University of Sheffield

Exploring Deep Learning for Named Entity Recognition

Nokessa Ilham Melika Soumahoro

Supervisor: Professor Robert Gaizauskas

This report is submitted in partial fulfilment of the requirement for the degree of BSc in Artificial Intelligence and Computer Science

in the

Department of Computer Science

May 8, 2024

Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Nokessa Ilham Melika Soumahoro

Abstract

This report explores Named Entity Recognition (NER), a natural language processing task which involves identifying and classifying words and sequences of words in a text into various predefined entity classes. Examples of such entity classes are people (e.g., Boris Johnson, Vladimir Putin), locations (e.g., London, Abidjan), and organisations (e.g., Google, International Business Machines Corporation). The report outlines the project's objective to implement and evaluate at least two distinct approaches to NER, leveraging a standard benchmark dataset for assessment. It provides a comprehensive literature review, an insightful analysis of the NER problem and the different approaches developed to date, an overview of the design and implementation process, and the results obtained followed by a discussion.

Acknowledgements

I would like to thank my supervisor Professor Rob Gaizauskas for his help and guidance throughout this project. I am thankful for the opportunity to benefit from his extensive expertise and knowledge. I would also like to thank my parents for their encouragements.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Named Entity Recognition (NER) is an important task. It serves as a critical component in information extraction from articles, books and newspapers and often acts as an initial step in a range of natural language processing (NLP) tasks. For instance, NER plays a key role in question answering systems by identifying essential entities within the text and enhancing the precision of answers. In sentiment analysis, it helps highlighting the target of a consumer's opinion, providing businesses with deeper insights on their products or services (Jurafsky and Martin, 2023). Other areas benefiting from named entity recognition include document retrieval as well as the development and training of chat-bots and virtual assistants. Despite its widespread recognition and extensive research, NER still faces several challenges. One major issue is type ambiguity; *America* can refer to a continent or an actress depending on the context. Another difficulty lies in segmentation, as entities could be single words but also spans of text. It is therefore necessary to accurately determine which parts of a text constitute an entity.

1.1 Aims and Objectives

The aim of this project is to design, implement, and evaluate a variety of Named Entity Recognition (NER) systems using the data-set from the 2003 Conference on Computational Natural Language Learning (CoNLL-2003) shared task. This includes a thorough review of current natural language processing and machine learning tools and frameworks, as well as an investigation into both traditional and deep learning approaches for NER.

Several systems are being built to compare the performance of different methodologies under varying conditions and configurations. They will fundamentally differ in aspects such as the underlying algorithms and data pre-processing strategies. The purpose of building more than one system is to determine which approaches are most effective for the NER task in terms of accuracy, speed, and robustness across different types of text. By comparing different approaches, this project aims to uncover insights into the strengths and weaknesses of each method, contributing to a deeper understanding of NER technologies.

1.2 Overview of the Report

This report begins with an overview of the named entity recognition task, including a description of the data provided. Chapter 2 then provides a literature review discussing common approaches and key ideas relevant to named entity recognition. Next, chapter 3 describes the analysis that has been done to date and outlines the rationale behind choosing specific solutions for our own systems. Chapter 4 covers the design decisions behind these approaches, and chapter 5 describes their implementation. Finally, chapter 7 presents the results and draw conclusions.

Chapter 2

Literature Survey

This chapter aims to methodically explore the critical aspects of the dissertation. It first presents and details the Named Entity Recognition task and the latest advances in the domain. Henceforth, it offers a comparative analysis of two distinct approaches to NER, analysing their methodologies, and performance. Finally, it conducts a comprehensive evaluation of the available tools for natural language processing, identifying those that could be effective for Named Entity Recognition. Therefore, the literature review offers a critical examination of the domain, blending an overview of cutting-edge developments and a comparison of methodologies, providing an understanding of the current research landscape.

2.1 The Named Entity Recognition task

2.1.1 Definition

The Named Entity Recognition concept was initially introduced at the Sixth Message Understanding Conference, becoming a cornerstone in natural language processing (Liu et al., 2022). NER is a sub-task of information extraction and essentially consists of identifying and categorising distinct proper nouns from unstructured text. It encompasses several predefined types; for instance, names of individuals, locations, and organisations. Liu et al. (2022) explains that over time, the academic community has refined these categories, distinguishing, for example, geographic locations into more specific groups such as countries, provinces, states, and cities.

2.1.2 The Conference on Computational Natural Language Learning

Tjong Kim Sang and De Meulder (2003) reported that the goal of the 2003 shared task of the Conference on Natural Language Learning was to develop machine learning systems for named entity recognition, more specifically language-independent entities. To achieve this result, a data set has been provided. It contains annotation data for two languages, English and German, across eight distinct files: a training file, a development file, a test file and a large file with unannotated data for each language. The English data was sourced from the Reuters

Corpus which consists of news stories from August 1996 to August 1997 and the German data originates from the German newspaper Frankfurter Allgemeine Zeitung retrieved from the Rundshau ECI Multilingual Text Corpus. In the following sections, the focus will be particularly on the English data. The named entities tagged in the CoNLL-2003 data belong to four categorical classes which are organisations, locations, persons and miscellaneous for names of entities that don't fall in any of these three categories. They are represented respectively by *ORG*, *LOC*, *PER*, and *MISC*. For this project, the CoNLL-2003 data set has been downloaded from the *Hugging Face* website.[1]

English data	Sentences	Tokens
Training set	14,041	203,621
Development set	3,250	51,362
Test set	3,453	46,435

Table 2.1: Number of sentences and tokens in each partition of the data set.

English data	LOC	MISC	ORG	PER
Training set	7,140	3,438	6,321	6,600
Development set	1,837	922	1,341	1,842
Test set	1, 668	702	1,661	1,617

Table 2.2: Number of instances of each named entity type per data set.

Table 2.1 shows the partition of the data set. For each set, a row represents a sentence as a list of words along with its id and the part of speech, chunk and named entity recognition tags for each of the tokens also stored in different lists. It can be seen that the training data is more voluminous than both the development and test set which can pose an over-fitting problem. Furthermore, the majority of the tokens in a text do not belong to any named entity type and performing training on the data-set without any adjustments can cause a class imbalance problem, where the model might become biased towards predicting the tag "O". Table 2.2 indicates that Location is the most common entity type and the less common type is Miscellaneous in each subset.

2.1.3 BIO and NER tagging

The CoNLL-2003 data-set employs the BIO tagging scheme to detail the position of tokens within named entities, indicating whether a token is at the Beginning, Inside, or Outside of a named entity. This approach allows NER to be treated as a sequence labeling task (Jurafsky and Martin, 2023).

The utilisation of BIO tagging is instrumental in defining the extent of entities throughout the classification process, effectively capturing both the entity's boundaries and its category, as demonstrated by the examples provided in table 2.3 . The systems to be developed should

[1] https://huggingface.co/datasets/conll2003

Token	NER tag
The	O
Finance	B-ORG
Ministry	I-ORG
raised	O
the	O
price	O
for	O
tap	O
sales	O

Table 2.3: Tokens in a sentence of the training data and their NER tags.

therefore follow this scheme. Any token at the start of a span of interest is labelled B, tokens that occur inside a span are tagged with an I, and those outside are labeled O. Distinct B and I tags are given for each named entity class (Jurafsky and Martin, 2023). In this case, there are thus nine different tags.

2.2 Feature engineering approach

Feature engineering is the process of converting raw data into meaningful information representing a certain attribute of the observations. This information is then fed into a classifier (Verdonck et al., 2021). Examples of such features for a word in the NER domain include embeddings, part of speech tags, and the presence of the word in a gazetteer.

This process is helpful for developing machine learning models that are both accurate and efficient. By carefully selecting and modifying relevant data attributes, feature engineering allows for the extraction of deeper insights and the identification of significant patterns within the data. This method relies on domain knowledge and experimentation to optimise the feature space. The goal is to create features that are effective substitutes for intricate characteristics.

2.2.1 Multinomial logistic regression

This section is based upon Jurafsky and Martin (2023). Logistic regression is a type of probabilistic classifier algorithm that uses supervised learning. Its application to more than two classes is called multinomial logistic regression. The classifier requires a training data-set containing input features mapped to their corresponding output labels. The input-output pairs are **(x,y)** where **x** refers to the feature vector for a training instance and **y** is a one-hot vector representing the true class label, *i.e.* the NER tag. **x** is usually a vector of features, for example, in our case, word embedding, part-of-speech tags or capitalisation. For a **K** classes problem, the goal of the classifier is to generate a prediction vector $\hat{\mathbf{y}}$. Specifically, for each class k, the classifier estimates the value $\hat{y}_k$ which is the probability that y_k is the true class label given the input features. The class with the highest $\hat{y}_k$ is returned.

This is achieved by learning two parameters, a weight matrix **W** and a bias vector **b,** using the labelled training corpus. The weight matrix allows each input feature to contribute differently to the outcome based on its class and relevance because each row k of W corresponds to the vector of weights w_k of class K and is associated with the input features vector **x** , while the bias helps fit decision boundaries. To decide the appropriate class of an input, the classifier makes use of the following equation described by Jurafsky and Martin (2023).

$$\mathbf{z} = \mathbf{W} \cdot \mathbf{x} + \mathbf{b} \tag{2.1}$$

A non linear function is then applied to the vector z to compute the probability $\hat{y}_k$ of each class. A function commonly used for multinomial logistic regression is the softmax function. For the vector previously defined, the softmax of one entry is:

$$softmax(\mathrm{z}_i) = \frac{exp(z_i)}{\sum_{j=1}^{K} exp(z_j)} \tag{2.2}$$

Softmax is a generalisation of the sigmoid that takes a K-dimensional vector $z = [z_1, z_2, ..., z_K]$ and maps its values to a probability distribution; each value is in the range [0,1] and all the values sum to 1. The final output of the classifier is:

$$\hat{\mathbf{y}} = softmax(\mathbf{W} \cdot \mathbf{x} + \mathbf{b}) \tag{2.3}$$

Adjusting W and b minimises prediction error on the training examples. One way of doing that is by combining a loss function and gradient descent. For multinomial logistic regression, the cross entropy loss function is commonly used. It estimates how close the classifier output $\hat{y}$ is to the true class label y. The cross entropy loss for multinomial logistic regression is generalised from binary logistic regression to our K classes problem:

$$L_{CE}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{k=1}^{K} \mathrm{y_k} log \hat{\mathrm{y}}_\mathrm{k} \tag{2.4}$$

$$L_{CE}(\hat{\mathbf{y}}, \mathbf{y}) = -log \frac{exp(\mathrm{w_c} \cdot \mathrm{x} + b_c)}{\sum_{i=1}^{K} exp(\mathrm{w_i} \cdot \mathrm{x} + b_i)} \tag{2.5}$$

In equation 2.5 c is the correct class of the sample. Since $\mathrm{y}_c = 1$ and all others $\mathrm{y}_j = 0, \forall j \neq c$, all the terms in the sum in equation 2.4 evaluate to 0 except for the term corresponding to the true class. The objective is to find a set of weights which minimises the loss function defined above. Gradient descent finds a local minimum of a function by determining in which direction the function's slope is rising the most steeply, and moving in the opposite direction. The gradient of the loss function will combine the derivatives for all the weights; but for each class k, the partial derivative of the loss with respect to the weight of the i-th element of the input vector x, $w_{k,i}$ is:

$$\frac{\partial L_{\mathrm{CE}}}{\partial \mathrm{w}_{k,i}} = -\left(\mathbf{y}_k \hat{\mathbf{y}}_k\right) \mathbf{x}_i = -\left(\mathbf{y}_k - \frac{\exp\left(\mathbf{w_k} \cdot \mathbf{x} + b_k\right)}{\sum_{j=1}^{K} \exp\left(\mathbf{w_j} \cdot \mathbf{x} + b_j\right)}\right) \mathbf{x}_i \tag{2.6}$$

2.2.2 Conditional Random Fields

The following section is based upon Jurafsky and Martin (2023). A Conditional Random Field (CRF) is a discriminative model that, similar to multinomial logistic regression, is grounded in a log-linear framework. In contrast, a CRF operates as a sequence model. This discussion will focus on the linear-chain CRF, the variant most frequently applied in language processing tasks. Linear chain CRFs are highlighted for their ability to model the conditional probability of label sequences given observation sequences, making them suitable for tasks like named entity recognition. They are preferred for their efficiency in incorporating context and dependencies through features, thus improving prediction accuracy by directly modelling the label sequence in relation to the observed data.

A CRF does not calculate a probability for each tag at every time step but evaluates log-linear functions across a set of pertinent features at each step. These local features are then combined and normalised to generate a global probability of an entire output sequence Y given an input sequence X. The sequence of output tags is computed by directly calculating the posterior probability $p(Y|X)$.

$$\hat{Y} = P(Y \mid X) \tag{2.7}$$

A CRF can be thought of as an expanded version of what multinomial logistic regression accomplishes for a single token. Let's suppose we have K features, where each feature F_k is associated with a weight w_k:

$$p(Y \mid X) = \frac{\exp\left(\sum_{k=1}^{K} w_k F_k(X, Y)\right)}{\sum_{Y' \in Y} \exp\left(\sum_{k=1}^{K} w_k F_k\left(X, Y'\right)\right)} \tag{2.8}$$

Typically, the equation is reframed by isolating the denominator into a function $Z(X)$:

$$p(Y \mid X) = \frac{1}{Z(X)} \exp\left(\sum_{k=1}^{K} w_k F_k(X, Y)\right) \tag{2.9}$$

$$Z(X) = \sum_{Y' \in Y} \exp\left(\sum_{k=1}^{K} w_k F_k\left(X, Y'\right)\right) \tag{2.10}$$

We refer to these K functions $F_k(X, Y)$ as global features, as each is characteristic of the entire input sequence X and output sequence Y. We calculate them by breaking down into a sum of local features for each position i in Y.

$$F_k(X, Y) = \sum_{i=1}^{n} f_k\left(y_{i-1}, y_i, X, i\right) \tag{2.11}$$

In a linear-chain CRF, each local feature f_k can utilise the current output token y_i, the preceding output token y_{i-1}, the entire input string X or any subpart of it, and the current position i. The restriction to only the current and previous output tokens, y_i and y_{i-1}, defines

a linear-chain CRF. This limitation enables the use of efficient algorithms such as the Viterbi and Forward-Backward algorithms.

2.2.3 Choosing features

As mentioned above, the input of the classifier is a vector of features. A variety of feature types can be considered to capture syntactic, lexical, semantic patterns and even word shapes, all of which are valuable for identifying named entities. Jurafsky and Martin (2023) present common features for a feature-based NER system which include the following: the specific word w_i, the words adjacent to w_i, embeddings for w_i and its neighbouring words, the part of speech for w_i and its neighbours, the presence of w_i in a gazetteer, specific prefixes in w_i, specific suffixes in w_i, the word shape of w_i and its neighbouring words, a simplified word shape of w_i and its neighbouring words.

A gazetteer is a massive list of place names. It typically offers millions of entries with detailed geographical and political information and can be used as a binary feature indicating whether a phrase is present in the list. Other entity dictionaries, including corporate listings and name-lists can also be employed(Jurafsky and Martin, 2023).

Moreover, the CoNLL-2003 data set already stores some features for each of the tokens namely the part-of-speech tags and the chunk tags. There are 46 part-of-speech (POS) tags categorising words according to the role played in a sentence; the data set from Hugging Face uses the Penn Treebank corpus. For example *VB* denotes a verb base and *SYM* is a symbol (Jurafsky and Martin, 2023). On the other hand, chunk tags are non-overlapping, non-recursive phrases including noun phrases, verb phrases, and adjective phrases, which also use the BIO tagging (Sang and Veenstra, 1999).

Token	POS tag	Chunk tag
EU	NNP	B-NP
rejects	VBZ	B-VP
German	JJ	B-NP
call	NN	I-NP
to	VB	B-VP
boycott	VB	I-VP
British	JJ	B-NP
lamb	NN	I-NP

Table 2.4: POS and chunk tags of the first sentence of the training data.

POS tags can reflect the likelihood for those tokens to be part of a named entity. As an example, a proper noun is likely to be a person, an organisation, or a location. Additionally verbs and prepositions provide context for segmentation and span boundaries. The POS and chunk tags highlight grammar and syntax cues which are useful in information extraction tasks and allow the system to encode broader linguistic notions that characterise entities instead of just memorising word patterns (Sang and Veenstra, 1999). The model can identify new or rare entities having similar tag patterns to known entities.

Additionally, another useful feature to consider is word embeddings: the representation of words as real-valued vectors. Since training word embeddings is computationally expensive, it is preferable to use pre-trained models. Numerous models for semantic vector spaces exist, but the focus will be particularly on *GloVe*. GloVe, or Global Vectors for Word Representation, is an unsupervised learning model which blends the strengths of two approaches: global matrix factorisation and local context window techniques (Toshevska et al., 2020).

Probability and Ratio	k = solid	k = gas	k = water	k = fashion
$P(k \mid \text{ice})$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k \mid \text{steam})$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k \mid \text{ice})/P(k \mid \text{steam})$	8.9	8.5×10^{-2}	1.36	0.96

Table 2.5: Co-occurrence probabilities for words ice and steam (Pennington et al., 2014).

As shown in table 2.5, the core concept is that the co-occurrence ratio, *i.e.* the frequency with which two words appear in each other's context, holds information about these words. GloVe uses the statistics of global word-word co-occurrence matrices and the model is fine-tuned so that the dot product of any two word vectors is equal precisely to the corresponding words' probability of co-occurrence.

2.3 Deep learning approach

Vargas et al. (2017) describe Deep learning as a cutting-edge field within machine learning research. It enables computational models consisting of numerous processing layers to learn data representations across various levels of abstraction. By leveraging the back-propagation algorithm deep learning methods can uncover complex structures within vast datasets (LeCun et al., 2015). In contrast to the feature engineering approach which involves manually selecting features, deep learning techniques learn how to represent features directly while training on data. The back-propagation algorithm guides the adjustment of a model's internal parameters, allowing each layer to refine its representation based on the information from the preceding layer. This approach has led to a significant improvement of performance in natural language processing tasks, setting new benchmarks. The subsequent sections discuss the two highest- performing architectures. Section 2.3.1 introduces recurrent neural networks, which initially set the standard for performance, only to be eclipsed by transformers described in section 2.3.2.

2.3.1 Recurrent neural networks

A recurrent neural network (RNN) is a type of network that includes a cyclical connection in its architecture: the output of a unit, at a given time, can serve as an input to the same unit at a later time, either directly or indirectly (Jurafsky and Martin, 2023).

Similarly to logistic regression, in recurrent neural networks an input vector x_t is multiplied by a weight matrix. This product then goes through a non-linear activation function to

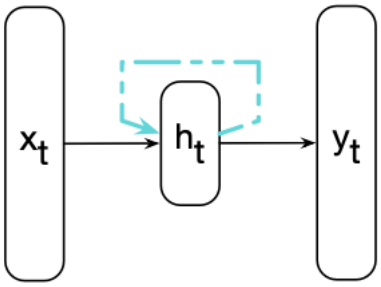

Figure 2.1: A simple recurrent neural network (Jurafsky and Martin, 2023).

determine the values for a layer of hidden units. These values from the hidden layer are subsequently utilized to compute the corresponding output y_t. The main difference is a recurrent link, as shown in figure 2.1, augmenting the input to the computation at the hidden layer with the value of the hidden layer from the previous time point. The hidden layer from the preceding time gives context for later times t (Jurafsky and Martin, 2023).

2.3.2 Transformers

One limitation of the RNN architecture highlighted by Tunstall et al. (2022) is the *information bottleneck* created by the encoder's final hidden state. It must encapsulate the entire input sequence's meaning, as it is the only information available to the decoder for generating output. This becomes particularly problematic with longer sequences, where initial sequence information may be lost. However, this bottleneck can be alleviated by enabling the decoder to access all the hidden states of the encoder using transformers.

Transformers refer to a groundbreaking model architecture that was introduced by Vaswani et al. (2017). This model outperformed RNNs and has significantly advanced the handling of natural language processing tasks due to its effectiveness and efficiency. The key feature of transformers is the *Attention mechanism*. Unlike previous models that processed data sequentially, transformers use self-attention mechanisms to evaluate the significance of different words in a sentence. Tunstall et al. (2022) emphasises that Attention allows transformers to focus on the most pertinent input tokens at every step. As a result, attention-based models are able to learn contextual relationships between words regardless of their relative position. They can be trained on larger data sets more efficiently, leading to better performance on complex language tasks. Furthermore, the transformer architecture demonstrates great scalability with respect to data and computational resources, making it suitable for large-scale tasks. Popular transformers such as Generative Pre-trained Transformer (GPT) and Bidirectional Encoder Representations from Transformers (BERT), have set new standards. Models are often pre-trained on a large corpus of text and then fine-tuned for specific NLP tasks including NER. Transformer models achieve state-of-the-art results in natural language processing.

2.4 Tools and tool-kits

A number of software tools and libraries have been developed specifically for natural language processing and can be used for Named Entity Recognition. This section will survey some of the leading open-source tool-kits. Frameworks have different features and capabilities and meet different requirements in terms of language support, accuracy, processing efficiency or ease of integration.

2.4.1 Language Processing tool-kits

A language processing toolkit is designed for processing and understanding human language data including text. These tool-kits come equipped with pre-built functions and models tailored for most NLP tasks such as tokenisation, parsing, part-of-speech tagging, and named entity recognition. They are engineered to handle the complexities of human language, covering syntax, semantics, and context.

spaCy

spaCy is an open-source library for natural language processing in Python renowned for its efficiency. The library includes a range of pre-trained models and word vectors in multiple languages, capable of performing various tasks including tokenisation, part-of-speech tagging, and named entity recognition. spaCy's models are frequently updated and bench-marked against other NLP tools (Explosion AI, 2023). It makes advanced NLP accessible, while allowing integration with other models and training on custom data-sets. The library integrates with deep learning frameworks like PyTorch and TensorFlow.[2]

Pipeline	Accuracy for NER task
en_core_web_trf	89.8
en_core_web_lg	85.5

Table 2.6: Full pipeline accuracy on the OntoNotes 5.0 corpus (Explosion AI, 2023).

Table 2.6 compares the accuracy of different spaCy pipelines when evaluating their performance for Named Entity Recognition on the subset of the OntoNotes 5.0 corpus. It is a large corpus containing various genres of text including news, conversational telephone speech, weblogs in three languages English, Chinese, and Arabic annotated with syntactic, semantic and discourse information (Pradhan et al., 2013). Both en_core_web_lg, which is spaCy English pipeline optimised for CPU, and en_core_web_trf, the English transformer pipeline achieve relatively high performance.

[2]https://spacy.io/usage/facts-figures

NLTK

NLTK, the Natural Language Toolkit in Python, is a library for applications in computational linguistics and NLP. It offers user-friendly access to more than fifty corpora and lexical databases, including WordNet, and a collection of text processing tools for tasks like classification, tokenization, stemming, tagging, parsing, and semantic analysis. It also features a pre-built NER chunker, a POS tagger and custom NER models while allowing integration with other tools.[3]

Hugging Face transformers

Tunstall et al. (2022) presents the Hugging Face transformers, a library that offers models but also provides code and tools for tailoring these models. It is compatible with leading deep learning frameworks: PyTorch, TensorFlow, and JAX. Moreover, it is possible to fine-tune transformers, for example, for the named entity recognition task, so that models can be trained and tested faster.[4]

2.4.2 Machine learning frameworks

A machine learning framework is a library or a collection of libraries which provides a versatile structure and resources for the development, training, and deployment of machine learning models. Such frameworks offer high-level abstractions for complex algorithms, simplifying the process for developers and researchers to craft advanced systems. Commonly, these frameworks are equipped to manage data, construct models, train, and make inferences.

Scikit-learn

Scikit-learn is a machine learning library for Python. It features various classification, regression, and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. It can be used to implement logistic regression for Named Entity Recognition. However, since it is related to general machine learning, it has some constraints. To use logistic regression for NER, manual feature engineering would be needed as scikit-learn doesn't process text data. After converting the tokens to numerical features, logistic regression can then be applied. While this method might not be close to the sophistication of deep learning models, it can still provide meaningful insights.[5]

PyTorch

PyTorch is an open-source machine learning library based on the Torch library, used for different applications including computer vision and natural language processing. It is primari-

[3]https://www.nltk.org
[4]https://huggingface.co/docs/transformers/index
[5]https://scikit-learn.org

ly developed by Facebook's AI Research lab (FAIR). PyTorch provides two high-level features. On one hand , tensors which compute with strong acceleration via graphics processing units. Tensors are a type of data structure used in linear algebra, and they are very similar to arrays and matrices. In PyTorch, tensors are optimised for automatic differentiation, which is a key feature for deep learning algorithms where gradients need to be calculated and back-propagated through neural network layers On the other hand, it offers deep neural networks allowing for dynamic creation and modification of neural networks during runtime, which is particularly useful for research and development of complex models. PyTorch's dynamic computational graph means that the network's behaviour can be changed programmatically at runtime, unlike in static graph frameworks where the graph is defined and optimised before the model is run. PyTorch is known for its simplicity, flexibility, and user-friendly interface.[6]

TensorFlow

TensorFlow is an open-source software library for numerical computation using data flow graphs. Originally developed for the purposes of conducting machine learning and deep neural networks research, TensorFlow is now widely used for a variety of computational tasks. It provides a comprehensive ecosystem of tools, libraries, and community resources that allows researchers to push the state-of-the-art in machine learning, and developers to easily build and deploy machine learning powered applications. Computational operations, variables, and data are all represented as nodes in a graph of data flows. This representation allows for efficient parallel computation and optimisation of the processes involved in learning and inference. It is designed to scale from small to very large computations, from single devices to clusters of devices, with optional distributed computing support. TensorFlow is used for a wide range of tasks but has particular strengths in classification, perception, understanding, discovering and predicting.

Keras

Keras is an open-source neural network library written in Python designed to enable fast experimentation. It provides a straightforward, high-level interface for building and training deep learning models, making complex concepts more accessible. Models are assembled module-by-module, layer-by-layer; this modular design is highly flexible and allows for total customisation. Keras comes with numerous pre-built layers and models for a variety of common tasks, such as convolutional networks, recurrent networks, and combinations of the two. This extensive library of building blocks facilitates the construction and experimentation of different models with ease. The design philosophy emphasises ease of use, modularity, and extensibility making it popular for beginners learning about deep learning.[7]

[6]https://pytorch.org
[7]https://keras.io

Chapter 3

Analysis

This chapter defines the essential requirements and specifications for the project, detailing the programming tools needed and assessing potential risks involved. Additionally, it will cover ethical, professional, and legal considerations relevant to the project.

3.1 Project Requirements

This project aims to evaluate the effectiveness of deep learning approaches for the NER task by comparing them to other approaches but also to state-of the-art results. The initial step requires choosing an appropriate dataset, critical for the project's success. The CoNLL-2003 dataset is designed specifically for the named entity recognition task in the English language. This dataset is already partitioned for training, testing and validation and includes annotations for named entities namely persons, organisations, locations, and miscellaneous names within sentences. Although other datasets were considered, the CoNLL 2003 dataset was selected as it is a well-known benchmark across the field and offers a rich corpus with more than ten thousand training sentences.

To fulfill the objectives of this project, four approaches to named entity recognition are implemented: Multinomial logistic regression, Conditional Random Field, Transformers, and a SpaCy pipeline. The code is written in Python and executed in both the terminal using a 24 GB Apple M2, and Jupyter notebook. Python is a simple, interpreted, object-oriented and versatile programming language commonly used for machine learning tasks. Python's extensive library ecosystem includes useful tools and allows for the development and implementation of algorithms.

3.2 Algorithms choice

Various algorithms were implemented based on their strengths and how they match the needs of the project. First, spaCy is chosen, as it is beginner-friendly and widely used in the industry. It is therefore a good starting point for designing solution to the NER task. Furthermore, multinomial logistic regression and conditional random fields are selected

to compare the effectiveness of token-based versus sequence-based labelling . Multinomial logistic regression, though powerful, has a notable limitation as it does not account for the dependencies between tokens:it does not consider the output of the previous token while CRFs analyse the whole sequence. Lastly, to illustrate the latest advancements in natural language processing, transformer models were also employed. Their ability to handle long-range dependencies and their state-of-the-art performance in numerous NLP tasks made them an indispensable part of this project. Each of these choices was strategic, aiming to uncover specific algorithmic strengths to address different aspects of the NER task.

3.3 Tools choice

spaCy has been chosen for its efficiency and customisation capabilities. It excels in fast text processing and offers extensive NLP features like tokenisation and entity recognition, making it ideal for our task. It incorporates deep learning techniques to enhance its NLP capabilities and integrates custom components into its processing pipeline.[1]

Additionally, scikit-learn is the chosen tool for both multinomial logistic regression and CRF approaches. It provides resources for the specified algorithms, therefore simplifying the models development process. It is known for its simplicity, versatility, and performance for classification tasks making it ideal for building and training machine learning models on text data.

Finally, the Hugging Face's transformers library has been chosen for the deep learning approach. Its vast array of pre-trained models, including BERT, GPT, and RoBERTa, provides state-of-the-art performance in named entity recognition. The library's flexibility is another key feature, allowing extensive customisation and fine-tuning to suit specific data-sets and domains. Moreover, its integration with leading machine learning frameworks like PyTorch and TensorFlow facilitates the incorporation into diverse workflows.

Together, these tools cover a comprehensive range of NLP needs, from initial text processing to the development and deployment of sophisticated machine learning models. For each implementation, the text has to be processed into the required format before the model can generate predictions and be evaluated. It is also vital to ensure that the system outputs adhere to the same naming conventions as the data-set, to guarantee the accuracy of performance measurements.

3.4 Evaluation metrics

Accuracy, precision and recall commonly serve as metrics to evaluate the effectiveness of classification systems, allowing for comparison with previous outcomes. These measures can be defined in terms of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN).

[1] https://spacy.io/usage/facts-figuresbenchmarks

Accuracy is defined as the proportion of correctly classified instances relative to the total number of instances classified (Novaković et al., 2017).

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{3.1}$$

Novaković et al. (2017) emphasise the primary drawbacks of using accuracy as an evaluation measure namely its failure to differentiate between types of errors and its dependence on the distribution of classes in the dataset.

In contrast, precision refers to the proportion of entities that a system identifies correctly out of the total entities it returns. Recall, on the other hand, indicates the proportion of relevant entities that a system successfully identifies and returns (Derczynski, 2016).

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

According to Derczynski (2016), these metrics, precision and recall, can be manipulated in such a way that a system exhibits behaviours yielding high scores with minimal practical value. A system could theoretically return every conceivable answer guaranteeing that all correct answers are included. This approach would result in a perfect recall score. Similarly, for precision, if the system is designed to return one correct answer it would achieve the highest possible precision score due to the absence of false positives and at least one true positive. Therefore using either one of them by itself is meaningless.

It's a standard approach to merge precision and recall into a single measure using a weighted harmonic mean, known as the F-score, to provide a more balanced evaluation.

$$F_\beta = \frac{(\beta^2 + 1) * precision * recall}{\beta^2 * precision + recall} \tag{3.4}$$

β can be varied and recall is considered β times as important as precision. The CoNLL-2003 shared task measures performance with $F_{\beta=1}$ (Tjong Kim Sang and De Meulder, 2003).

3.5 Ethical, Professional and Legal Issues

The project utilises an open-source data set, which ensures freedom from legal issues commonly associated with the use of proprietary data. It complies with specific licenses that permit its unrestricted use, alteration, and sharing, as long as any stipulated terms are followed. Moreover, the project upholds ethical research practices by avoiding any unfair methods. This ethical commitment includes a focus on accuracy, transparency, and impartiality in all aspects of data processing and analysis. Additionally, to ensure the reproducibility of the work, a detailed documentation of the methodologies will be maintained, access to the code will be granted, and version control systems will be used for all datasets and software used.

3.6 Risk Analysis

No.	Risk	Likelihood	Severity	Mitigation
1.	Failure to understand the objectives of the project	Low	High	Regular meetings with supervisor, read relevant materials and ask questions when needed
2.	Lose files and progress	Low	High	Set up a Github repository for the project and make regular backups
3.	Difficulties implementing one of the approaches	Medium	High	Read about similar implementations, use pre-trained models, ask for help
4.	Data-set being subjected to privacy and copyright	Medium	Medium	Get the data-set from an open source website
5.	Running out of time	High	High	Focus on the approach with the most progress, expanding its depth and implications, while acknowledging the unexplored approach as a direction for future research

Table 3.1: Risk register

Table 3.1 presents the potential risks associated with this project, their likelihood and severity, and the measures taken to mitigate their effect.

Chapter 4

Design

This chapter introduces the architecture and design of the algorithms implemented for the task. It presents the libraries and tools employed in crafting these NER systems and provides insight into the choices made at each step of the development process. Additionally, the reasoning behind these choices is discussed, highlighting the trade-offs and decisions that were essential in shaping the implementation strategy.

4.1 Libraries

The core Python libraries used in the development process are Scikit-learn, spaCy, and Transformers. They provide a comprehensive suite of tools that facilitate the intricate process of NER, from initial data handling to the final stages of entity classification. This section will walk through the specific functionalities of each library, detailing how they are applied within the project.

4.1.1 Scikit-learn

Multinomial Logistic Regression

Multinomial logistic regression extends binary logistic regression to multi-class problems. In Scikit-learn, this classifier is implemented under `LogisticRegression` with the option multi_class set to multinomial. It handles single tokens rather than whole sequences. The parameters of the model are typically estimated using maximum likelihood, with the help of optimisation algorithms. This result in a prediction of the probability distribution over multiple classes for a given observation after training.

Conditional Random Fields

Scikit-learn's crfsuite is a machine learning library implementing Conditional Random Field models, designed for sequence labelling. Features such as word tokens, parts of speech, or other contextual cues can manually be defined from data to enhance the performance.

The learning process involves training the model to understand the relationship between the sequence's label and the optimisation of parameters using the chosen algorithm. The optimisation of the CRF model can be carried out using algorithms like Stochastic Gradient Descent, which are configurable within sklearn-crfsuite. The parameters are fine-tuned during this process to prevent the model from over-fitting, maintaining a balance between model complexity and performance on unseen data. Post training, the model uses learned weights to decode the most likely label sequence for new input data.

4.1.2 spaCy

The en_core_web_lg model from spaCy is a large English language featuring a wide range of linguistic annotations including named entity recognition. When processing text for NER, the pipeline first tokenises the text, breaking it down into words, punctuation, and other elements. It then predicts the syntactic dependencies of each token and uses these along with other token-level features such as word shapes and the surrounding context to predict entity labels. The model covers more entity types than the four mentioned in previous sections including Cardinal, Date, Event, Fac (Facility), GPE (Geo-political entity), Language, Law, Loc (Location), Money, NORP (Nationalities or religious or political groups), Ordinal, Org (organisation), Percent, Person, Product, Quantity, Time, Work of art. It applies a transition-based approach to ensure consistent label sequences making it robust for complex and varied text inputs.

4.1.3 Hugging Face Transformers library

The Hugging Face Transformers library provides a powerful and flexible platform for natural language processing tasks including NER. It offers models such as BERT, and DistilBERT. Specifically, bert-base-NER is a model fine-tuned on the English version of the CoNLL-2003 data-set ready for immediate use[1]. The process involves tokenising text into sub-words, feeding these tokens into the transformer model, training the model, and then predicting entity labels for each token.

4.2 General algorithm

For the multinomial logistic regression, the CRF and transformers approaches, and the spaCy pipeline, the general process can be thought of as being the same. Each system begins with the dataset loading, which is already partitioned into training, test, and validation sets. Following this initial step, the training data undergoes a pre-processing phase which includes its formatting for each approach, ensuring that the data is in the appropriate form for the subsequent stages. Multinomial logistic regression, CRF and BERT models are then trained on the data-set and used to effectively classify the sentences which are processed as lists of tokens. The spaCy model does not require training. The final phase is evaluation, which is

[1] https://huggingface.co/dslim/bert-base-NER

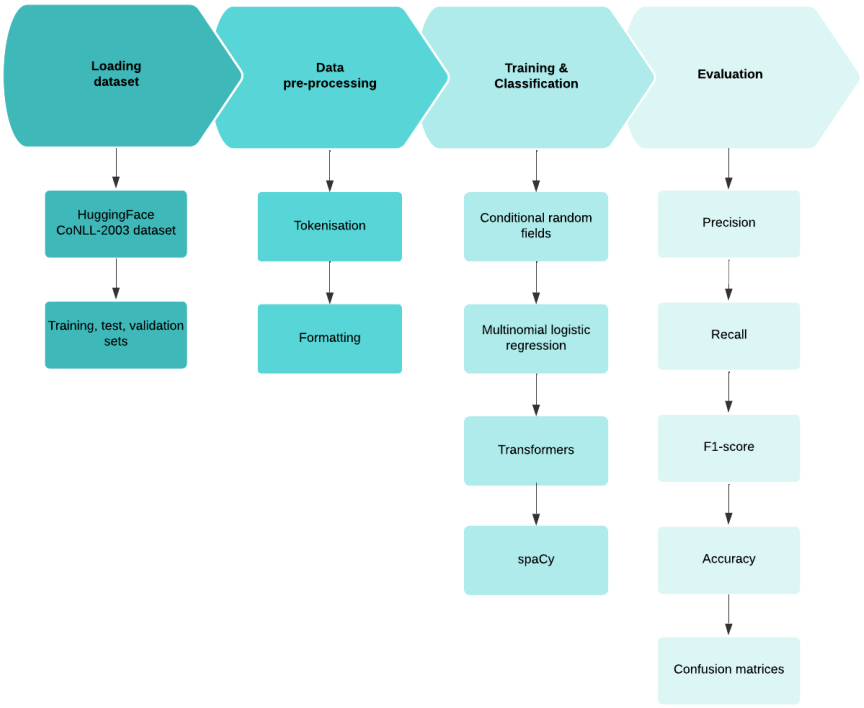

Figure 4.1: Flow chart of the project

critical to ascertain the performance of the system. This is accomplished through precision, recall, and the F1-score, with confusion matrices providing a detailed representation of the system's classification accuracy. Figure 4.1 illustrates the process.

Chapter 5

Implementation and Testing

This chapter details the implementation of the Multinomial Logistic Regression, Conditional Random Field, and Transformers algorithms along with the integration of the spaCy pipeline. The methodologies adopted for each approach is detailed, offering a transparent view of the procedures and requirements necessary to use the models. This includes an overview of the pre-processing steps, feature extraction techniques, and the nuances of the algorithms training.

5.1 Data collection

The Hugging Face CoNLL-2003 dataset, as detailed in prior discussions, serves as the data-set for this project. It categorises text data into four distinct named entity types: persons, locations, organisations, and a miscellaneous category for entities that don't neatly fall into the first three categories. Structurally, the dataset is formatted such that each partition of the data-set consists of rows of sentences. There are five columns: the id of the sentence, the sentence as a list of tokens, with accompanying annotations POS tags, syntactic chunk tags, and NER tags, respectively. Table 5.1 shows the different features of an item in the dataset. This structure allows items to be accessed and exploited efficiently. The data-set can be retrieved by a simple python import.

id	tokens	pos tags	chunk tags	ner tags
1	Peter, Blackburn	22, 22	11, 12	1, 2
2	BRUSSELS, 1996-08-22	22, 11	11, 12	5, 0

Table 5.1: Entries 1 and 2 of the training set.

POS, NER, and chunk tags are stored as integers and can be converted to their literal values via provided dictionaries as highlighted by 5.2.

Entity	Integer label
O	0
B-PER	1
I-PER	2
B-ORG	3
I-ORG	4
B-LOC	5
I-LOC	6
B-MISC	7
I-MISC	8

Table 5.2: NER tags integer correspondences

5.2 Data pre-processing and text representation

The approaches developed employ various strategies for text representation and tokenisation, reflecting the multifaceted nature of natural language processing. These differences underscore the need for careful alignment with the dataset formatting. The systems can then effectively use the information contained within the data, ultimately enhancing the performance of the models.

5.2.1 Scikit-learn Multinomial Logistic Regression

For this model to be used, the provided numeric tags are converted to string labels; the tokens are then represented as numeric features using the `CountVectorizer` from scikit-learn. This vectoriser converts the text data into matrix of token counts, essentially creating a bag-of-words model where each column represents a unique token and each row represents a sentence. Since the classifier does not support sequence labeling, the input consist of two flat lists: one containing the labels, and the other the extracted features. Multinomial logistic regression operates at token level rather than sentence level.

5.2.2 Scikit-learn Conditional Random Field

The process of preparing data for the scikit-learn conditional random field begins with the reformatting of the data-set for feature extraction including converting NER tags to their string equivalents, and pairing each token with its corresponding NER tag into tuples.

Features encompassing both the intrinsic properties of the word such as lowercase form, suffix, title casing and the contextual features derived from neighbouring words are retrieved as highlighted by figure 5.1. Additionally, the model receives positional context through tags specifying the start (BOS) or end (EOS) of a sentence.

```python
def word2features(sent, i):
    word = sent[i][0]
    features = {
        'bias': 1.0,
        'word.lower()': word.lower(),
        'word[-3:]': word[-3:],
        'word[-2:]': word[-2:],
        'word.isupper()': word.isupper(),
        'word.istitle()': word.istitle(),
        'word.isdigit()': word.isdigit(),
    }
    if i > 0:
        word1 = sent[i-1][0]
        features.update({
            '-1:word.lower()': word1.lower(),
            '-1:word.istitle()': word1.istitle(),
            '-1:word.isupper()': word1.isupper(),
        })
    else:
        features['BOS'] = True  # Beginning of Sentence

    if i < len(sent)-1:
        word1 = sent[i+1][0]
        features.update({
            '+1:word.lower()': word1.lower(),
            '+1:word.istitle()': word1.istitle(),
            '+1:word.isupper()': word1.isupper(),
        })
    else:
        features['EOS'] = True  # End of Sentence
```

Figure 5.1: Features for the CRF model.

5.2.3 spaCy pipeline

The main challenge of using a spaCy model for our task is the difference of tokenisation. The `SpacyToConllConverter` class is utilised to adjust the tokenisation. Figure 5.2 presents a tokeniser instance which uses regular expressions to define token boundaries and replaces the default token handling. The tokenisation in spaCy diverges from the CoNLL format, particularly in how hyphenation is handled. As indicated by table 5.3, while hyphenated words are treated as single tokens in the data-set, spaCy's tokeniser splits them into multiple tokens. Special processing of hyphens interferes with that of contractions, requiring the custom tokeniser to simultaneously address both aspects. The function allows the application of predefined custom tokenisation rules within the processing pipeline.

Sentence id	Word	CoNLL-2003 token	spaCy tokens
4	first-time	first-time	first, -, time
101	Ben-Elissar	Ben-Elissar	Ben, -, Elissar
304	1996-08-22	1996-08-22	1996, -, 08, -, 22

Table 5.3: CoNLL-2003 and spaCy hyphenated words splitting comparison

Figure 5.2 demonstrates the setup for the custom tokeniser. It identifies English contractions using regular expressions for prefixes, suffixes, and infixes to refine how tokens are recognised and separated. This tokeniser is then integrated into a spaCy language model, modifying its

default behaviour.

```python
def custom_token_match(text):
    # Regex to match contractions like "n't", "'re", "'s", etc.
    contraction_regex = re.compile(r"\b\w+'t\b|\b\w+'re\b|\b\w+'s\b|\b\w+'d\b|\b\w+'ll\b|\b\w+'ve\b")
    if contraction_regex.search(text):  # Use search instead of match to find the pattern anywhere in the text
        return True
    return False

# Adapted from
# https://stackoverflow.com/questions/51012476/spacy-custom-tokenizer-to-include-only-hyphen-words-as-tokens-using-infix-regex

def custom_tokenizer(nlp):
    """
    Initialises a new Tokenizer with custom_token_match as the token matching function.

    Parameters:
    - nlp (spacy.lang): The spaCy language model.

    Returns:
    spacy.tokenizer.Tokenizer: A custom tokenizer instance for the specified spaCy language model.
    """
    return Tokenizer(nlp.vocab,
                     token_match=custom_token_match)
```

Figure 5.2: spaCy entity types mapping.

The wide range of entity types that spaCy is able to detect also poses a problem for classification. The converter in figure 5.3 maps spaCy tokens to the CoNLL-2003. For example, spaCy uses multiple entity types to represent locations; a specific location such as "London" would be classified as GPE and subsequently mapped to Location. The corresponding numerical indices are then returned. This step is necessary to align spaCy's predicted labels with the actual labels from the dataset for further processing and evaluation.

```python
# Determine if it's the start of a new entity
def is_start_of_entity(token: Token) -> bool:
    return token.ent_iob == 3

# Map spaCy's entity types to CoNLL format
def spacy_ent_to_conll(token: Token) -> str:
    # The considered entity types
    retained_entities = {'ORG', 'LOC', 'PERSON'}

    # Check if the token is part of an entity
    if token.ent_type_:
        # Check if the entity type is one of the retained types
        if token.ent_type_ in retained_entities:
            prefix = "B-" if is_start_of_entity(token) else "I-"
            return f"{prefix}{token.ent_type_}"
        else:
            # Label everything else as 'MISC'
            prefix = "B-" if is_start_of_entity(token) else "I-"
            return f"{prefix}MISC"
    else:
        return "O"
```

Figure 5.3: spaCy custom tokeniser implementation.

5.2.4 Hugging Face Transformers

For the Transformers approach, tokenisation and label alignment is handled by the function `tokenise_and_align_labels`. BERTs have a propensity to tokenise input text into sub-words, which may not neatly correspond to the word-level NER tags. Within this function, each tokenised input undergoes mapping, whereby each sub-word token is traced back to its original word, if applicable, and is then assigned the appropriate NER tag. Should a sub-word not align with a starting word in the original text or represent a special token like [CLS] or [SEP], it receives a label of -100. [CLS], denotes classification and is placed at the beginning of the input sequence. It represents the classification of the entire input sequence for tasks like text classification or sentence-level tasks. Essentially, it functions as a distinctive token encapsulating information pertinent to the entire sequence. On the other hand, [SEP], stands for separator and is interposed between pairs of sentences or sequences of tokens. It helps BERT understand when one sequence ends and another begins. These signify that the token should be disregarded in loss computations throughout the model training phase.

5.3 Evaluation

To asses the performance of the approaches previously described, it is necessary to employ a rigorous evaluation methodology. Therefore, recognised libraries specifically tailored for machine learning tasks are utilised. Scikit-learn offers a classification report which provides a detailed token level evaluation through precision, recall, accuracy, and f1-score. Furthermore, for sequence labeling approaches, namely CRFs, and transformers, the seqeval library is used to compute metrics for the full named entity types.

Model	F1	Precision	Recall	Accuracy
elastic/distilbert-base-uncased-finetuned-conll03-english	0.989	0.99	0.99	0.99
elastic/distilbert-base-cased-finetuned-conll03-english	0.987	0.99	0.99	0.98

Table 5.4: Highest Token Classification performance on CoNLL-2003 dataset.

For reference, table 5.4 presents the models achieving the highest performances to date on the CoNLL-2003 English dataset[1]. DistilBERT is described as a transformers model, smaller and faster than BERT, Both models are pre-trained on the same corpus but DistilBERT is self-supervised and learnt from the BERT base model. The difference between the two variants of DistilBERT mentioned is that one is case sensitive and the other is not.

[1] https://paperswithcode.com/sota/token-classification-on-conll2003

Chapter 6

Results and discussion

The main objective of the project was to conduct a comprehensive comparison and analysis of various approaches to the Named Entity Recognition task. In this chapter, the performance of the different systems and the results obtained are presented. Conclusions are drawn followed by a consideration of further work.

6.1 Results comparison

Four models were employed to perform named entity recognition on the CoNLL-2003 dataset: Multinomial Logistic Regression, Conditional Random Fields, BERT, and a spaCy pipeline. The overall performance of each of these approaches is evaluated at the token level using standard metrics, including accuracy and weighted average, precision, recall, and F1-score as illustrated by table 6.1. CRF and BERT models achieve the best performance across all metrics with F1 scores of 0.96 and 0.98 respectively.

Model	Accuracy	Precision	Recall	F1-score
spaCy	0.82	0.91	0.82	0.85
MLR	0.89	0.87	0.89	0.86
CRF	0.96	0.96	0.96	0.96
BERT	0.98	0.98	0.98	0.98

Table 6.1: Accuracy and average precision, recall, and f1-score comparison

Each of these models was selected to provide a broad overview of different approaches ranging from traditional machine learning techniques to advanced deep learning methods, allowing for a detailed comparison of their respective capabilities and performances. The spaCy pipeline's relatively poor performance in this context could be attributed to its customisation limitations and the need to adapt its broad entity recognition capabilities, which are trained on a variety of entity types, to a more constrained set of only four categories. This forced simplification likely hampers its ability to leverage its full analytical capabilities, leading to decreased accuracy. In comparison, Multinomial Logistic Regression, despite using

a simpler bag-of-words approach, shows a surprisingly higher performance with a F1 of 0.86. However, its fundamental limitation lies in treating each token in isolation, disregarding any potential semantic or syntactic relationships between them. This lack of context can affect the model's ability to accurately predict entities based on surrounding tokens, a feature that more sophisticated models can handle effectively. In contrast, Conditional Random Fields approach the NER task by considering it as a sequence labelling problem. This perspective allows the CRF model to utilise a broader range of features, including those that capture dependencies between tokens in a sentence. Such features significantly enhance effectiveness by enabling the model to understand and use the context in which tokens appear, thereby improving performance compared to logistic regression.

Lastly, BERT's outstanding performance, which is very close to the state-of-the-art, can be largely attributed to its pre-training on extensive text corpora and subsequent fine-tuning for the specific task of NER. This extensive pre-training enables BERT to develop a deep understanding of language nuances and context, which is further refined during the fine-tuning process. By adjusting its parameters specifically for the NER task, BERT can effectively apply its sophisticated understanding of language to achieve high accuracy. This capability not only makes it highly effective but also versatile in handling various NER challenges, setting a high standard for both precision and recall in complex datasets.

6.2 Visualisation and error analysis

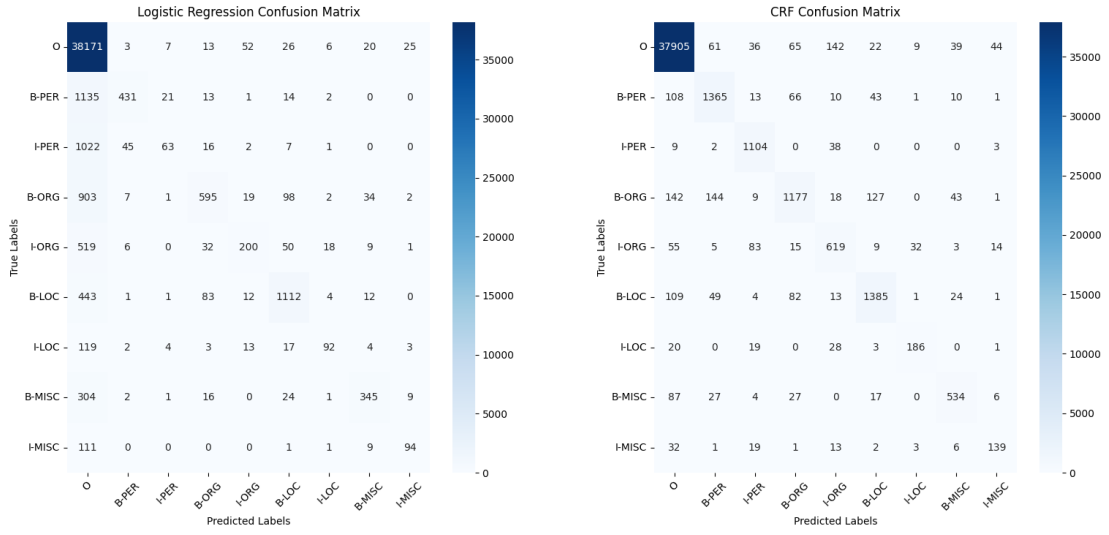

Figure 6.1: Logistic regression and CRF confusion matrices

Figures 6.1 and 6.2 show the confusion matrices for the four different approaches. For all models, most tokens outside named entities are accurately identified, as they are the most prevalent in the training set. Additionally, the Multinomial Logistic Regression model

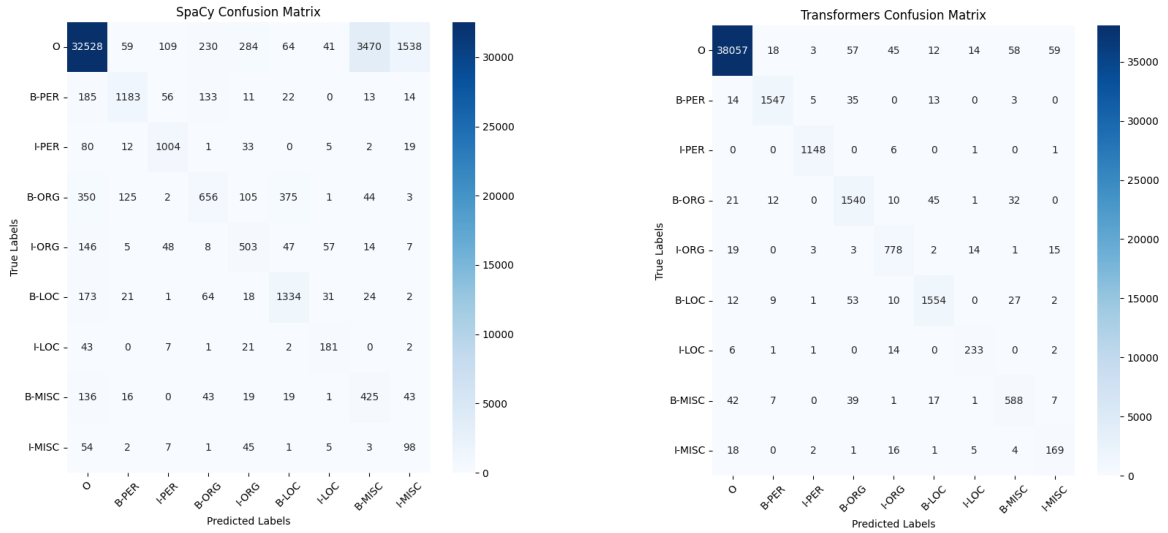

Figure 6.2: spaCy and BERT confusion matrices

exhibits more classification errors than correct classifications, demonstrating a significant bias as it frequently assigns the majority class to most tokens. Conversely, the Conditional Random Field model achieves better accuracy with most tokens well-classified. However, there are some specific confusion issues, such as the beginning of organisations tokens being confused with locations and persons tags. This suggests some overlap in the feature representations of these entity types. Furthermore, spaCy faces a considerable number of classification errors even for O tokens , with 3470 tokens misclassified as beginning of a miscellaneous entity and 1538 as inside a miscellaneous entity. The confusion between these three tags might stem from the fact that MISC is not a standard spaCy entity type. Consequently, entities that do not fit the defined named entity types are manually mapped to this type. A number of beginning of organisations tokens are also mistakenly classified as locations,and there is a tendency to mis-classify tokens from across all classes as being outside named entities. This could occur when organisation names contain geographic identifiers or are inherently linked to locations. Ultimately, the results from the transformer-based model are nearly perfect, but there are occasional confusions between locations and organisations, and vice versa. This indicates some challenges in distinguishing between these two types of entities, possibly due to similar contextual clues or overlapping training examples.

6.3 Full named entity performance

Investigating the full entity performance in addition to token-level analysis is useful for NER as it provides more accurate measures of a system's capability to identify and outline complete entities. It accounts for fragmentation errors; for example mistakenly identifying "Wall Street" as separate entities instead of the single "Wall" "Street". Full entity evaluation

simplifies the assessment process by concentrating on the end result, thus making the evaluation results easier to interpret and communicate. Additionally, this approach helps understanding weaknesses in entity boundary detection or in handling certain types of entities, facilitating targeted improvements. In this section, the highest performing systems are compared.

Entity Type	Metric	BERT	CRF
LOC	Precision	0.93	0.86
	Recall	0.93	0.82
	F1-Score	0.93	0.84
MISC	Precision	0.81	0.79
	Recall	0.83	0.74
	F1-Score	0.82	0.76
ORG	Precision	0.88	0.77
	Recall	0.91	0.67
	F1-Score	0.90	0.72
PER	Precision	0.96	0.82
	Recall	0.95	0.84
	F1-Score	0.96	0.83

Table 6.2: Performance of BERT and CRF models across different entity types.

Compared to the metrics computed for token classification in table 6.1, table 6.2 shows that precision, recall, and F1-scores tend to drop when considering whole entities. BERT still achieves the best results for all entity types but is better at recognising persons. As it is pre-trained on large corpora that frequently include text about people, it could learn person-related contextual cues. Moreover, person names tend to exhibit less ambiguity and follow more predictable linguistic patterns than other types. The lowest precision, recall, and F1 scores are observed for miscellaneous entities, which are the least common type in the training set. Conversely, the CRF model performs better on location entities. This could be explained by location being the most common type of entity in the training set. The model can learn a robust set of rules and contextual dependencies, leading to higher performance.

Lower performance for full entity evaluation compared to token-level evaluation can be attributed to the increased complexity and strictness of recognising entire spans of text. This is more challenging than merely labeling tokens, as it involves understanding contextual relations and interactions between tokens. Fragmentation and combination errors, where entities are incorrectly broken up or merged, are more penalised. Moreover, data sparsity and imbalance in training data can exacerbate these challenges, particularly for less common or more complex entities.

6.4 Discussion

The objective of this project was to design, implement, and evaluate various Named Entity Recognition systems using the CoNLL-2003 shared task dataset. Four distinct approaches

were implemented: Multinomial Logistic Regression, CRF, BERT, and one utilising a spaCy pipeline. These systems were rigorously tested to determine which methodologies performed best at both token and entity levels, measured through accuracy, precision, recall, and F1-score. First, among the models compared, the BERT system consistently showed superior performance across all metrics, hereby verifying the hypothesis that deep learning techniques are the most suited for the NER task. It is followed by the CRF model. It can then be concluded that factors contributing to high performance include sequence modelling rather than individual token classification, the utilisation of semantic and syntactic features, whether handcrafted or directly learned, and label transition learning, which are integral to models like CRF and BERT. Moreover, data pre-processing also plays a crucial role: different systems require data in varying formats. For instance, the spaCy pipeline demonstrated that tokenisation methods are not universal and must be tailored to the specific characteristics of each system. Techniques such as token normalisation enhanced the performance, underscoring the critical role of data quality in Natural Language processing. Finally, the evaluation further highlighted that pre-training with a diverse range of text samples, not just the provided training set, is beneficial. This diversity in training helps in building a more robust model capable of handling a wide array of text variations. Fine-tuning for specific tasks also leads to better results.

6.5 Further work

Building on the insights from this project, areas for further research have been identified, aiming to refine and expand the current research. First, exploring cross-domain adaptability could be useful to determine how well NER systems can perform across diverse text genres. Research could also be extended to multilingual and cross-linguistic named entity recognition, broadening the focus beyond predominantly English language systems. This expansion would address the need for applications of natural language processing in a variety of languages, reflecting the global use of technology and the diversity of data sources. Such research would not only support the development of more inclusive technologies but also improve model performance across different linguistic structures, which can vary significantly from English. Furthermore, implementing under-sampling strategies could help prevent overfitting, a common problem where models perform well on training data but poorly on unseen data. By ensuring a more balanced dataset, under-sampling could enhance the generalisation capabilities of models, making them more reliable when deployed in diverse settings. Finally, adjusting the parameters of models like BERT could further optimise performance for specific tasks or datasets. This fine-tuning involves altering learning rates, the number of layers, or hidden unit sizes to better suit the specific characteristics of a task or data, potentially leading to significant improvements in accuracy and efficiency. These adjustments require a deep understanding of both the model architecture and the dataset characteristics.

Bibliography

Leon Derczynski. 2016. https://aclanthology.org/L16-1040 Complementarity, F-score, and NLP evaluation. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 261–266, Portorož, Slovenia. European Language Resources Association (ELRA).

Explosion AI. 2023. https://spacy.io/ spacy 3.0: Industrial-strength natural language processing in python. Accessed: 2023-12-01.

Daniel Jurafsky and James H. Martin. 2023. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 3rd edition.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. https://doi.org/10.1038/nature14539 Deep learning. *Nature*, 521(7553):436–444.

Xing Liu, Huiqin Chen, and Wangui Xia. 2022. http://ojs.bbwpublisher.com/index.php/JCER Overview of named entity recognition. *Journal of Contemporary Educational Research*, 6(5).

Jasmina D. Novaković, Alempije Veljović, S. Ilić, Siniša, Željko Papić, and Milica Tomović. 2017. https://www.proquest.com/scholarly-journals/evaluation-classification-models-machine-learning/docvi Evaluation of classification models in machine learning. *Theory and Applications of Mathematics Computer Science*, 7(1):39–46.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. https://doi.org/10.3115/v1/D14-1162 GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. https://aclanthology.org/W13-3516 Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. http://arxiv.org/abs/cs/9907006 Representing text chunks.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. https://doi.org/10.3115/1119176.1119195 Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, page 142–147, USA. Association for Computational Linguistics.

Martina Toshevska, Frosina Stojanovska, and Jovan Kalajdjieski. 2020. https://doi.org/10.5121/csit.2020.100402 Comparative analysis of word embeddings for capturing word similarities. In *6th International Conference on Natural Language Processing (NATP 2020)*, NATP 2020. Aircc Publishing Corporation.

L. Tunstall, L. von Werra, and T. Wolf. 2022. https://books.google.co.uk/books?id=pNBpzwEACAAJ *Natural Language Processing with Transformers: Building Language Applications with Hugging Face*. O'Reilly Media.

Rocio Vargas, Amir Mosavi, and Ramon Ruiz. 2017. https://eprints.qut.edu.au/127354/ Deep learning: A review. Working paper.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. http://arxiv.org/abs/1706.03762 Attention is all you need. *CoRR*, abs/1706.03762.

Tom Verdonck, Bart Baesens, María Óskarsdóttir, and Seppe vanden Broucke. 2021. https://doi.org/10.1007/s10994-021-06042-2 Special issue on feature engineering editorial. *Machine Learning*.

Appendices

Appendix A

Detailed classification reports

	precision	recall	f1-score	support
0	0.97	0.85	0.90	38323
B-PER	0.83	0.73	0.78	1617
I-PER	0.81	0.87	0.84	1156
B-ORG	0.58	0.39	0.47	1661
I-ORG	0.48	0.60	0.54	835
B-LOC	0.72	0.80	0.76	1668
I-LOC	0.56	0.70	0.63	257
B-MISC	0.11	0.61	0.18	702
I-MISC	0.06	0.45	0.10	216
accuracy			0.82	46435
macro avg	0.57	0.67	0.58	46435
Neighted avg	0.91	0.82	0.85	46435

Table A.1: spaCy classification report.

	precision	recall	f1-score	support
B-LOC	0.94	0.94	0.94	1668
B-MISC	0.83	0.85	0.84	702
B-ORG	0.90	0.93	0.91	1661
B-PER	0.96	0.95	0.96	1617
I-LOC	0.87	0.91	0.89	257
I-MISC	0.65	0.75	0.69	216
I-ORG	0.89	0.92	0.91	835
I-PER	0.99	0.99	0.99	1156
0	1.00	0.99	0.99	38323
accuracy			0.98	46435
macro avg	0.89	0.91	0.90	46435
weighted avg	0.98	0.98	0.98	46435

Table A.2: bert-base-NER classification report.

	precision	recall	f1-score	support
O	0.89	1.00	0.94	38323
B-PER	0.87	0.27	0.41	1617
I-PER	0.64	0.05	0.10	1156
B-ORG	0.77	0.36	0.49	1661
I-ORG	0.67	0.24	0.35	835
B-LOC	0.82	0.67	0.74	1668
I-LOC	0.72	0.36	0.48	257
B-MISC	0.80	0.49	0.61	702
I-MISC	0.70	0.44	0.54	216
accuracy			0.89	46435
macro avg	0.77	0.43	0.52	46435
weighted avg	0.87	0.89	0.86	46435

Table A.3: Logistic regression classification report.

	precision	recall	f1-score	support
O				
B-PER	0.99	0.99	0.99	38323
I-PER	0.83	0.84	0.83	1617
B-ORG	0.86	0.96	0.90	1156
I-ORG	0.82	0.71	0.76	1661
B-LOC	0.70	0.74	0.72	835
I-LOC	0.86	0.83	0.85	1668
B-MISC	0.80	0.72	0.76	257
I-MISC	0.81	0.76	0.78	702
	0.66	0.64	0.65	216
accuracy			0.96	46435
macro avg	0.81	0.80	0.81	46435
weighted avg	0.96	0.96	0.96	46435

Table A.4: CRF classification report.