universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Named Entity Recognition in government domain with reports from the City of Vienna Court of Audit“

verfasst von / submitted by

Soojeong Jang

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2023 / Vienna 2023

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 977
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Business Analytics UG2002
Betreut von / Supervisor:	Univ.-Prof. Dr. -Ing. Benjamin Roth, B.Sc. M.Sc.

Abstract

Today, a huge amount of unstructured data from various sources is available. Natural language processing (NLP) techniques using such data are applied to several tasks such as named entity recognition (NER). This study is designed to investigate the applicability of the NER task to the German language government domain. Nowadays, various government documents are digitized, creating opportunities to utilize the information in the text. We selected documents created by the City of Vienna Court of Audit as a data resource. We extract text from documents by using two methods hand-labelling and weak-labelling with skweak library. Using the data, we customize and fine-tune five SpaCy and BERT models that achieve high F1-scores with CoNLL03 German datasets. Then, we compare results by models and by data type which means the data generated by hand-labelling or weak-labelling. The highest F1-score was a 95.57% achieved by the SpaCy transformer xlm-roberta-base model by using hand-labelled data and weakly-labelled data together as a training dataset. Furthermore, we discuss the case that this model can be adapted to the real-world business problems.

Kurzfassung

Heutzutage sind enorme Mengen unstrukturierter Daten aus verschiedenen Quellen verfügbar. Techniken der Natural Language Processing (NLP), die solche Daten nutzen, werden für diverse Aufgaben wie die Named Entity Recognition (NER) eingesetzt. Diese Studie untersucht die Anwendbarkeit der NER-Aufgabe im Regierungsbereich der deutschen Sprache. Regierungsdokumente werden heutzutage in verschiedenster Form digitalisiert, was neue Möglichkeiten bietet, um die Informationen im Text nutzen zu können. Als Datenquelle wurden Dokumente des Stadtrechnungshofs Wien ausgewählt. Der Text der Dokumente wird mit Hilfe zweier Methoden extrahiert, dem "Hand-Labeling" und "Weak-Labeling" unter Verwendung der "skweak". Anschließend werden fünf SpaCy- und BERT-Modelle mit den Daten angepasst, die hohe F1-Scores mit den CoNLL03 German-Datensätzen erreichen. Die Ergebnisse werden dann nach Modellen und Datentyp verglichen, was bedeutet, dass die Daten entweder durch Hand-Labeling oder Weak-Labeling generiert wurden. Der höchste F1-Score lag bei 95,57 % und wurde durch das SpaCy-Transformator-Modell "xlm-roberta-base" erzielt, indem sowohl Hand- als auch Weak-Labeling-Daten gemeinsam als Trainingsdatensatz verwendet wurden. Zudem wird diskutiert, wie das Modell auf reale Geschäftsprobleme angewendet werden kann.

Contents

List of Tables

List of Figures

1 Introduction

Natural Language Processing (NLP) is a subfield of computer science in which machines learn, understand, and produce human language content using computer technology (Hirschberg & Manning, 2015). Nowadays, enormous amounts of unstructured data from a variety of sources are available, such as social media, e-mail, news articles, and conversation data (Shelar et al., 2020). NLP technologies that are using those data are applied in several tasks such as question-answering, information retrieval, information extraction, etc (Sazali et al., 2016).

The first basic technique and one of the most essential sub-tasks of NLP is named entity recognition (NER) (Sazali et al., 2016). NER identifies the references of designators, which are called named entities, from texts belonging to predefined semantic types such as people, locations, organizations, etc (Yadav & Bethard, 2019). Although early NER was designed with rule-based methods, machine learning such as Conditional Random Fields and Hidden Markov Model and deep learning including Recurrent Neural Networks and Transformers are employed yielding state-of-the-art performance (Li et al., 2020; Sun et al., 2018).

Although available models known for their successful performance pretrained with general texts such as newspaper articles are not always suitable for domain-specific NER tasks (Zöllner et al., 2021). Of interest in recent years has been the application of domain-specific NER due to the growth of digital information in various fields (Cariello et al., 2021) such as laws and medicines. Hence, models can be fine-tuned on the target domain corpus to perform better, but the scarcity of labelled data hinders this process. In order to overcome this problem, weakly-labelled data can be used (Syed & Chung, 2021; Yu et al., 2020).

In this context, this study is designed to investigate the applicability of NER tasks for the government domain in German. The objective of the study is to identify the referred organization in reports issued by the City of Vienna Court of Audit ("Stadtrechnungshof Wien") for business usage. From the research objective, the main research question (RQ) arises as follows:

What is the best NER model to detect governmental organization entities (ORG)?

In order to find out an answer for the main research question, we create our own dataset from the City of Vienna Court of Audit and the five models are customized and fine-tuned on the dataset. The best model achieves a 95.57% F1-score.

The detailed sub-questions are developed and enlisted in section 1.2. The remainder of the paper is organized as follows. Chapter 2 elaborates on the theoretical background of the experiment including NER and Weak supervision. The state-of-the-art models are explained as well. In chapter from 3 to 5, details of the setups of experiments are provided. The results of the experiments are presented in chapter 6. Finally, in Chapter 7, conclusions of the research and opportunities for future research are provided.

1.1 Motivation

Currently, various data are being digitized, and in line with this trend, documents from the Vienna government are also digitized. In the past, it was not easy to utilize hand-written or printed documents as data. Now, there is a chance to conduct various tasks such as natural language processing on digitized documents. However, these documents are not used in any other way than posting this content on their website. Therefore, we want to get further interpretations from government reports and seek new directions to gain insights from them.

Reports of the City of Vienna Court of Audit are chosen as the source of the analysis since those reports contain content about finance, which is one of the most important parts of the government. There could be numerous pieces of information in their reports, but here we focus on organizations referred to in the documents. This is because the report includes audits of organizations that are financially related to the government and other companies that are related to the agenda of each document. Therefore, exploring the organizations and the number of times those are mentioned gives information on the importance of the organizations and an overall view of the network of the associated organization. By finding organizations from documents, we would improve the management and information retrieval of the documents and it is expected that the relationship between organizations that were not clearly visible can be captured.

Therefore, this study aims to derive the best NER model to detect organizations (ORG) from the corpus.

1.2 Task definition

The experiment starts with getting the data from the text source and training models with the data. Then the ORG entities are extracted from the text source, and the relationship between entities will be represented as their co-occurrence in a document. All results will be represented in the graphs. The research questions (RQs) formulated according to the steps of the experiments are as follows:

RQ1. What is mor effective way to get the training dataset?

RQ2. Which model shows the best performance for domain-specific NER?

RQ3. How can the results be adapted in real-world business situation?

RQ1 is discussed in chapter 3 including the source of raw data, and the method for labelling. For the RQ2, the models and the experiments are explained in chapters 4 and chapter 5, and the best model is described in chapter 6 with the result of the experiments. In the case of RQ3, a tool for visualization and case study is provided in chapter 5 as well.

2 Theoretical Backgrounds

This chapter elaborates on the theoretical background of machine learning, named entity recognition, and weak supervision for experiments of this research. The first part of this chapter briefly discusses machine learning and deep learning. The second part describes the definition and approaches of NER as a basic understanding. Then, tools of weak supervision and related works are introduced in the last part of this chapter.

2.1 Machine learning

Machine learning, one of today's fastest-growing technology fields, is at the heart of data science (Jordan & Mitchell, 2015). Due to the fast-growing 'big data' phenomenon, the ability of network mobile computing systems to generate, collect and transmit vast amounts of data, machine learning has widely adapted in a number of data-intensive fields to mining insights, predictions, and decisions from these datasets (Jordan & Mitchell, 2015; Qiu et al., 2016).

2.1.1 Introduction of Machine Learning

Machine learning is a field of research about the theory, performance, and properties of learning systems and algorithms (Qiu et al., 2016). In this context, learning refers to the improvement of algorithms based on ‘experiences’ without external assistance (Das & Behera, 2017). Machine learning algorithms can be classified into supervised and unsupervised learning according to the experiences the algorithms are allowed to have in the learning process (Goodfellow et al., 2016).

Supervised learning requires a dataset consisting of input features and corresponding target features, called labels (Goodfellow et al., 2016; Mahdavinejad et al., 2018; Qiu et al., 2016). Its objective is to predict the appropriate output for a given input (Mahdavinejad et al., 2018), There are famous supervised machine learning algorithms such as decision trees(Quinlan, 1986) and Support Vector Machines(Cortes & Vapnik, 1995). Decision trees are tree-like models consisting of the chosen output of the model in

their nodes (Mahesh, 2020), and Support Vector Machines are binary classifiers dividing hyperplane between both classes with maximum margin (Mahdavinejad et al., 2018).

On the other hand, in the case of unsupervised learning, the training set has no labels (Qiu et al., 2016). The major object of unsupervised learning is clustering, grouping data points based on their similarity (Mahdavinejad et al., 2018), and one of the clustering algorithms is K-Means clustering(MacQueen, 1967) that combines data close to one of the k centres into one cluster to create k clusters (Mahesh, 2020). Furthermore, there is weak supervision that is using a small amount of labelled data and a lot of unlabelled data for the training model. This is covered in detail in section 2.3.

There is a fundamental challenge in machine learning called overfitting (Goodfellow et al., 2016; Ying, 2019). An algorithm should be generalized which means that it must perform well on not only unseen input but also the own dataset used for training (Goodfellow et al., 2016). There are several techniques to avoid overfitting such as regularization, dropout, and early stopping. L1 regularization is utilized by adding the absolute value of the weight to cost function, and L2 regularization adds the squared value of the weight so that the weight is learned in a direction that is not too large. Dropout reduces interdependency between units by ignoring output of part of network units. At last, early stopping is a setting a trigger to stop training when the validation loss stops decreasing.

2.1.2 Deep learning

One of the remarkable advancements in learning algorithms is the evaluation of artificial neural networks summarized as deep learning (Goodfellow et al., 2016). Deep learning is a subfield of machine learning. It is called deep because it uses multiple layers in a neural network performing a mathematical operation on its input to map the non-linear relationship between input and output (Hodnett, 2019). A neural network is a series of algorithms mimicking the way the system of neurons in the human brain operates to recognize relationships in the dataset (Mahesh, 2020).

The basic architecture of the feedforward neural network is shown in Figure 1. The process called forward propagation is that information feed into the model as inputs

(x) and through the intermediate computations in the hidden layer, it finally yields output (y) without feedback connection (Goodfellow et al., 2016). In Figure 1, the circles are nodes, and the lines are connections between nodes (Hodnett, 2019). The input layer takes the data from the network, and the hidden layer process received information from the input layer (Imran & Alsuhaibani, 2019). An output of one node is computed as a matrix operation on the input values to the node and the weights (Hodnett, 2019) which is denoted by a symbol ∑. By applying a function called activation function to this output value, represented by *f* in the figure, nonlinearity is introduced to determine whether a neuron is activated or not, and the result is passed to the next node until the output node (Hodnett, 2019). At last, the model calculates error by using the loss function to compute the difference between the actual value and predicted value by the model for measuring its performance.

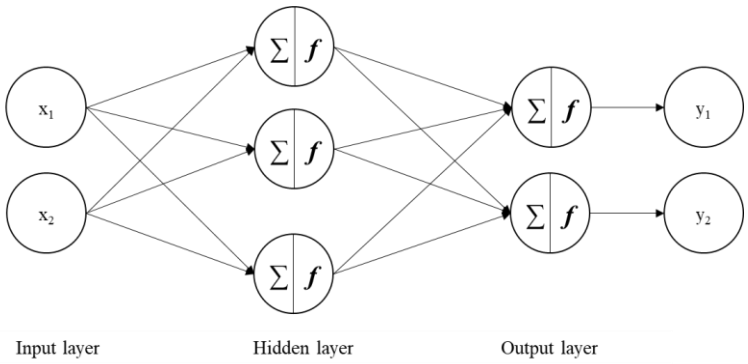


Figure 1. A neural network architecture
(source: Own illustration based on Zaccone (2018))

If the forward propagation is directed from the input layer to the output layer, the backpropagation updates the weight by calculating from the output layer toward the input layer. Learning in artificial neural networks refers to repeating forward and backward propagation with the aim of finding weights that minimize errors (Cole, 2018).

2.1.3 Natural Language Processing and deep learning

The concept of neural networks rooted in artificial intelligence is quickly gaining popularity (Mahesh, 2020), in particular, deep learning has made great strides in solving problems in various tasks of natural language processing(NLP) such as topic

classification, emotion analysis, question and answering, and language translation (LeCun et al., 2015). Deep learning frameworks outperform other traditional machine learning approaches such as SVM and logistics regression (Young et al., 2018). Therefore nowadays, deep learning architectures such as Recurrent Neural Networks (RNNs), Long Short Term Memory (LSTMs), and transformers are used widely for NLP tasks.

RNNs Recurrent Neural Networks (RNNs) are processing input and output in a sequence of values (Goodfellow et al., 2016). Unlike the feedforward neural network where information is passed through the hidden layer and the output layer, RNNs send the result from the activation function of the current state to the next state in the hidden layer. Therefore weights are shared by all time steps, and outputs at a time step are indirectly dependent on all previous time steps (Liu, 2019). This provides a short-term memory for the model that helps the model more accurately determine the next data and preserve the previous information (Kumar Tiwari, 2021). This is why RNNs are widely used in NLP problems dealing with sequential data such as named-entity recognition (NER) and part-of-speech (PoS) tagging (Liu, 2019).

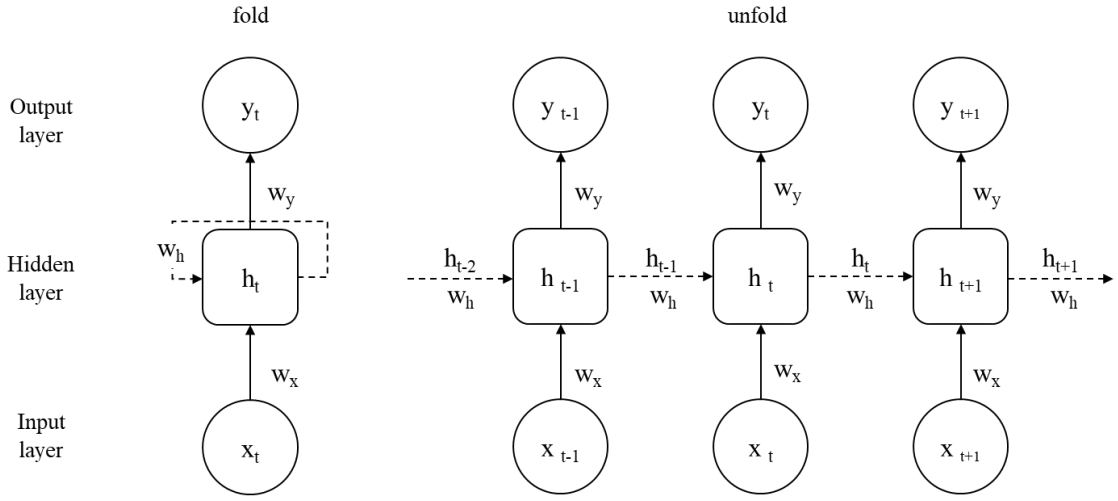


Figure 2. A recurrent neural network architecture
(Source: own illustration based on Goodfellow et al. (2016), Liu (2019))

LSTMs Basic RNNs are effective only for relatively short sequences because as step t is longer, the information cannot be passed backward sufficiently. Gradient descent and backpropagation are used during training RNN (Menshawy, 2018), the gradient may gradually decrease as you go to the input layer during the

backpropagation process, and the weights are not updated properly in layers close to the input layer. Long Short-Term Memory (LSTM) networks are designed to overcome the vanishing error problem (Hochreiter et al., 2001).

LSTM architectures are similar to RNNs, but the hidden layer is substituted as a memory cell (Huang et al., 2015), and the cell status is added depicted as in Figure 3. In order to update hidden state and cell state value, three gates, erasing unnecessary memories and deciding what to remember, are used each called the input gate, output gate and, forgot gate. LSTMs are not affected by the disappearance of gradients during the training procedure (Hochreiter & Schmidhuber, 1997), therefore they are more suitable for finding and exploiting long-range dependencies in the data such as long sentences or paragraphs (Huang et al., 2015).

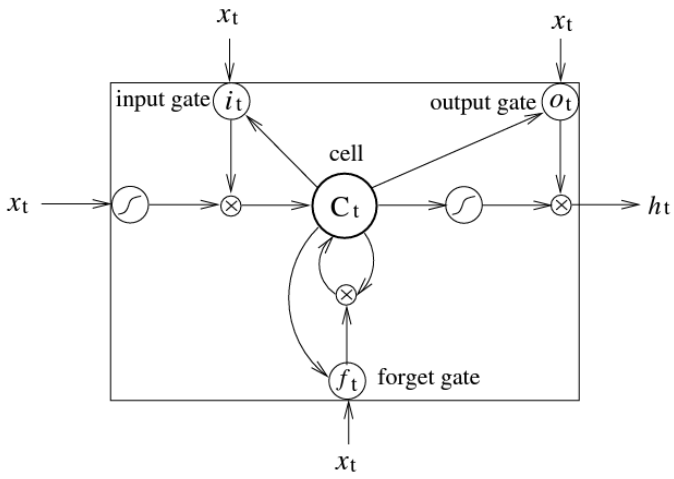


Figure 3. A Long Short-Term Memory Cell.
(Source: *Huang et al. (2015)*)

Bidirectional LSTM Bidirectional LSTM (BiLSTM), a model proposed by Graves et al. (2013), consists of two LSTM layers, one takes input forward direction (forward LSTM) and the other takes input backward direction (backward LSTM) (Huang et al., 2015). Compare to a simple LSTM, BiLSTM catches context information from past and future (Shaalan et al., 2017). One of the commonly used variations of BiLSTM is BiLSTM-CRF which is a model adds Conditional Random Fields (CRF) layer on the top of the BiLSTM model for using tag information of neighbours to predict the current tag more precisely (Huang et al., 2015).

Transformers Vaswani et al. (2017) proposed a transformer based entirely on attention mechanisms. The transformer model is designed to process sequential data such as natural languages, however, unlike RNNs and LSTMs, the transformer processes the entire input at once parallelly. The attention mechanism provides contexts for all positions in the input sequence. It reduces training time and allows train models on larger datasets (Vaswani et al., 2017).

Attention is the process of finding a key (K) that has a value similar to a query (Q) and obtaining that value (V). In other words, the attention functions compute the similarity with all keys for a given query respectively and map each value to the key using this similarity as a weight. Through this, it is possible to calculate which words are highly related to other words in the input sentence.

The transformer has an encoder-decoder structure as shown in Figure 4. In order to input a sentence into a transformer, the sentence is converted into a vector that the model can compute. In addition, the transformer does not receive word input sequentially, therefore it is necessary to inform the location information of the word. For this purpose, location information is added to the embedding vector of each word and used as input to the model, which is called positional encoding. Even if it is the same word, the embedding vector value depends on the location in the sentence.

The encoder is composed of a stack of 6 identical layers containing multi-head attention layers and feed-forward neural networks. Residual learning is used to prevent learned information from being lost in the data processing process, and a normalization process is performed. After processing 6 stacks of layers sequentially, the output of the last encoder layer is transferred to every decoder layer.

The decoder is similar to the encoder, however, look ahead mask is required in the decoder. Since the input is not sequential, words that are not used in inference are masked to prevent the model from cheating. This allows the model to pay attention only to words from previous locations when trying to predict the next token. The vector output of the decoder must be converted to the final output. One fully connected layer converts the decoder output into a matrix of logits, and the softmax function converts it into a probability distribution for the words.

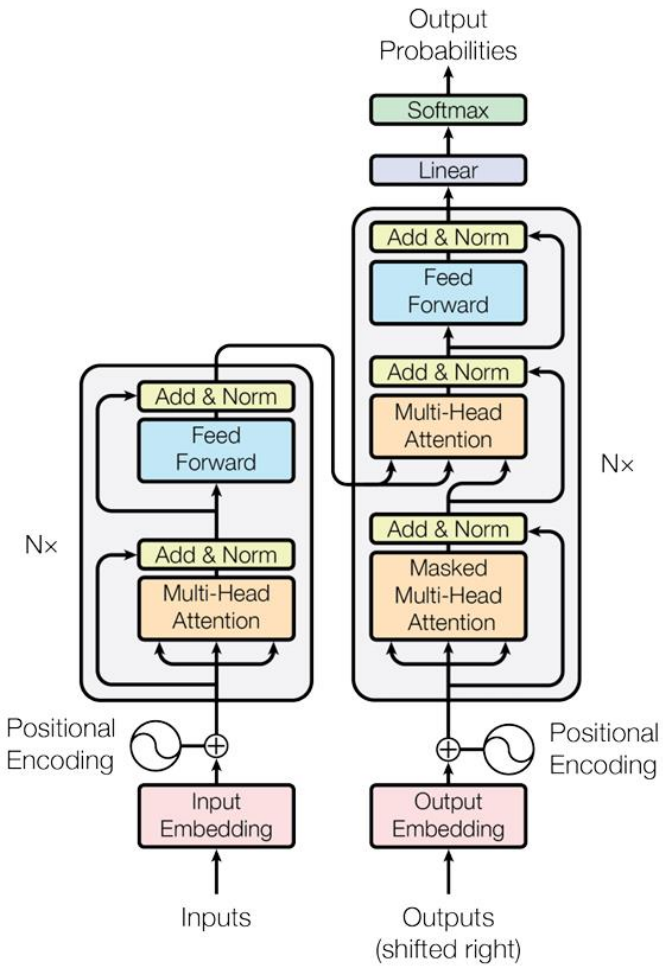


Figure 4. The Transformer model architecture
(source : (Vaswani et al., 2017))

BERT Devlin et al. (2018) proposed BERT(Bidirectional Encoder Representations from Transformers) which is a pre-training language model with unlabelled large text that can be used for various NLP tasks only through simple fine-tunning steps without designing substantial task-specific architecture as shown in Figure 5. For pre-training, BERT used two methods called Masked Language Model (MLM) and Next Sentence Prediction (NSP) that allow BERT to be learned to better understand the context.

The Masked Language Model (MLM) is the task of predicting a word given a series of words. Instead of predicting all the next tokens, few tokens are randomly masked from the input and put them in the transformer structure to predict the tokens that are

masked in the context of the surrounding words. By performing MLM, BERT develops the ability to understand the context of intra- sentence. The Next Sentence Prediction (NSP) is a method of predicting whether the second sentence is the one immediately following the first sentence in the learning process to understand the relationship in inter-sentences. (Devlin et al., 2018).

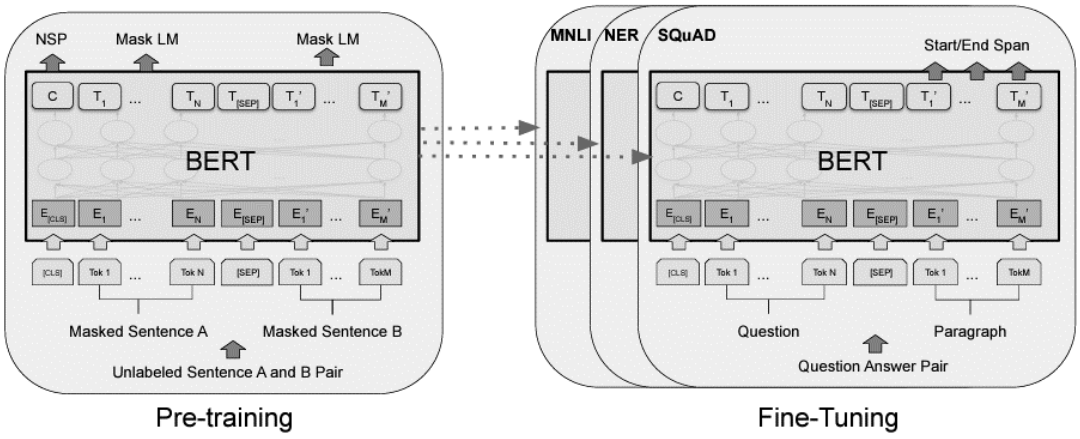


Figure 5. Overall pre-training and fine-tuning procedures for BERT.
Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers) (Source : *Devlin et al. (2018)*)

BERT uses only the encoder among the encoder and decoder of the transformer for pre-training. BERT can be trained with case-sensitive data or data converted to lowercase. Typically, there are 2 architectures of BERT according to its size, called $BERT_{BASE}$ and $BERT_{LARGE}$. Devlin et al. (2018) denote the number of layers as L, the hidden size as H, and the number of self-attention heads as A. Hyperparameters of those two architectures are as below (Devlin et al., 2018):

$BERT_{BASE}$: L=12, H=768, A=12, Total Parameters=110M

$BERT_{LARGE}$: L=24, H=1024, A=16, Total Parameters=340M

The self-attention mechanism is simple to fine-tune because BERT can model many downstream tasks by exchanging appropriate inputs and outputs. By plugging task-specific inputs and outputs into BERT, we can finetune all the parameters end-to-end and create state-of-the-art models (Devlin et al., 2018).

2.2 Named Entity Recognition

Named entity recognition is the process of identifying and classifying named entities such as person, location, and organization in text. The term 'Named Entity' was developed by the sixth Message Understanding Conferences (MUC-6) for information extraction tasks (Grishman & Sundheim, 1996). Since MUC-6, NER plays an important role as the first step of various NLP applications particularly information extraction, information retrieval, text summarization, question and answering, and machine translation (Li et al., 2020; Yadav & Bethard, 2019), because NER is a tool that identifies elements with specific semantics of interest for adding structures to unstructured data (Marrero et al., 2013).

NER system recognizes named entities from a sequence of tokens $s =< \omega_1, \omega_2, \dots, \omega_N >$ as Figure 6. The outputs of the system are tuples $< I_s, I_e, t >$, where I_s and I_e are the start and end index of the named entity tokens and t is the predefined categories like person or location (Li et al., 2020).

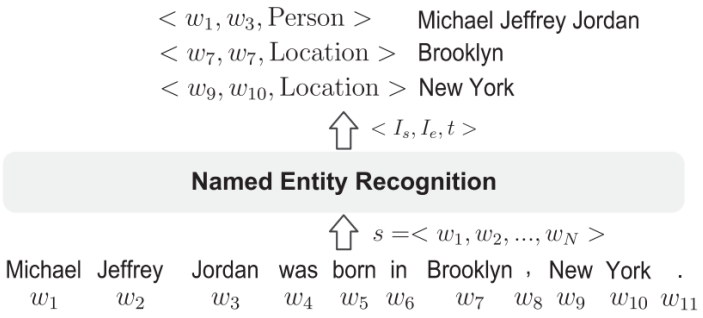


Figure 6. An illustration of the named entity recognition task
(Source : Li et al. (2020))

2.2.1 Approaches

Approaches for NER are commonly divided into three categories: Rule-based system, supervised learning, and neural networks (Goyal et al., 2018; Li et al., 2020; Nadeau & Sekine, 2007; Yadav & Bethard, 2019).

Rule-based systems Earlier systems are rule-based that rely on semantic and syntactic rules written by domain expertise. Rules can be generated based on patterns or syntactic structures in the text (Li et al., 2020). These systems usually show high precision, however, there are several disadvantages that they need domain experts for maintaining the knowledge resources and the recall is low due to the limitation of the domain-specific dictionary (Yadav & Bethard, 2019). Furthermore, a system made for one domain cannot be implanted into another domain (Goyal et al., 2018).

Supervised learning Replacing hand-crafted rules, supervised learning systems generate outputs with machine learning algorithms by using given annotated data as training inputs. There are applicable algorithms such as Hidden Markov Models (HMM), Decision Tress, Maximum Entropy Models, Support Vector Machines (SVM), and Conditional Random Fields (CRF) (Goyal et al., 2018; Li et al., 2020; Nadeau & Sekine, 2007; Yadav & Bethard, 2019).

Neural Networks Supervised learning systems show high performances, however, neural networks can be more effective because they can learn complex patterns in the data through a non-linear transformation (Chiu & Nichols, 2016; Li et al., 2020). Li et al. (2020) suggested the general architecture of recent deep learning in NER that consists of distributed representation for input, context encoder, and tag decoder as shown in Figure *7*. The input sentences are changed to vectors in dense format through word- or character-level embeddings. Encoders capture context dependencies using networks such as CNN, LSTM, and transformers. Tag decoders predict entity types (tags) of input sequences (Li et al., 2020).

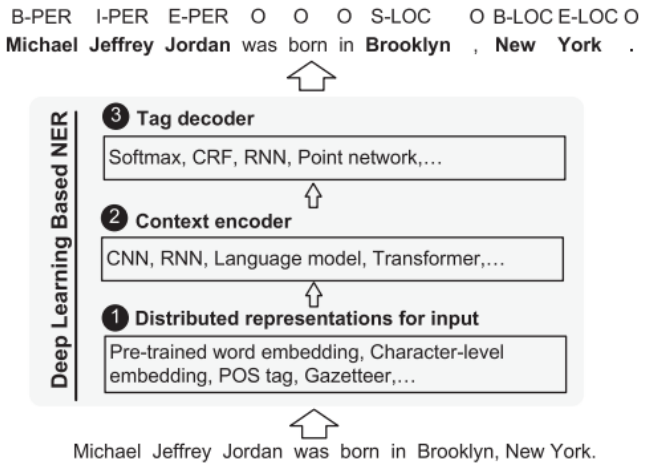


Figure 7. The taxonomy of DL-based NER
(Source : Li et al. (2020))

2.2.2 Related works

In this section, related works especially domain-specific NER are examined. All following studies creates their own dataset and train models such as BiLSTM and BERT with the dataset.

BIOfid dataset In 2019, Ahmed et al. (2019) launched biological data from historical scientific pieces of literature about plants, animals, and insects in German libraries for over the past 250 years. They annotated German text converted from printed text using optical character recognition (OCR) techniques. The final dataset consists of six types of named entities, person, location, organizations, time, taxon, and others, from 15,833 sentences. Then they used several neural models for NER such as BiLSTM, BERT. The best result of NER was an 80.23% F1-score with the BiLSTM-CRF model (Ahmed et al., 2019).

A Dataset of German Legal Documents Leitner et al. (2020) developed a dataset for NER in German federal court decisions. Data contains human-annotated 54,000 entities and the types of entities are 19: person, judge, lawyer, country, city, street, landscape, organization, company, institution, court, brand, law, ordinance, European legal norm, regulation, contract, court decision, and legal literature. They evaluated the dataset with BiLSTM models and achieved a 95.95% F1-score (Leitner et al., 2020).

A Dataset of Dutch Archaeology Domain Brandsen et al. (2022) tested a dataset that they created (Brandsen et al., 2020) to implement a text retrieval engine for a

large archaeological text collection. The source of data was DANS (Digital Archiving and Networked Services) in the Netherlands, a total of 15 documents were selected as annotation candidates (~42,000 tokens) (Brandsen et al., 2020). The dataset contains 43,000 annotated entities across six entity types: artifacts, time periods, locations, contexts, materials, and species. They fine-tuned BERT and custom SpaCy pipeline (see section 4.1.2), and get a 73.5% F1-score with a model named ArcheoBERTje (Brandsen et al., 2022).

2.3 Weak Supervision

Recently, there has been an enormous increment of online data and interest in machine learning systems, especially in the field of natural language processing and image analysis (Ratner et al., 2017). However, labelled data that is massively needed to train systems is scarce because it is expensive to create, and this phenomenon is exacerbated when the data is from uncommon languages or domains (Lison et al., 2021). In order to alleviate the problem of scarcity of labelled data, weak supervision which is a cheaper way to label data has emerged (Ratner et al., 2017).

Concretely, weak supervision is an approach in which often noisier sources of supervision are used to create large training sets, that are generated by noisy and heuristically with knowledge bases, patterns, rules, and classifiers, much quicker than manual labelling. Weak supervision is stronger than rule-based classification because it is possible to generalize for those similar to specified rules. It can be applied if domain-related resources exist, or label heuristics can be explained (Snorkel, n.d.).

2.3.1 Software frameworks

Several software frameworks have been released as tools for applying weak supervision to NLP tasks (Lison et al., 2021) as follows.

skweak A python toolkit for the use of weak supervision for NLP tasks could be applied to not only sequence labelling but also text classification. The general procedure consists of 4 steps: 1) applying labelling functions on texts, 2) aggregating results of labelling functions, and 3) training a discriminative model on the labels.

There are 4 categories for labelling functions: Heuristics that are detecting entities under pre-defined rules, gazetteers that are searching lists of words or phrases, machine learning models that are using transfer learning from trained models on data from others, and document-level functions that are relying on the document-level context to capture new supervision signals. The outputs of labelling functions are aggregated by using a statistical model such as the Hidden Markov Model that automatically estimates the relative accuracy of predictions of each labelling function (Lison et al., 2021).

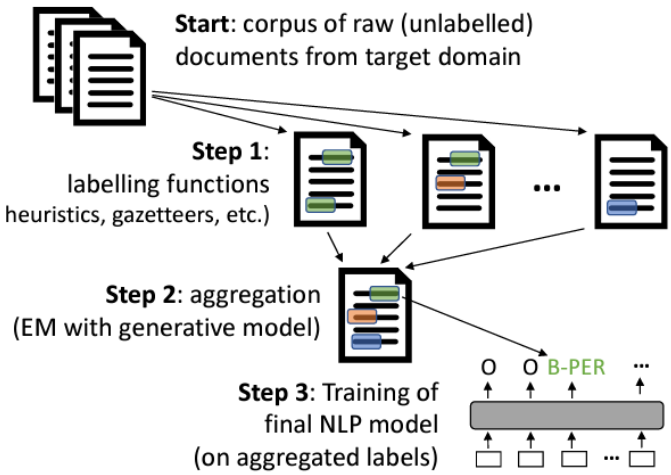


Figure 8. General overview of skweak
(Source : Lison et al. (2021))

Flying Squid According to Fu et al. (2020), it is a framework that uses triplets to learn the potential correlations of multiple noisy label sources without any ground truth data. A fast and accurate model can be generated because the parameters are obtained by a closed-form solution instead of relying on stoichiometric gradient descent. When learning the label model, parameters are computed by looking at all possible triplets and taking the mean or median, so it is to be more stable than just looking at a single triplet. This framework can be used for video analysis as well as NLP (Fu et al., 2020).

The pipeline of this framework is shown in Figure 9. Users build multiple weak supervision sources that assign noisy labels to data. these weak supervision sources can vote or abstain on individual data points; this lets users express high-precision signals. These sources are noisy and may conflict with each other, so a latent variable model, which we refer to as a label model, is used to express the accuracies of and correlations

between them. Once its parameters are learned, the model is used to aggregate source votes and generate probabilistic training labels, which are in turn used to train a downstream discriminative model (Fu et al., 2020).

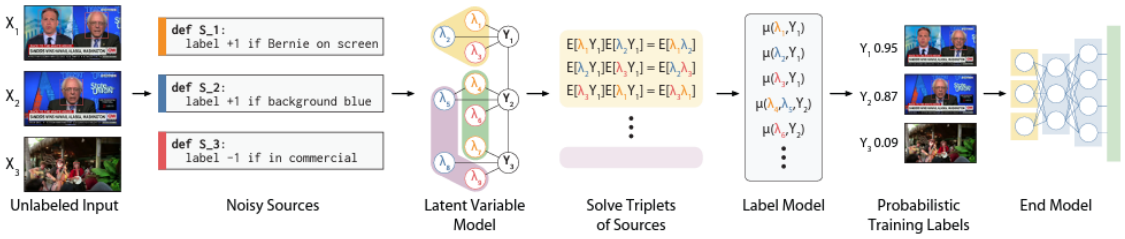


Figure 9. The FLYINGSQUID pipeline
(Source: Fu et al. (2020))

Snorkel It is an end-to-end system for combining weak supervision sources, offering a flexible interface layer for writing labelling for NLP and image/video classification tasks. There is an optimizer accelerating pipeline executions for deciding when to model the accuracy of modelling functions and when to skip learning in favour of a simple majority vote, and additionally, which of the labelling function are used to model correlation.

The overview of the Snorkel architecture is in Figure 10. The first step is to write labelling functions with weak-supervised sources, such as patterns, heuristics, and external knowledge bases, to assign labelling to unlabelled data programmatically. Next, Snorkel automatically learns a generative model over the labelling functions that compute accuracies of labelling functions and combine outputs into a single training label per data point. Then, the probabilistic labels are trained to derive a discriminative model that is generalized beyond the labelling functions, maintaining accuracy on unseen data.

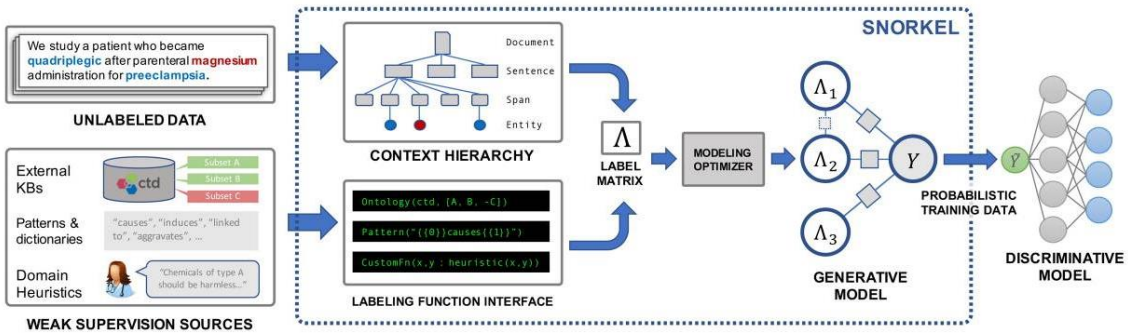


Figure 10. An overview of the Snorkel system
(Source : (Ratner et al., 2017))

Each software framework has a slightly different structure and operation; however, all frameworks help to learn the accuracies of weak supervision sources without access to ground truth using a generative model. In this study, skweak is used to get training data, and the detailed method is elaborated in Chapter 3.

2.3.2 Methods

Weak supervision can be applied to broad tasks, but we focus on recent research for NER tasks in this section. Zeng et al. (2020) proposed a method called 'Counterfactual Generator' which removes part of the fake correlation between the context representation and the output label. The NER model augmented with counterfactual examples yields results an average increase of 8.68 % points compared to when the method was not used. Wang et al. (2020) created dataset on the COVID-19 Open Research Dataset Challenge (CORD-19) corpus by using weak supervision. With their weakly supervised methods, new entity types were successfully recognized with only several seed examples. The work of Lison et al. (2020) presented a weakly supervised method using a hidden Markov model for producing probabilistic predictions from labelling functions and the final neural sequence model shows over a 70% F1-score at the entity level on the CoNLL03 dataset.

3 Data

In this chapter, we figure out the answers for the first research question, an efficient way to get the large training dataset. Raw data is text data extracted from the audit report of the City of Vienna Court of Audit (Stadtrechnungshof Wien). In order to use them for analysis, the texts were extracted from documents first, and training data was generated through hand labelling and weak labelling on a part of the extracted texts. The detailed description of the text is covered in sections 3.1 and 3.2, and the hand-labelling and weak labelling methods are explained in sections 3.3 and 3.4 respectively.

3.1 Text Source

In this study, an audit report issued by the City of Vienna Court of Audit was use. The City of Vienna Court of Audit is an external public audit institution in Vienna delegated by law to audit financial management and safety (Stadtrechnungshof Wien, n.d. -a). Its task is auditing the accuracy, regularity, and economic feasibility of financial management of municipality, funds, foundations, and institutions that have legal personality (Stadtrechnungshof Wien, n.d. -b).

The audit report is the result of an audit conducted by the City of Vienna Court of Audit and the full report can be found online (Stadtrechnungshof Wien, n.d. -c). The City of Vienna Court of Audit annually conducts audits based on internationally recognized auditing standards. A standard audit (Standardprüfungen) is basically 1) notice of audit of the facility under audit 2) conducting an audit (data collection and evaluation or evaluation) 3) commentary procedures, 4) submitting a final report to the city audit committee and 5) publishing.

According to the activity reports (Taetigkeitsbericht) issued by City of Vienna Court of Audit, the types of reports are 1) Audit report (Prüfungsberichte), 2) Statement by the audited organization (Stellungnahmen der gepruefteN stellen), 3) Reports on audits of notifications of measures (Berichte über Prüfungen von Maßnahmenbekanntgaben), and 4) Statement by the audited organization on the notifications (Stellungnahme einer

geprüften Einrichtung). The number of reports issued for each year is shown in the Table 1.

Year	2017	2018	2019	2020	2021
1) Audit report	92	86	87	55	101
2) Statement by the audited organization	158	96	109	57	149
3) Reports on audits of notifications of measures	25	26	15	5	13
4) Statement by the audited organization on the notifications	6	9	7	2	1
Total	281	217	218	119	264

Table 1. The number of reports of the latest 5 years
(Source: the number of reports are from 'Taetigkeitsbericht' of each year)

The fields of Audit are as follows: Municipality of Vienna, Equity investments of the municipality of Vienna, Subsidy recipients, Institutions, foundations and funds, and Audit competencies not governed by Wiener Stadtverfassung, the Vienna City Statutes (Stadtrechnungshof Wien, n.d. -d). On average, more than 90 institutions are audited annually, and the number of institutions is in Figure 11.

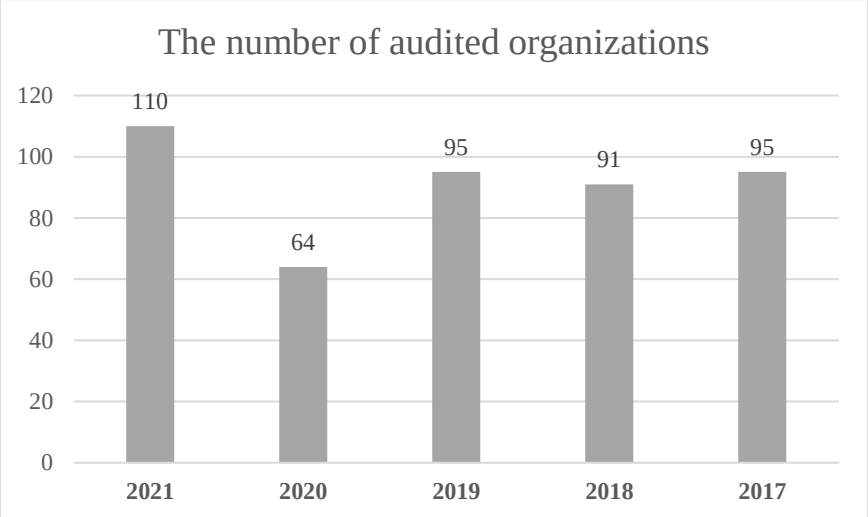
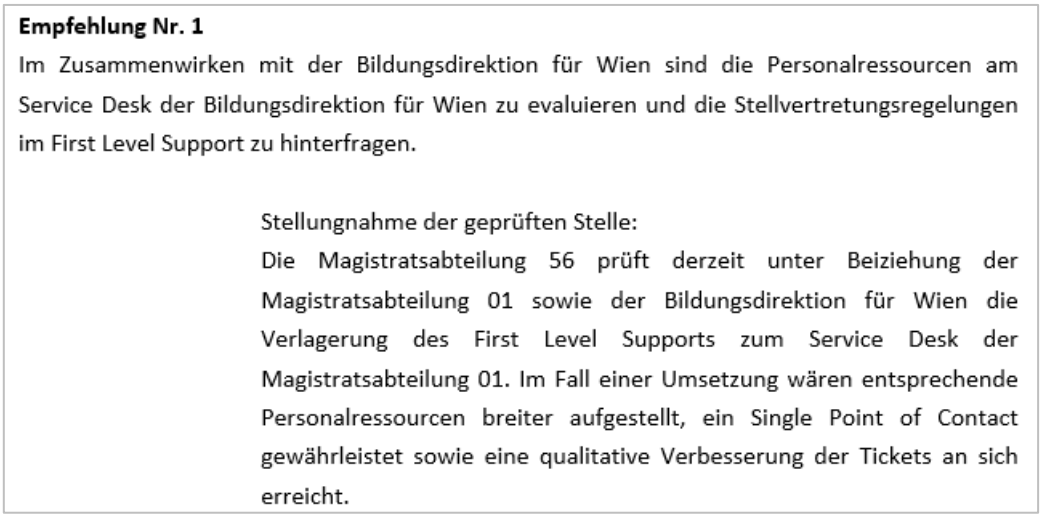


Figure 11. The number of the audited organization per year of the latest 5 years
(Soucre: the number of reports are from 'Taetigkeitsbericht' of each year)

The number of unique organizations is 175, excluding those audited in duplicate over five years. In particular, the following agencies were audited annually: Fonds Soziales Wien, Kuratorium Wiener Pensionisten-Wohnhäuser, Unternehmung Stadt Wien - Wiener Wohnen, Unternehmung Wiener Krankenanstaltenverbund, Wien Energie GmbH, Wiener Netze GmbH. A detailed list of organizations subject to annual audits can be found in Appendix. In addition to the institutions included in the list, there are additional institutions mentioned while auditing the institutions, and by analysing this text, it is expected that the relationship between government institutions, corporations, and organizations related to them can be found.

3.2 Text extraction

Paragraphs were extracted using the 'python-docx' library from each word file (.docx extension) format report. Extracting paragraphs based on line-breaking is easier and more accurate than extracting sentence units with ambiguous boundaries. In addition, since Paragraph is a unit with a single theme, co-occurrence within the paragraph help to check the strength of the relationship between entities.



Empfehlung Nr. 1

Im Zusammenwirken mit der Bildungsdirektion für Wien sind die Personalressourcen am Service Desk der Bildungsdirektion für Wien zu evaluieren und die Stellvertretungsregelungen im First Level Support zu hinterfragen.

Stellungnahme der geprüften Stelle:

Die Magistratsabteilung 56 prüft derzeit unter Beiziehung der Magistratsabteilung 01 sowie der Bildungsdirektion für Wien die Verlagerung des First Level Supports zum Service Desk der Magistratsabteilung 01. Im Fall einer Umsetzung wären entsprechende Personalressourcen breiter aufgestellt, ein Single Point of Contact gewährleistet sowie eine qualitative Verbesserung der Tickets an sich erreicht.

Figure 12. Example of “Empfehlung” and “Stellungnahme der gepruefteN stelle” from a report. StRH I – 4/19 MA 56, Maßnahmenbekanntgabe zu MA 56 und MA 01, Prüfung des IT Servicemanagements des Schulverwaltungsprogramms “WiSion” (2021.01)

Notably, the opinions of the city audit (Empfehlung) and the statement of the audited organizations (Stellungnahme dergepruften stell) were written in different paragraphs as shown in Figure 12 , but they are covering the same agenda. Therefore, it

is treated as a paragraph and extracted as a paragraph if an indent is included after the paragraph beginning with ‘Empfehlung No.’.

Some of the reports for 2020 and 2021 are randomly selected and used as modelling data. Some of the 2021 data were manually labelled with hand-labelling, and the 2020 data was created with weak labelling. The labelling detailed process is described in the following section.

3.3 Hand labelling

Hand labelling was performed on the raw text data extracted by the method mentioned in the above section. Labelling was carried out using ‘Doccano’ by adding ORG annotation to organization in a sentence. Doccano[1] is an open-source data labelling tool for machine learning practitioners, and it offer intuitive interface and easy to use like Figure 13.

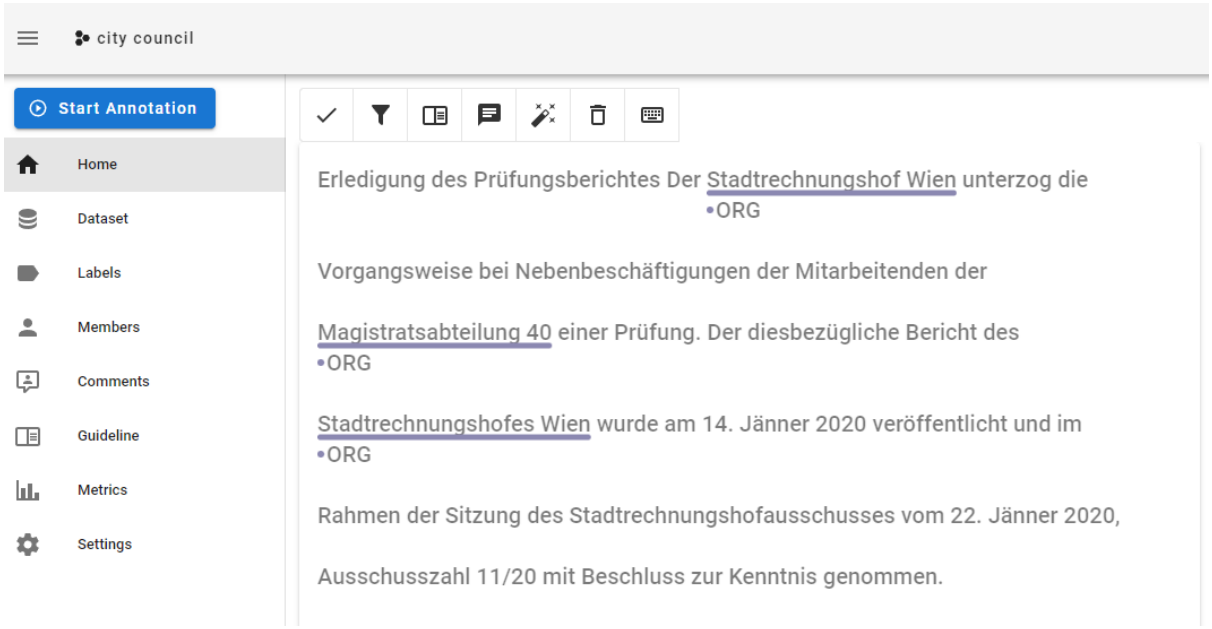


Figure 13. Screenshot of Doccano

The labelled dataset is exported as a jsonl format file containing text and entity list:

[1]https://doccano.github.io/doccano/

```
{
 "id":26618,
 "text":"Erledigung des Prüfungsberichtes der Stadtrechnungshof Wien
 unterzog die Vorgangsweise bei Nebenbeschäftigungen der Mitarbeitenden
 der Magistratsabteilung 40 einer Prüfung. Der diesbezügliche Bericht des
 Stadtrechnungshofes Wien wurde am 14. Jänner 2020 veröffentlicht und im
 Rahmen der Sitzung des Stadtrechnungshofausschusses vom 22. Jänner 2020,
 Ausschusszahl 11/20 mit Beschluss zur Kenntnis genommen.",
 "entities":
 [
  {"id":51632,"label":"ORG","start_offset":37,"end_offset":59},
   {"id":51633,"label":"ORG","start_offset":135,"end_offset":157},
    {"id":51634,"label":"ORG","start_offset":204,"end_offset":228}
 ]
}
```


At last, all data are changed into a txt file containing list of pairs of a token and a tag as below because it is easy to use for training NER models. BIO scheme is opted as a tagging scheme here. ‘B’ stands for Begin, where a named entity begins, ‘I’ stands for Inside of a named entity, and ‘O’ stands for Outside where the token is not a named entity.

```
...
Der O
Stadtrechnungshof B-ORG
Wien I-ORG
unterzog O
...
```

3.4 Weak labelling

As aforementioned in Chapter 2.3, weakly supervised labelling using noisier sources of supervision is adapted to create large training sets. The ‘skweak’ framework in python is employed for this study because it is easy to construct, apply, and aggregate labelling functions for sequence labelling tasks (Lison et al., 2021).

Then four labelling functions were defined as described in Table 2 for weak labelling with the skweak library. All labelling functions are aggregated using a majority voter, along with results using the HMM (Lison et al., 2021). After the aggregation,

cleansing step is added which is removing detected entities starting with lowercase or being longer than 15 tokens are removed.

Function Name	Type	Rules
MA detector (lf1)	Heuristic	Tokens start with ‘MA’ or ‘Magistrats-abteilung’, which mean municipal department, and the next token is number.
ORG detector (lf2)	Gazetteers	Tokens are matched to list of orgs extracted from activity reports (Tätigkeitsberichte) 2013 – 2022.
Company detector (lf3)	Heuristic	Noun chunks end with 'GmbH’, ‘Gesellschaft mbH’, ‘Gesellschaft mit beschränkter Haftung’, etc.
Verein detector (lf4)	Heuristic	Noun chunks start with ‘Verein’ which means club or associations.

Table 2. Labelling functions for skweak

The outputs of the weak labelling could be visualized as depicted in Figure 14. Finally, all the labelling data are exported as dictionaries consisting of text and entity span.

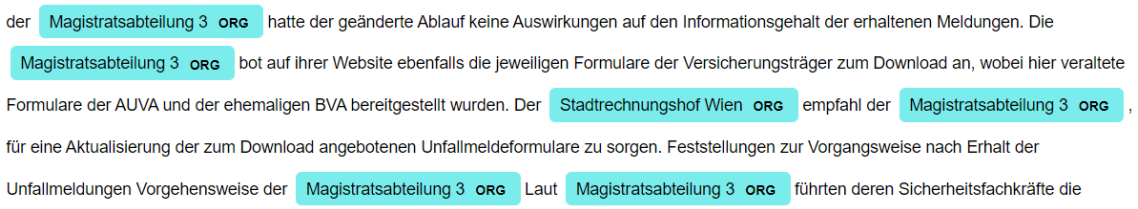


Figure 14. Visualized outputs of weakly-labelled data

Before applying this model to unlabelled data, the model was tested with labelled data to figure out how the model detects tokens correctly. Entity-level f1-scores (explained in section 5.3) are shown in the Table 3. The model with the best result of using all labelling functions with cleansing has a 0.88% f1-score.

Model	B-ORG		I-ORG		O		Total
	#	*F1*	#	*F1*	#	*F1*	*F1*
Hand labelled	1,927	-	2,233	-	84,434	-	-
All lf + Cleansing	**1,796**	**0.84**	2,340	0.82	**84,468**	**0.99**	**0.88**
All lf	1,796	0.84	**2,340**	**0.83**	84,468	0.99	0.88
Only lf1	860	0.5	854	0.5	86,880	0.98	0.68
Only lf2	1,039	0.59	1,404	0.6	86,151	0.98	0.73
Only lf3	61	0.0	243	0.1	88,290	0.98	0.36
Only lf4	1	0.0	1	0.0	88,591	0.98	0.33

Table 3. Entity-level f-score. # is the number of detected entities, and the F1 is f1-score computed by using the true labels from hand labelled data. Total F1 is macro average f-score. All lf is aggregated all labelling functions using HMM. All lf + cleansing is added cleansing step on the result of the aggregated all labelling functions.

3.5. Final dataset for training

Both data from hand labelling and weak labelling are used for experiments. The total number of documents, paragraphs, and tokens are shown in Table 4.

	Hand labelling	Weak labelling	Total
The number of documents	87	120	207
The number of paragraphs	6,131	10,178	16,309
The number of tokens	297,945	523,989	821,934
- B-ORG	5,715	8,635	14,350
- I-ORG	7,598	14,249	21,847
- O	284,642	501,105	785,747

Table 4. Information of each labelling data (hand labelling and weak labelling)

Each dataset which is a set of tokens from a paragraph text and their tags is split into train, validation, and test set which are required to allow a computer to learn the patterns (Zaccone & Karim, 2018). The training set is the knowledge base to train a

model, and the validation set is testing trained model during training in order to tune the parameters of the model. The test set, on the other hand, is used to evaluate thee performance of model on unseen data (Zaccone & Karim, 2018). A commonly used ration for data splitting is that 80% of the data is for training and 20% for testing (Joseph, 2022). However, weakly labelled data are added to the training dataset, the ratio 60:20:20 was used for splitting hand labelling data which means 60% for training dataset, 20% for validation, and 20% for testing. The model trained with weak labelling data will be tested on hand labelling test dataset which is the ground truth. **Error! Reference source not found.** shows the result of data splitting.

Split dataset		**# Paragraphs**	**# Tokens**			
			Total	**B-ORG**	**I-ORG**	**O**
Train	**Hand**	3,678	177,176	3,421	4,502	169,263
	Weak	10,178	523,989	8,635	14,249	501,105
	Sum	13,856	701,165	12,056	18,751	670,368
Validation	**Hand**	1,227	60,416	1104	1,492	57,820
Test	**Hand***	1,226	60,353	1190	1,604	57,559

Table 5. The result of data splitting into train/validation/test dataset. # means 'the number of', 'Hand' means hand-labelled data, 'Weak' means weakly-labelled data. Test-Hand dataset is used for testing model trained on hand-labelled data and trained on weakly-labelled data.

4 Methods

The detailed explanations of the trained models and training process are offered in this chapter. In section 4.1, the architectures of models including SpaCy and BERT are covered. In the next section, the steps of training the models that are introduced in the previous section are described.

4.1 Models

In this study, models commonly used for NER tasks such as BERTs and SpaCy pipelines are trained. Details of each model are described in following sections.

4.1.1 BERT

After BERT (see section 2.1.3) was introduced in 2018, variants that trained the BERT model with various data and methods were presented. Among them, we examine two models related to German, 'bert-base-german-cased' and 'xlm-roberta-base', and fine-tune the two models fit our data. Both models are $\text{BERT}_{\text{BASE}}$ models, each of them has 12 layers, 768 hidden sizes, and 12 self-attention heads.

***bert-base-german*-cased** The bert-base-german-cased (German BERT) is an open-source BERT model created by Deepset (Deepset, n.d.). According to Deepset (n.d.), German Wikipedia dump (6GB of raw txt files), the OpenLegalData dump (2.4 GB), and news articles (3.6 GB) were used for training model. The model was trained 810k steps with a batch size of 1024 for sequence length 128 first, 30k steps with sequence length 512 at last, and the learning rate = 1e-4. This model outperforms Google's multilingual BERT model on downstream NLP tasks, and it showed an 80.4% F-score on CONLL03 dataset (see section 5.1) (Deepset, n.d.).

xlm-roberta-base In 2019, a robustly optimized BERT pretraining approach (RoBERTa) is proposed by Liu et al. (2019), is a replication study of BERT. There was several measurements to improve the performance of BERT, such as using more training data (pretrain over 160GB of text), removing NSP parts, training on longer sequence, and

using dynamic masking pattern for MLM (Liu et al., 2019). The xlm-roberta-base model is a multilingual version of RoBERTa pretrained on 2.5TB of newly created clean CommonCrawl data in 100 languages including German (Conneau et al., 2019). It showed an 84.6% F-score on CONLL 03 dataset (Conneau et al., 2019).

4.1.2 SpaCy

SpaCy is an open-source python library for NLP introduced in 2015. SpaCy offers basic tasks of NLP such as tokenizer, tagger, and parser. Furthermore, production-ready training systems are provided that can be used for named entity recognition, sentence segmentation, text classification, etc (Honnibal, 2015).

SpaCy Tok2Vec First, we customize a SpaCy Tok2Vec pipeline. The pipeline consists of two parts, Tok2Vec and NER. Tok2Vec is a step for processing inputs, and NER is the one for detecting named entities. Tok2Vec has two sub-networks, one for embedding and the other for encoding as depicted in Figure 15.

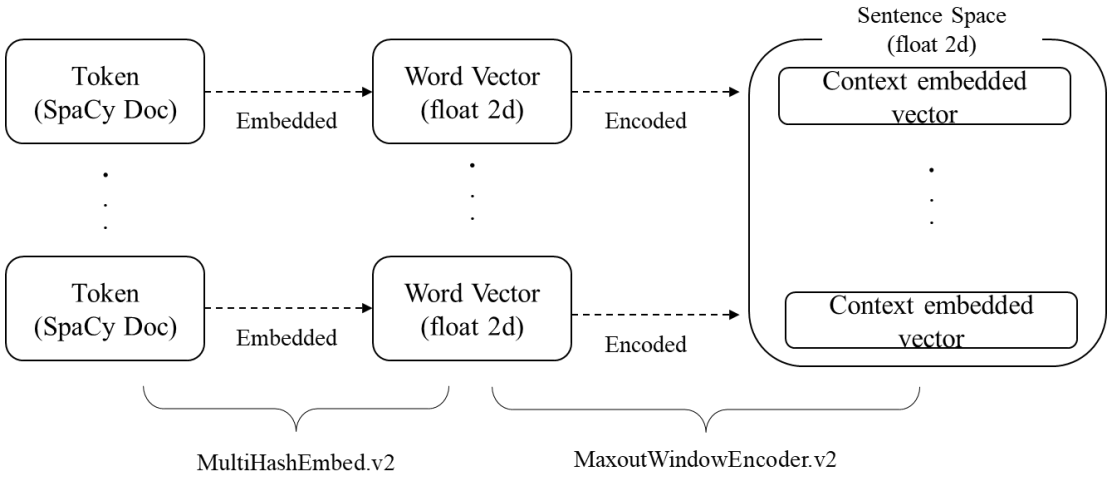


Figure 15. The SpaCy Tok2Vec process.
(Source: own illustration based on (Honnibal, 2016))

Tokens are transformed into word vectors in the embedding network. The model used for embedding 'MultiHashEmbed.v2' that the tokens are separately embedded as several lexical attributes using hash embedding, concatenates the results (Honnibal, 2016). Words that are hashed a 64-bit key are stored in four rows in a small hash table, and the sum of rows are the final embedding results (Honnibal, 2022). After the embedding, the

encoded as a ‘sentence matrix’ that each row represents the meaning of each token in the context. A model called ‘MaxoutWindowEncoder.v2’ using convolutions with maxout activation, layer normalization, and residual connections is used for context encoding.

As a sub-network of the NER model, ‘Tok2VecListener.v1’ model is deployed to pass the Tok2Vec components which are embedded and encoded inputs into NER model. For NER, ‘TransitionBasedParser.v2’ is used for predicting the parsing structure by reading words sequentially and combining them into syntactic structures (Dyer et al., 2015). The predicting named entity is performed with a simple multi-perceptron (SpaCy, n.d.).

SpaCy Transformers SpaCy transformers package offers convenient access to state-of-the-art transformer architecture. By using transformers which are large and powerful neural networks that give you better accuracy instead of Tok2Vec, the SpaCy transformer can increase the performance of the models. In this study, we use two models discussed in the previous section, bert-base-german-cased and xlm-roberta-base in the SpaCy transformer architecture. This model transfers the output of transformers with ‘TransformerListener.v1’ to the ner component of the architecture, and the ner component carries out parsing and detecting the final named entity as mentioned in section ***SpaCy Tok2Vec***.

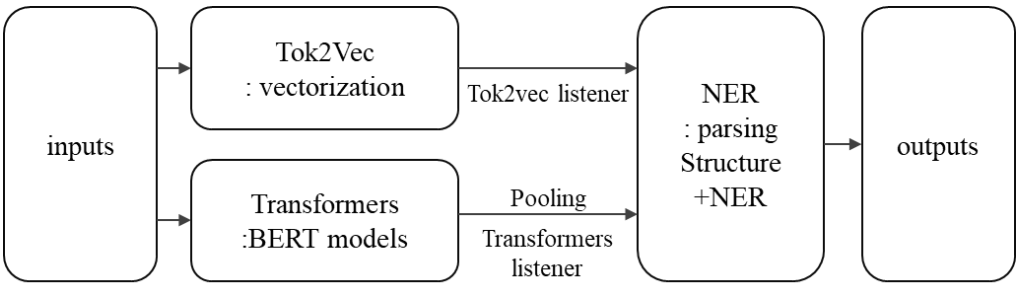


Figure 16. Pipelines of SpaCy Tok2Vec and Transformers. In case of Tok2Vec(Top), inputs transfer through Tok2Vec components from NER components. When transformers are used (bottom) inputs go through transformer components using selected BERT models, and then the outputs of transformers are reduced by pooling steps to convey them to NER components. After passing through NER components the outputs which are named entity tags are generated.

Hence, when customizing SpaCy transformers shows different results from when we just train the BERT model. Additionally, the outputs of transformer models do not align against spaCy tokens, which is why the reduction operation called pooling is needed

to communicate the transformer with SpaCy ner components. Figure 16 contains the difference between Tok2Vec pipeline and the transformers pipeline.

4.2 Training Process

The method of training the network is to use gradient-based optimization algorithms by having a cost function for calculating losses in the output layer and adjusting weights and biases (Larochelle et al., 2009). In this study, the Adam optimization algorithm (Kingma & Ba, 2014) is used instead of stochastic gradient descent (SGD) because the Adam algorithm converged faster than SGD. Hyperparameters such as weight decay and dropout are described in section 5.2, and the detailed training methods for models are explained below.

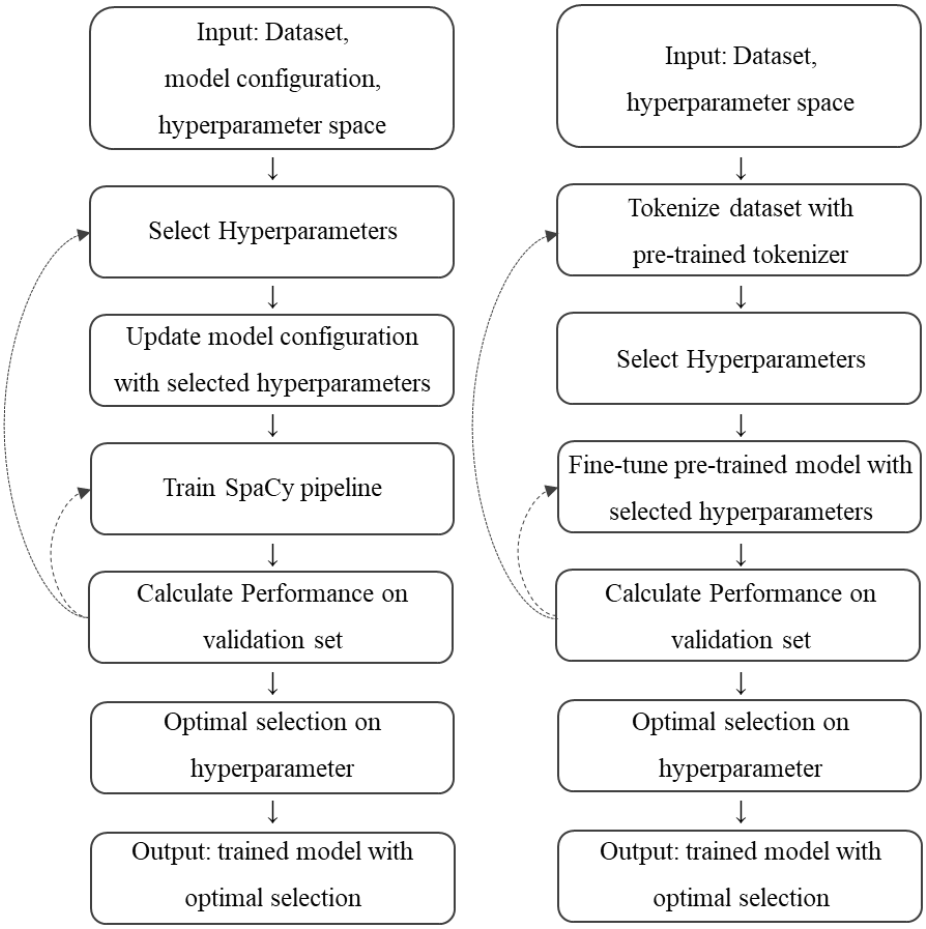


Figure 17. Training process of SpaCy Tok2Vec model(left) and BERT pre-trained models(right). The loops depicted with dashed-line is training loops repeated every evaluation (steps or epochs), with dotted-line is hyperparameter searching loops repeated when the new hyperparameters are selected.

For SpaCy, we defined the pipeline and then trained models with the SpaCy 3.0 library (version 3.4.4). SpaCy pipeline adapts Mean Square Error (MSE) to calculate training and validation losses. For training BERT models, we fine-tune the pretrained models mentioned in the previous section. The training process is performed with the python library Transformers (version 4.25.1) which provides APIs and tools to easily utilize pretrained models. Cross entropy (Zhang & Sabuncu, 2018) finding the difference between the probability distribution of the actual data and the prediction of the model is employed as a loss function.

The basic concepts of the training process are the same for both, SpaCy and pre-trained BERT models. The hyperparameter is selected in hyperparameter space, and the model is trained with selected hyperparameters. During the training, calculating training loss and validation loss, and back-propagation are performed for adjusting weights and biases. After training, the performance is measured and the optimal hyperparameter is determined showing the best performance on the validation dataset. The output is the model trained with optimal hyperparameters. The inputs and detailed processes are slightly different by models as depicted in Figure 17. Additionally, the SpaCy model calculates loss for every 200 steps, and the early stopping with 3 patients is adapted for training steps because it converged to the optimum quickly and tends to overfit when evaluated every epoch. On the contrary, the training loop is repeated every epoch in the case of BERT models, because the hyperparameter candidates were offered in the original BERT paper (Devlin et al., 2018).

5 Experimental Setup

This chapter covers experimental setups that are essential for implementing experiments. With the experiments, the answers to the second research question, and the best model for domain-specific NER are examined. Inputs of models and how those are processed before feeding to the model are described in section 5.1. The explanation of hyperparameter space for models is offered in section 5.2. At last, in section 5.3, the metric for evaluation is clarified.

5.1 Model input

The model input is basically the dataset hand-labelled dataset and the sum of the hand-labelled dataset and weakly-labelled dataset as explained in section 3.3.5. Additionally, CoNLL-2003(CoNLL03) dataset is used in order to examine the model architectures that are applicable to not only our dataset but also other datasets. CoNLL03 is a NER dataset released with CoNLL-2003 shared task (Sang & De Meulder, 2003). There are English and German data, and only German data is used in this study. The source of data comes from ECI Multilingual Text Corpus and German newspaper Frankfurter Rundshau and all texts are tokenized and manually annotated. Four entity types are detected: persons (PER), locations (LOC), organizations (ORG), and names of miscellaneous (MISC) entities (Sang & De Meulder, 2003). The dataset consists of three subsets: training, development, and test as explained in Table 6.

	Articles	**Sentences**	**Tokens**	**LOC**	**MISC**	**ORG**	**PER**
Training	553	12,705	206,931	4,363	2,288	2,427	2,773
Development	201	3,068	51,444	1,181	1,010	1,241	1,401
Test	155	3,016	51,943	1,035	670	773	1,195
sum	909	18,789	310,318	6,579	3,968	4,441	5,369
Own dataset	-	16,309	785,747	-	-	14,350	-

Table 6. The number of articles, sentences, tokens and entities in CoNLL03 dataset (source: based on (Sang & De Meulder, 2003)) and own dataset which is sum of hand-labelled and weakly-labelled data

However, in the case of BERT models, data is converted into vectors to help the machine understand the input. Tokenizers from each pre-trained model are used for tokenizing and embedding. Tokens are changed to numbers and create input id, and the token_type_ids are generated containing information about sentences. At last, to differentiate tokens and padding, the attention mask vector is added to the result of tokenizing. The tags should be converted to numbers because of the same reason for processing tokens. The tags are converted into numbers such as `{1: 'O', 2: 'I-PER', 3: 'I-LOC', 4: 'I-ORG', 5: 'I-MISC', 6: 'B-MISC', 7: 'B-LOC', 8: 'B-PER', 9: 'B-ORG', 0: 'PAD'}` as an example.

5.2 Hyperparameters

A step to find the optimal configuration for each model is essential, hence the hyperparameter tuning was performed during the training with Weights & Biases (Wandb)[2]. Wandb is a machine learning platform for tracking experiments, tuning hyperparameters, and evaluating models. The selected method for this experiment is 'random' because random search is faster than other search methods such as 'grid' and 'Bayesian'. Additionally, as the lack of computing power, the search is only repeated 10 times for each case. The goal is to maximize F1-score (see chapter 5.3). The five hyperparameters are considered:

Learning Rate: the rate for determining how much to change parameters at each time updating parameters

Batch size: the number of data samples passed to the network

Epochs: the number of times to iterate training over all training dataset

Dropout rate: the rate that randomly selected neurons are ignored (dropped out) during training

Weight decay: a factor to limit overfitting by reducing the complexity of the model in the back-propagation step is optimized

[2] https://docs.wandb.ai/

The hyperparameter candidates and selected optimal parameters are as below.

BERT For the BERT models, the hyperparameters proposed in the original paper (Devlin et al., 2018), epochs, batch size, and learning rate were borrowed. Additionally, weight decay is optimized as well.

SpaCy In the case of training SpaCy Tok2Vec pipeline, learning rate, batch size, and dropout are considered tuned hyperparameters. Variations from a basic configuration provided by SpaCy were set as candidates. In the case of SpaCy, early stopping with 3 patients is used instead of setting the number of epochs, which is why there is no number of epochs in the hyperparameters. On the contrary, the SpaCy transformer models are a mixture of the SpaCy tok2vec and BERT models. The learning rate and weight decay are adapted from the BERT, and the rest of the hyperparameters are from the SpaCy Tok2Vec models. Furthermore, when we train SpaCy transformers, the epoch is not considered a hyperparameter since early stopping is used as well.

Hyperparameters	**Range**		
	SpaCy Tok2Vec	**BERT**	**SpaCy Transformers**
Epoch	-	2,3,4	-
Learning rate	0.001 - 0.01	2e-5, 3e-5, 5e-5	2e-5, 3e-5, 5e-5
Batch size	32, 64, 128, 1000	32, 64, 128	32, 64, 128, 1000
Drop out	0.0, 0.1, 0.2, 0.3, 0.4, 0.5		
Weight decay	-	0.0, 0.1, 0.2, 0.3, 0.4, 0.5	0.0, 0.1, 0.2, 0.3, 0.4, 0.5

Table 7. Hyperparameter range of each model

5.3 Evaluation metrics

The output of the trained model is compared to human annotations which is the ground truth and the Precision, Recall and F1-score can be computed based on true positive (TP) and false positive (FP), and false negative (FN) are calculated first (Li et al., 2020). The formulas of precision, recall, and F1-score are defined below.

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

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

$$F1 - score = \frac{2 * precision * recall}{precision + recall}$$

Precision shows how the NER system presents only correct entities, and recall takes all entities into account to measure the ability of a NER system, and the F1-score is a balanced metric between precision and recall (Li et al., 2020). F1-score is most commonly used for evaluating NER models (Li et al., 2020), which is why F1-score is used as a metric in this study. If there are more than 2 entities, the F1-score must access all entity types. Macro-averaged f1-score computes F1-score for all entity types respectively and then take an average. Micro-averaged f1-score, however, calculates the total TP, FP, and FN of outputs altogether regardless of their entity types (Li et al., 2020).

6 Results

With models and the experiment step as explained in chapters 4 and 5, the experiments are implemented, and we get the results of each model. The performances of models were measured as entity level F1-score and macro average F1-score as described in section 5.3. Hence, in this chapter, the results of the models are analysed, and the deployment of the best model is introduced.

6.1 Performances

The results of the trained models are shown in Table 8. We used 3 different training datasets: one is hand-labelled data only, the second one is weakly-labelled data only, and the last one is the sum of the hand-labelled dataset and weakly-labelled dataset as explained in Table 5. However, the validation, and test sets consist of hand-labelled data only because the model should be tested on the golden dataset. the hand-labelled dataset. Additionally, there is only one entity type, ORG, the macro-average F1-score and the entity level F1-score for ORG type were the same.

Train on / **Models**	**Hand**	**Weak**	**All data**
SpaCy tok2vec	65.20%	55.59%	81.50%
Fine-tuned bert-base-german-cased	85.73%	66.45%	81.83%
Fine-tuned xlm-roberta-base	85.58%	69.68%	80.69%
SpaCy Transformers bert-base-german-cased	92.62%	64.36%	92.77%
SpaCy Transformers xlm-roberta-base	95.05%	67.98%	**95.57%**

Table 8. Entity level f1-score of each model on test dataset. The test dataset is hand-labelled dataset. Hand means the model is trained on the hand-labelled data, Weak means the model is trained on weakly-labelled dataset and the All means the models is trained on the sum of hand-labelled data and the weakly-labelled data.

Among the five models, the SpaCy transformers xlm-roberta-base model trained on all data with a 95.57% F1-score outperforms the SpaCy Tok2Vec model and fine-tuned BERT model.

In more detail on the data side, using only weakly-labelled data as input results in less performance than using only hand labelled data as input. This is because weakly-labelling method does not perfectly detect entities as we discussed in section 3.4, it leads to low performance compared to using the golden data. Furthermore, when we use both of hand-labelled and weakly-labelled data, the performances are not better than only using hand-labelled dataset. It comes from the same reason as when only using weakly-labelled data.

First, we expected the larger dataset might prevent overfitting and lead to obtaining the better models as commonly known. Hence, we guess the bigger dataset, the sum of hand and weakly-labelled dataset, leads to better performance. However, with the lower accuracy of annotation of the weakly-labelled input data, the results were not as good as hand-labelled dataset.

Interestingly, however, there was one exception. When we use SpaCy Transformers for two models, bert-base-german-cased and xlm-roberta-base, models trained on all data get higher F1 scores than on hand-labelled data. This is because the NER components in SpaCy Transformers, parsing syntactic structures of sentences, might revise inaccuracy caused by weakly-labelled data. Although this inference would have been more robust if the same results were found with only weakly-labelled data, SpaCy Transformers models trained with only weakly-labelled data perform slightly lower than without SpaCy. However, it can be interpreted that SpaCy itself may not be robust to too noisy input.

To focus on the models, first, the SpaCy tok2vec model shows the worst performance as we expected because this model is adapting a simple multi-perceptron model with the vectorization step for processing input. The other models are based on BERT model, which is using more high-level architecture that makes those models outperform SpaCy tok2Vec.

Comparing bert-base-german-cased and xlm-roberta-base, it was expected that xlm-roberta-base showed better performance as were the experiments on the CoNLL03 dataset from their papers. In the case of SpaCy transformer the same results were obtained in the experiments on our dataset, xlm-roberta-base outperformed. However, without SpaCy, xlm-roberta-base is slightly worse perform. The difference might come from the

hyperparameter optimization. Since the optimization loops were tried only 10 times as mentioned in section 5.2 it could miss the global optimum that we could get if we tried all combinations of hyperparameter candidates.

On the other hand, the SpaCy transformer models show the best f1-score. Since the SpaCy NER component examines dependencies of tokens and syntactic structures, it could raise the accuracy of detecting entities with more information that we gain in this step. Hence, customizing the SpaCy transformers model shows better results than fine-tuning BERT models.

After the training model, we test a simple sentence as presented below by using the best model among trained models with the pipeline method from the SpaCy transformer library. It detects 2 organization entities, ‘Stadtrechnungshofes Wien’ (the City of Vienna Court of Audit) and ‘Magistratsabteilung 3’ (Municipal Department 3), correctly.

> 'Auf Rückfrage des **Stadtrechnungshofes Wien(ORG)** teilte die **Magistratsabteilung 3(ORG)** mit, dass in den vergangenen 15 Jahren kein tödlicher Dienst- bzw'

Furthermore, the results of the training on CoNLL 03 dataset with the same training process as our own dataset are found in Table 9. Since the CoNLL 03 dataset has 4 entity types, LOC, MISC, ORG, and PER, the entity-level F1-score and macro-average F1-score are derived separately.

	LOC	**MISC**	**ORG**	**PER**	**AVG**
SpaCy tok2vec	75.49%	61.51%	64.76%	81.67%	72.79%
Fine-tuned bert-base-german-cased	81.91%	69.54%	72.44%	91.85%	78.94%
Fine-tuned xlm-roberta-base	83.07%	71.68%	73.57%	92.88%	80.29%
SpaCy Transformers bert-base-german-cased	80.88%	69.35%	69.35%	91.96%	80.25%
SpaCy Transformers xlm-Roberta-base	81.90%	73.35%	78.95%	90.49%	**81.92%**

Table 9. Entity level f1-score and macro average f1-score of each model on validation dataset and the test dataset. LOC, MISC, ORG, PER mean the entity-level F1-score of each entity type and the AVG is the macro-average F1-score.

For all models, an entity type PER is detected most accurately than other entity types, on the contrary, MISC shows the worst F1-score. As the same as our own dataset, the SpaCy xlm-roberta-base model outperforms with an 81.92% micro-average F1-score.

6.2 Case study

In the previous section, we obtain the best model for detecting named entities (ORG) in Vienna governmental reports. In this section, the best model is deployed for detecting organizations in reports from the City of Vienna Court of Audit in 2022 as an example. Additionally, we examine how the detected organizations can be usefully used in real-world business situations to solve the third research question.

6.2.1 Setups

The process of deployment of the model for detecting entities from the new dataset is described in Figure 18. We have text data that are paragraphs extracted from reports produced by the City of Vienna Court of Audit in 2022. The process of extracting text from the original documents is the same as explained in section 3.2, and the additional information of each document such as the year and month that the document is written, file name, and the title of the document for achieving documents.

Next, the extracted text data is fed to the model and the model detects entities in each paragraph. However, there is a problem that the situation that the same entity is denoted in a different way because of the changes in the articles or using the abbreviation, such as ‘Stadtrechnungshof Wien’ and ‘Stadtrechnungshofes Wien’. When this problem causes because of the articles or plural forms, the edit distance is applied to calculate the similarity of the entities. For this step, the NLTK library is used, and the NLTK edit distance function is calculating the Levenshtein edit distance between two strings which is the number of characters that must be replaced, inserted, or deleted to convert the first string to the second string. But when the string is long, the edit distance is also can be

larger compared to the shorter entities, so we divided the edit distance by the length of the string as denoted below:

$$\mathbf{ED} = \left[\frac{\boldsymbol{NLTK\ edit\ distance}}{\boldsymbol{max}\ (\boldsymbol{len}(\boldsymbol{org}),\ \ \boldsymbol{len}(\boldsymbol{compared_{org}})\ )}\right]$$

The threshold is set as 0.25, hence if the ED is smaller than 0.25, the two strings are considered to be the same organization, so the shorter one is converted by the longer string. This method is not efficient for abbreviations such as 'MA' and 'Magistrats-abteilung', because the edit distance between the original entity and the abbreviation is too large. This case should be treated manually by matching the entity and its abbreviation. After cleansing entities, we count the frequency which means the occurrence of the entity of a document.

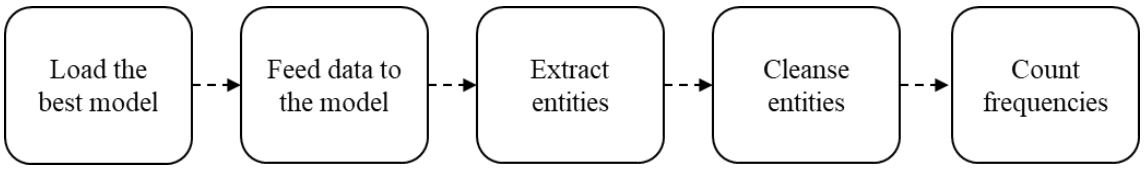


Figure 18. The process of the model deployment

6.2.2. Results

The frequency of the entity is the sum of the occurrence in a document as shown in Table 10.

Document id	Organization	Frequency
2022012	Community TV GmbH	225
2022012	MA 13	167
2022012	Stadtrechnungshofes Wien	135
2022012	OktoLab GmbH	117
2022012	stadt wien	25
2022012	Wirtschaftstreuhand und Steuerberatungs GmbH	13
2022012	MA 5	5

Table 10. The example of frequency of organization in a document

Table 11 shows an example of the result of the computing co-occurrences of entities. Doc ID is the document id, Organization 1 and Organization 2 is the pair of the organizations, and the Frequency is the count of the co-occurrence of the pair of

organizations. Therefore, ‘Community TV GmbH’ and ‘MA 13’ co-occurred in one paragraph 13 times in document 2022012.

Doc ID	Organization 1	Organization 2	Frequency
2022012	Community TV GmbH	MA 13	65
2022012	Community TV GmbH	OktoLab GmbH	58
2022012	MA 13	Stadtrechnungshofes Wien	37
2022012	MA 13	OktoLab GmbH	20
2022012	Stadtrechnungshofes Wien	stadt wien	8

Table 11. The example of the co-occurrences of entities

Up to this point, the process of deployment of the best model and obtaining entities in the document is discussed. In the next section, how the extracted entities can be utilized in the real-world is examined.

6.2.3 Analysis

The detected entities and their frequencies make digital documents more accessible in a variety of ways.

Meta data

First, we can manage the data of documents more efficiently by storing the meta data. For now, the documents are listed on the website as shown in Figure 19, and one only can read the title of the document without any other information.

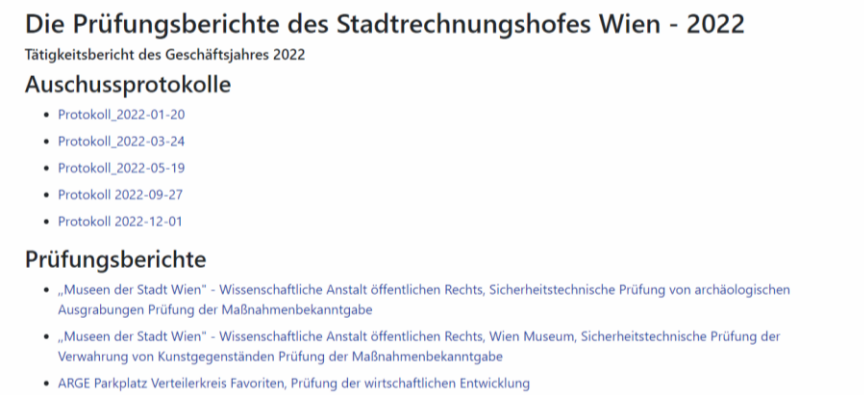


Figure 19. The list of the reports(Fruefungsberichte) on the website : https://www.stadtrechnungshof.wien.at/berichte/2022/inhaltsverzeichnis.htm (last visit : 15.01.2023)

However, by storing metadata of the documents, the summarized information of the document can be found easily, and this can be used for document search as well. The metadata containing the document id, produced year and month, file name and the title of the document, and the list of the organizations are mentioned in the document as an example in Table 12.

doc_id	2022010	2022011	2022012
Year	2022	2022	2022
Month	1	1	1
File name	StRH-I-1-21.docx	StRH-I-10-20.docx	StRH-I-12-20.docx
Title	MA 10 und Katholischer Familienverband der Erzdiözese Wien, Prüfung des Projektes Omadienst; Subventionsprüfung	MA 10, Prüfung des Kindergarten Vormerksystems	MA 13 und Community TV-GmbH, Prüfung der Community TV-GmbH Prüfungsersuchen gemäß § 73e Abs. 1 WStV vom 25. September 2020
Entities	[('Katholischen Familienverbandes Wien', 115), ('MA 10', 53), ('Stadtrechnungshofes Wien', 30), ('stadt wien', 13), ('KFVW', 5)]	[('MA 10', 101), ('Stadtrechnungshofes Wien', 36), ('stadt wien', 14), ('MA 23', 3), ('MA 1', 2), ('MA 18', 2)]	[('Community TV GmbH', 225), ('MA 13', 167), ('Stadtrechnungshofes Wien', 135), ('OktoLab GmbH', 117), ('stadt wien', 25), ('Wirtschaftstreuhand und Steuerberatungs GmbH', 13), ('MA 5', 5), ('Bundeshauptstadt Wien', 2)]

Table 12. The example of the metadata of the documents

With this metadata, one can search the document by the organization names, and the result of the documents can be listed by relevance which can be represented by its occurrence in the document. Plus, the similarity of the documents also can be calculated by entity lists. The more similar the organizations mentioned in the two documents, the more they can be treated as related documents.

For example, we can assume which documents are more related by examining graphs representing the relationship between the documents and the organizations with Neo4j[3], a graph data application. Neo4j can search nodes and edges from csv files and connect them as pre-defined relations with 'Cypher' as below:

```
LOAD CSV WITH HEADERS FROM 'file:///doc_ent_freq_1.csv' AS row
WITH row.id AS id, row.doc_id AS docid, row.org_name AS org,
toInteger(row.freq) AS freq
WHERE (docid = "2022011" or docid = "2022015" or docid = "2022014")
and (org <> "stadt wien" and org <> "Stadtrechnungshofes Wien")
MERGE (orgname:Organization {name: org})
MERGE (docu:Document {id: docid})
CREATE (orgname)-[r: containing {weights:freq}] -> (docu)
RETURN docu, r, orgname
```

With the cypher, a graph representing a relationship between document ids and organizations is shown in Figure 20. The red nodes are documents id, and the green nodes are organizations. If an organization is mentioned in a document, the edge 'containing' from the document id (red node) to the organization (green node) is created. In the case of document 2022014, it shares more organization nodes with document 2022015 than document 2022011. Therefore, we can say documents 2022014 and 2022015 is containing more similar information than 2022014 and 2022011.

[3] https://neo4j.com/

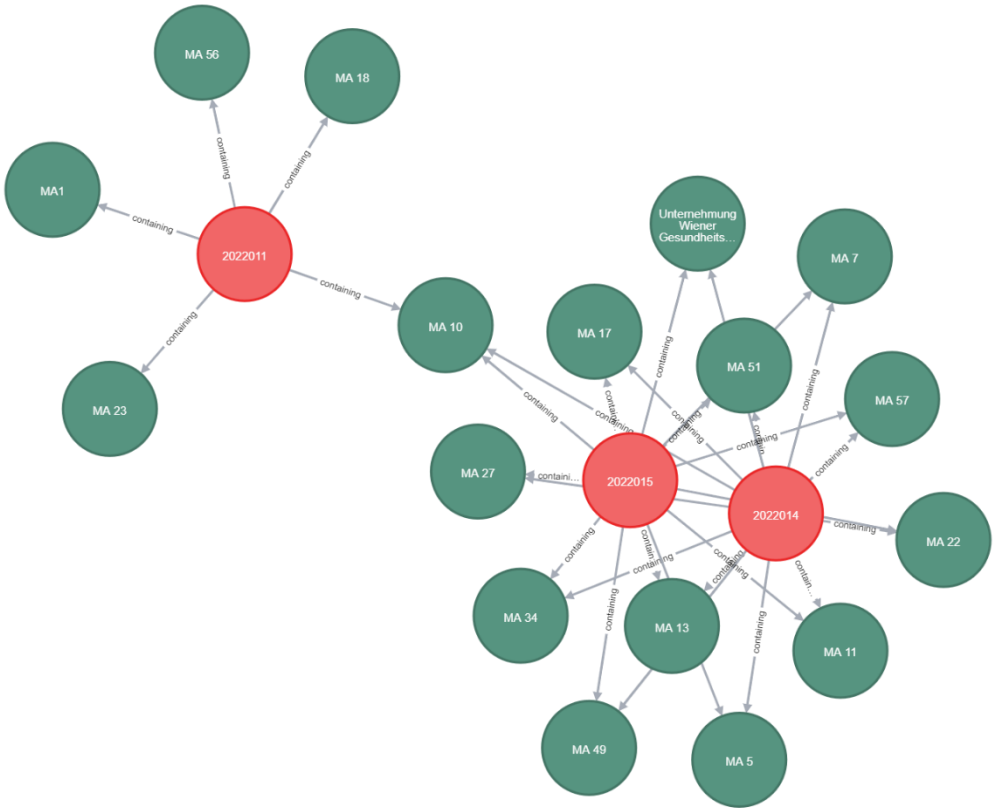


Figure 20. A graph of occurrence of the organizations in documents

Furthermore, the documents that are related to the organizations also can be searched. Figure 21 shows the result of searching documents that are mentioned in MA 13 more than once. The red nodes are document IDs and the edge weights are the frequency of occurrence of the MA 13 in each document.

Figure 21. A graph of documents that are mentioned of the organization, MA 13

In this way, the extracted meta data and entities can enhance the convenience of managing data and information retrieving.

Relation between organizations

Not only the relationship between the documents and the organizations, but we can also derive the relationship among organizations by using the co-occurrence data that we collect in section 6.3.1. The higher the frequency that the two organizations appear together in documents, the stronger the connection between the two organizations can be inferred as shown in Figure 22.

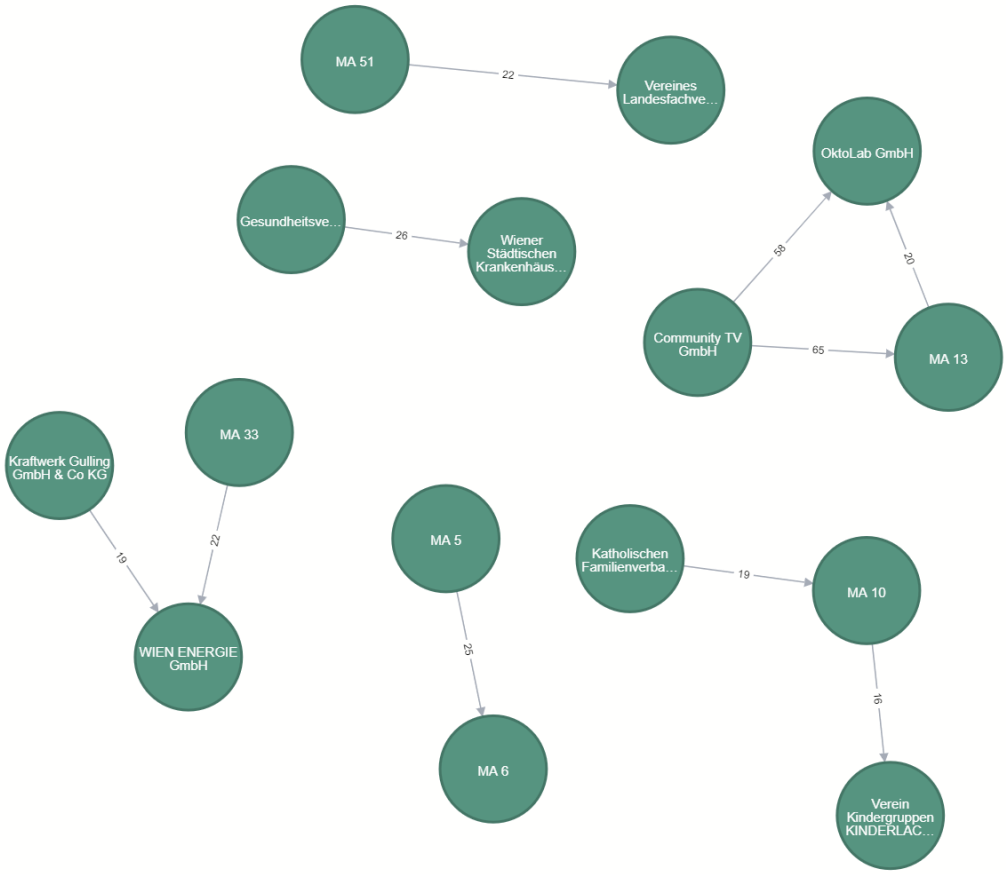


Figure 22. A graph of relationship between organizations in 2022 where co-occurrence is bigger than 15

An obviously strong relationship can be found such as MA 5 (Finance) and MA 6 (Accounting and taxes), and there are also relationships between organizations that are hard to infer from their names that they are strongly related such as MA 13 (Education

and Youth) and Community TV GmbH. Additionally, if one adjusts the frequency lower, then the rare but important relationship can be discovered.

We could develop models for solving more complex business problems. The model is to extract only organization entity types and we have counted the frequencies of their co-occurrences. We assume that the higher frequency shows a stronger relationship, however, there could be different aspects. To be clear about this problem, the tokens showing relationships between organizations can be extracted as well. Furthermore, besides the organization entity type, we could extract various entity types from the reports. Especially, since the contents of the reports from the City of Vienna Court of Audit are regard to government finance, ‘money’ can be one of the important entity types. These additional tasks will make NER more useful in the business.

7 Conclusion

In this study, we examined models for NER in the government domain in German. The current state that documents from the government are digitized make opportunities to utilize information from the text. However, the underuse of digitized assets and the lack of applicable models for the documents written in Germen in the government domain led us to this study. The three research questions are derived from this situation.

The first question is about an effective method for getting a training dataset. We create a dataset by extracting text from the documents created by the City of Vienna Court of Audit, and labelling organizations on the text using two methods, hand labelling, and weak labelling. Using these two methods, a dataset consisting of over 800K tokens is created. The second research question is finding the best model for the dataset we have created. With the data, we customized and fine-tuned five models that achieved high scores with the CoNLL03 German dataset. The model that achieves the highest F1-score was the SpaCy transformers xlm-roberta-base model. The last research question is exploring how the model can be deployed and the outputs can be utilized for real-world business. By using the model for NER in 2022 reports from the City of Vienna Court of Audit, organizations, frequency in a document, and the co-occurrence representing relationships between organizations are extracted. Neo4j is used for easier searching relationships that are meaningful for the business and visualization. We explore the relationship between organization and documents, or among the organization with the outputs from our model. Adding additional entity types or relationship will be enhance the usability of this model in the real-world business problem.

References

Ahmed, S., Stoeckel, M., Driller, C., Pachzelt, A., & Mehler, A. (2019, 2019). BIOfid Dataset: Publishing a german gold standard for named entity recognition in historical biodiversity literature.

Brandsen, A., Verberne, S., Lambers, K., & Wansleeben, M. (2022). Can BERT Dig It?–named entity recognition for information retrieval in the archaeology domain. *Journal on Computing and Cultural Heritage (JOCCH)*.

Brandsen, A., Verberne, S., Lambers, K., Wansleeben, M., Calzolari, N., Béchet, F., Blache, P., Choukri, K., Cieri, C., & Declerck, T. (2020, 2020). Creating a dataset for named entity recognition in the archaeology domain.

Cariello, M. C., Lenci, A., & Mitkov, R. (2021, 2021). A comparison between named entity recognition models in the biomedical domain. Proceedings of the Translation and Interpreting Technology Online Conference,

Chiu, J. P. C., & Nichols, E. (2016). Named Entity Recognition with Bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, *4*, 357-370. https://doi.org/10.1162/tacl_a_00104

Cole, M. R. (2018). *Hands-on neural network programming with C# : add powerful neural network capabilities to your C# enterprise applications*. Packt. http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&AN=1905987

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2019). Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273-297.

Das, K., & Behera, R. N. (2017). A survey on machine learning: concept, algorithms and applications. *International Journal of Innovative Research in Computer and Communication Engineering*, *5*(2), 1301-1309.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dyer, C., Ballesteros, M., Ling, W., Matthews, A., & Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

Fu, D., Chen, M., Sala, F., Hooper, S., Fatahalian, K., & Ré, C. (2020, 2020). Fast and three-rious: Speeding up weak supervision with triplet methods.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. The MIT Press

MIT P. http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&AN=2565107

Goyal, A., Gupta, V., & Kumar, M. (2018). Recent Named Entity Recognition and Classification techniques: A systematic review. *Computer Science Review*, *29*, 21-43. https://doi.org/https://doi.org/10.1016/j.cosrev.2018.06.001

Graves, A., Mohamed, A.-r., & Hinton, G. (2013, 2013). Speech recognition with deep recurrent neural networks.

Grishman, R., & Sundheim, B. M. (1996, 1996). Message understanding conference-6: A brief history. The 16th International Conference on Computational Linguistics,
Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. *Science*, *349*(6245), 261-266. https://doi.org/10.1126/science.aaa8685
Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In: A field guide to dynamical recurrent neural networks. IEEE Press In.
Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural computation*, *9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735
Hodnett, M. (2019). *Deep Learning with R for Beginners : Design Neural Network Models in R 3. 5 Using TensorFlow, Keras, and MXNet*. Packt Publishing, Limited. https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&AN=2142582
Honnibal, M. (2015). *Introducing spaCy*. explosion. Retrieved Feburary 05 from https://explosion.ai/blog/introducing-spacy
Honnibal, M. (2016). *Embed, encode, attend, predict: The new deep learning formula for state-of-the-art NLP models*. Explosion. Retrieved Feburary 05 from
Honnibal, M. (2022). *Compact word vectors with Bloom embeddings*. Retrieved Feburary 05 from https://explosion.ai/blog/bloom-embeddings
Huang, Z., Xu, W., & Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.
Imran, M., & Alsuhaibani, S. A. (2019). Chapter 7 - A Neuro-Fuzzy Inference Model for Diabetic Retinopathy Classification. In D. J. Hemanth, D. Gupta, & V. Emilia Balas (Eds.), *Intelligent Data Analysis for Biomedical Applications* (pp. 147-172). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-815553-0.00007-0
Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255-260. https://doi.org/10.1126/science.aaa8415
Joseph, V. R. (2022). Optimal ratio for data splitting. *Statistical Analysis and Data Mining: The ASA Data Science Journal*.
Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
Kumar Tiwari, A. (2021). *Deep learning and its applications*. Nova Science Publishers. https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&AN=3052657
Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring strategies for training deep neural networks. *Journal of machine learning research*, *10*(1).
LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436-444.
Leitner, E., Rehm, G., & Moreno-Schneider, J. (2020). A dataset of german legal documents for named entity recognition. *arXiv preprint arXiv:2003.13016*.
Li, J., Sun, A., Han, J., & Li, C. (2020). A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, *34*(1), 50-70.
Lison, P., Barnes, J., & Hubin, A. (2021). skweak: Weak Supervision Made Easy for NLP. *arXiv preprint arXiv:2104.09683*.
Lison, P., Hubin, A., Barnes, J., & Touileb, S. (2020). Named entity recognition without labelled data: A weak supervision approach. *arXiv preprint arXiv:2004.14723*.

Liu, Y. (2019). *Hands-On Deep Learning Architectures with Python : Create Deep Neural Networks to Solve Computational Problems Using TensorFlow and Keras*. Packt Publishing, Limited. https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&AN=2116438

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

MacQueen, J. (1967, 1967). Classification and analysis of multivariate observations.

Mahdavinejad, M. S., Rezvan, M., Barekatain, M., Adibi, P., Barnaghi, P., & Sheth, A. P. (2018). Machine learning for internet of things data analysis: a survey. *Digital Communications and Networks*, *4*(3), 161-175. https://doi.org/https://doi.org/10.1016/j.dcan.2017.10.002

Mahesh, B. (2020). Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, *9*, 381-386.

Marrero, M., Urbano, J., Sánchez-Cuadrado, S., Morato, J., & Gómez-Berbís, J. M. (2013). Named Entity Recognition: Fallacies, challenges and opportunities. *Computer Standards & Interfaces*, *35*(5), 482-489. https://doi.org/https://doi.org/10.1016/j.csi.2012.09.004

Menshawy, A. (2018). *Deep Learning By Example: A hands-on guide to implementing advanced machine learning algorithms and neural networks*. Packt Publishing Ltd.

Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, *30*(1), 3-26.

Qiu, J., Wu, Q., Ding, G., Xu, Y., & Feng, S. (2016). A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*, *2016*(1), 67. https://doi.org/10.1186/s13634-016-0355-x

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, *1*(1), 81-106.

Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J., Wu, S., & Ré, C. (2017, 2017). Snorkel: Rapid training data creation with weak supervision.

Sang, E. F., & De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Sazali, S. S., Rahman, N. A., & Bakar, Z. A. (2016, 23-24 Aug. 2016). Information extraction: Evaluating named entity recognition from classical Malay documents. 2016 Third International Conference on Information Retrieval and Knowledge Management (CAMP),

Shaalan, K., Hassanien, A. E., & Tolba, F. (2017). *Intelligent natural language processing: trends and applications* (Vol. 740). Springer.

Shelar, H., Kaur, G., Heda, N., & Agrawal, P. (2020). Named entity recognition approaches and their comparison for custom ner model. *Science & Technology Libraries*, *39*(3), 324-337.

Sun, P., Yang, X., Zhao, X., & Wang, Z. (2018, 15-17 Nov. 2018). An Overview of Named Entity Recognition. 2018 International Conference on Asian Language Processing (IALP),

Syed, M. H., & Chung, S.-T. (2021). MenuNER: Domain-Adapted BERT Based NER Approach for a Domain with Limited Dataset and Its Application to Food Menu Domain. *Applied Sciences*, *11*(13).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Wang, X., Song, X., Li, B., Guan, Y., & Han, J. (2020). Comprehensive named entity recognition on cord-19 with distant or weak supervision. *arXiv preprint arXiv:2003.12218*.

Yadav, V., & Bethard, S. (2019). A survey on recent advances in named entity recognition from deep learning models. *arXiv preprint arXiv:1910.11470*.

Ying, X. (2019). An Overview of Overfitting and its Solutions. *Journal of Physics: Conference Series*, *1168*(2), 022022. https://doi.org/10.1088/1742-6596/1168/2/022022

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent Trends in Deep Learning Based Natural Language Processing [Review Article]. *IEEE Computational Intelligence Magazine*, *13*(3), 55-75. https://doi.org/10.1109/MCI.2018.2840738

Yu, Y., Zuo, S., Jiang, H., Ren, W., Zhao, T., & Zhang, C. (2020). Fine-tuning pre-trained language model with weak supervision: A contrastive-regularized self-training approach. *arXiv preprint arXiv:2010.07835*.

Zaccone, G. (2018). *Deep Learning with TensorFlow : Explore neural networks and build intelligent systems with Python, 2nd Edition* (2nd . ed.). Packt Publishing. http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&AN=1789473

Zaccone, G., & Karim, M. R. (2018). *Deep learning with tensorFlow: Explore neural networks and build intelligent systems with python*. Packt Publishing Ltd.

Zeng, X., Li, Y., Zhai, Y., & Zhang, Y. (2020, 2020). Counterfactual generator: A weakly-supervised method for named entity recognition.

Zhang, Z., & Sabuncu, M. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, *31*.

Zöllner, J., Sperfeld, K., Wick, C., & Labahn, R. (2021). Optimizing small BERTs trained for German NER. *Information*, *12*(11), 443.