the two main categories of ANNs.

2.5.1 Perceptron

Perceptrons are the basic classifier units that artificial neurons use in order to explain inputs. The fundamental structure is to build a weighted linear combination of inputs and pass them through the activation function. Figure[perceptron] shows the basic structure of a perceptron. The objective is to develop a threshold that is tolerant of slight fluctuations while allowing the signal—the important portion of the input—to get through it and not the noise. As a non-linear function with the additional task of normalizing the output, the activation function's selection has a significant impact on how well the task performs.

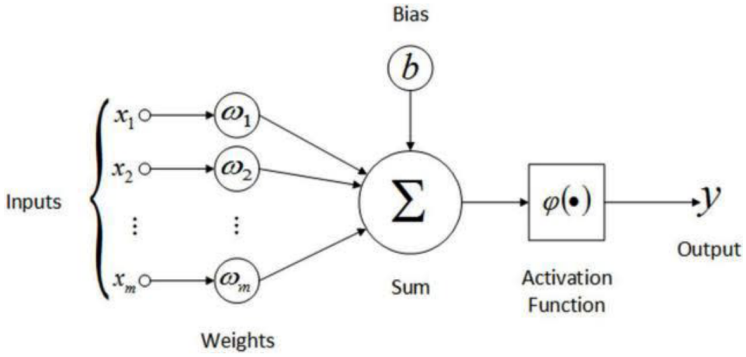

Figure 2.12: Perceptron schema [2]

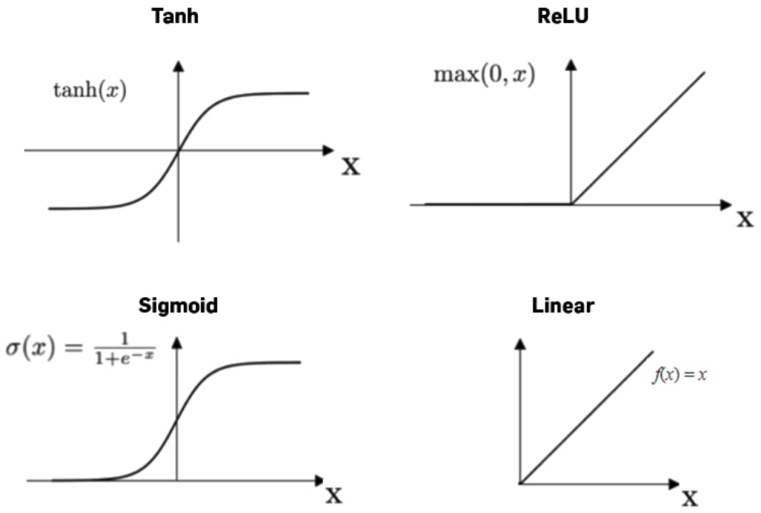

Figure 2.13: Activation function [2]

2.5.2 MultiLayer Perceptron

The multilayer perceptron is the most well-known and supportive structure after the basic element; it is frequently employed in classification frameworks where data is not linearly separable. This structure is made up of layers, which are collections of neurons.
These neurons are often completely linked, meaning that each output or hidden unit accepts as input all of the outputs from the units at the layer above. Depending on the task and level of complexity, a single structure may have one or more hidden layers in addition to an input layer, which accepts input and has a number of

neurons equal to the number of dataset features, a classification layer, which is the output layer and has a number of neurons equal to the number of classes, and one or more hidden layers.
Depending on the task and the number of hidden layers, we can categorize them into *Shallow architecture* and *Deep architecture*. Where the former consists of feature extractors and a trainable classifier, which usually embeds a generic ML algorithm, and the latter, exists in a number of hidden layers, with the capacity to learn features through filters at various levels of semantic abstraction based on layer level, structuring in such a way as to create a feature hierarchy.
The number of steps in the MLP's or NN's overall training phase, known as epochs, is a user-defined hyperparameter that depends on the goal and the results. Each phase includes the creation of the output as well as a method for the network to learn, which is accomplished through the use of loss computing, which will be explained in detail in the next section.

2.5.3 Loss Function

In order to increase the accuracy of the results provided by the network, learning entails modifying its weights and biases. This is done by minimizing errors up until a point where the network can no longer be improved by lowering its error rate. The learning process continues as long as the cost function's output keeps decreasing, which is accomplished by first constructing a cost function that is frequently reviewed during learning. The cost is frequently defined as a statistic, and some examples include the Mean Squared Error (MSE), Binary Cross Entropy (BCE), etc. The choice of the loss function needs to happen on a per-case basis, according to the task that has to be learned.
The MSE is a measure for computing the distance between two quantities, which in learning is the distance of predicted values from the ground truth.

$$MSE_i = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i; W))^2 \tag{2.5}$$

where $f(x_i; W)$ represents the values predicted by the network with weights W taking as input data x_i, and y_i indicates the ground truth, or label.

The Cross Entropy loss function, also called Binary Cross Entropy in binary classification tasks, is a measure of the difference between two distributions.

$$BCE_i = -\frac{1}{n}\sum_{i=1}^{n} y_i \log\left(f(x_i; W) + (1 - y_i)\log\left(1 - f(x_i; W)\right)\right) \tag{2.6}$$

Once a loss function $\mathcal{L}$ is defined, the goal of the learning task is to find the set of parameters, weights W^* and biases b^*, that minimize the function:

$$W^* = \arg\min_W \mathcal{L}(W) \tag{2.7}$$

2.5.4 Back-Propagation

It is a chain rule-based process to compute the gradients of the loss with respect to parameters in a multi-layer network in order to penalize the NN's behavior, since the network learns if it minimizes the loss(plus some regularization terms) with respect to parameters over the training set. A common method for minimizing the loss function is called gradient descent. To better understand the gradient descent algorithm, consider the following hypothetical representation of a simple loss function $\mathcal{L}(w_0, w_1)$, where each pair (w_0, w_1) is associated to a value of the loss function.

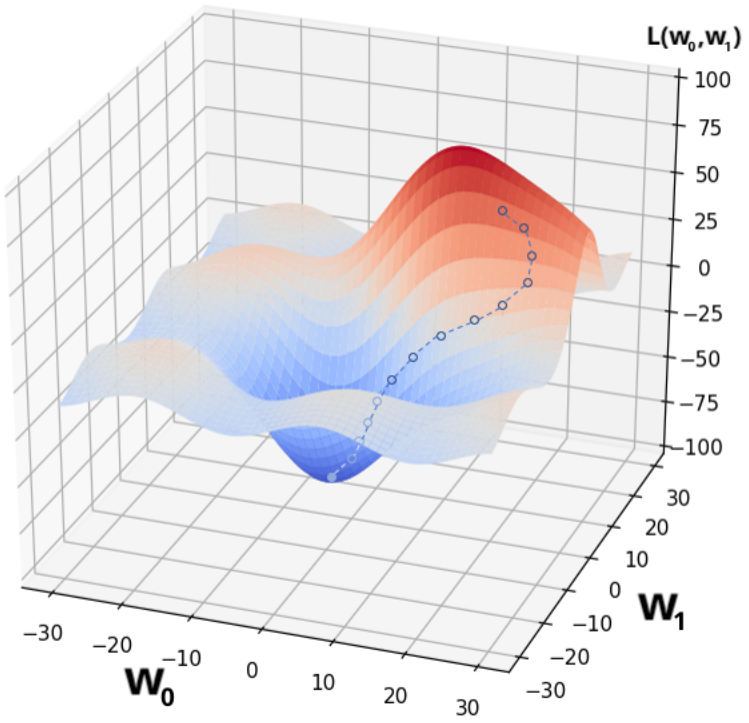

Figure 2.14: Example of gradient descent

The theory deriving the algorithm states, starting from any point in the loss function space, one could follow the direction that allows to locally minimizes the function $\mathcal{L}(w_0, w_1)$ by following the direction of the most rapidly decreasing gradient, $-\nabla\mathcal{L}$. Although gradient descent is a pretty simple optimization technique, it functions similarly to other, more complicated ones in theory. There are

other methods that take into account the possibility of getting stuck in these local minima, such as stochastic gradient descent, adam, and adagrad. The gradient descent algorithm is not particularly efficient in avoiding local minima, which would cause the learning process to stagnate and stop improving the model.

Consider a simple MultiLayer Perceptron with a single hidden layer h, having as input x and output $\hat{y}$, with weights w_0 between the input layer x and hidden layer h, and weights w_1 between the hidden layer h and the output layer $\hat{y}$. Since improving the network is effectively achieved by updating the weights, it's necessary to compute the gradient with respect to each set of weights. While the gradient with respect to w_1 can be directly calculated as:

$$\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_1} \tag{2.8}$$

Here the impact of chain rule can be observed, as it allows us to express the gradient in terms of w_0 as:

$$\frac{\partial \mathcal{L}}{\partial w_0} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_0} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h} \frac{\partial h}{w_0} \tag{2.9}$$

In this way, it's possible to calculate all gradients, from output to input layers.

2.5.5 Learning Rate

The learning rate is a significant hyperparameter employed throughout the learning phase. The amount of the steps the model takes to account for errors is determined by the learning rate. One could imagine the learning rate as the distance between two updates of the loss function, using the gradient descent algorithm previously described as an example.
A lower learning rate results in a longer training period but the possibility for greater accuracy, while a higher learning rate reduces the training period but results in a lower final accuracy. In order to start the learning process by making large corrections, schedulers that are adaptively change the value of the learning rate are frequently used.
Once the model has reached a point where its accuracy is stagnating, the schedulers gradually reduce the learning rate in order to increase accuracy. In order to retain a reliance from earlier updates, some learning rate schedulers also use the idea of momentum, a hyperparameter that enables balancing the weight update by taking into account both the current and prior gradients.

2.6 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of FFNNs usually utilized in NLP and for computer vision purposes. Since the connectivity network between neurons mirrors how the neurons are arranged in a person's visual cortex, CNNs were inspired by biological processes. In the following points, we will focus on CNN architecture:

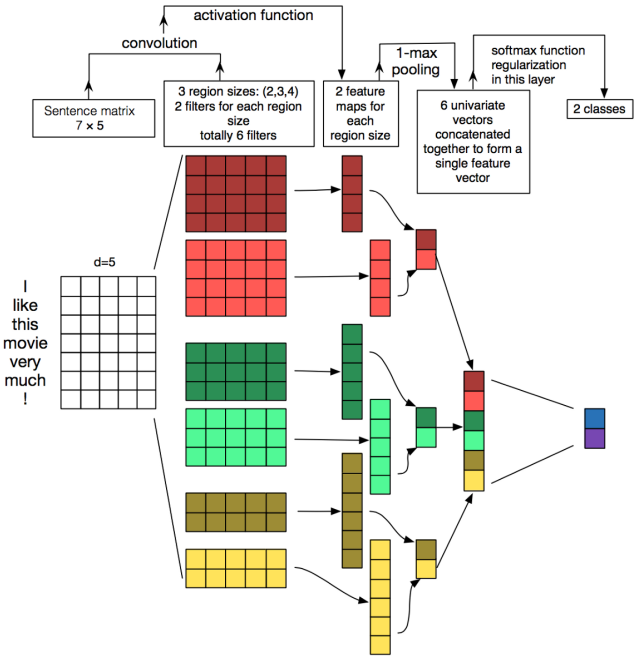

Figure 2.15: Example of CNN schema

- **Convolutional layers:** Considering the large number of parameters analyzed in a deep learning network, CNNs have convolutional layers, where each neuron is responsible for a particular sub-window of data that "strides" through the set. A dot-product is then performed between the extracted sub-window and a matrix known as the kernel or filter, yielding a value that will make up the feature map. The layer then applies the same activation pathway as a fully connected layer.
- **Pooling layer:** A pooling layer may be added after the convolutional output has been generated. This layer's job is to divide the output into additional sub-windows and then collapse each of them into a single data point with a particular function (often average, sum, or max), improving the network.

- **Normalization Layer:** In order to avoid a reduction in the speed and accuracy of the model, a normalization layer is added for re-scaling tensors.
- **Non-linearity functions:** In convolutional layers, the existence of non-linear function, such as *ReLU* or *Leaky ReLU* are to address issues including output saturation, vanishing gradient, and collapsing of fully connected layers. For example, leaky ReLU utilizes the following in the network:

$$f(\mathbf{x}) = \begin{cases} 0.01\mathbf{x} & \text{if } \mathbf{x} < 0, \\ \mathbf{x} & \text{if } \mathbf{x} \geq 0 \end{cases} \tag{2.10}$$

- **Softmax layer:** The softmax function is used as the activation function in the output layer of CNNs that predict a multinomial probability distribution. The Softmax layer is used as the activation function for multi-class classification problems where class membership is required on more than two class labels.

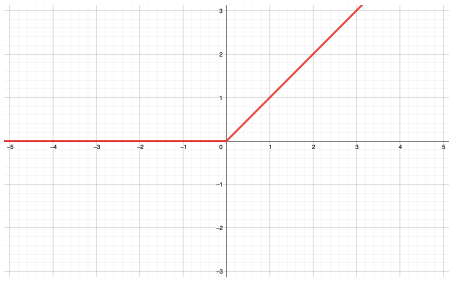

Figure 2.16: ReLU

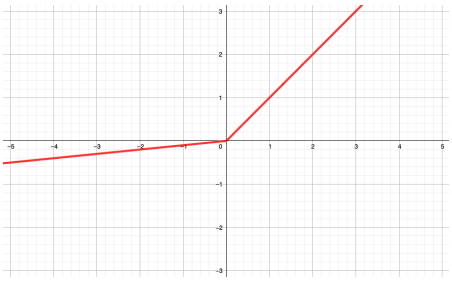

Figure 2.17: Leaky ReLU

2.7 Transformers

According to the issue statement, a transformer model essentially assists in changing one sequence of input into another. With the aid of a combined encoder and decoder model, this transformation can take the shape of a translation system (from one language to another) or an answer (output sequence) to a question (input sequence), among other things.
Research has long used complex and deep NN models, particularly convolutional and recurrent neural networks, for the mentioned tasks. The number of operations required to relate signals from two positions in the input sequence rises as the distance between positions decreases, making it harder for complex nets to learn the dependencies between elements, despite the fact that they have been used as fundamental building blocks for state-of-the-art networks. Since then, the state-of-the-art has advanced as a result of the use of encoder-decoder models.

An *encoder* is a network (FC,CNN, RNN, or other) that takes the input and generates a feature vector as an output; this feature vector holds the information that represents the input. On the other hand, the decoder is again a network (usually with the same network structure as the encoder but in the opposite orientation) that takes the feature vector and gives the closest match to the actual input or intended output. The *Transfomer* first introduced by [6] is an encoder-decoder model based on the attention mechanism, especially the *self-attention* which is a component relating different positions of a single sequence in order to compute a representation of the sequence.
The **encoder** maps the input sequence to a representation sequence and is made up of 6 identical layers, each of which is further divided into two smaller layers: the first is a *multi-head attention layer*, and the second is a *feed-forward fully connected layer*. Each sub-layer receives a layer normalization and residual connections (i.e., blocks that enable data to reach later layers of the neural network by skipping some layers).
The **decoder** also has 6 layers, but each layer in this case is made up of 3 sub-layers: the first is a multi-head attention layer, the last is a feed-forward network, and the second in-between layer is another attention layer that receives input from both the encoder output and the true output (offset by one position). Additionally, it is altered with a masking strategy to stop positions i from attending to subsequent positions.
This method makes sure that the forecast for a point only relies on the data that is known for positions below i. There are residual connections and normalizing layers available for all three sub-layers.

Multi-head attention layer

This block is the fundamental contribution of the Transformers. After extraction of input embedding components and passing them to output, The *attention function* utilizes a query and a key-value pair mapped into the previous output. [Figure] displays a quick summary of the structure. The weights in the object are computed by a measure of compatibility between the key associated with the value and the query. This mapping procedure is carried out by computing a weighted sum of the values.

In short, a dot-product between queries and keys is performed, with a normalization term that depends on the key vector dimension. The output of the softmax function, which receives the product of this multiplication, is the weight associated with the values. To prevent a situation where there is no convergence, an additional scale factor is applied. The following softmax function may push the multiplication into region with small gradient values, which may provoke a slow convergence or

no convergence at all.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.11}$$

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{2.12}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.13}$$

where $\sqrt{d_k}$ is the dimension of the key vector k and query vector q. Q, K and V are the matrices related respectively to queries, keys, and values; the first two have d_k dimensions while the latter has d_v dimensions. These matrices are extracted from the input dimensional embedding through a linear projection layer with proper weight matrices. This process is repeated in the multi-head layer where The heads enable the model to jointly gather data from various representation sub-spaces located at various positions. After being concatenated, the outputs of the several heads—whose initial work's h value is 8—are once more projected in N-dimensional model space as mentioned in the formula.

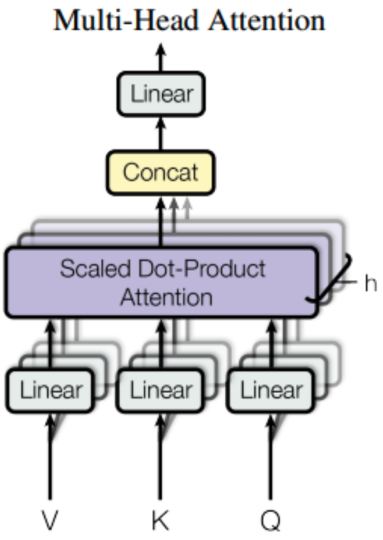

Figure 2.18: Multi-head attention layer structures [6]

After the conversion of sequences to N-dimensional vectors, a linear transformation and a softmax are in order. This is to obtain a probability distribution about the word predictions. In addition, There is a positional encoding layer that adds its output to the input embedding at the beginning of both the encoder and the decoder. This additional procedure is used to inject information about the relative

and absolute positions of the tokens. Both forms of locations are encoded by using sinusoidal functions to calculate the value for the i-th dimension, which depends on both the position of the token and the element's dimension in the positional array:

$$PE_{pos,2i} = Sin(\frac{pos}{10000^{2i/d_{model}}})PE_{pos,2i+1} = Cos(\frac{pos}{10000^{2i/d_{model}}}) \tag{2.14}$$

As it can be seen from the formulas, in essence, two tokens close together in a sentence will have similar values when the sinusoidal frequency is low and when it is high, but their difference in position will be noticeable at very higher values. In contrast, two tokens far apart are more likely to have different positional encoding values at much lower frequencies. This method involves computing each element of the positional encoding array for each token before adding it to the appropriate embedding vector.

2.7.1 BERT

After the rise of Transformers, they were introduced to different NLP research scopes and one of their biggest impacts was on language models. These include sentence-level tasks such as natural language inference and paraphrasing, which aim to predict the relationships between sentences by analyzing them holistically, as well as token-level tasks such as named entity recognition and question answering,where models are required to produce fine-grained output at the token level.
There are two existing strategies for applying pre-trained language representations to down-stream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo, uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer, introduces minimal task-specific parameters and is trained on downstream tasks by simply fine-tuning all pre-trained parameters.
The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations [7].

Bidirectional **E**ncoder **R**epresentations from **T**ransformers(BERT) proposes methods to improve the fine-tuning approach. BERT uses a MLM pre-training objective. The MLM masks some input tokens at random, and the goal is to predict the masked word's original vocabulary id based solely on its context. The MLM objective, in contrast to left-to-right language model pre-training, enables the representation to integrate the left and right context, allowing us to pre-train a deep bidirectional Transformer. In addition to the masked language model.
Respecting the original paper [7], BERT's model architecture is a multi-layer bidirectional transformer en-coder based on the original implementation described in

[6] and released in the `tensor2tensor` library. It is worth mentioning that the use of Transformers has become common and our implementation is almost identical to the original. The number of layers(i.e., transformer blocks) denoted as **L**, the hidden size as **H**, and the number of self-attention heads as **A**. Considering the two model sizes: BASE (L = 12, H = 768, A = 12, Total Parameters = 110M) and LARGE (L = 24, H = 1024, A = 16, Total Parameters = 340M). BASE was chosen to have the same model size as common benchmarks for comparison purposes.

To make BERT handle a variety of downstream tasks, its input representation is able to unambiguously represent both a single sentence and a pair of sentences(e.g. Question and Answer) in one token sequence. Throughout this work, a "sentence" can be an arbitrary span of contiguous text rather than an actual linguistic sentence. A "sequence" refers to the input token sequence to BERT, which may be a single sentence or two sentences packed together. A 30,000-token vocabulary of WordPiece embeddings is used for this purpose.
Each sequence's first token is always a classification token `[CLS]`. The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks. Sentence pairs are packed together into a single sequence. The sentences are differentiated in two ways. First, divide them with a special token `[SEP]`. Second, add a learned embed-ding to every token indicating whether it belongs to sentenceA or sentenceB. As shown in [**Figure**],The input embedding is denoted as E, the final hidden vector of the special `[CLS]` token as C,and the final hidden vector for the input token as T. For a given token, its input representation is constructed by summing the corresponding token, segment, and position embeddings.

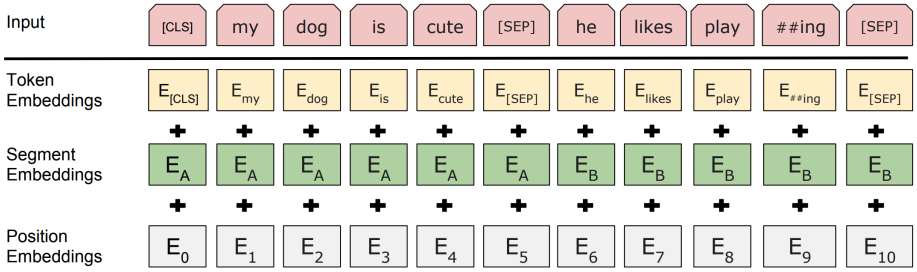

Figure 2.19: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings[7]

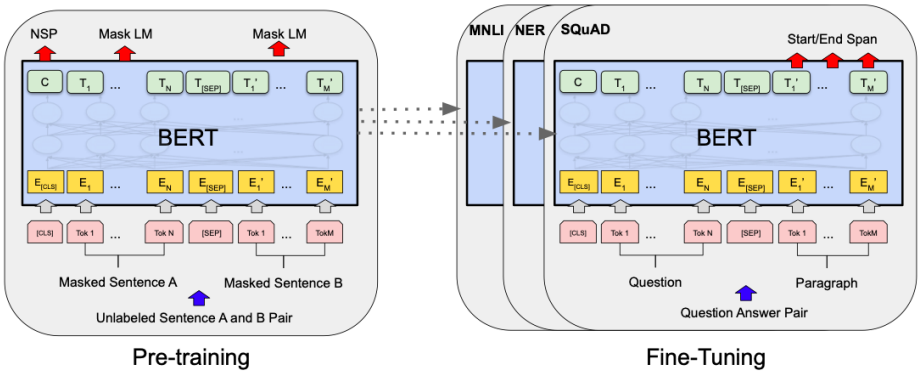

Figure 2.20: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings[7]

2.7.2 Training a BERT architecture

Pre-training is where BERT learns the "linguistic model" of a given language, and fine-tuning is all the dedicated work, including Transfer Learning, to train the model for any particular task (such as QA, classification, or any other task). Pre-training is therefore a once-in-a-lifetime requirement. The same model can also be adjusted based on other problems. The model doesn't need to learn every feature from scratch for every problem because it has previously been trained; instead, it only needs to learn a few new ones that are appropriate for the problem, saving hardware resources and latency.

The Masked Language Model (MLM) and Next Sentence Prediction (NSP) jointly pre-train the text-pair representations of BERT. More thoroughly, considering the bidirectional aspect of the model, prediction of the target word is done in a multi-layered context. So to train, we simply mask some percentage of the input tokens at random, and then predict those masked tokens.
In this instance, the output softmax over the vocabulary is fed the final hidden vectors corresponding to the mask tokens, as in a typical language model. In BERT's original pre-training, The training data generator chooses 15% of the token positions at random for prediction. If the i-th token is chosen, we replace the i-th token with (1) the `[MASK]` token 80% of the time (2) a random token 10% of the time (3) the unchanged i-th token 10% of the time. Then, T_i will be used to predict the original token with cross entropy loss. This procedure is called masked language modelling.
Understanding the relationship between tokens is important for next steps and applications. In order to train a model that understands sentence relationships, BERT is pre-train for a binarized NSP task that can be trivially generated from

any monolingual corpus [7]. Specifically, when choosing the sentences A and B for each pre-training example, 50% of the time B is the actual next sentence that follows A(labeled as `IsNext`), and 50% of the time it is a random sentence from the corpus (labeled as `NotNext`). As shown in Figure 2.20, C is used for next sentence prediction. The NSP task is closely related to representation-learning. However, in prior work, only sentence embeddings are transferred to downstream tasks, where BERT transfers all parameters to initialize end-task model parameters.

This effort is easier since the Transformer acts in encoding a concatenated text pair with self-attention effectively includes bidirectional cross-attention between two sentences, which prepares BERT for downstream tasks. With this, it's just needed to insert the job-specific inputs and outputs into BERT and fine-tune all the parameters end-to-end for each task.

Chapter 3

State of the Art analysis

In this chapter we will investigate the current State-of-the-Art (SotA) solutions to the NER problem, focusing on neural network-based and BERT-based methods which currently offer the best performance thanks to their computational power and complexity. Deep Learning architectures allow to create extremely complex models that, through a learning procedure, are able to produce flexible results compared to the traditional rule-based techniques showcased in the previous chapter.

3.1 Solutions to the lack of data problem

Referencing to problem statement in Chapter 1. Since Data Annotation is an expensive task, adopting a data re-compensation technique is the way to go. In this section we analyze the SotA methods like Data Augmentation which effectively improve the model's performance.

3.1.1 Gazetteers

One of the most gainful techniques is utilizing Gazetteers. Gazetteers are built upon a large list of entries and boost finding matches in a document; which in turn improves the NER. Building a data-driven gazetteer also requires large amounts of labeled data, and in this case, having a structured and carefully labeled dataset is of even greater importance. The generation and employment of gazetteers in NER was first done by Riloff *et al.* [22]. They utilized mutual bootstrapping to build a small set of entities using a small number of lexical patterns.
Furthermore, Lin *et al.* [23] demonstrate how to use an unsupervised technique to construct massive clusters of semantically linked terms. They came up with the notion after looking at words that had comparable syntactic dependency ties. Their

method, however, does not identify the semantic class labels, which is a common drawback of clustering algorithms.
Following this, Etzioni *et al.* [24] developed an algorithm that outperforms all prior approaches for the task of automatically generating a large list for a specific type of entity or semantic class, which will be the task of automatic automatic gazetteer generation. This idea was further exploited by Nadeau *et al.* [25] where their unsupervised method was built on past work by Collins and Singer and Etzioni *et al.* [24]. The effectiveness of gazetteers in NER has long been recognized. Building and maintaining high-quality gazetteers, on the other hand, requires time.
Many ways of overcoming this challenge by automatically building gazetteers from large amounts of text have been presented (Etzioni *et al.* [24]; Nadeau *et al.* [25]; Toral and Monzu [26]). However, to extract high-quality gazetteers, these methods necessitate sophisticated pattern-induction or statistical methods; Which in view of modern Neural Networks is considered inefficient.

3.1.2 Knowledge Bases and Entity Linking

The growth of open knowledge-base DBs in the past few years makes them a suitable choice for building a gazetteer upon them. The idea of using Wikipedia's named entities for this goal was first exploited by Kazama *et al.* [27]. They were motivated by the fact that:

> *Since Wikipedia aims to be an community-driven encyclopedia, most articles are about named entities, and they are more structured than raw texts. So, extracting knowledge such as gazetteers from Wikipedia will be much easier than from raw texts or from usual Web texts.* [27]

An approach proposed by Radford *et al.* [28] was to use document-specific KB tags, a series of key words from a specific document domain, and created a KB out of it. This helped in improving the domain-specific approach by lowering the training time. However, they just collect aliases with the KB, where it could be used to also harvest related entities for better disambiguation.
Therefore, we have explored this area by carefully analysing the recent findings in both Ontology Mapping and Graph Enrichment [29]. The outcome shows this deficiency in literature can be addressed by linking the named entities to structured databases with clean semantics. And by taking a closer look into available data, it's noticeable that DBpedia and Wikidata Ontologies offer a competitive advantage w.r.t. other DBs. This arises from the hand-generated mappings of Wikipedia info-boxes since 2008, where it soon evolved into a successful shallow cross-domain ontology where classes have multiple superclasses [30]. Radford *et al.* [28] and Delpeuch [31] suggested promising leads in Gazetteer Generation and the EL method, respectively.

Furthermore, SotA introduces Distant-Supervised Learning (DSL). DSL is a new learning approach that correctly takes advantage of KBs.

3.1.3 Distant-Supervision

To overcome the lack of labeled data, weak or distant supervision methods have become popular,which automatically annotate unlabeled, raw text. Even in low-resource settings, unlabeled text is often available, and research has shown that automatically annotated labels can be a useful training resource in the absence of expensive, high-quality labels. For NER, a widespread approach is to use lists, dictionaries, or gazetteers of named entities (e.g. a list of person names or cities). Each word in the corpus is assigned the corresponding named entity label if it appears in this list of entities.
While distant supervision performs very well on high-resource languages, it has been shown to be more difficult to leverage in real low-resource settings due to the lack of external information. Additionally, several difficulties arise when applying it in a practical way, such as obtaining these dictionaries(e.g. a list of city names in Yoruba) or adapting the matching procedure to the specific language and domain (e.g. deciding for or against lemmatization and, thus, trading off recall and precision). Distant supervision can only be beneficial and save resources if it is easy to use and fast to deploy [32].
In the area of information extraction, the tools by Dalvi et al.[33] allow the user to create rules or patterns, e.g. "[Material] conducts [Energy]".They can, however, require a large amount of manual rule creation effort to obtain good coverage for NER. NER is closely related to entity linking. Zhang et al.[34] presented a system to link entities in many languages automatically but focus on disaster monitoring and,therefore, only consider persons, geopolitical entities, organizations, and locations. This is the problem with many SotA fine-grained NER models which are trained and presented with specific goal and domain.

Another way to approach DSL is a method that pairs a knowledge graph (a graph of entities connected by edges labeled with relation classes) with an unstructured corpus to generate labeled data automatically. This method was first exploited by Mintz *et al* [35] and the recent work of Hogan *et al* [36] showed its significance in Relation Extraction. They propose Fine-grained Contrastive Learning (FineCL). Contrastive learning is a technique to train a model to learn representations of sentences such that similar samples are close in the vector space, while dissimilar ones are far apart.
Figure 3.1 illustrates the end-to-end data flow for the FineCL method. First, automatically labeled relation data is used to train an off-the-shelf language model (e.g., RoBERTa, or another domain-specific BERT relative) via cross-entropy. During

training, the learning order of relation instances is recorded. A training instance is considered "learned" when the model first correctly predicts the corresponding instance. They show that the order of learned instances corresponds to label accuracy: clean, accurately labeled relation instances are, on average, learned first, followed by noisy, inaccurately labeled relation instances. They call this "fine-grained" contrastive learning since it leverages additional, fine-grained information about which instances are and are not noisy to produce higher-quality relationship representations.The representations learned during pre-training are then used to fine-tune the model on gold-labeled data.

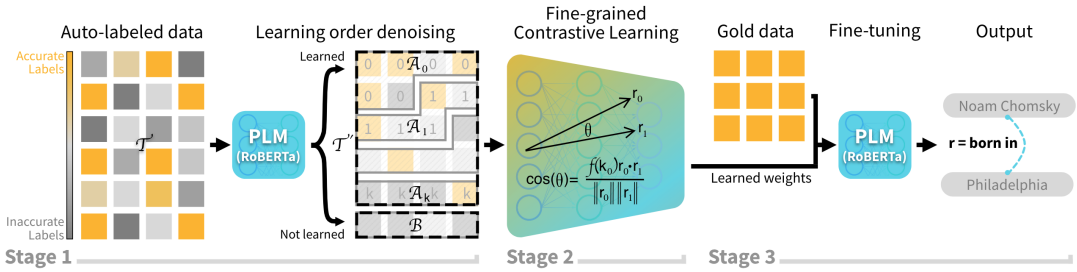

Figure 3.1: The FineCL framework [36]. distantly supervised data isused to train a PLM via cross-entropy to collect ordered subsets of learned and not learned instances over k epochs. Stage 2: function $f(k)$ weighs relation instances relative to their learning order in a contrastive learning pre-training objective that uses cosine similarity to align similar relations. Stage 3: the model is adapted to a discriminative task.

However, since not all phrases will reflect a relationship, employing this method to automatically label data produces a noisy training signal in the form of false positives. Therefore, a significant percentage of the effort that uses distant supervision focuses on creating cutting-edge denoising techniques. This invites us to use the structured characteristics of knowledge bases and ontologies to use the expert driven relationships to better select and extract data. Therefore, it is evident these approaches still do not answer the gap in the literature regarding the use of ontologies and their semantics as a reliable source of structured data for improving Fine-grained NER.

3.1.4 Ontology in NER

The most common knowledge management tool is the search engine which includes a broad range from web search engine to native query systems. having knowledge

retrieval as its core motive. The most effective and popular tool for representing knowledge is a graph. Although knowledge extraction is still the main issue, it should be specifically built and trained for various types and formats of data sources. Ontology is an abstract knowledge modeling, which treats the knowledge as concepts, associated attributes, and relations. Ontology can be divided into four categories: application ontologies; domain ontologies; generic ontologies and representation ontologies [37].
According to the survey by [38] "The knowledge acquisition can be automatically or semi-automatically conducted using the reasoning mechanisms involved in the application ontologies. Domain ontologies focus on a specific domain for conceptualizations. In this situation, the primary task is to eradicate the misperception among the concepts". They study the use of ontologies to to extract the knowledge embedded in the paper abstracts from four aspects (background, objective, solutions, and findings) and their relations.

Janowic *et al* [39] argue that gazetteers can benefit from an ontological approach to typing schemes, providing a formalization. These will better support gazetteer applications, maintenance, interoperability, and semi-automatic feature annotation and also discuss the process of developing such an ontology as a modification of an existing feature type thesaurus; the difficulties in mapping from thesauri to ontologies are described in detail. To demonstrate the benefits of a categorization based on ontologies.
In the recent work of Wang *et al* [40] leverages the chemistry type ontology structure to generate distant labels with novel methods of flexible KB-matching and ontology-guided multi-type disambiguation. It significantly improves the distant label generation for the subsequent sequence labeling model training. Experimental results show that CHEMNER is highly effective, outperforming substantially the state-of-the-art NER methods.

3.1.5 Data Augmentation

The term "Data Augmentation" describes techniques used to expand the amount of data by adding copies of current data that have been significantly updated or by generating brand-new synthetic data from existing data. Deep learning has recently attracted a lot of interest and demand since these methods address situations when deep learning techniques may not perform well due to data shortage.
One of the main focuses of the DA methods is to improve the diversity of training data, thereby helping the model to better generalize to unseen testing data. Large numbers of DA methods have been proposed recently and different analysis and surveys are dividing the categories w.r.t methods(e.g *back-translation and model-based techniques*). Among all DA methods in NLP we are interested in the ones

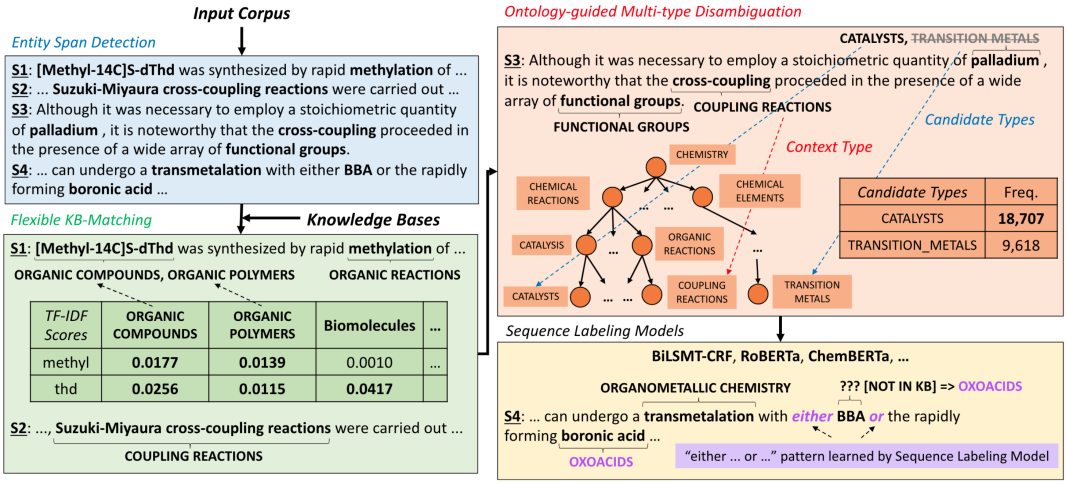

Figure 3.2: The overall framework of CHEMNER [40]. It includes a distant label generation (entity span detection, flexible KB-matching, and ontology-guided multi-type disambiguation) and a sequence labeling model training.

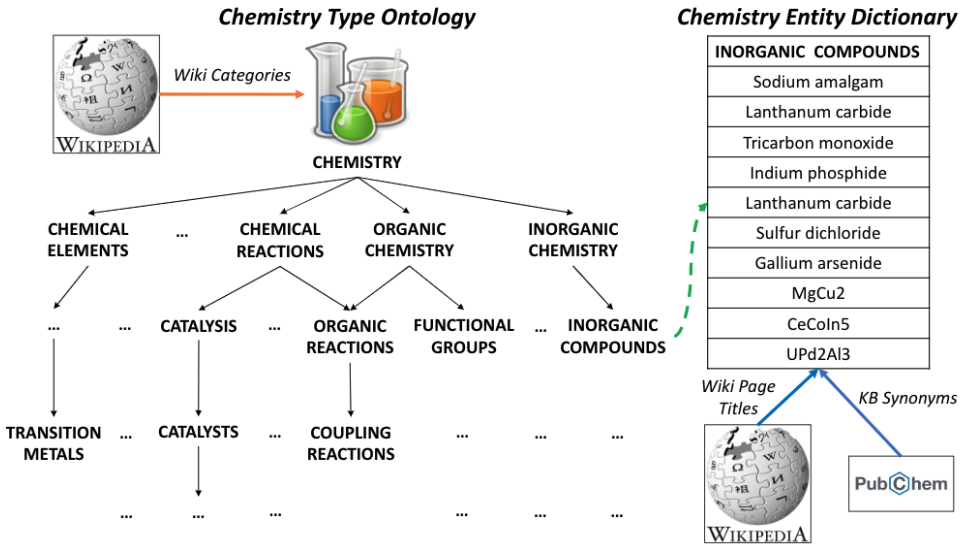

Figure 3.3: CHEMNER [40]: Illustration of the chemistry type ontology construction and dictionary collection.

targeted at text classification and NER tasks.

Different from these sentence-level classification tasks, NER does predictions on the token level. That is, for each token in the sentence, NER models predict a label indicating whether the token belongs to a mention and which entity type the mention has. Therefore, applying transformations to tokens may also change their labels. Due to this difficulty, data augmentation for NER is comparatively less studied. Inspired by sentence-level and sentence-pair DA techniques, Dai *et al.* [41] show that simple augmentation can boost performance for both recurrent and transformer-based models, especially for small training sets.
By investigating DA methods like Label-wise token replacement, Synonym replacement, Mention replacement, and Shuffle within segments , their results leads to three notable conclusions. First, all DA methods can improve the baseline regardless of model being recurrent or transformer. Second, applying all DA methods together outperforms any single DA on average. Although this combination of DA methods may reflect a trade-off between diversity and validity of augmented instances, applying all DA together prevents overfitting via producing diverse training instances. And last, considering the significant improvements when using pre-trained transformer models, it is important to investigate the effectiveness of techniques also on pre-trained models, such as BERT, considering they are supposed to capture various knowledge via self-supervision learning.

Chen *et al.* [19], investigated the possibility of leveraging data from high-resource domains by projecting it into the low-resource domains. Taking into consideration the text in the newswire domain is long and formal while the text in the social media domain is short and noisy, often presenting many grammar errors,spelling mistakes, and language variations. they proceeded with the hypothesize that even though the textual patterns are different across domains, the semantics of text are still transferable.
Additionally, there are some invariables in the way the named entities appear which the model can learn from them. Their novel neural architecture [19] introduce a cross-domain auto-encoder model capable of extracting the textual patterns in different domains and learning a shared feature space where domains are aligned. Although their work is more dedicated to investigate domain similarities and cross-domain performance improvements(Newswire[NW] to Social Media[SM], the results show by training the model on reconstruction loss, it can extract the textual patterns in each domain and learn a feature space where both domains are aligned. The effectiveness of their proposed method by evaluating a model trained on the augmented data for NER, concluding that transforming text to low-resource domains is more powerful than only using the data from high-resource domains. A note worthy chart of their performance improvement w.r.t other DA methods is presented on Figure 3.4 .

Exp ID	Method	NW → SM				SM → NW			
		Number of Training Samples				Number of Training Samples			
		1K	2K	3K	4K	1K	2K	3K	4K
Exp 2.0	Baseline (No Augmentation)	34.71	35.47	35.69	35.69	25.94	28.83	29.54	30.20
Exp 2.1	Keyboard Augmentation	34.83 ↑	35.69 ↑	36.13 ↑	36.69 ↑	27.01 ↑	27.87 ↓	28.21 ↓	28.43 ↓
Exp 2.2	Swap Augmentation	29.49 ↓	30.54 ↓	31.36 ↓	32.07 ↓	27.33 ↑	28.62 ↓	29.13 ↓	29.56 ↓
Exp 2.3	Delete Augmentation	28.59 ↓	29.56 ↓	30.01 ↓	30.38 ↓	27.81 ↑	28.95 ↑	29.16 ↓	29.20 ↓
Exp 2.4	Spelling Augmentation	34.97 ↑	35.51 ↑	35.95 ↑	36.24 ↑	28.09 ↑	29.68 ↑	30.42 ↑	30.92 ↑
Exp 2.5	Synonym Replacement	34.77 ↑	35.95 ↑	36.31 ↑	36.63 ↑	28.94 ↑	29.18 ↑	29.84 ↑	30.66 ↑
Exp 2.6	Context Replacement	24.89 ↓	26.04 ↓	27.04 ↓	27.60 ↓	26.98 ↑	26.95 ↓	28.03 ↓	28.06 ↓
Exp 2.7	DAGA (Ding et al., 2020)	32.75 ↓	33.62 ↓	34.17 ↓	34.32 ↓	28.57 ↑	29.29 ↑	29.95 ↑	30.54 ↑
Exp 2.8	Ours (Domain Mapping)	**43.19** ↑	**43.85** ↑	**44.29** ↑	**44.82** ↑	**35.98** ↑	**38.33** ↑	**39.55** ↑	**40.84** ↑

Figure 3.4: Comparison of [19]'s cross-domain mapping method with previous data augmentation methods for NER task. Scores are calculated with the F1 metric. The best score for each column is in **bold**

Although the aforementioned methods produce improvements in NER performance, their performance under standard in-domain conditions is relatively lower than SotA. Recent advancements in knowledge bases and knowledge graphs has driven research on the use of this as an extra reliable source of data. This in turn introduced the usage of expert-guided heuristics and template-based DA methods. The work of Cui *et al.* [42] leverages template-based few-shot NER using BART. In contrast to the traditional sequence labeling methods, their method is more powerful on few-shot NER.
Experiment results show that their model achieves competitive results on a rich-resource NER benchmark, and outperforms traditional sequence labeling methods. This approach has caught our attention since it can be fine-tuned for the target domain directly when new entity categories exist, which is close to our goal.
However, there is still a gap in addressing the solution and method for collecting data entities and sequences for the target domain. Inspired by the SotA visions and approaches explained here, we step into details alongside our contributions and novel method for data re-compensation in 4.

3.2 Fine-grained NER methods

According to literature [40][43][44], a vast range of information for scientific discovery is provided by fine-grained named entity recognition. Medical research, for instance, must examine hundreds of distinct, fine-grained entity types(e.g protein types). Making accurate and uniform annotation is challenging, even for domain experts. Nevertheless, the performance of relation extraction and knowledge graph creation

has recently been proposed to be greatly improved by using fine-grained NER. Filtering away candidate KB's *relation types* that do not match to the type constraint is made easier by being aware of the fine-grained entity categories.
Additionally, it offers increased details to help match the query with potential replies(candidate data satisfying the query), enhancing the effectiveness of question responding. Yet, the decision to utilize a certain standard file for the keywording has an impact on the entire process because it affects both the level of atomicity at which these entities are disambiguated as well as how the entities are defined. On the basis of how well the expressions in an KB fit the specific topic, not how well they are related to the Linked Data cloud.
As mentioned in section 4, for the task of Fine-grained NER, SotA reveals the best performing models leverage DSL. Moreover, having in mind the goal of creating a generalizable and adaptable framework leads to our focus being on leveraging KBs and ontologies rather than learning the deeper relations and heuristics in sequences of a specific domain. Amongst them, the ones related to our research scope are presented in the following section.

3.2.1 AutoNER: Learning Named Entity Tagger using Domain-Specific Dictionary

With regard to the solutions on replacing manual annotations, DSL in combination to external dictionaries proved outstanding results, but the generated noisy labels pose significant challenges on learning effective neural models.
Shang *et al.* propose two neural models to suit noisy distant supervision from the dictionary.

> *" First, under the traditional sequence labeling framework, we propose a revised fuzzy-CRF layer to handle tokens with multiple possible labels. After identifying the nature of noisy labels in distant supervision, we go beyond the traditional framework and propose a novel, more efficient neural model, AutoNER, with a new Tie or Break scheme. In addition, we discuss how to refine distant supervision for better NER performance. Extensive experiments on three benchmark datasets demonstrate that AutoNER achieves the best performance when only using dictionaries* ***with no additional human effort*** *and delivers competitive results with SotA supervised benchmarks."*[45]

Existing DSL NER models every unmatched token will be tagged as a non-entity. However, as most existing dictionaries have limited coverage of entities, simply ignoring unmatched tokens may introduce false-negative labels. AutoNER's data collection method takes this into consideration and proposes to extract high-quality out-of-dictionary phrases from the corpus, and mark them as potential entities with

a special "unknown" type. Moreover, every entity span in a sentence can be tagged with multiple types, since two entities of different types may share the same surface name in the dictionary. To address these challenges, also propose and compare two neural architectures with customized tagging schemes.

Tie or Break Tagging Scheme: For every two adjacent tokens, the connection between them is labeled as (1) `Tie`, when both tokens are matched to the same entity; (2) `Unknown`, if at least one of the tokens belongs to an unknown-typed high-quality phrase; and (3) `Break`, otherwise. An example can be found in Figure 3.5. The distant supervision shows that "ceramic unibody" is a matched `AspectTerm` and "8GB RAM" is an unknown-typed high-quality phrase. Therefore, a `Tie` is labeled between "ceramic" and "unibody", while `Unknown` labels are put before "8GB", between "8GB" and "RAM," and after "RAM".[45]

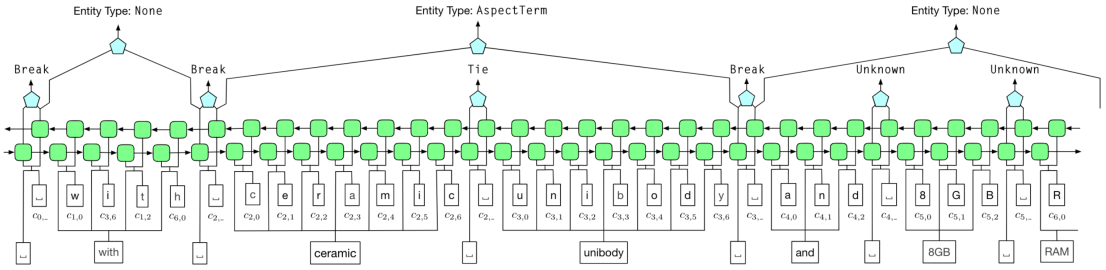

Figure 3.5: The illustration of AutoNER with `Tie or Break` tagging scheme. The named entity type is `AspectTerm`. “ceramic unibody” is a matched `AspectTerm` entity and “8GB RAM” is an unknown-typed high-quality phrase. `Unknown` labels will be skipped during the model training. [45]

Another noteworthy contribution’s of AutoNER’s work is the well-defined comparison of their distant-supervised learning method with the SotA supervised benchmarks which show the viability of DSL methods as indicated in Figure 3.6. They apply DSL by modifying the original dictionary and eliminating entities whose canonical names never appear in the provided corpus in order to fit it to a corpus-related subset. The idea is that in order to clear up any confusion, people will probably mention the official name of the entity at least once. Next, the training is done by their proposed Fuzzy-LSTM-CRF network which we treat as a benchmark for our models comparison.

3.2.2 BOND: BERT-Assisted Open-Domain Named Entity Recognition with Distant Supervision

Liang *et al.* [46] study the open-domain NER problem under distant supervision. And to address the incomplete and noisy distant labels they propose BOND, a

Method	Human Effort other than Dictionary	BC5CDR			NCBI-Disease		
		Pre	Rec	F1	Pre	Rec	F1
Supervised Benchmark	Gold Annotations	88.84	85.16	**86.96**	86.11	85.49	**85.80**
SwellShark	Regex Design + Special Case Tuning	86.11	82.39	84.21	81.6	80.1	**80.8**
	Regex Design	84.98	83.49	**84.23**	64.7	69.7	67.1
Dictionary Match	None	93.93	58.35	71.98	90.59	56.15	69.32
Fuzzy-LSTM-CRF		88.27	76.75	82.11	79.85	67.71	73.28
AutoNER		88.96	81.00	**84.8**	79.42	71.98	**75.52**

Figure 3.6: NER Performance Comparison on the BC5CDR and NCBI-Disease datasets. SwellShark has no annotated data, but for entity span extraction, it requires pre-trained POS taggers and extra human efforts in designing POS tag-based regular expressions and/or hand-tuning for special cases. [45]

framework which leverage a two stage training algorithms.

> *"In the first stage, we adapt the pre-trained language model to the NER tasks using the distant labels, which can significantly improve the recall and precision; In the second stage, we drop the distant labels and propose a self-training approach to further improve the model performance."*[46]

taking a closer look, first BERT and RoBERTa is used to predict a set of pseudo soft-labels (*soft* labels are attached with a score that indicates their likelihood since they an be a member of multiple classes) for all data w.r.t the distantly-matched labels. Next, replace the distantly-matched labels with the pseudo soft-labels and design a *teacher-student* framework to further improve the recall. The *student* model is first initialized by the model learned in the first stage and trained using pseudo soft-labels. Then, the *teacher* model updated from the *student* model in the previous iteration to generate a new set of pseudo-labels for the next iteration to continue the training of the *student* model.

Another novel contribution of Liang *et al.* is the generation of distant labels which is another aspect to analyze for comparison to our method. By first using hand-crafted rules and POS tagging, potential entities are located. Then with SPARQL [19] a query from Wikidata is performed to determine the entities types. Afterwards, in order to better match additional data tokens, gazetteers are collected from other web resources.
Figure 3.8 conveys a better understanding of the distant-label generation. After finding the potential entities, through POS-tagger, e.g., NLTK [47]. Then, using SPARQL to query the parent categories of an entity in the knowledge tree. For entities with ambiguity (e.g., those with meanings), during the matching process

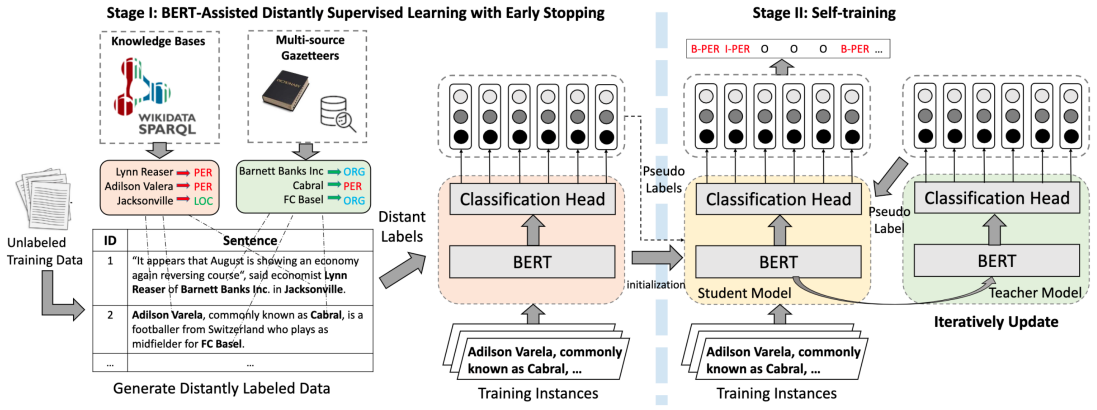

Figure 3.7: The two-stage BOND framework. In Stage I, the pre-trained BERT is adapted to the distantly supervised NER task with early stopping. In Stage II, a student model and a teacher model are first initialized from the model learned in Stage I. Then the student model is trained using pseudo-labels generated by the teacher model. Meanwhile, the teacher model is iteratively updated by the early-stopped student.[46]

they are discarded and assigned with type `O`). For next stage, a gazetteer is built based on crawling online data sources for each selected entity types; which in turn will be used for matching an entity with a type if the entity appears in the gazetteer for that type. Yet, if a token is not matched with the aforementioned method it will be passed to a set of hand-crafted rules for matching. Matching of a potential entity with a type is done if there exists a stamp word in this entity that has frequent occurrence in that type. For example, "Inc." frequently occurs in `organization` names, thus the appearance of "Inc." indicates that the entity labels of words in the "ABC Inc." should be `B-ORG` or `I-ORG`)[46].

The experiments are conducted on multiple datasets and our attention is to the CoNLL03 [10] results. Furthermore, for distant label generation, entity types are matched to external KBs such as the Wikidata corpus and gazetteers gathered from various online sources. The model is compared with different groups of baseline methods. Namely, Fully-supervised Methods, Distantly-supervised Methods, and KB-matching methods which fully demonstrate the robustness of BOND's approach and methodology, and their method consistently achieves the best performance under the distant supervision scenarios, in F1-score, precision, and recall. And more importantly, on the CoNLL03 dataset, compared with baselines that use different sources, the model also outperforms them by significant margins.

Another highlight BOND's research work is they realizing their work also relevant to **semi-supervised learning**. emphasising on semi-supervised learning

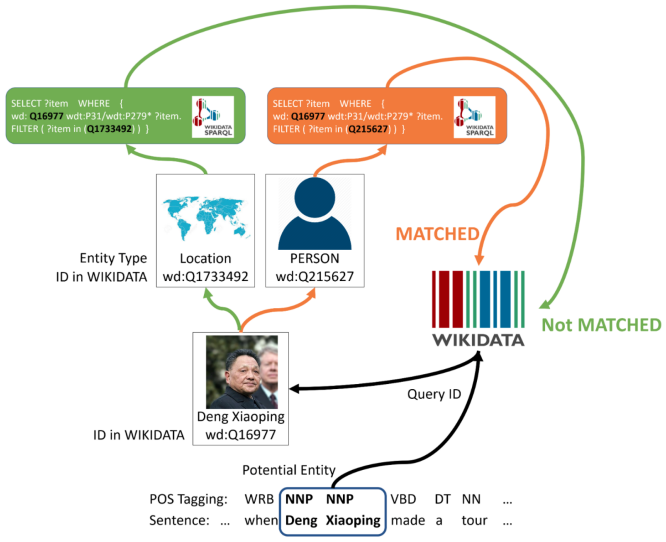

Figure 3.8: Illustration of matching entities from Wikidata. [46]

methods having a posterior set of labeled data. They rely on the labeled data to train a sufficiently accurate model. With this aspect in mind unlabeled data is used to induce certain regularization in order to improve generalization performance. In contrast to distant supervision that considers the settings with only noisy labels. They also imply that existing semi-supervised learning methods such as Mean Teacher and Virtual Adversarial Training can only marginally improve the performance.

Method	CoNLL03	Tweet	OntoNote5.0	Webpage	Wikigold
Entity Types	4	10	18	4	4
KB Matching	71.40(81.13/63.75)	35.83(40.34/32.22)	59.51(63.86/55.71)	52.45(62.59/45.14)	47.76(47.90/47.63)
Fully-Supervised (Our implementation)					
RoBERTa	90.11(89.14/91.10)	52.19(51.76/52.63)	86.20(84.59/87.88)	72.39(66.29/79.73)	86.43(85.33/87.56)
BiLSTM-CRF	91.21(91.35/91.06)	52.18(60.01/46.16)	86.17(85.99/86.36)	52.34(50.07/54.76)	54.90(55.40/54.30)
Baseline (Our implementation)					
BiLSTM-CRF	59.50(75.50/49.10)	21.77(46.91/14.18)	66.41(68.44/64.50)	43.34(58.05/34.59)	42.92(47.55/39.11)
AutoNER	67.00(75.21/60.40)	26.10(43.26/18.69)	67.18(64.63/69.95)	51.39(48.82/54.23)	47.54(43.54/52.35)
LRNT	69.74(79.91/61.87)	23.84(46.94/15.98)	67.69(67.36/68.02)	47.74(46.70/48.83)	46.21(45.60/46.84)
Other Baseline (Reported Results)					
KALM †	76.00(– / –)	–	–	–	–
ConNET ◇	75.57(84.11/68.61)	–	–	–	–
Our BOND **Framework**					
Stage I	75.61(83.76/68.90)	46.61(53.11/41.52)	68.11(66.71/69.56)	59.11(60.14/58.11)	51.55(49.17/54.50)
BOND	81.48(82.05/80.92)	48.01(53.16/43.76)	68.35(67.14/69.61)	65.74(67.37/64.19)	60.07(53.44/68.58)

Figure 3.9: BOND's experiment results in comparison with different groups of of baseline methods: F1-Score (Precision/Recall). [46]

Chapter 4

Fine-Grained Named Entity Recognition

The contents of this chapter will go over the details of this particular thesis project, from the motivations behind the annotated dataset and generated labels to the models analyzed for this specific task, and a new proposed pipeline, based on the application of entity linking and knowledge graphs to enhance our data augmentation.

4.1 Task description

Nowadays, Despite the growth of chatbot technology, they still have certain fundamental flaws that limit the extent of their applications. One of their shortcomings is when processing Named Entities, which leads to their performance being dependent on understanding human in-query. To better describe the goal and vision of our research work, a brief introduction of Recruitment chatbots is necessary. Recruiting chatbots, also known as hiring assistants, are used to automate communication between recruiters and candidates. After candidates apply for jobs from the career pages, recruiting chatbots can obtain candidates' contact information, arrange interviews, and ask basic questions about their **experience and background**. This will introduce multiple challenges for the AI model.
Before explaining in detail our contributions, it is noteworthy to explain the source of raw data and how Hubert's recruitment chatbot provides us with the data for research work. The data is gathered by Hubert's chatbot, which is designed to conduct interviews with a huge number of job applicants. The job type and domain are specified by Hubert's client. A section of the interview is dedicated to asking for the applicant's background information. This includes multiple branches of asking for educational background, skills, and experience. This drives the need

for a NER system that can recognize entities corresponding to curtain domains. And, considering both the variability of the applicant's background areas and the recurring change in the domain of job with the change of client and job type, a NER model is needed that is able to recognize fine-grained entities, e.g. *University name* and *Field of study* or *Skill type* and not just `[PER]`, `[ORG]`, and `[LOC]`, which is the usual case for NER research scope.
As previously mentioned, two of the SotA models described have been picked for investigation in this particular use case of Fine-grain NER, in order to offer our contribution: AutoNER[45] and BOND[46]. These models have been chosen as they have been alleged to be a fairly advanced basis to study and propose improvements w.r.t our research scope. Moreover, both leverage DSL, where training labels are generated by matching mentions in a document with the concepts in the KBs. However, this kind of KB-matching suffers from two major challenges: *incomplete annotation* and *noisy annotation* as discussed in Chapter 3.
In order to generate an objective and reliable estimate of the objective function, distantly-supervised NER employs positive and unlabeled learning (PU-learning). However, utilizing PU-learning for distantly-supervised NER has two limitations. First, PU-learning uses the prior distribution for each entity type, which is estimated from an existing human-annotated test set and is not always available for new entity types. Second, the heuristically derived class-imbalance rate for each entity type has a significant impact on how well PU-learning performs. Due to the two aforementioned restrictions, it is challenging to apply PU-learning to distantly-supervised NER tasks on new entity types from new domains.

AutoNER's "tie-or-break" tagging offers an "unknown" type that can be skipped during training to reduce the effect of false negative labeling with DSL and incomplete KB-matching. However, their phrase mining method(AutoPhrase [45]) can miss low-recurring words for unknown-tag generation.
BOND introduces self-training approach to iteratively integrate more training labels and improve the NER performance, These approaches do not work well with fine-grained NER that has severe low precision and low recall with KB-matching. This takes into account the fact that earlier techniques for generating distant-labels assumed a high degree of precision and reasonable KB-matching coverage. For instance, the CoNLL03 dataset's KB-matching[10] revealed over 80% precision and over 60% recall in BOND. Additionally, they essentially overlook the issue of noisy annotation by merely excluding those many labels from the KB-matching process.

4.2 Method and proposed solution

The thorough investigation of open research topics reveals, the major topic related to this area is NER, and more specifically, fine-grained NER. As mentioned in Section 2.1, NER is a crucial natural language understanding task for many downstream tasks such as question answering and retrieval. Hence, the first and largest obstacle is the need for large annotated inputs for acceptable results. As stated in Chapter 3, whilst this area is exploited by various approaches, it still requires much further research and analysis.
The literature review clearly shows a lack of research on the potential of Knowledge Graphs and Entity Linking in investigating entity relationships and utilizing these relations as a basis to find the correct relation that leads to the target entity. This type of entity extraction helps in improving domain-specific KBs and NER models. Additionally, all discovered research works are on a curtain domain, and by "fine-grained NER model" they refer to a NER system capable of entity recognition on labels and sub-labels from that only domain, such as CHEMNER [40], Chemistry labels.
Evidently, there is no research on enabling a model to directly fetch new training data from an online reliable KB through an EL pipeline that has the ability to generalize to new domains. This gap in literature explains another challenge to be experimented is to build a training pipeline capable of switching to new fields of employment and their required keywords with limited human effort. Possible solutions include training transformer models with label-specific data, manual data annotation, use of open-source data, and pre-trained tools.
Yet, all these are highly task-specific and do not contribute to forming a robust generalisable model to tackle the problem objectively. Therefore, we have explored this area by carefully analysing the recent findings in both Ontology Mapping and Graph Enrichment [29].

With this project, we plan to both address the two mentioned challenges in fine-grained NER and also address the shortcomings of research work in a domain-adaptable NER, so we propose a proof of concept novel approach in Data Augmentation to improve the performance of our NER model. Our goal is to first utilize a framework that incorporates a KG-linking method to fetch new data based on the domain and area of the entity recognition task (e.g. education, experience, skill, etc.) and also give us the ability to change domain with limited cost and effort in retraining the model for new labels.
To explore this path, with the goal of reducing human effort in data annotation and expert evaluation, we propose Ontology Guided Name Entity Recognition (**OG-NER**), a robust semi-supervised NER method for fine-grained adaptable-domain NER tasks.

Our method attempts to solve the lack of data by using a technique that constructs an initial domain-specific KG based on the pre-defined target classification labels related to that domain. Then it further semantically enriches the KG by querying Wikidata. Next, an analysis on the most typical sentence structures of that domain provides us a series of blank sentences. Finally, the entities from the populated KG are inserted into their corresponding positions in the blank sentences by data augmenter as stated in Section 4.4.2. With this approach, we have also changed the learning method from distant to semi-supervised.

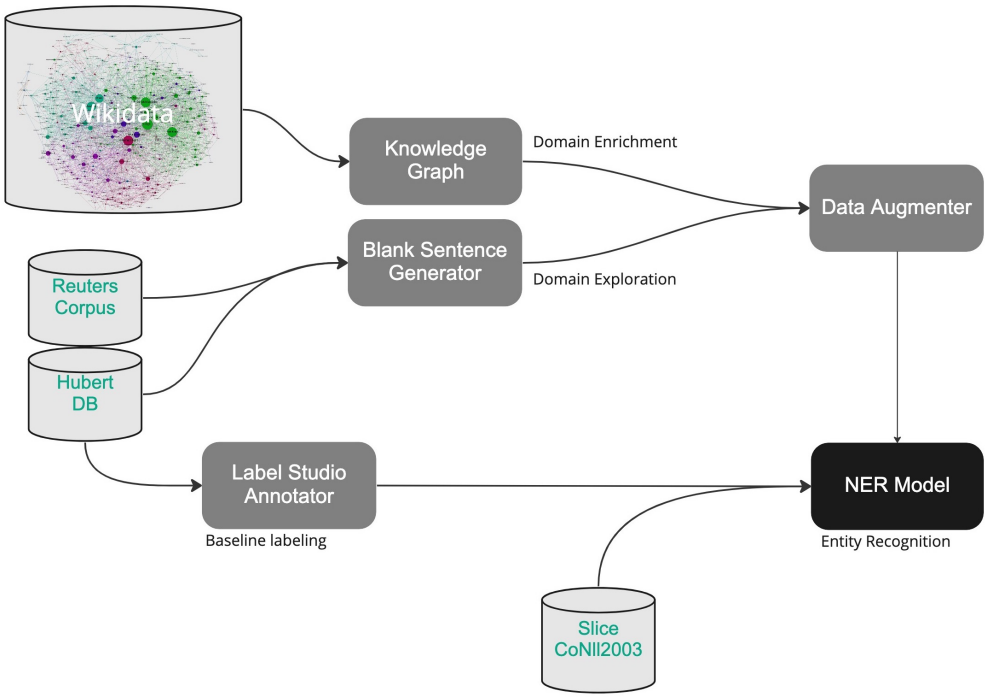

Figure 4.1: OG-NER Project Framework

4.3 Dataset creation

Data is the raw material of today's world and yet in most cases it is hard to obtain. Every business has a ton of digitally stored data and information, such as manuals, FAQs, rules, regulations, or databases. But in most businesses, these gems are dispersed among various divisions. Many algorithms for solving important real-world issues are based on the analysis of named entities, such as applicant monitoring, ATS prediction, and auto-score development, to name a few.
Most of these algorithms could benefit from the availability of high volumes of

expert-annotated data, which is usually not publicly available due to the high costs involved with both annotation and precise evaluation. Data scarcity is also an issue which leads to little or no labeled training data, or insufficient data for a particular label in comparison to other labels (a.k.a. data imbalance). Larger technological firms typically have access to a wealth of data, but they could run into a data imbalance. On the other hand, what makes a NER model stand out from other performant models is the cost of its training, which includes both the cost of building the training dataset and the cost of the training process itself.

The NER task of this thesis project is to show the impact of KG in a NER architecture capable of recognizing custom-labeled named entities and also offer competitive results in relation to existing SotA models. The choice of the dataset for this specific thesis work was made considering the vision the company had for this research work and also the data the company could provide with respect to GDPR and the signed NDA. According to our preliminary research work and literature review and respecting company goals for this thesis project, we decided to use English language and the educational background domain data.
The challenge of lowering the cost of building training data, One of the most effective and academically acceptable solutions is to use reference open data. Since we working with text data, the most straightforward choice is the use of open community-driven databases. These type of databases are available as structured form of knowledge bases and knowledge graphs(e.g. Wikidata, DBPedia, Yago, etc). The choice of a suitable KB is discussed in detail in the following chapters.

The first step and most important step of any dataset creation is label set definition. Since we started the research with the goal of analyzing an applicant's background information with the recognition of named entities in their chats. While different clients would need an assessment of skills from different domains, all would need an assessment of the applicant's educational background. Furthermore, since the recognition of these named entities has the future goal of being processed for scoring and ranking of each applicant, it is important that in addition to recognition of their school/university of studies, the education program, field of study, education level,location of school, and duration of studies be acknowledged and assessed.
This design of labels also relate to the scope of research being focused on fine-grain NER, since the labels are introduced in a hierarchy w.r.t education domains and if this convention of custom labels didn't existed a base NER model(e.g. `bert-base-uncased`) would recognized both *education program* and *education level* as *Miscellaneous* label.

- **Label set 1(Open domain - CoNLL03):**

Person [PER], Organization [ORG], Location [LOC], Miscellaneous [MISC]

- **Label set 2(Educational background domain):**
  Name of university/school [INS], Education program [EDUPROGRAM], Education level [EDULVL], Duration of study [DURATION], Location of university/school [LOC]

On the other hand, since we are comparing our learning method and NER network to the SotA benchmarks, we must also comply with the datasets that those models are trained on and perform best on. According to the literature, the general benchmark to evaluate any NER model is the CoNLL03 [10]. Consequently, we will be experimenting and comparing our NER to the mentioned SotA NER models with respect the two above-mentioned label sets.

4.3.1 Entity annotation

Training NER models in a semi-supervised fashion still implies the necessity of a dataset that includes carefully gathered ground truths in that specific domain. To kick-start the project and have a baseline dataset for start experimenting on SotA NER models and with respect to the defined label set, we proceeded with annotating on the section of the applicant's chats regarding their educational background. The time invested for annotation closely revealed the aforementioned annotation cost and also defined the need for collecting annotated data for further training and improving the NER model.
Our annotation process was carried out using "Label Studio" [48]. It is an open source data labeling platform which provides multiple beneficial tools that help toward the annotation process. To name a few highlights, its Configurable layouts and templates adapt to datasets and workflows. Additionally, it connects to cloud object storage and label data there directly and it has a fairly easy to work GUI.

4.3.2 Analysis of available services

According to the survey [49], pre-training language models on unstructured text can acquire certain factual knowledge and large-scale pre-training can be a straightforward way to inject knowledge. However, rethinking the method of knowledge aggregation in an efficient and interpretative manner is also of significance. As mentioned in previous chapter there are various databases and knowledge bases available and each them of them is powerful in curtain aspects, e.g., specific knowledge acquisition tasks include Knowledge Graph Completion, Triple Classification, Entity Recognition, and Relation Extraction.
Based on [29], there are many benefits to Wikidata that makes it suitable for our work considering the task at hand:

- Wikidata provides the most complete ontology and expert-evaluated data through its open-source community-driven features which makes it a reliable source for our KG data collection
- Wikidata has the most widespread and complete set of instance and subgroup categories, which makes it the perfect match for the need of having a source for quickly fetching data when switching to a new domain.
- Wikidata's native query language(SPARQL) is the inert solution to our need for key data collection. By carefully selecting the type of relation we are looking for, it gives us access to a pool of knowledge in the target domain.
- Wikidata's is offered in multiple languages and each node and link are also connected to their corresponding features in the other language. This enable us to query data from other languages for future work purposes.

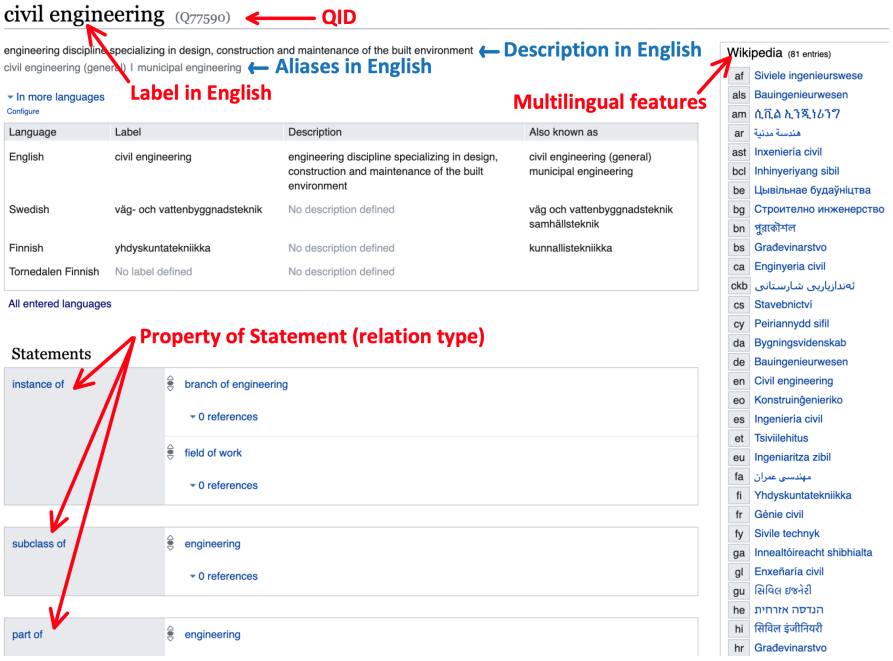

Figure 4.2: Wikidata KB.

4.4 Main task

A considerable obstacle in the path NER is that a named entity can have different meaning with regard to the place it is being used; This is academically realized as entity ambiguity. Although NER recognises the mention and assigns general semantic categories or labels, it is unable to identify the precise entity to which it refers. Traditionally, ambiguity resolution has been handled apart from NER as a sense disambiguation task. However, theoretically, the two have comparable objectives. Name disambiguation can be seen as a further step of "recognition" in which the true identity of a name mention is discovered, whereas NER can be thought of as disambiguation at a higher level. Disambiguation is frequently a necessary post-process in practice to make NER output useful for other sophisticated NLP applications.
Humans usually resolve ambiguities based on context and this overture has also driven AI algorithms to introduce a disambiguation techniques by linking a named-entity mention to an instance in a KB, typically Wikipedia-based resources like DBpedia or Wikidata. This is acceptable since literature checks Wikipedia's instances as a reliable source because of its community-driven features. Hence, the normal procedure toward EL is first recognizing an entity and then link it to its Wikipedia URL.
Yet, In light of our data collection method, we leverage this EL mechanism with a bottom-up fashion. To be more precise, we treat the origin KB as a reference and reliable source of labeled data ans use the same entity linking approach to query and collect the required data.
This section presents a more detailed insight into our proposed method to tackle the challenge of lack of training data and how to compensate for it. We take a closer look into how the correct labels are extracted and how to prepare adequate sentence for these keyword to be placed upon. In short, preliminary steps for tuning the target domain and context for DA. This task consists of different steps which will be discussed in the following sections.

4.4.1 Domain exploration

Taking into account the importance of context and domain, to correctly design our Data Augmenter, a collection of descriptive information regarding that context will be required. Moreover, having in mind that our model works with individual sentences from chat answers and not whole paragraphs, we understand that it is more reasonable to focus our collection on the acquisition of the most typical sentence structures used in a context. This is with the key insight that curtain questions will have a series of common correct answers. This introduces the notion

that, with limited expert input and supervision, we can investigate and design the most common sentences (in our case study: answers) in the selected domain and prepare a set of blank sentences to be filled and used by data augmenter.

Corpus collection and analysis

More than 80% of the data we generate every day is unstructured and not in a predefined way, making it extremely difficult to analyze and process. This fact shows the importance of the intention to avoid starting from scratch and design a comprehensive set of common sentences from a domain. There exist some routine keyword extraction tools that are the key to helping automatically index data, summarize text, or generate tag clouds with the most representative keywords. Also, available services like SpaCy's Sense2vec [4], Rapid API's Word Association, and MonkeyLearn [50] help with this goal.
Yet, they are all built to provide semantically related words(e.g. synonyms, hypernyms, hyponyms, and etc.), which makes them limited to only word-level . For our goal, it is necessary to go beyond word-level and utilize the tool for sentence-level work. Following this, the best source to initiate research on a domain's sentence-level statistics is a corpus relevant to that domain.
The selection of corpora requires some preemptive study of the domain to find the most suitable corpus for a determined goal. In our case, for the background education, the analysis of the most common sentences was carried out on Hubert's chat data, which was already structured and didn't need to pass through the whole process. Nevertheless, to fully examine our system, we carried out the data collection for the CoNLL03 labelset by analyzing Reuters Corpus [51].
Reuters Corpus has made the global information, news, and technology group available to research communities worldwide. The Reuters Corpus Volume 1 includes over 800,000 news stories typical of the annual English-language news output of Reuters. whereby each story is annotated for topic, region, and industry sector. And considering the CoNLL03 [10] was derived from the same corpus, it makes this corpus a great basis for testing our data collection scheme.

Analysis of most typical sentence structures in a domain

There are different techniques to analyze and compare to build an effective framework for acquiring key words from a corpora. From simple statistical approaches that detect keywords by counting word frequency, to more advanced machine learning approaches that create even more complex models by learning from previous examples.
There are different types of statistical approaches, including word frequency, word

collocations and co-occurrences, TF-IDF (short for term frequency–inverse document frequency), and RAKE (Rapid Automatic Keyword Extraction). These methods can extract a text's most significant keywords without the need for training data which is beneficial since we are using these for the initial steps of data collection toward data re-compensate method. In the following we explain in detail the techniques we chose for determine and extract most common sentence structures from a corpora:

Word Frequency: Listing the words and phrases that appear most frequently in a text constitutes word frequency. This can be helpful for a variety of general insights into the corpus. Word frequency techniques, on the other hand, ignore important factors linked to the meaning, structure, syntax, and order of words and treat manuscripts as little more than a "bag of words." For instance, this term extraction method misses very important information when trying to discover synonyms.

Word Collocations and Co-occurrences: Also referred to as N-gram statistics, aid in comprehending the semantic organization of a text and count different words as one. Words that regularly go together are called collocations. Yet, notice in literature collocation refer to meaningful N-gram and not any co-occurrence of words. The most frequent collocations are tri-grams (a group of three phrases, like "as a result " or "chief executive officer") and bi-grams (two terms that appear next to each other, like "customer service," "video calls," or "email notice").
Additionally, since we were interested in sentence-level analysis we also took a look into four-grams. Contrarily, co-occurrences relate to words that frequently appear together in a corpus. Although they don't have to be close to one another, they do have a semantic proximity.
Different measures are used to rank collocations. The simplest method is to rank the most frequent bi-grams or tri-grams is counting frequencies of adjacent words with part of speech filter. However, a common issue with this is adjacent spaces, stop words, articles, prepositions or pronouns are common and are not meaningful. A solution is to filter out the one that do not contain stop words. Yet, since we are using these as a preliminary step for our concordance search, this solution is not appropriate for our work.
A better measure is to use Pointwise Mutual Information (PMI) score. The purpose of PMI is to measure the likelihood that two words will appear together while accounting for the possibility that this co-occurrence may be influenced by the frequency of the individual words. So, using equation 4.1, the algorithm calculates the (log) chance of co-occurrence scaled by the product of the single probability of occurrence.
The primary intuition is that it calculates how much more frequently words appear

together than they would independently. It is, nevertheless, extremely sensitive to uncommon word combinations. For instance, if the bigram "abc xyz" appears at random and neither "abc" nor "xyz" appear elsewhere in the text, "abc xyz" will be classified as a very significant bigram even though it may simply be a chance typo or a word that is too uncommon to be classified as a bi-gram in general. As a result, a frequency filter is combined with this technique.

$$PMI(x;y) = \log \frac{p(x,y)}{p(x)p(y)} \tag{4.1}$$

Other measures like hypothesis testing with a chi-square test are used for comparison. To determine which measure for ranking seems to make the most sense for a certain dataset, we also run several tests with different measurements and intersect the outcomes of those lists. Lastly, in order to establish a proper threshold for the top occurring collocations, the list is manually scanned and inspected to filter out items until they no longer make sense or are of value for creating a sentence.

Concordance search: To highlight and extract the complete sentences that include the most recurring collocations, we use `nltk`'s concordance search. A concordance view shows us every occurrence of a given word, together with its surrounding context. This preview will give the expert a general overview of how these most common sentence structures are used and how they include and position named entities.
Yet, a major challenge in this step was introduced by the original concordance search being designed to search for one token in the corpora. And to re-design the algorithm to our needs, which includes typical sentence structures(i.e., ngrams) would require the design of a multi-token concordance search. This solution is another honored contribution of our work.

Blank sentence: To perform the mentioned techniques, we use `nltk` library that efficiently provides tools for each purpose. We unify prior approaches into a comprehensive framework that combines all the rankings driven by each studied measure to filter the adequate collocations for the most common sentences. Since this step is taking place prior to having a fine-tuned NER model in that domain, a limited contribution from an expert is required.
This act manifests in the use of human intuition for understanding the objective and use case of common sentences based on the concordance search's output by recognizing the determined target named entities and tagging their position in the sentence. Next, compose a few similar sentences, leaving blank the position of named entities so they can be filled by data augmenter with proper entities in the proceeding step.

Sentence #1	Blank Augmented	"I have studied <EDUPROGRAM> at <INS>." "I have studied **Masters** at **KTH**."
Sentence #2	Blank Augmented	"My <EDULVL> is in <EDUPROGRAM> from the <INS>." "My **Bs** is in Engineering from the **Stockholms universitet**."

Table 4.1: Example of a generated instance of blank sentence and corresponding result after augmentation.

4.4.2 Domain enrichment

Ontology analysis

The pre-training of language models on unstructured text can pick up specific factual knowledge. Moreover, pre-training on a large scale might be an easy technique to provide knowledge. However, it is also important to re-evaluate the method of knowledge aggregation in an efficient and interpretable manner is also of significance. This is where ontologies and KBs will come into the spotlight. In this section we describe our novel approach in using SPARQL to collect data from a KG and semantically populate our KG.
Ontologies include an appropriate place for domain-specific concepts in the existing taxonomy of the same domain. In particular, the Wikification oriented ontology, gives us this tidy structure of classes of nodes and links, which is very advantageous for traversing a KG correctly and fetching the determined data.
Another noteworthy feature of using ontologies is the discovery of "Gold Links". These are relations between entities that are completely guaranteed by an expert and rarely suffer from incomplete data across Wikidata and also different languages. We can use those links to acquire reliable data points(named entities). Reliable data is of great importance when analysing sensitive background information about an applicant.

Entity linking and Knowledge Graph creation

A rewarding utilization of this existing ontology is through the entity linking process, where instead of textual features generated from input documents or text corpora, the system extracts complex features that take advantage of the information graph topology or exploit multi-step relations between entities that would otherwise go undetected by simple text analysis. This linking process has been used in different ways in literature, yet there is limited research that exploits this as a method for collecting keywords for fine-grained NER.
This can be answered by combining the EL and graph traversing ideas with querying from knowledge bases. The significance of this approach is that we are exploring

and collecting the linked knowledge across a reference KB to compensate for the lack of annotated data for our NER needs. This approach is executed by building graphs using Wikidata Query Service [19] and D3.js [52] considering the following steps:

The first step is the identification *relation types*. This enables us to correctly navigate an ontology and reach the appropriate target data points. This can be utilized in different ways. To name an example, when we want to populate the KG with data from the same label(e.g. `[EDUPROGRAM]`), starting from the Qid of a random education field(e.g. *engineering:Q11023*) we investigate which links are suitable for guiding us into the parent node. Possible reasonable link class for this purpose are *subclass of:P279*, *instance of:P31*. By traversing these links with a bottom-up advance, we can reach the parent entity which in this case will be *applied science:Q28797* and *economic sector:Q3958441*. With this reveal, we have a good insight in what can be the candidate nodes for collecting education fields similar to *engineering*.
Hence, by following a top-down with the same link relation classes, we reach our target data points. A presentation of this workflow is illustrated in Figure 4.3.When the iteration steps are increased, the graph generator is tasked with traversing deeper levels of the KB. This results in a crowded KG and is not readable for the purpose of visualization.
It is also worth noting that, when analysing conversational text, people tend to use acronyms for named entities. Considering the fact that named entities often refer to something well-known, results in the acronym being recognizable by both parties in the conversation. This results in confusion in the network and the need for a disambiguation technique applied to the NER model for recognizing such entities. Yet, by collecting the data with our approach, we can use the relation of *alternative name:P4970* in Wikidata to also collect all the other aliases that a named entity is known by, thus providing a solution for acronyms that are being used in sentences.

Data Augmentation

As discussed in Chapter 3, while there exist different DA methods for NLP tasks and only some are applicable to NER, they still cannot reach the level of performance equal to having more adequate data for training. Moreover, considering the goal being to minimize the effort required for tuning the augmenter when switching to new domains, the necessity to explore new DA approaches arises.
Additionally, NER systems often perform well on in-distribution data but poorly on instances taken from a shifted distribution. In the literature on this problem, both adversarial methods and expert-guided heuristics [53] are applied to solve

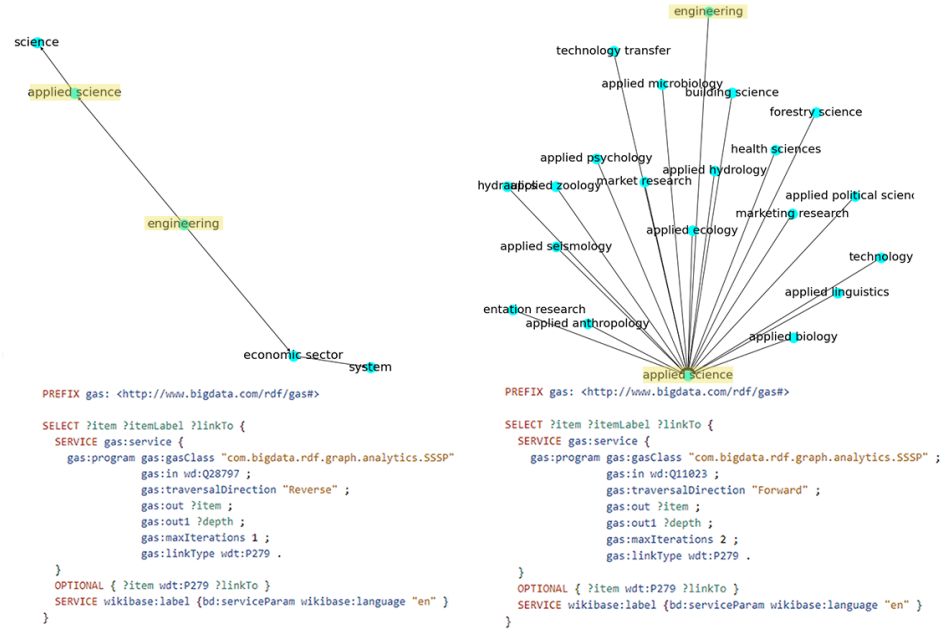

Figure 4.3: Illustration of Wikidata entity query using SPARQL.

it. A DA method that leverages a transition-based entity swapping approach to improve out-of-domain generalizations is used as a solution. It receives as input the generated blank sentences and the expert evaluated entities through the knowledge graph, to be used in generating new training data.
The pipeline receives the training data from the blank sentences and use the KG instances to replace corresponding mentions. Algorithm 1 shows a detailed overview of steps and attributes implemented to perform data augmentation.

4.5 Model architecture

With respect to the literature review and investigation of benchmark results, which was also backed by Lothritz *et al.* [8] research works conclusion, "transformer-based models do indeed outperform the BiLSTM-CNN-CRF model with regards to F1 score, with BERT yielding the highest results over-all".
Moreover, SotA performance in many natural language processing tasks has been attained using pre-trained language models, such as BERT. These models are essentially very large neural networks trained entirely unsupervised using open-domain data and based on bi-directional transformer architectures. Deep contextual information can be captured by the stacked self-attention modules of the transformer designs, and the training can scale to include a lot of open-domain data thanks to their non-recurrent structures. More notably, many pre-trained language models

In : $D = (source, target)$
$E = (label\ name :\ Observed\ Entities)$
$G = (label\ name :\ Gazetteer\ Entities)$
$V = E + G$
$Exs = Human\ example\ sentences$
$B = (source, empty\ target)$
γ: augmentation greediness
ϵ: decay rate
ρ: synthesize ratio
Func *Synthesizer(D, E, Exs)*
return B
Out: $D' = Augmented\ dataset$
begin
Initialize Transition Probabilities ($\tau \in T$) based on
$freq_{ij}$ = Frequency of j^{th} entity at i^{th} sentence in B
$\hat{freq}_{ij}$ = Pre-set frequency of entities that will be pulled
$P = Inversed(freq/n)$ # Probability of entities being pulled
$\tau_{B_i(E_j,\ E(j+n))}$ = Conditional probability of $(j+n)^{th}$ entity appearing when j^{th} entity was pulled previously
C_i = Set limit of augmentation for each sentence B_i with $(size(D), \gamma, \rho)$
for $i \in size(B)$ **do**
$counter = 0$
while $C_b < counter$ **do**
$b = B[i]$
$\hat{e}_0$ = pulled entity based on $P_{(b,\ e_0 \ldots e_n)}$
$\hat{b} = Fill(b, \hat{e}_0)$
for $blank \in blanks$ **do**
$\hat{e}_k$ = pulled entity based on $\tau_(b,\ \hat{e}_{(k-1)})$
$\tau \leftarrow$transition prob * ϵ
$\hat{b} = Fill(b, \hat{e}_k)$
end
D $\leftarrow$ Append()
$counter = counter + number\ of\ blanks$
end
end
end

Algorithm 1: DA's pseudocode(Hubert's Transition based entity swapping)

have been publicly accessible online which helps in not needing to train them from scratch.
Furthermore, BERT is capable of handling sequences up to 512 tokens long, and while the vast majority of both Hubert's and CoNLL03's paragraphs are below that value, some sequences are longer than 512 tokens. In addition, the padding strategy of the WordPiece Tokenizer, splits the following string "Jim Henson was a puppeteer" into ['Jim', 'Hen', 'son', 'was', 'a', 'puppet', 'eer']. This creates flexibility, as the tokenizer can always create tokens for a given sequence, regardless if the word has been seen previously by the model. This is especially useful for NER as some names may be very unusual and not occur in the training dataset.
Considering these we implemented BERT for pre-trained model of our NER network. The model is written in `pytorch` framework which alongside allowing for ingesting BERT model weights, also include several other auxiliary functions used during pre-training or handling hardware acceleration. Additional consideration include adding a dropout layer for regularization and overfitting prevention. A visualisation of the implemented network is displayed in Figure 4.4.
For loss function, we use Cross-entropy loss, which is builds upon the idea of information theory entropy and measures the difference between two probability distributions for a given random variable/set of events. Cross-entropy loss is designed to measure the performance of a classification model whose output is a probability value between 0 and 1. As it is evident in equation 4.2, the significance of this loss function is that the penalty is logarithmic, yielding a large score for large differences close to 1 and a small score for small differences tending to 0.

$$\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{4.2}$$

4.6 Experiments and results

4.6.1 Experiment settings

A ML model is simply a mathematical model that learns its parameters by training process. Hyperparameters, on the other hand, are a different class of parameter that cannot be directly learned through routine training. Usually, they are fixed before the start of the training itself. These parameters describe crucial model characteristics including complexity and learning speed.
To provide a fair basis for comparison, we kept the same parameters and input configuration for all the models. Having an understanding of how a model performs across different settings enables a better design process, as hyperparameters can be a problematic aspect of fine-tuning a model. A total of 32 different and the most performant results for each model are reported below.

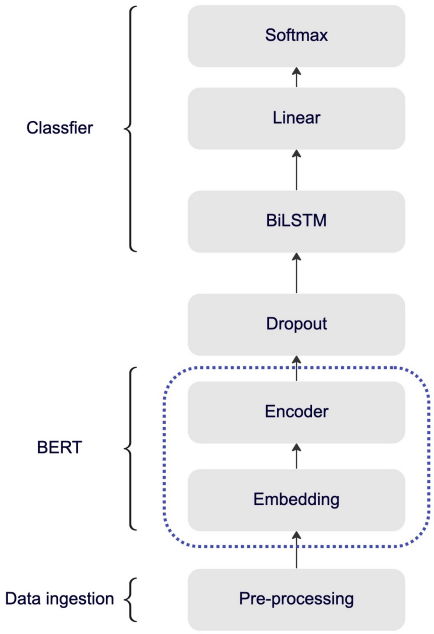

Figure 4.4: NER network visualization.

Additionally, experience from [7] showed that a too large learning rate can introduce problems of convergence. All models have been subjected to experiments on the same datasets and label sets and were trained using an NVIDIA RTX 2070 GPU. Batch size must be a multiple of eight, as required by the GPUs. The final fixed parameters of our experiments are as follows[1]:

- pre-trained model: bert-base-cased
- synthesize ratio: 35
- augmenter greediness: 10
- epochs: 17
- learning rate: 5e-05
- batch size: 16
- dropout: 0.2

[1]The experiment results for hyperparmeter tuning is presented in appendix

4.6.2 Evaluation method

According to the evaluation scenarios presented in section 2.1.3, more complex evaluation metrics have been introduced by the Message Understanding Conference (MUC) and the International Workshop on Semantic Evaluation (SemEval) that analyze at a full-entity level and can measure the performance for correct, incorrect, partial, missed and spurious predictions in different ways. So, they can address not only the first three but all the scenarios.
However, as evidenced by the literature in our scope, all benchmark NER research works continue to evaluate their performance using the Computational Natural Language Learning method, and because we are comparing our work to those SotA NER models, we must also comply with their evaluation method(i.e Standard method). The implementation of this evaluation method has been performed by using `seqeval 1.2.2` testing framework.

4.6.3 Results

The models were evaluated on both the validation and test sets. Doing this enables a comparison of both the performance of the classifier and the robustness of the models for different hyperparameter settings. According to the explanations with regard to the evaluation method, we follow the standard method and report on precision, recall, and f1-score at a token level.
Highlighting the **Overall Results** on Table 4.4, The transformer-based models significantly outperform the other models with regards to recall in the fine-grain NER task. In fact, both BOND and OG-NER significantly outperform BiLSTM-CNN-CRF in both domains.
This difference in recall scores also explains the higher F1 scores for the transformer-based models. On the other hand, BiLSTM-CNN-CRF shows its strength in terms of precision, and this model acts as a trade-off between CRF and the transformer-based models. Where this drived us to employ BiLSTM-Linear with drop-out into our network.

Furthermore, we observe improvement when introducing new label sets and training data with respect to different domains in all investigated models. While this can be the result of switching from a wider domain to a more concentrated one, it can also suggest that the transformer-based model's language understanding may not be outstandingly different than rule-based methods(i.e., they would not systematically perform well/badly for the same domains).
It is worth noting that the KB-Matching method of BOND suffers from lower precision and lower recall for labeling fine-grained entities, which greatly limits the performance of the NER methods that use KB-Matching for distant supervision.
Since limited benchmarks are available for this fine-grained NER task, we have

evaluated our work by simulating the same experimental configurations on available benchmarks to create a fair environment for comparison.
The results indicate that incorporating a semantically populated KG of relevant named entities and deploying them through the DA method and semi-supervised training leads to considerable improvement of the NER network for recognizing fine-grained entities. Table 4.2 visualise a sample of how different models perform tagging of named entities. Table 4.3 shows the ablation study on impact of KG enrichment on NER performance which show the significance if a rich KG on NER performance.

Sentence #1	...I am currently studying for my **Master's**$_{\texttt{[EDULVL]}}$ in **Language Technology**$_{\texttt{[EDUPROGRAM]}}$.
BOND	...I am currently studying for my **Master's**$_{\texttt{[O]}}$ in **Language Technology**$_{\texttt{[O]}}$.
OG-NER	...I am currently studying for my **Master's**$_{\texttt{[EDULVL]}}$ in **Language Technology**$_{\texttt{[EDUPROGRAM]}}$.
OG-NER$_{base}$	...I am currently studying for my **Master's**$_{\texttt{[EDULVL]}}$ in **Language Technology**$_{\texttt{[INS]}}$.
Sentence #2	...I am going to study **SFI**$_{\texttt{[EDUPROGRAM]}}$ in **lund university**$_{\texttt{[INS]}}$.
BOND	...I am going to study **SFI**$_{\texttt{[O]}}$ in **lund university**$_{\texttt{[INS]}}$.
OG-NER	...I am going to study **SFI**$_{\texttt{[EDUPROGRAM]}}$ in **lund university**$_{\texttt{[INS]}}$.
OG-NER$_{base}$	...I am going to study **SFI**$_{\texttt{[EDUPROGRAM]}}$ in **lund university**$_{\texttt{[INS]}}$.
Sentence #3	...I have **certifikat in degree**$_{\texttt{[EDULVL]}}$ for **bilolycka**$_{\texttt{[EDUPROGRAM]}}$ but kno I am studieg studieg\u00e5ng **SFI**$_{\texttt{[EDUPROGRAM]}}$ the whole week.
BOND	...I have **certifikat in degree**$_{\texttt{[O]}}$ for **bilolycka**$_{\texttt{[O]}}$ but kno I am studieg studieg\u00e5ng **SFI**$_{\texttt{[INS]}}$ the whole week.
OG-NER	...I have **certifikat in degree**$_{\texttt{[EDULVL]}}$ for **bilolycka**$_{\texttt{[EDUPROGRAM]}}$ but kno I am studieg studieg\u00e5ng **SFI**$_{\texttt{[EDUPROGRAM]}}$ the whole week.
OG-NER$_{base}$	...I have **certifikat in degree**$_{\texttt{[EDULVL]}}$ for **bilolycka**$_{\texttt{[O]}}$ but kno I am studieg studieg\u00e5ng **SFI**$_{\texttt{[EDUPROGRAM]}}$ the whole week.

Table 4.2: Examples showing how OG-NER improves the fine-grained NER performance. The ground truth labels are in green, model's correct predictions are in blue and wrong predictions in red.

Configuration	KG enrichment source	Prec	Rec	F1
no KB	-	53	66	59
Weak KB	Scrapped data	60	74	66
Strong KB	Wikidata	93	95	94

Table 4.3: Ablation study on impact of KG enrichment on NER performance.

Learning method	Model	Data compensation method	Labelset 1[2]			Labelset 2 [3]		
			Prec	Rec	F1	Prec	Rec	F1
Unsupervised	SpaCy-lg	Prodigy annotator	45	43	44	41	23	30
Distant-supervised	AutoNER	Tie-or-break tagging	53	55	54	51	67	58
	BOND	KB Matching + self-training	45	58	50	60	69	64
Semi-supervised	OG-NER	KB + Data Augmentation	86	88	**87**	93	95	**94**

Table 4.4: Overview of experimented results on NER models.

[2] **Labelset 1(CoNLL03):** `[PER]`, `[ORG]`, `[LOC]`, `[MISC]`.

[3] **Labelset 2(Educational background):** `[INS]`, `[EDUPROGRAM]`, `[EDULVL]`, `[DURATION]`, `[LOC]`.

Chapter 5

Conclusions

In this document, we presented the research and experimentation of OG-NER, a framework that is capable of improving fine-grained NER using its data augmentation method. The system is trained both on a specific domain of educational background and also on the benchmark open domain of newswire(Conll) for comparison with SotA.
Along with the neural network architecture, we propose a unique method with the objective of training a model that can be easily re-trained and fine-tuned with limited preemptive human effort. The data gathered from open-source online KBs plays an important role in the implications of successful applications. We highlight the lower efficiency of benchmark NER models when being exposed to a new domain and new labels with limited annotated data, even though they seem to perform relatively acceptable in open-domain.
The work proposed for this thesis project is also motivated by the fact that although fine-grain NER still generates a lot of coverage in research, it usually lacks the flexibility to produce convincing results in newly introduced domains. Furthermore, the literature is focused on KB-matching and distant-supervised learning, which rely on the use of heavy models to partially solve problems.
Our approach introduces new way to utilize KGs to adequately leverage the gathering of new named entities from KBs and perform semi-supervised learning to avoid the DSL and domain adaptation challenges.
Following the experimentation, it was evident that our method requires less contribution from experts in its fine-tuning and new domain adjustments compared to benchmark studied NER models. To demonstrate the usefulness of performing the ontology-guided graph enrichment technique, we initialized our experiments with provided chat interview data by "Anna and Hubert labs" regarding analysis of their applicant's educational background.
We conclude that, while the SotA has proposed outstanding fine-grain NER models, they cannot fill the gap when being exposed to new domains and a limited amount

of annotated data. We proposed a proof of concept study focusing on the ability to switch to a new domain with limited effort for re-training the network and achieve acceptable results.
Additionally, the proposed data augmentation approach performs noticeably well, implying the possibility of deploying this model successfully for real-world tasks in the NER field. The use of suitable metrics and architectures, geared towards the application domain, holds an important role in comparison of the above-mentioned performances.

5.1 Future work

Explore new and complex domains and labelsets - The different use cases determined by the company had the goal of evaluating an applicant's background in mind. While it's an important use case, it may not be the ideal case on which to focus all our attention. We can investigate our model's performance in recognizing scientific named entities like chemistry or medicine domains.
Different method for selection of most common sentences - As it can be observed by the experiments, the word frequency and n-grams metric are not the best indicators for judging the most common sentences in a context domain . As discussed in the introduction of these metrics, they are based on statistics and more complex ML-assisted methods can be utilized to enhance them.
Different method for generating blank sentences - Our next in-line step is to investigate alternative frameworks used to automate sentence generation and configure those for generating blank sentences. An example of utilizing this approach is to first adopt transformer models like GPT-2 to generate the sentence and then use graph embedding to determine the target entities' positions and replace them with blank tags to be filled with named entities later in the augmenter.
Different NER network architecture - In our work, the original BERT-cased model has been studied and validated on datasets. Since this field is evolving swiftly with the advent of transformers and language models, detailed investigation of the novel approaches of different transformer-based language models like T5, ELECTRA, etc.

.1 Appendix: Hyperparameter tuning

With respect to the explanations in Section 4.4.2, it is important to find the correct combination of parameters to further tune the data augmenter and its impact on the NER performance. Considering the Synthesize_ratio determines the ratio of synthesize data compared to training data we and Augmenter_greediness sets the number of maximum limit of augmented sentences after up-sampling calculation. These hold the balance of correct usage of data augmenter in order to improve the performance while avoiding the overfitting of the model. A visualization of the attempt to find the best parameters is provided below.

Parameters			**Results**		
Epoch	**Synthesize_ratio**	**Augmenter_greediness**	**Prec**	**Rec**	**F1**
10	20	3	60	74	66
10	30	3	68	79	74
15	30	3	73	82	77
15	40	3	71	80	76
15	35	3	70	81	75
15	35	3	69	80	74
20	30	3	74	83	78
10	30	3	72	82	77
5	35	3	79	80	79
17	35	5	83	87	85
20	35	5	82	86	84
19	35	5	81	86	83
18	35	5	81	87	84
17	35	5	82	86	84
16	35	5	82	86	84
15	35	5	82	87	85
15	30	10	87	89	88
15	35	10	90	92	91
17	35	10	92	95	93

Table 1: Hyperparameter tuning of DA

Bibliography

[1] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning : from theory to algorithms*. 2014. ISBN: 9781107057135 1107057132. URL: http://www.worldcat.org/search?qt=worldcat_org_all&q=9781107057135 (cit. on p. 7).

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org (cit. on pp. 7, 26).

[3] Mihan Gupta. *A Review of Named Entity Recognition (NER) Using Automatic Summarization of Resumes*. URL: https://towardsdatascience.com/a-review-of-named-entity-recognition-ner-using-automatic-summarization-of-resumes-5248a75de175 (cit. on p. 7).

[4] Matthew Honnibal and Ines Montani. «spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing». To appear. 2017 (cit. on pp. 8, 61).

[5] Hailin Zhang, Hai Zhu, Junsong Ruan, and Ruoyao Ding. «People Name Recognition from Ancient Chinese Literature Using Distant Supervision and Deep Learning». In: *2021 2nd International Conference on Artificial Intelligence and Information Systems*. ICAIIS 2021. Chongqing, China: Association for Computing Machinery, 2021. ISBN: 9781450390200. DOI: 10.1145/3469213.3470270. URL: https://doi.org/10.1145/3469213.3470270 (cit. on p. 9).

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. «Attention Is All You Need». In: (June 2017). URL: http://arxiv.org/abs/1706.03762 (cit. on pp. 8, 32, 33, 35).

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». In: (Oct. 2018). URL: http://arxiv.org/abs/1810.04805 (cit. on pp. 9, 10, 34–37, 69).

[8] Cedric Lothritz, Kevin Allix, Lisa Veiber, Tegawendé F Bissyandé, and Jacques Klein. *Evaluating Pretrained Transformer-based Models on the Task of Fine-Grained Named Entity Recognition*. 2020, pp. 3750–3760. URL: https://primer.ai/blog/a-new-state-of-the-art-for-named-entity-recognition/ (cit. on pp. 10, 66).

[9] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. «A Survey on Deep Learning for Named Entity Recognition». In: (Dec. 2018). URL: http://arxiv.org/abs/1812.09449 (cit. on p. 11).

[10] Erik F Tjong, Kim Sang, and Fien De Meulder. «Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition». In: (). URL: http://lcg-www.uia.ac.be/conll2003/ner/ (cit. on pp. 12, 13, 50, 54, 58, 61).

[11] Wikipedia. *Machine Learning*. URL: https://en.wikipedia.org/wiki/Machine_learning (cit. on p. 14).

[12] Valohai. *What is a Machine Learning Pipeline?* URL: https://valohai.com/machine-learning-pipeline/ (cit. on p. 14).

[13] Neptune. *Data Augmentation in NLP*. URL: https://neptune.ai/blog/data-augmentation-nlp (cit. on p. 17).

[14] Tianyu Liu, Jin-Ge Yao, and Chin-Yew Lin. *Towards Improving Neural Named Entity Recognition with Gazetteers*. 2019, pp. 5301–5307. URL: https://en.wikipedia.org/wiki/ (cit. on p. 18).

[15] Simone Magnolini, Valerio Piccioni, Vevake Balaraman, Marco Guerini, and Bernardo Magnini. *How to Use Gazetteers for Entity Recognition with Neural Models*. 2019. URL: https://github.com/XuezheMax/NeuroNLP2 (cit. on p. 19).

[16] Fabien Gandon. «A Survey of the First 20 Years of Research on Semantic Web and Linked Data». In: *Revue des Sciences et Technologies de l'Information - Série ISI : Ingénierie des Systèmes d'Information* (Dec. 2018). DOI: 10.3166/ISI.23.3-4.11-56. URL: https://hal.inria.fr/hal-01935898 (cit. on p. 20).

[17] Medium-Mohit Mayank. *A guide to Knowledge Graphs*. URL: https://towardsdatascience.com/a-guide-to-the-knowledge-graphs-bfb5c40272f1 (cit. on pp. 21, 23, 24).

[18] Ontotext. *what are ontologies*. URL: https://www.ontotext.com/knowledgehub/fundamentals/what-are-ontologies/ (cit. on p. 22).

[19] Yanji Chen, Mieczyslaw M. Kokar, and Jakub J. Moskal. «SPARQL Query Generator (SQG)». In: *Journal on Data Semantics* 10 (3-4 Dec. 2021), pp. 291–307. ISSN: 18612040. DOI: 10.1007/s13740-021-00133-y (cit. on pp. 23, 45, 46, 49, 65).

[20] Wei Shen, Jianyong Wang, and Jiawei Han. «Entity linking with a knowledge base: Issues, techniques, and solutions». In: *IEEE Transactions on Knowledge and Data Engineering* 27 (2 Feb. 2015), pp. 443–460. ISSN: 10414347. DOI: 10.1109/TKDE.2014.2327028 (cit. on p. 24).

[21] Xianpei Han and Le Sun. «A Generative Entity-Mention Model for Linking Entities with Knowledge Base». In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 945–954. URL: https://aclanthology.org/P11-1095 (cit. on p. 24).

[22] Ellen Riloff and Rosie Jones. *Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping*. 1999. URL: www.aaai.org (cit. on p. 39).

[23] Dekang Lin and Patrick Pantel. «Induction of Semantic Classes from Natural Language Text». In: (2001). URL: www.cs.ualberta.ca/~lindek/demos.htm. (cit. on p. 39).

[24] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. «Unsupervised Named-Entity Extraction from the Web: An Experimental Study». In: (2005) (cit. on p. 40).

[25] David Nadeau, Peter D Turney, and Stan Matwin. «LNAI 4013 - Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity». In: *LNAI* 4013 (2006), pp. 266–277. URL: http://balie.sourceforge.net. (cit. on p. 40).

[26] Antonio Toral, Rafael Muñoz, and Mu~ Muñoz. «A proposal to automatically build and maintain gazetteers for Named Entity Recognition by using Wikipedia». In: (2006). URL: http://simple.wikipedia.org/wiki/Portugal (cit. on p. 40).

[27] Jun ' Ichi Kazama and Kentaro Torisawa. «Exploiting Wikipedia as External Knowledge for Named Entity Recognition». In: (2007), pp. 698–707. URL: http://hyperestraier.sourceforge.net/index.html (cit. on p. 40).

[28] Will Radford, Xavier Carreras, and James Henderson. «Named entity recognition with document-specific KB tag gazetteers». In: (2015), pp. 17–21. URL: https://github.com/goldsmith/ (cit. on p. 40).

[29] Cedric Möller, Jens Lehmann, and Ricardo Usbeck. «Survey on English Entity Linking on Wikidata». In: (Dec. 2021). URL: http://arxiv.org/abs/2112.01989 (cit. on pp. 40, 55, 58).

[30] Gregor Titze, Volha Bryl, Cäcilia Zirn, and Simone Paolo Ponzetto. «DBpedia Domains: augmenting DBpedia with domain information». In: (2014). URL: http://en. (cit. on p. 40).

[31] Antonin Delpeuch. «OpenTapioca: Lightweight Entity Linking for Wikidata». In: (Apr. 2019). URL: http://arxiv.org/abs/1904.09131 (cit. on p. 40).

[32] Michael A. Hedderich, Lukas Lange, and Dietrich Klakow. «ANEA: Distant Supervision for Low-Resource Named Entity Recognition». In: (Feb. 2021). URL: http://arxiv.org/abs/2102.13129 (cit. on p. 41).

[33] Bhavana Dalvi, Sumithra Bhakthavatsalam, Chris Clark, Peter Clark, Oren Etzioni, Anthony Fader, and Dirk Groeneveld. «IKE - An Interactive Tool for Knowledge Extraction». In: *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*. San Diego, CA: Association for Computational Linguistics, June 2016, pp. 12–17. DOI: 10.18653/v1/W16-1303. URL: https://aclanthology.org/W16-1303 (cit. on p. 41).

[34] Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. «Cross-lingual Name Tagging and Linking for 282 Languages». In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1946–1958. DOI: 10.18653/v1/P17-1178. URL: https://aclanthology.org/P17-1178 (cit. on p. 41).

[35] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. «Distant supervision for relation extraction without labeled data». In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Association for Computational Linguistics, Aug. 2009, pp. 1003–1011. URL: https://aclanthology.org/P09-1113 (cit. on p. 41).

[36] William Hogan. «An Overview of Distant Supervision for Relation Extraction with a Focus on Denoising and Pre-training Methods». In: (July 2022). URL: http://arxiv.org/abs/2207.08286 (cit. on pp. 41, 42).

[37] Andrew Thomas Bimba, Norisma Idris, Ahmed Al-Hunaiyyan, Rohana Binti Mahmud, Ahmed Abdelaziz, Suleman Khan, and Victor Chang. «Towards Knowledge Modeling and Manipulation Technologies». In: *Int. J. Inf. Manag.* 36.6 (Dec. 2016), pp. 857–871. ISSN: 0268-4012. DOI: 10.1016/j.ijinfomgt.2016.05.022. URL: https://doi.org/10.1016/j.ijinfomgt.2016.05.022 (cit. on p. 43).

[38] Hainan Chen and Xiaowei Luo. «An automatic literature knowledge graph and reasoning network modeling framework based on ontology and natural language processing». In: *Advanced Engineering Informatics* 42 (Oct. 2019). ISSN: 14740346. DOI: 10.1016/j.aei.2019.100959 (cit. on p. 43).

[39] Krzysztof Janowicz and Carsten Keßler. «The Role of Ontology in Improving Gazetteer Interaction». In: *International Journal of Geographical Information Science fttontology-final-draft International Journal of Geographical Information Science* 00 (00 2007), pp. 1–24. DOI: 10.1080/1365881YYxxxxxxxx. URL: http://www.tandf.co.uk/journals (cit. on p. 43).

[40] Xuan Wang, Vivian Hu, Xiangchen Song, Shweta Garg, Jinfeng Xiao, and Jiawei Han. *CHEMNER: Fine-Grained Chemistry Named Entity Recognition with Ontology-Guided Distant Supervision*. 2021, pp. 5227–5240. URL: https://en.wikipedia.org/wiki/ (cit. on pp. 43, 44, 46, 55).

[41] Xiang Dai and Heike Adel. «An Analysis of Simple Data Augmentation for Named Entity Recognition». In: (Oct. 2020). URL: http://arxiv.org/abs/2010.11683 (cit. on p. 45).

[42] Leyang Cui, Yu Wu, Jian Liu, Sen Yang, Yue Zhang, Zhejiang University, and Microsoft Research Asia. *Template-Based Named Entity Recognition Using BART*, pp. 1835–1845 (cit. on p. 46).

[43] Huiming Zhu, Chunhui He, Yang Fang, and Weidong Xiao. «Fine Grained Named Entity Recognition via Seq2seq Framework». In: *IEEE Access* 8 (2020), pp. 53953–53961. ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2980431 (cit. on p. 46).

[44] Khai Mai, Thai-Hoang Pham, Nguyen Minh Trung, Nguyen Tuan Duc, Danushka Bolegala, Ryohei Sasano, and Satoshi Sekine. *An Empirical Study on Fine-Grained Named Entity Recognition*. 2018, pp. 711–722. URL: https://fgner.alt.ai/duc/ene/testsets/comp/en/ (cit. on p. 46).

[45] Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. *Learning Named Entity Tagger using Domain-Specific Dictionary*. 2018. URL: https://github.com/shangjingbo1226/ (cit. on pp. 47–49, 54).

[46] Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. «BOND: BERT-Assisted Open-Domain Named Entity Recognition with Distant Supervision». In: Association for Computing Machinery, Aug. 2020, pp. 1054–1064. ISBN: 9781450379984. DOI: 10.1145/3394486.3403149 (cit. on pp. 48–51, 54).

[47] Edward Loper and Steven Bird. *NLTK: The Natural Language Toolkit*. 2002. DOI: 10.48550/ARXIV.CS/0205028. URL: https://arxiv.org/abs/cs/0205028 (cit. on p. 49).

[48] Heartex. *Label Studio* (cit. on p. 58).

[49] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. «A Survey on Knowledge Graphs: Representation, Acquisition, and Applications». In: *IEEE Transactions on Neural Networks and Learning Systems* 33 (2 Feb. 2022), pp. 494–514. ISSN: 21622388. DOI: 10.1109/TNNLS.2021.3070843 (cit. on p. 58).

[50] Gonz Saavedra. *monkeylearn*. https://github.com/monkeylearn. 2020 (cit. on p. 61).

[51] Tony Rose, Mark Stevenson, and Miles Whitehead. «The Reuters Corpus Volume 1 -from Yesterday's News to Tomorrow's Language Resources». In: *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*. Las Palmas, Canary Islands - Spain: European Language Resources Association (ELRA), May 2002. URL: http://www.lrec-conf.org/proceedings/lrec2002/pdf/80.pdf (cit. on p. 61).

[52] D3. *Data-Driven Documents*. https://github.com/d3/d3/wiki. 2021 (cit. on p. 65).

[53] Aaron Reich, Jiaao Chen, Aastha Agrawal, Yanzhe Zhang, and Diyi Yang. *Findings of the Association for Computational Linguistics Leveraging Expert Guided Adversarial Augmentation For Improving Generalization in Named Entity Recognition*. URL: https://github.com/GT-SALT/ (cit. on p. 65).