UNIVERSITY OF CAPE TOWN

M.Sc. Data Science

Department of Statistical Sciences

Named Entity Recognition with Recurrent Neural Networks

Author: Dean Hope Robertson

Supervisor: Stefan Britz

University Of Cape Town

A thesis submitted in partial fulfilment of the requirements for the degree of Masters in Data Science

October, 2021

London, United Kingdom

Declaration

I, Dean Hope Robertson, declare that this project titled, "Named Entity Recognition with Recurrent Neural Networks" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a master's degree at this University.
- Where any part of this dissertation has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project is entirely my own work.
- I have acknowledged all main sources of help.
- Where the project is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: [illegible]

Date: 22/10/2021

Abstract

Neural networks have become widely used in a variety of machine learning applications, ranging from image recognition to forecasting. This complex computing framework has more recently been applied to natural language processing (NLP) tasks, more importantly, named entity recognition (NER). Entity recognition traditionally requires extensive feature engineering and a large prior knowledge base in the form of lexicons. In recent years, NER systems have been utilizing a deep learning neural network architecture, known as a bidirectional long short-term memory (Bi-LSTM) with a conditional random field (CRF) layer, in an effort to build the next state-of-the-art language model. Throughout the years, models have been using more external data in order to attain the leading edge, by utilizing entity gazetteers, trigger words, word embeddings, and hand-crafted features. This research study aims to compare a traditional CRF model that relies heavily on engineering features to the deep learning bidirectional-LSTM-CRF model, which eliminates the dependence on engineered features without compromising performance.

In this study, we show that bidirectional LSTM–CRF architecture, given only publicly available pre-trained word embeddings, can outperform a baseline CRF model (77.73%) in the task of named entity recognition with an F1 score of 82.81% on the CoNLL-2003 dataset. The performance surpasses the baseline CRF model by effectively using long-range dependencies between past and future tokens instead of hand-crafted features.

Acknowledgements

Special thanks of gratitude for his guidance and expertise of my supervisor, Stefan Britz, and for financial assistance to Bloomberg LP. To my family and partner for their constant love and support, and putting up with me through this time.

Contents

List of Tables

List of Figures

Chapter I

Introduction

1.1 Problem Description

Natural language processing (NLP) continues to be at the forefront of computer science research with the exponential growth in text data generation. This growth has been driven by the ever expanding integration of internet applications and social media into everyday life. Named entity recognition, commonly known as NER, has become a major area of research within NLP, falling into a sub-task of information extraction. The task focuses on recognising and extracting key entities such as people entity recognition (PER), location entity recognition (LOC) and organisations entity recognition (ORG) from a document of text. The purpose of NER systems is to detect, extract and link named entities from everyday text. This has a variety of applications including recommender systems, powering online search algorithms, and machine translation systems (Goyal, Kumar, and Gupta, 2017).

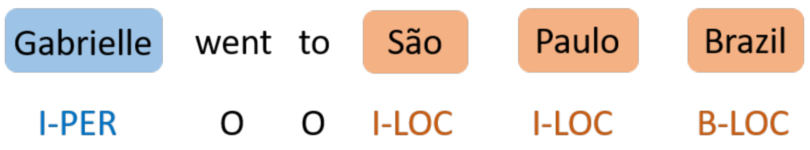


Figure 1.1: People (PER) and location (LOC) entities detected in a sentence.

As shown from the example above in Figure 1.1, a NER system is able to detect entities such as persons (PER) and locations (LOC) within the sentence. The entity labels are comprised of two parts, namely a boundary (“I-”) and the entity label (“PER”). This Inside-Outside-Beginning (IOB) tagging scheme was originally developed by Ramshaw and Marcus (1995), and has been used as the standard data annotation for NER datasets. Within the NLP, text is broken down into smaller units called tokens which can be defined as words, characters (e.g !,&,?..) or sub-words. The boundary with an I-prefix ("I-") indicates that the tokens belong to the same entity. When two separate named entities with the same entity label are located next to each other, the start of the second entity is denoted by the B-prefix ("B-") to indicate the beginning of a separate entity. This boundary partition can be seen in the Figure 1.1 where the city and country entities are located next to one another.

An NER model’s performance is commonly evaluated on precision, recall and F1 score (J. Li et al., 2018). Since the introduction of the CoNLL-2003 shared task challenge (Tjong Kim Sang and De Meulder, 2003) a stringent "exact-match" criteria has been used to evaluate NER systems. The exact-match criteria only awards credit when the entity label ("PER") and entity boundary ("B-") are both correctly predicted. However, identifying named entities is not as trivial as it might seem, considering that a token in one entity’s name (e.g “Burger” in “Burger King”) might be a non-entity word used in everyday text.

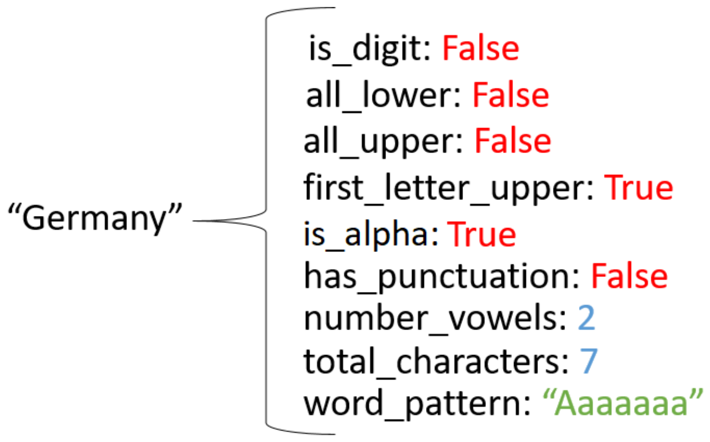


Figure 1.2: An example of feature engineering on the word "Germany".

A traditional approach to NER is to conduct extensive feature engineering on the text data, in an effort to extract further information from the words, before applying a supervised learning algorithm such as conditional random fields (CRF). An example of the type of feature engineering can be seen above in Figure 1.2. The biggest drawback of this approach is the tedious effort required to hand-craft the rules as well as the knowledge expertise needed to not only construct, but also maintain the rules. Another disadvantage of this approach is the lack of flexibility. Rule-based models are unable to adapt to a new domain task and are thus non-transferable. Models built for detecting entities in financial documents may struggle to detect entities in sports articles, resulting in the need to build a custom model for each specific domain (Chiticariu, Y. Li, and Reiss, 2013). Furthermore, this approach is known to be reliant on the presence of capital letters which are not always found in free-form-text (e.g a typical google search). This is where deep learning can help provide a more flexible solution.

Deep learning with the assistance of open sourced pre-trained word embeddings offers an alternative solution to feature engineering. Bidirectional long short-term memory cells with a conditional random field output layer (also known as BiLSTM-CRF) aim to map long range dependencies between words and expose hidden features within text. It is this ability to recognise patterns between words within a text, combined with deep learning's ability to expose non-linear characteristics, that reduces the need for engineered features.

1.2 Objectives of Investigation

The race to develop the next state-of-the-art language model to achieve the optimal performance on the CoNLL-2003[1] dataset (Tjong Kim Sang and De Meulder, 2003) has seen a variety of new techniques and neural network architectures (including BiLSTM-CRF) developed to solve this task (Lample et al., 2016; Huang, Xu, and Yu, 2015). In doing so, attempts have increasingly added additional data in the form of character embeddings, part-of-speech tags, name gazetteers and engineered features in an effort to attain the best results (Panchendrarajan and Amaresan, 2018).

This minor dissertation explores the potential use of BiLSTM-CRF model architecture with pre-trained word embeddings for named entity recognition, without the use of any additional data.

[1]A database of Reuters news stories in which named entities have been annotated. We expand on this dataset in Chapter 3.

In this research study the objective is to determine whether a BiLSTM-CRF model with only lowercase word features can outperform a conditional random field model with over-engineered features. The author of this minor dissertation is not aware of any published investigation into the comparison of these two models given the applied limitations to the BiLSTM-CRF model.

1.3 Layout of Dissertation

Chapter 2 provides an overview of all relevant research conducted on named entity recognition. The chapter takes a look into common metrics used for evaluating NER models' performance as well as an in-depth overview of the various techniques and approaches that have been developed to solve this problem.

Chapter 3 provides an overview of the CoNLL-2003 dataset used in this dissertation, including the format of the data, summary statistics, as well as definitions and distributions of entities.

Chapter 4 outlines the methodology used for developing both the BiLSTM-CRF and conditional random fields models. The data preprocessing and model architecture used in each model are outlined, as well as the framework for hyperparameter tuning.

Chapter 5 provides an overview of the results attained from the hyperparameter tuning and final model evaluations on unseen test data.

Chapter 6 covers the conclusions and recommendations derived from the results in the previous chapter, as well the limitations and potential improvements.

In this chapter the mechanics of the problem for recognising named entities such as people, locations and organisations from text are outlined. Traditional approaches to NER such as CRF models suffer from a number of drawbacks, namely the reliance on engineered features which are both tedious to create and not easily transferable. Deep learning can offer a more flexible solution by means of exploiting the potential of BiLSTM-CRF architecture to expose the hidden long-range dependencies between tokens in a sentence. The next chapter highlights all the related works in the field of named entity recognition that is relevant to achieving this study's objectives.

Chapter II

Literature Review

This chapter contains an overview of all related works in the field of named entity recognition. This chapter will primarily focus on the history of the named entity recognition models, their relevant applications, as well as various metrics used to measure their performance.

2.1 Approaches to Named Entity Recognition

Named entity recognition (NER) has been around since the early 1990's and was first introduced at the Message Understanding Conference (MUC-6) held in the United States. At the time, the MUC focused on information extraction and realised that recognition of entities such as people, locations, organisations and units would become a vital component for information extraction in the future. Since then, NER has developed into one of the most important subtasks of information extraction with a large variety of techniques developed over time to recognise and extract entities. Techniques include rule-based models and supervised learning (Nadeau and Sekine, 2007), later moving toward a deep learning approach with the help of recurrent neural network architectures and word embeddings (Lample et al., 2016). In this section, we review the development of NER systems over time as well as the evaluation metrics.

2.1.1 Rule-Based

The earliest attempts to develop NER systems were predominately rule-based techniques that did not require any annotated training data that you would expect to see with supervised learning models. Instead, rule-based approaches relied heavily on extensive dictionaries called lexicons and handcrafted rules for identifying entities within text (Etzioni et al., 2005). A variety of lexicons could be used to look up pre-defined entity names to help the NER system recognise potential named entities such as:

- Person's first and second name (e.g. George Washington)
- Name of organisation (e.g. Microsoft)
- Names of locations (e.g. Amazon River, Eiffel Tower, Oxford Street, Mount Everest)
- Currencies (e.g. Euro)

In addition to lexicons, trigger words such as "Dr." and "Mrs." were used to identify a potential person's name as well as other entities (Abuleil, 2004). Stop words and modifier lists are less commonly used but can assist by removing noise words that are insignificant like "and", "of" or "if" (Antonio Moreno, 2005). Rules were then handcrafted and used in conjunction with the various lists and lexicons to locate entities within a document. The rules are constructed to heuristically explore the language syntax using extensive knowledge of the specific field or utilising domain expertise. An example of how rules interacted with these lists would be to identify two consecutive words (which started with an uppercase) as a person's name if it followed a trigger word such as "Mr." (Zaghouani, 2012).

Rule-based approaches generally result in high precision since the lexicons recognise the exact predetermined entity name, but due to the specificity of the rules and incomplete dictionaries often result in low recall (see Section 2.3) (J. Li et al., 2018). While errors are generally easy to locate and fix, the biggest pitfall with this approach is the amount of time and and effort required to hand-craft the rules as well as the knowledge expertise needed to construct them. The second downside to a rule-based approach is the lack of flexibility that comes with handcrafting rules for a specific domain, since rule-based NER models are not easily transferable from one knowledge domain to another (Chiticariu, Y. Li, and Reiss, 2013).

2.1.2 Supervised Learning

It was not long before supervised learning approaches such as Support-Vector Machines (Hearst et al., 1998), Conditional Random Fields (Lafferty, McCallum, and F. C. N. Pereira, 2001), Hidden Markov Models (Eddy, 1996) and decision trees (Quinlan, 1986) were applied to NER, setting high performances on NER datasets. Unlike the rule-based approach mentioned in the previous section, these supervised learning models predict entities in a sentence based on labelled training data and engineered features.

A Markov language model is a dominant statistical sequence tagging model based on the Markovian system, which classifies a sequence of tokens in which the probability of the token being a named entity depends on the state of the preceding token (Jurafsky and Martin, 2000). However, in real application of named entity recognition, the states are not observed but rather inferred, resulting in a hidden Markov model architecture (Jurafsky and Martin, 2009).

Humans use hidden observations in every day decision-making. For example, when trying to determine if it is windy outside, people do not carry anemometers on their person but instead look to see to what extent the trees are blowing outside. Here the inferred observations are being used to help people make a decision about the weather conditions outside. Hidden Markov Models (HMMs) permit the input data to not only contain the token from the sentence but also additional (hidden) orthographic features. These orthographic features are engineered to break down the tokens, and expose deeper characteristics of the words such as the presence of capital letters, digits or other special characters.

In 2002, a HMM architecture along with 11-orthographic word features, a list of semantic trigger words, and entity gazetteers (i.e. lists of entity names) scored F1 scores of 96.6% and 94.1% on the MUC-6 and MUC-7 English NER test datasets respectively (Zhou and Su, 2002).

Here, trigger words (like "River", "Dollars", "Ltd" etc.) are used together with orthographic (hidden) features to assist the identification of named entities. In contrast to Bayesian rule-based HMMs, Max Entropy models (also known as MaxEnt) are discriminative and based on the principles of maximum entropy. MaxEnt models are commonly used to tackle various sequence tagging tasks such as named entity recognition and do so by building probability distributions for each class (e.g. each named entity). The predicted entity ($\hat{C}$) for each word in the document is the class with the highest probability (Jurafsky and Martin, 2009).

$$\hat{C} = \underset{c \in C}{\operatorname{argmax}} \Pr(C \mid \text{word}) \tag{2.1}$$

Bender, Och, and Ney (2003) explored the implementation of MaxEnt models for NER with a limited window of two words $\left(W_{n-2}^{n+2}\right)$ around the current word (W_n) and two preceding entity labels $\left(C_{n-2}^{n-1}\right)$. The limited window accounted for a wider contextual approach, looking for patterns in word relationships between a broader set of words. The model can be presented as follows:

$$\Pr\left(C_1^N \mid W_1^N\right) = \prod_{n=1}^{N} \Pr\left(C_n \mid C_1^{n-1}, W_1^N\right) \tag{2.2}$$

$$\underset{\text{model}}{=} \prod_{n=1}^{N} \Pr\left(C_n \mid C_{n-2}^{n-1}, W_{n-2}^{n+2}\right) \tag{2.3}$$

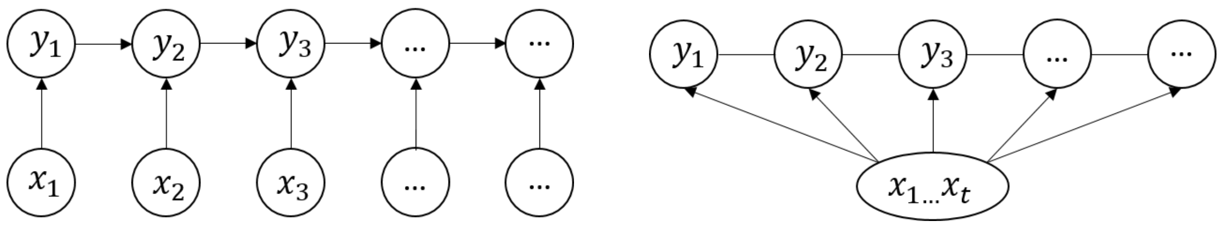
where:

$W_{n-2}^{n+2} = W_{n-2}, W_{n-1} ... W_{n+2}$

$C_{n-2}^{n-1} = C_{n-2}, C_{n-1}$

The model then used the maximum entropy framework developed by Berger, V. J. D. Pietra, and S. A. D. Pietra (1996) where a set number of feature functions (lexical, word, transition, prior, compound and dictionary features) are used as input data. The word features only contain three orthographic features (capitalisation, digits, prefixes and suffixes). The goal of the model is to build a distribution by continuously adding features, which highlights different attributes from subsets of the training data. The model is trained on the observations, feature-functions and corresponding states to maximise the log-likelihood on the CoNLL-2003 corpus, achieving an F1 score of 83.92% and 68.88% on the English and German test sets respectively.

Unfortunately, Maximum Entropy Markov Models (McCallum, Freitag, and F. C. N. Pereira, 2000) suffer from a fundamental limitation whereby the models favour states with fewer outgoing transitions, which for NLP tasks results in a model having a label bias towards the most frequently used words in the training set. The solution to this label bias is Conditional Random Fields (CRFs) (Lafferty, McCallum, and F. C. N. Pereira, 2001), which overcomes the problem by implementing a global normalisation that looks at state sequences globally rather than locally. The difference between these two models is shown in Figure 2.1.



(a) Hidden Markov Model. (b) Conditional Random Field.

Figure 2.1: A comparison of HMM and CRF architectures.

It was not until McCallum and W. Li (2003) that CRFs were applied to named entity recognition where the approach resulted in an improved overall performance on the CoNLL-2003 dataset compared to HMM and Maximum Entropy Markov Models (MEMM), with F1 scores of 84.04% and 68.11% on the English and German test sets respectively. McCallum and W. Li (2003) adopted quasi-Newton methods, such as L-BFGS optimisation algorithm which differed from the Lafferty, McCallum, and F. C. N. Pereira (2001) initial iterative scaling methods. Quasi-Newton methods (also known as variable-metric methods) proved to be significantly superior to alternative optimisation methods such as conjugate gradient (Malouf, 2002). The approach implemented by McCallum and W. Li (2003) expands on the use of lexicons with a technique known as “WebListing”, which automatically creates lexicons by scraping html code from the internet.

Support Vector Machines (SVMs) were implemented on the dataset during the CoNLL-2003 shared task challenge (Tjong Kim Sang and De Meulder, 2003) by Mayfield, McNamee, and Piatko (2003) in an effort to extend the quantity of features used within the model. SVMs only use a subset of vectors to determine the hyper-plane that separates the classes, referred to as support vectors. This allows the SVM model to handle a much higher dimensionality within the training data. The approach uses a larger window size (3 words) and additional features such as the Parts-of-Speech (POS) tags, chunk tags, as well as lemma features for the German language dataset. The model achieved F1 scores of 84.67% and 69.96% for the English and German test sets respectively.

The winners of the CoNLL-2002 shared task competition (Tjong Kim Sang, 2002), decided to apply a similar approach for the CoNLL-2003 edition using Adaboost (Freund and Schapire, 1997), an adaptive boosting algorithm that combines weak learners (shallow decision trees) into a weighted stronger learner. In their approach, Carreras, Marquez, and Padro (2003) used a wide range of input features including lexicons, POS tags, chunk tags, orthographic features, affixes (prefix and suffices of words), trigger words, bag-of-words, gazetteer features etc. All of the above features were implemented with a window size of between {-3, +3} and {-5, +5} as can be seen in Figure 2.2. All the studies on the CoNLL-2002 and CoNLL-2003 datasets used a predefined train and test set therefore rendering the performances of each approach comparable.

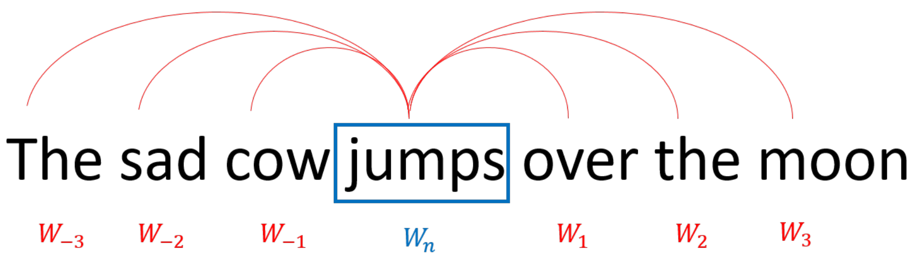


Figure 2.2: An example of a target word (W_n) with a window range of {-3, +3}.

Since the model uses an extensive amount of external knowledge, it was decided that two F1 scores should be presented in order to compare performance against other alternative models. The Adaboost model achieved a score of 85.00% and 84.12% on the English test dataset with and without external knowledge respectively.

Other supervised learning approaches that achieved better performance in the CoNLL-2003 shared task challenge include Florian et al. (2003) who used a combination of four classifiers, as well as extensive additional features including POS tags, chunk tags, affixes, a large gazetteer, and the output of two other NER models trained on an external 1.7 million labelled words. This approach accomplished the highest score in the shared task challenge with an F1 score of 88.76% on test set. The runner up for the shared task challenge used a MaxEnt model (Chieu and Ng, 2003), which used an extensive array of both local and globally hand-crafted features: rare words, bi-grams, case sequences, class suffixes etc. In order to compare against other approaches, external knowledge was excluded to return an F1 score of 86.84% on test set.

2.2 Deep Learning

The application of deep learning (Ivakhnenko and Lapa, 1965) for NLP tasks such as named entity recognition was first introduced by Collobert and Weston (2008). Today the deep learning approaches dominate the space with well-known language models such as BERT (Devlin et al., 2019) and Flair (Akbik, Blythe, and Vollgraf, 2018) attaining state-of-the-art results. In this approach, all words are reduced to lowercase, sentences converted to vectors in a d-dimensional space, and the orthographic features kept as separate features. There are two clear advantages in applying deep learning to a problem like named entity recognition. Firstly, deep learning approaches allow for hidden features within text to be discovered automatically and remove the need to manually engineer features as well as the domain expertise needed to create the rules in the first place. Secondly, neural networks (NN) have the option to make use of non-linear activation functions (such as ReLU (Nair and Hinton, 2010)), which are able to pick up non-linear characteristics within the input data. This is a significant advantage over linear models such as HMM and other supervised learning approaches that are not able to detect non-linearity within the data(J. Li et al., 2018).

Typical deep learning architecture for NER is structured in three distinct layers (J. Li et al.,

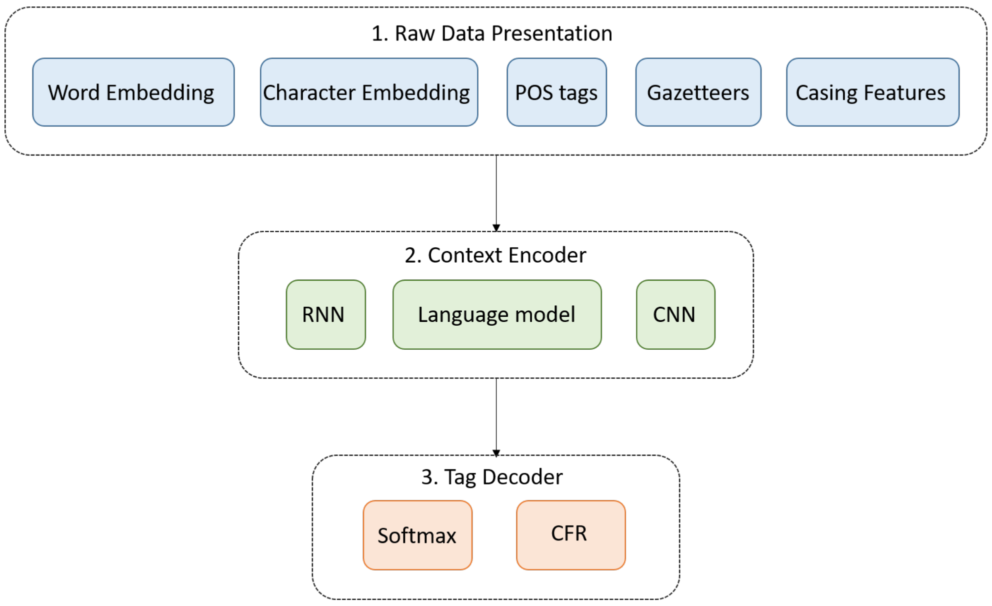


Figure 2.3: The generic deep learning structure for NER.

2018) as can be seen in Figure 2.3. The first layer considers a variety of data representations such as word and character level embeddings, POS tags and external entity gazetteers that are normally associated with rule-based approaches. This layer utilises a variety of data presentations as well as the actual raw words themselves. The second layer, a context encoder, aims to uncover the dependencies between these representations in the first layer before passing these dependencies to the final output layer. A CNN refers to a Convolutional Neural Network; Recurrent Neural Networks (RNNs) are explained in the next section. The last layer, tag decoder, is responsible for predicting the entities in the input sequences of tokens and this is done either with a softmax (Chiu and Nichols, 2015) or conditional random field layer (Huang, Xu, and Yu, 2015).

2.2.1 Word Embeddings

It was not until 2011 that word embeddings (Collobert, Weston, et al., 2011) in conjunction with neural network architecture were applied in NER systems. Word embeddings (Mikolov et al., 2013) play a vital role as they convert each token into a continuous vector, which is able to show the similarity between tokens and thus give tokens context to each other. The similarity between two tokens is calculated by means such as cosine similarity (see formula below). Words whose vectors are orthogonal to one another (90 degrees) are dissimilar and have a cosine value of 0, whilst similar words attain a cosine value closer to 1.

$$\cos(\theta) = \frac{A \times B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.4}$$

Word embeddings are good at capturing general syntactic and semantic information, making them efficient for tackling core NLP tasks. Pre-trained word embeddings are typically trained on massive corpora implementing an unsupervised algorithm such as continuous bag-of-words (CBOW) or skip-gram. The word embeddings (SENNA) were trained from scratch during Collobert, Weston, et al. (2011)'s research, which took over 7 weeks to train on large English corpora from Wikipedia and Reuters. The most commonly used word embeddings include Glove (Pennington, Socher, and Manning, 2014), Word2Vec (Mikolov et al., 2013), FastText (Joulin et al., 2016) and SENNA (Collobert, Weston, et al., 2011). The architecture proposed by Collobert, Weston, et al. (2011) consists of a convolution layer to process the sentences as sequences, which is then fed to a CRF layer, making the final predictions. Table 2.1 below shows the research results.

Approach	POS	CHUNK	NER
NN + WLL	96.31	89.13	79.53
NN + SLL	96.37	90.33	81.47
NN + WLL + LM1	97.05	91.91	85.68
NN + SLL + LM1	97.10	93.65	87.58
NN + WLL + LM2	97.14	92.04	86.96
NN + SLL + LM2	97.20	63.63	88.67

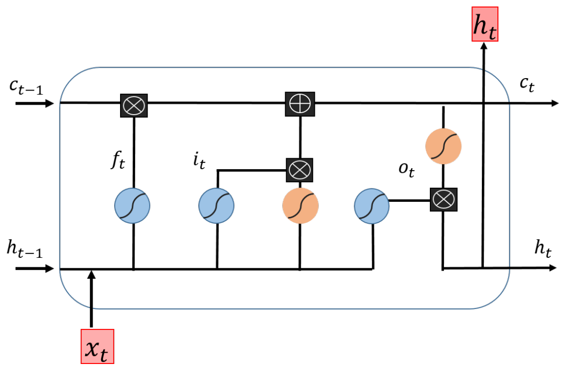
Table 2.1: A comparison of different approaches to three NLP tasks implemented by Collobert, Weston, et al. (2011). Performance for POS is reported as per-word accuracy rate and F1 score for other tasks.

The research paper looked at maximising both word-level log-likelihood (WLL) and sentence level log-likelihood (SLL) with stochastic gradient descent (Bottou, 1991) via with neural networks (NN) whilst utilising two different word embeddings: Language model (LM1) and Language model (LM2). The results above show that embeddings (LM1/LM2) significantly increase the model's performance for the task of NER, improving the baseline model F1 score of

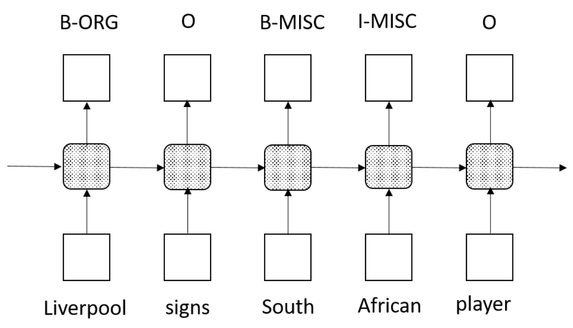
79.53 to 85.68. The second embedding, LM2, returns slightly better results on all the NLP tasks listed above, since LM2 was created by initialising the embeddings of LM1 and then training for an additional 3 weeks. The model architecture achieves an F1 score of 89.59 on the test set with the use of the CoNLL-2003 challenge supplied gazetteer containing 8000 named entities and the SENNA word embeddings. It is also important to note that Collobert, Weston, et al. (2011) also encoded capitalisation as an additional input feature during data preprocessing.

2.2.2 Context Encoders

Recurrent neural network (RNN) (Jain and Medsker, 1999) architecture has shown great success when applied to sequence tagging tasks especially in the area of natural language processing. RNN's have a simple structure with a built-in feedback loop which allows historical information to be passed on sequentially in the form of states. This is very different from a feed-forward neural network (FNN) which only allows the information to be passed in one direction from input to output. There are two popular variants that were developed to combat RNN's short-term memory and detect the long range dependencies between features. Gated recurrent units (GRU) (Cho et al., 2014) and long short-term memory (LSTM) (Hochreiter and Jürgen Schmidhuber, 1997) contain built-in memory cells that aid deep learning architecture to better find and utilise these long-range dependencies for sequence labeling.



(a) LSTM Cell.



(b) LSTM Network.

Figure 2.4: Overview of LSTM cell structure and model architecture.

An LSTM cell contains a network of gates (see Figure 2.4 a) that decide what information is important enough to keep or discard. An input feature (x_t) at a given time-step t feeds into the cell with the information from the previous hidden state (h_{t-1}) and moves through the forget gate (f_t). The forget gate consists of a sigmoid function which compresses the values between 0 and 1. Values which are multiplied by zero are seen as irrelevant information and are removed, whereas values multiplied by 1 are kept and passed on to the cell state (C_t). Of course there are intermediate values between 0 and 1 and so the forget gate is not a binary process. The next gate is the input gate (i_t) which consists of both a sigmoid and tanh activation function. The input data (x_t) and the hidden state (h_{t-1}) are fed to the sigmoid function which decides which values are important to update and which can be ignored. The same data is then fed to the tanh activation function which compresses values between -1 and 1. The sigmoid output is then multiplied by the tanh output, and the results are used together with the forget gate output to calculate the cell state (C_t). Finally, the input data (x_t) and the hidden state (h_{t-1}) are fed through the output gate (O_t). At the same time, the cell state (C_t) is fed from the top through the tanh function. The two function outputs are multiplied to return the next hidden state h_t, which is also the predicted variable h_t. The hidden state and cell states are then passed on to the next cell in order for the process to be repeated with the next input features. Figure 2.4 b) shows a simpler visualisation of how the LSTM network passes along information from the previous states and predicts the target entity for each token.

It wasn't until Graves and Jurgen Schmidhuber (2005) that bidirectional LSTM (BiLSTM) cells were investigated for sequence tagging. Here not only were LSTM cells proved to outperform conventional RNN, but directional LSTMs proved to be significantly superior to uni-directional LSTM cells scoring, 3.2% higher on the TIMIT speech database (Garofolo et al., 1993). The BiLSTM framework allows the model to make use of past features (via forward states) and future features (via backwards states) for a particular time frame, as displayed in Figure 2.5 below. This unique feature essentially gives the model the ability to read the entire sentence before predicting the entity for the single token, and thus encoding context.

In an attempt to achieve the state-of-the art performance without the use of any hand-engineered features or gazetteers, Lample et al. (2016), decided to develop a unique architecture that would run characters through an LSTM layer to create a character embedding. This character embed-

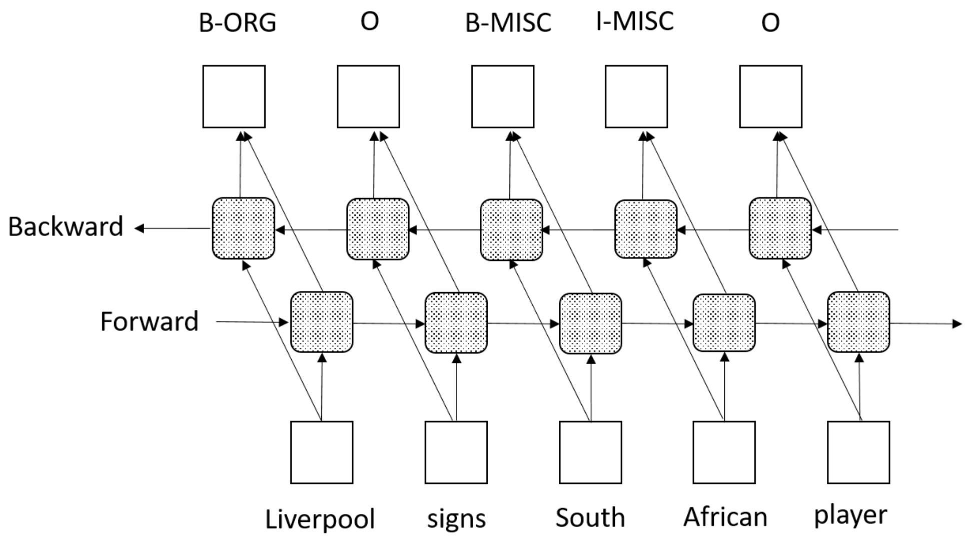


Figure 2.5: An example of a Bi-LSTM network.

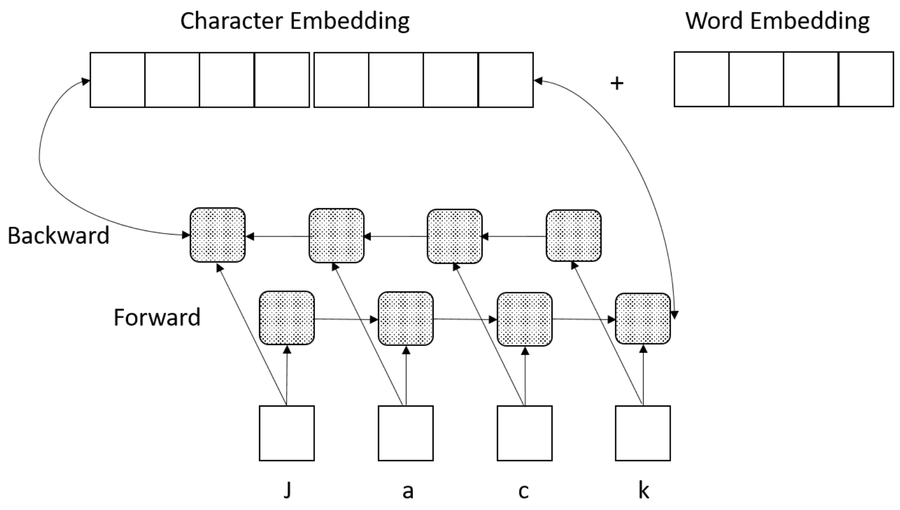


Figure 2.6: Encoding characters and words into a single embedding by Lample et al. (2016).

ding would extract the information on the characters such as casing/spelling features which was then concatenated with a pre-trained word embedding. The concatenated embedding would then be passed through another BiLSTM layer where the CRF was then used to predict the entities. The novel architecture (see Figure 2.6) attained an F1 score of 90.94% on the test set without the use of any external data.

Recurrent neural networks also come with challenges, as they are known for their difficulty to train. Two well-known issues which occur when training include the vanishing and exploding gradient problems (Bengio, Simard, and Frasconi, 1994). The vanishing gradient problem is removed by using an LSTM layer within the architecture, however LSTMs do not solve the exploding gradient problem. There are two proposed solutions to dealing with exploding gradient: gradient normalisation and gradient clipping. Gradient clipping (Mikolov, 2012) involves reducing (clipping) the gradient to a specific maximum or minimum value if the gradient migrates outside of this set range. Gradient normalisation is similar, however it involves clipping the norm of the gradient if above a specific threshold (Pascanu, Mikolov, and Bengio, 2012).

2.2.3 Tag Decoder

The final step of deep learning architecture consists of a decoding layer which is responsible for predicting the entity labels for an input sequence. Two prominent decoders include 1) softmax layer and 2) conditional random fields. A softmax layer will model the task as a multi-class classification problem, essentially decoding the probabilities of every entity label for each token in the input sequence. Chiu and Nichols (2015) connected a bidirectional LSTM network to a decoder consisting of a single linear layer followed by a softmax layer which was able to attain an F1 score of 91.62% on the CoNLL-2003 test dataset with the aid of publicly available external sources. As mentioned earlier in this chapter, conditional random fields have been widely adapted in feature based sequence labeling tasks. Most state-of-the-art approaches have utilised the CRF layer as the decoder. Ghaddar and Langlais (2018) matched the state-of-the art performance by utilising a BiLSTM-CRF network, which takes the feature engineering to new limits by implementing lexical similarity vectors and attaining a score of 91.76% on CoNLL-2003 test dataset.

2.3 Evaluating NER Systems

Many evaluation metrics (Chinchor and Sundheim, 1993) for named entity recognition have been developed over the years in an attempt to design a comprehensive metric on which to compare information extraction systems. NER systems are primarily evaluated on three key metrics: precision, recall and F1 score. For each entity category, precision (2.5) is defined as the number of correct predictions produced by the system divided by the total number of predictions made for that particular category. Therefore, precision is the percentage of predicted tokens that are correctly labelled for a particular entity. For each entity category, recall (2.6) is defined as the number of correct predictions produced by the system divided by the total number of human annotated labels. Therefore, recall is the percentage of that entity category that was actually labelled by the model.

$$Precision = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Positives\ (FP)} \tag{2.5}$$

$$Recall = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Negatives\ (FN)} \tag{2.6}$$

Precision and recall are often traded off against each other; typically as recall increases, precision tends to decrease or vice versa. Methods have been developed to quantify the precision-recall trade-off into a single measure such as Van Rijsbergen's F1 score (Rijsbergen, 1979), which was first introduced at the Fourth Message Understanding Conference (MUC-4) in 1992 (Grishman and Sundheim, 1996). The measure aims to find a balance between the two measures and is defined as a the harmonic mean of precision (P) and recall (R) (Sasaki, 2007). The difference between harmonic (H) and arithmetic (A) mean is shown below:

$$A = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{n}(x_i + x_2 + ... + x_n) = \frac{1}{2}(P + R) \tag{2.7}$$

$$H = \frac{n}{\sum_{i=1}^{n}\frac{1}{x_i}} = \frac{n}{\frac{1}{x_1} + ... + \frac{1}{x_n}} = \frac{2}{\frac{1}{P} + \frac{1}{R}} = \frac{\frac{2}{P+R}}{PR} = \frac{2PR}{P+R} \tag{2.8}$$

The parameter β is used to control the importance given to recall over precision. The F1 score favours the importance of recall if $\beta > 1$ is used, and alternatively favours precision if $\beta < 1$.

$$F_\beta = \frac{(\beta^2 + 1) \times Precision \times Recall}{(\beta^2 \times Precision) + Recall} \tag{2.9}$$

When recall and precision are weighted evenly, $\beta = 1$ is used, resulting in a harmonic mean and thus called the F_1 score.

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2.10}$$

The harmonic mean incentivises the development of the overall performance, seeing as the measure favours precision and recall values that are located towards the centre of the precision-recall curve and instead punishes the extremities (Rijsbergen, 1979). For example, suppose that an NER system attains a precision of 0.9 and recall of 0.1. Intuitively, the total performance of the model should be low considering that the NER model only correctly predicted 1/10 of the entities in the document. As can be calculated using the equations above, the arithmetic mean gives a score of 0.5 whilst the harmonic mean returns an F-score of 0.18, confirming the model has indeed under-performed. Despite alternative measures, this quality has made the F1 score broadly favoured in a large variety of NLP tasks such as named entity recognition.

Additional metrics for analysing error types can also be utilised in the form of error summaries. A study by B. A. Nguyen, K. V. Nguyen, and N. L.-T. Nguyen (2019) explores a variety of different error types, namely no extraction, no annotation and wrong tag errors. The errors are defined below:

No extraction: An error where the model does not extract a token as a named entity, which is in fact a named entity.

No annotation: An error where the model extracts a token as a named entity, which is not a named entity.

Wrong tag: An error where the model extracts a token as a named entity, however the entity label is incorrect (e.g."PER") but the boundary is correct (e.g."I-").

Wrong range: An error where the model extracts a token as a named entity, however the entity label is correct (e.g."PER") but the boundary is incorrect (e.g."I-").

Wrong range + tag: An error where the model extracts a token as a named entity and both the entity label (e.g."PER") and the boundary are incorrect (e.g."I-").

Overall, various approaches have been developed over time to tackle the task of NER from rule-based to deep learning. Deep learning has the unique ability to detect non-linear patterns within the data and expose dependencies between words with the help of LSTM cells. The generic architecture of a deep learning NLP model was outlined in addition to the importance of word embeddings and the significant improvement they can have on an NER system. Lastly, model evaluation metrics were reviewed with special focus on the calculation of the F1 score. The next chapter looks at the characteristics of the CoNLL-2003 dataset as well as the data format.

Chapter III

Data

This chapter provides a detailed overview of the dataset used to conduct the research for this dissertation. The selected dataset for this research study is the CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003) NER shared task which is comprised of a collection of news articles from Reuters Corpus and annotated by the people at the University of Antwerp. The dataset was licensed through the Reuters Organisational Usage Agreement (which can be found in Appendix A), allowing access to the RCV1 corpora for research purposes. The CoNLL-2003 dataset is commonly used as one of the benchmark datasets on which the modern named entity recognition (NER) systems are evaluated and compared.

3.1 Data Exploration

One of the biggest challenges with supervised learning tasks, such as NER, is the need for extensively labelled data. Labelled datasets utilised for NER systems require extensive resources to curate because each token in the corpus of documents has to be manually labelled with an entity label. The dataset itself contains named entities for both English and German in the form of a training, test, and a validation file. The CoNLL-2003 dataset has four named entity tags, namely locations [LOC], persons [PER], organisations [ORG] and miscellaneous named entities [MISC] that do not belong to the previous three groups.

	Train	Val	Test
Sentences	14041	3250	3453
Tokens	203621	51362	46435
Unique Tokens	21009	9002	8548

	LOC	ORG	PER	MISC
Train	8297	10025	11128	4593
Val	2094	2092	3149	1268
Test	1925	2496	2773	918

Table 3.1: The size of the CoNLL-2003 data sets, as well as the number tags per set.

The training set comprises approximately 80% of the sentences in the entire dataset and holds more than 70% of the unique tokens. As can be seen in Table 3.1, the training dataset contains 21,009 unique tokens with the entire corpus containing a total of 26,869 unique tokens, of which only 4,856 are non-alphabetic and comprised of primary numeric variations or punctuation. The average sentence length is 15 tokens with the shortest and longest sentences containing 1 and 124 tokens respectively. The distribution of sentence length across the entire dataset can be seen in Figure 3.1 below. The top 10 most frequent words can be found in Figure B.2.1 in Appendix B.

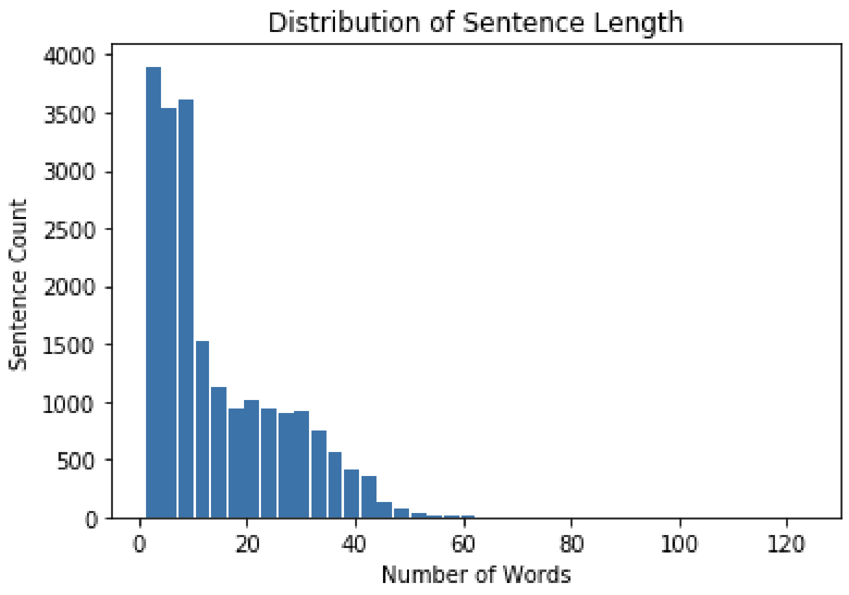


Figure 3.1: Sentence length distribution within CoNLL-2003.

Another important feature of the training data that cannot be overlooked is the inherent class imbalance between the "O" labels and the primary entity labels. This imbalance is displayed in Figure 3.2 below. The "O" labels fall outside the entity labels mentioned in the paragraph above and form the remainder of the sentence. The primary objective of named entity recognition is to locate and extract the primary entities (ORG, PER, LOC etc.); therefore it is important to take this into account when evaluating a NER system's performance. The shear number of "O" labels within the dataset would significantly skew the F1 score, elevating the system's performance. Therefore, in this study the "O" entity label is excluded from the F1 score calculation to reflect a more comprehensive model evaluation as is common practice (Lample et al., 2016; Huang, Xu, and Yu, 2015; Chiu and Nichols, 2015).

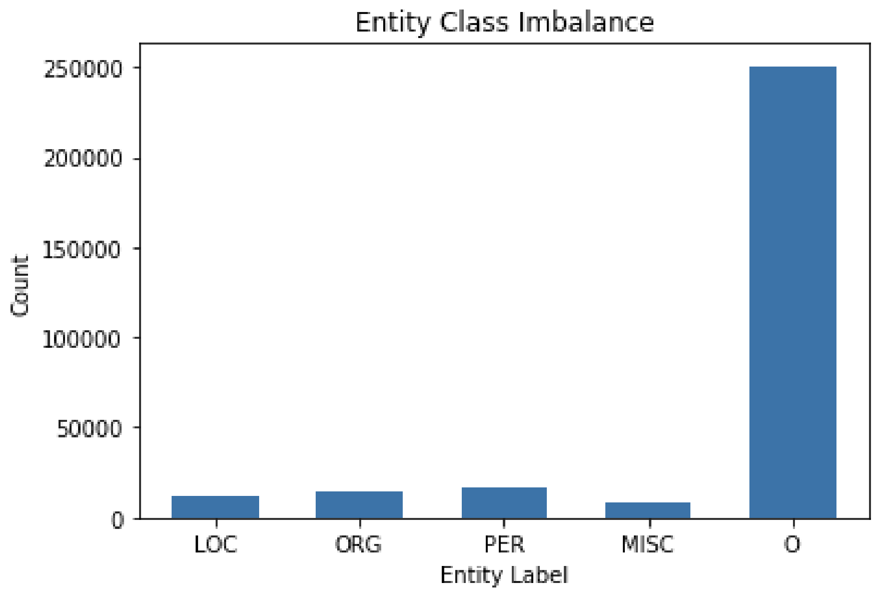
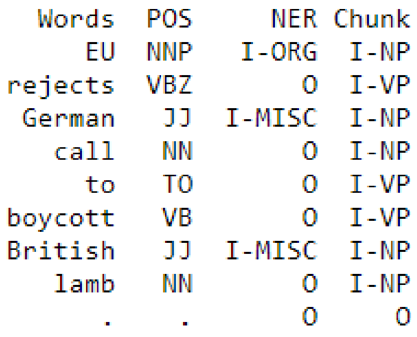


Figure 3.2: The imbalance of entities within CoNLL-2003.

3.2 Data Format

The three data files (ASCII text-formatted) consist of four columns separated by a single space. Each row in the data file contains a token, which belongs to a sentence. An empty row demarcates sentence boundaries as well as a “DOCSTART” string which defines the documents boundaries within the corpora. Any given row contains four data points: the token, the part-of-speech (POS), the syntactic chunk, and the named entity label. An example of this data format can be seen Figure 3.3.

The CoNLL-2003 shared task challenge also supplies an additional 8,000 entities gazetteer containing names of locations, organisations, people and miscellaneous entities. In this study,



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

Figure 3.3: CoNLL-2003 data format.

it has been decided to omit the use of these gazetteers as well as any additional information, including the part-of-speech (POS) and the syntactic chunks. Only the raw tokens shall be used as input data for training models in this dissertation.

In this chapter data exploration was conducted in an effort to understand the characteristics of the CoNLL-2003 dataset. The dataset has an inherent class imbalance between "O" labels and the primary entity labels. This class imbalance is addressed by excluding the "O" labels from the evaluation calculations. The following chapter outlines the methodology used to develop both NER models and the framework for hyperparameter tuning and model evaluation.

Chapter IV

Methodology

This chapter details the methodology followed in this dissertation. Specifically, the focus will be on the development of a Conditional Random Fields (CRF) model as well a Bidirectional-LSTM-CRF (BiLSTM-CRF) model. Firstly, the engineered features, window sizes and structure of the CRF model are outlined. Secondly, the BiLSTM-CRF model architecture is detailed, along with the preprocessing of data and selection of word embeddings. Finally, the framework for hyperparameter tuning and model evaluation is defined. The code base pertinent to this section can be found on this GitHub repository[1].

4.1 Conditional Random Fields

Conditional Random Fields (CRF), an undirected statistical graph model, is selected as the baseline model due to its impressive performance with sequence modelling tasks within natural language processing. This particular statistical model (single linear CRF) also forms an integral part of the final hybrid model (BiLSTM-CRF), of which the primary research is conducted. As explored in Chapter 2, exponential models such as conditional random fields contain convex likelihood functions and thus finding a global maximum is guaranteed (Lafferty, McCallum, and F. C. N. Pereira, 2001). This adds to the appeal for using conditional random fields as

[1]https://github.com/deanhoperobertson/Named-Enitty-Recognition

a baseline model. This baseline NER system was inspired by the earlier work published by McCallum and W. Li (2003).

4.1.1 Features

CRF is a feature-based supervised learning approach that requires a feature-engineering process that carefully extracts clues from the data, which the supervised learning algorithms utilise in order to find patterns. Feature engineering aims to create word-level features of each token within the text, extracting orthographic and grammatical information. Even though other NER models use lookup lists (lexical) and part-of-speech tags (syntactic) in conjunction with the extract features, these are not included in the input data for this baseline system. The features engineered in this experiment (see Table 4.1) can be separated into three categories: orthographic, morphological, and contextual. These features are extracted from any given token and used in conjunction with the lower case version of the token as the input data.

Orthographic	**Morphological**	**Contextual**
Start with capital letter	Prefix (window size 2 to 3)	Start of sentence
All upper case letters	Suffix (window size 2 to 3)	End of sentence
All lower case letters		
All digits		
All letters		
Mix of letters and numbers		
Has non initial capital letters		
Has punctuation		
Has apostrophe end ('s)		
Word pattern feature		
Word pattern summarisation		

Table 4.1: Summary of engineered features used in the CRF model.

Word pattern features are regular expressions that map capital letters, lower case letters, and digits to "A", "a" and "0" respectively. The word pattern summarisation goes one step further and aims to reduce the length of the pattern by removing any consecutive identical characters in the pattern. For example, "AAa0" is reduced to "Aa0".

4.1.2 Window Sizes

In order to give the CRF model context, a window is used to add additional neighbouring words located next to the primary word. When the window size is set to zero (W_0), no surrounding tokens are added and the input data only contains a single primary token at any instance. A window size of 1 ($W_{-1},\ W,\ W_{+1}$) allows for one token on either side of the primary token to be used as input data. During training, various window sizes were explored with a maximum window size of 3 ($W_{-3},\ W,\ W_{+3}$) similar to the window size used by Carreras, Marquez, and Padro (2003) and Mayfield, McNamee, and Piatko (2003).

4.1.3 Training

Conditional random fields are trained to maximise the likelihood of the training data in order to determine the corresponding vector weights. This is done more efficiently by maximising the log-likelihood conditional probability of any label (Y) for an observation (x).

$$\ell(\theta) = \sum_{i=1}^{N} \log\big(p(Y_i|x_i, \theta)\big) \tag{4.1}$$

This produces a convex optimisation function and therefore has a single global optima, thus CRFs are able to converge with relative ease. In early attempts at training, the CRF model tended to overfit the training data and thus a smoothing method, such as regularisation, is needed. Regularisation acts as a penalty to weight vectors that are too large and therefore reduce the chance of overfitting during training (Sutton and McCallum, 2012). There are two common forms of regularisation, namely L1 and L2 regularisation, which creates a trade-off between finding an optimal solution and the size of the weights, thus penalising weights for being too large.

$$\ell_2(\theta) = \sum_{i=1}^{N} \log\big(p(Y|x)\big) - \sum_{k} \frac{\theta_k^2}{2\sigma^2} \tag{4.2}$$

For L2 regularisation, the second sum in the equation is a Gaussian prior, where the λ_k is the learned weight of each feature (k) in the input data. This means that penalty terms are applied to each feature and subtracted from the log likelihood. The $1/\sigma^2$ penalty parameter determines to what extent the weights are penalised. Large values for $1/\sigma^2$ correspond to harsher penalties, whilst smaller values closer to zero implement no regularisation. The resulting optimisation function with the regularisation penalty is still convex in shape and therefore the penalty does not add to the computational effort needed to find a converged solution (Vail, Veloso, and Lafferty, 2007).

$$\ell_1(\theta) = \sum_{i=1}^{N} \log\big(p(Y|x)\big) - \alpha \sum_k |\theta_k| \tag{4.3}$$

L1 regularisation differs from L2 regularisation, which replaces the Gaussian prior with an exponential prior. The absolute value of the weights allows the weight parameters to be constrained completely to zero, and thus acting as a technique for feature selection, resulting in a sparser model (Sutton and McCallum, 2012).

Limited Memory BFGS (L-BFGS) (Liu and Nocedal, 1989) is selected as the training optimisation algorithm due to the fact that it uses second order likelihood derivations to estimate the gradients numerically. This results in a faster convergence when training CRFs compared to alternative algorithms (Sha and F. Pereira, 2003). One disadvantage of using L1 regularisation is that the likelihood function is non-differentiable at 0, adding to the complexity of training CRFs. In an effort to overcoming the lack of constraints, the Orthant-Wise Limited-memory Quasi Newton (Andrew and Gao, 2007) optimisation algorithm is used, which fixes the coordinate sign. An advantage of using L1 over L2 regularisation is that it has very little effect on model accuracy, but due to the nullifying of weight parameters can results in a more computationally efficient solution. Both L1 and L2 regularisation penalty parameters can be determined through grid searching via cross-validation.

In the next section, the framework for tuning the hyperparameters will be outlined in which both L1 and L2 coefficients are determined.

Hyperparameter Optimisation

Early stage grid searches reduced our penalty coefficient domain to a range between 0 and 1. Each domain is split into 11 possible values for each L1 and L2 regularisation coefficient, resulting in 121 possible combinations. During the hyperparameter tuning, the possibility of an elastic net regularisation (Zou and Hastie, 2005) was considered as both L1 and L2 regularisation could be applied simultaneously. A standard grid search is then used to determine the best L1 and L2 regularisation coefficients for each window size as shown in the order of operations for hyperparameter tuning below:

1. For every given window size (W_i) :
   (a) Grid search the parameter space (0, 1) for both regularisation coefficients.
2. Choose best performing combination of hyperparameters.

The class imbalance within the training data prohibits the use of smaller sample datasets, and instead hyperparameter tuning is conducted on the entire training dataset. The grid searches for L1 and L2 coefficients using 3-fold cross-validation – to ensure sufficient training samples are seen in each fold – results in fitting approximately 363 models per window size. Similar to the approach implemented by McCallum and W. Li (2003), the maximum number of iterations is limited to 10 of L-BFGS during hyperparameter tuning. This computationally expensive process was executed using Google Colab Pro's cloud computing platform and run on 4 parallel CPU's with a total of 25GB RAM.

Symbol	**Description**	**Type**	**Range**
$c1$	The coefficient for L1 regularisation	Continuous	(0, 1)
$c2$	The coefficient for L2 regularisation	Continuous	(0, 1)

Table 4.2: Summary of tunable hyperparameters for conditional random fields.

4.2 Bidirectional-LSTM-CRF

Bidirectional LSTM-CRF models have an advantageous characteristic for detecting patterns in context-dependant representations such as text. In particular, the bidirectional context encoding layer comprised of LSTM cells allows for information to be passed both forward (via forward states) and backwards (via backward states). This ability to not only look ahead but also behind a token allows for a single token to be encoded with information from the entire sentence at any point in time. The sequential text data contains strong dependencies between tokens and thus the output entity labels. To model these dependencies, the bidirectional LSTM network is fed directly into a conditional random field layer which allows for tagging decisions to be modelled jointly (Lample et al., 2016).

4.2.1 Preprocessing

Data preprocessing is required to transform the raw data into a standardised format in order to be passed through the neural network structure. In this section, data is preprocessed in an effort to prepare the data into a compatible format, and also handle certain tokens in a manner that reduces the feature set.

Formatting

The raw data is first processed from a static text file (.txt) into one single string object, which is then converted into a list of strings (sentences). No tokenization is needed seeing as the data has already been split into individual tokens comprised of English words, punctuation, dates and numbers. All these tokens contained within the training data make up the text corpus. In order for numerical machine learning to be applied to the text, the text data needs to be converted into vector representation through a process known as vectorization. In this process each token in the corpus is indexed into a vocabulary dictionary and then each token is converted into an integer. This process transforms a sentence into a vector. For the sake of simplicity, in this dissertation only uni-grams (single token combinations) are considered for vectorization and thus bi-grams and tri-grams are ignored. For any new token that may arise in the test set and is not indexed in the initial vocabulary, an "UNK" string is used to represent these out-of-vocabulary tokens.

As shown in the data exploration covered in Chapter 3, sentences vary in length ranging from a single token to 124 tokens. The input layer for the BiLSTM-CRF model is of a fixed shape and therefore set to a length of 124 to handle the longest sentence found in the training data. All sentences need to be the same length in order to be fed into the network. A process known as padding is used to pad the shorter sentences with zero's ("0") to make all the sentences in the corpus of equal length. Figure 4.1 illustrates this process.

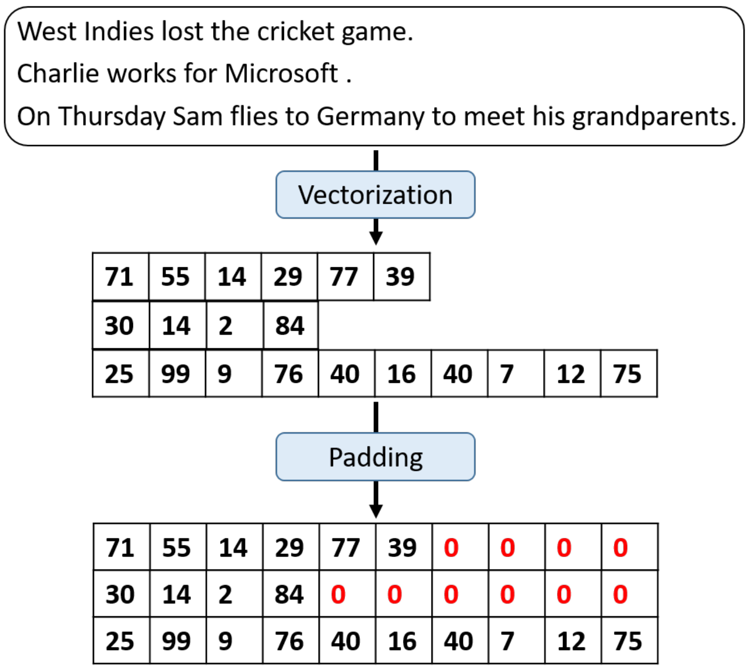


Figure 4.1: An example of text vectorization and sentence padding.

The named entity labels, being the target variables, are also vectorized in a similar fashion to the text data. Since the entity classes have no relationship to one another and are mutually exclusive, they can be transformed into a set of binary vectors through a process known as one-hot encoding. This process involves creating a binary vector representation where one element in the vector denotes an entity label and is allocated a "1", whilst the rest of the elements are allocated zero's. Table 4.2 displays how these entity classes are vectorized. All of the preprocessing covered in this section was implemented with the use of Keras's *preprocessing* and *utils* Python packages (Chollet et al., 2015).

I-PER	**1**	0	0	0	0	0	0	0	0
I-PER	0	**1**	0	0	0	0	0	0	0
B-LOC	0	0	**1**	0	0	0	0	0	0
I-LOC	0	0	0	**1**	0	0	0	0	0
B-ORG	0	0	0	0	**1**	0	0	0	0
I-ORG	0	0	0	0	0	**1**	0	0	0
B-MISC	0	0	0	0	0	0	**1**	0	0
I-MISC	0	0	0	0	0	0	0	**1**	0
O	0	0	0	0	0	0	0	0	**1**

Figure 4.2: One-hot encoding target variables.

Dimensionality reduction

There are over 23,623 unique tokens in the training set, which is reduced to 21,009 when all tokens are converted to lowercase. Casing can be a strong indicator as to whether a token is a named entity. However, upper casing can also be deceiving since words located in the beginning of a sentence are traditionally upper cased. Contradictory to the CRF approach, BiLSTM does not use any engineered features and instead relies on exposing the hidden relationships between tokens. Without capitalisation used in engineered features, there is little use for cased tokens. Therefore, all tokens within the text corpus are reduced to lowercase which in turn reduces the size of the vocabulary (Collobert, Weston, et al., 2011). A reduced vocabulary results in fewer words to look up within the word embeddings and thus directly reduces the dimensionality of the embedding layer, which is discussed in the following section.

Numbers are handled in a similar way to the word casings. The number, whether it represents a persons age or a team's score, are all a form of numerical magnitude. For the purpose of detecting entities within raw text, this scale of magnitude will make little difference. However, the location of these numerical values within the text will play a vital role in entity detection. Similar to the work by Chiu and Nichols (2015), all digits within the text corpus are replaced with a "__" token. This reduces the 3,865 unique digit tokens to one single generic token.

4.2.2 Word Embeddings

Word embeddings are distributed representations of words which capture the syntactic and semantic properties of a word. Word embeddings are good at representing the relationship between words and how they interact with each other. This unique property of word embeddings has been shown by Collobert, Weston, et al. (2011) to have a significant improvement on sequence tagging tasks such a NER. The two word embeddings this research focuses on are Standford's GloVe (Pennington, Socher, and Manning, 2014) and SENNA embeddings (Collobert, Weston, et al., 2011). Both embeddings are open source and publicly available online.

SENNA[2] embeddings were trained on approximately 850 million words from Wikipedia and Reuters RCV1 corpus (Lewis et al., 2004). The SENNA embeddings were trained using deep neural architecture where an embedding layer was trained using stochastic gradient minimisation of a pairwise ranking criterion. The downloaded files are processed using a custom Python processing script which can be found in file Senna.py in the GitHub repository. The script standardises the embedding into a single text file of a specified format. The SENNA embedding itself contains 130,000 tokens of which each token corresponds to a 50-dimensional continuous vector.

Unlike SENNA embeddings, GloVe[3] is not trained via deep learning (i.e. no hidden layers). GloVe was trained using gradient descent to perform matrix decomposition of the log co-occurrence matrix. The co-occurrence matrix is produced by counting the number of times two words are used in the same context. GloVe was trained on 6 billion words scraped from Wikipedia 2014 and Gigaword5. GloVe has a greater vocabulary (400,000 tokens) and offers embeddings with a greater vector length between 50 to 300.

[2]https://ronan.collobert.com/senna/
[3]https://nlp.stanford.edu/projects/glove/

Word	Cosine Sim
blackberry	0.754
chips	0.744
iphone	0.743
microsoft	0.733
ipad	0.733
pc	0.722
ipod	0.720
intel	0.719
ibm	0.715
software	0.709

(a) GloVe

Word	Cosine Sim
piggy	0.596
chicken	0.567
cookie	0.566
sandwich	0.561
nokia	0.561
spaghetti	0.556
safeway	0.551
peanut	0.548
rye	0.547
pretzel	0.546

(b) SENNA

Table 4.3: Ranked cosine similarity for the token "apple".

Both embeddings will have unique representations of the tokens as well as their relationships with each other. The difference between the two embeddings can largely be attributed to the difference in training methodology and corpora on which the embeddings were trained on. Table 4.3 above shows an example of how each embedding may differ in its representation of a token. In this example, a search is conducted to find the most similar word to the token "apple". To do this, the cosine similarity between every vector in the embedding and the "apple" vector is calculated. The tokens with the greatest cosine similarity are the most similar to the target token. As shown above, the GloVe embedding returns words associated with technology as such as "iphone", "pc" and "software", whilst the SENNA embedding returns words typically associated with food such as "chicken", "sandwich" and "peanut". This vastly different interpretation between the two embeddings can be attributed to the difference in age of their training data. The majority of the SENNA training data was acquired from a 2007 version of Wikipedia, whilst GloVe's training data is from 2014. Tokens such as "ipad" are not even present in the SENNA embeddings, since the ipad was only released in 2010, after the development of SENNA.

Figure 4.3 shows how similar the top 10 token vectors are to the "apple" vector. Principal component analysis (PCA) is a dimensionality reduction technique that simplifies large datasets into reduced set of summary dimensions which allow data to be easily visualized and analyzed (F.R.S., 1901). The 50-dimensional vectors are reduced to two dimensions by means of PCA.

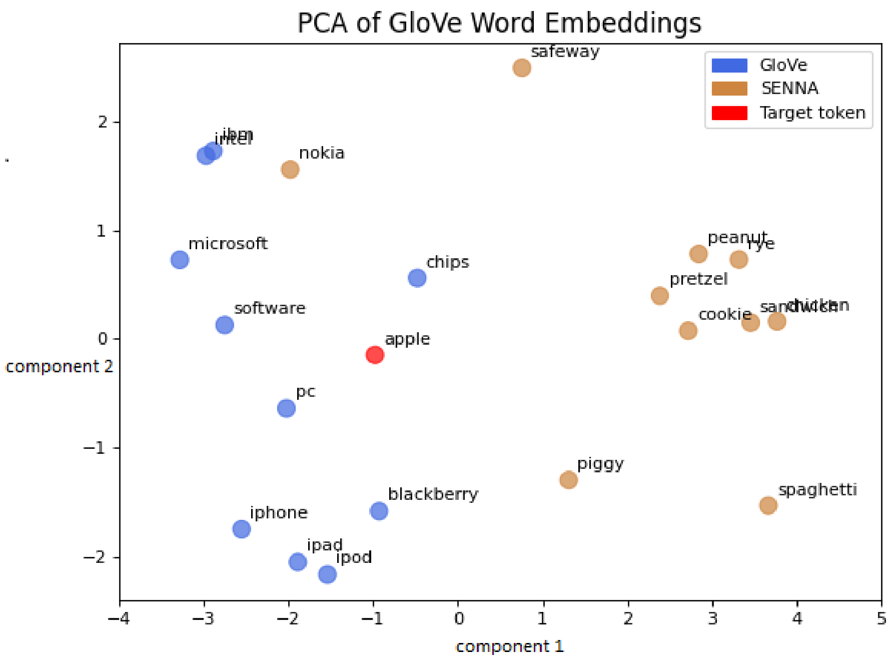


Figure 4.3: Visualisation of both embeddings via PCA.

From the figure below one can clearly see that the closest vectors to the target token ("apple") seem to be GloVe vectors. Due to this difference in composition between the two embeddings, this study will test both embeddings in an effort to see which embedding produces a higher performance on the CoNLL-2003 dataset. It is important to note that these embeddings do not contain every token in the CoNLL-2003 dataset and that there is a small percentage of missing tokens. Table4.4 shows the percentage of tokens missing from each embedding. The reason for some missing tokens ranges from spelling mistakes to unique person names that would typically not be found in a word embedding.

Embeddings	**Tokens missing**	**Missing %**
GloVe	1218	7.10%
SENNA	1554	9.05%

Table 4.4: Tokens missing in both GloVe and SENNA embeddings.

4.2.3 Network Architecture

A bidirectional LSTM with a conditional random fields layer is the selected network architecture to be explored in this study. BiLSTM architecture has been favoured for sequential tagging tasks such as NER due to its ability to uncover hidden features within raw text and reduce the dependencies on the domain expertise and over-engineered features. As mentioned in Chapter 2, the architecture comprises of three distinct layers, namely: input layer, context encoder and tag decoder. This section will cover the structure of each layer and the relationship that exists between them.

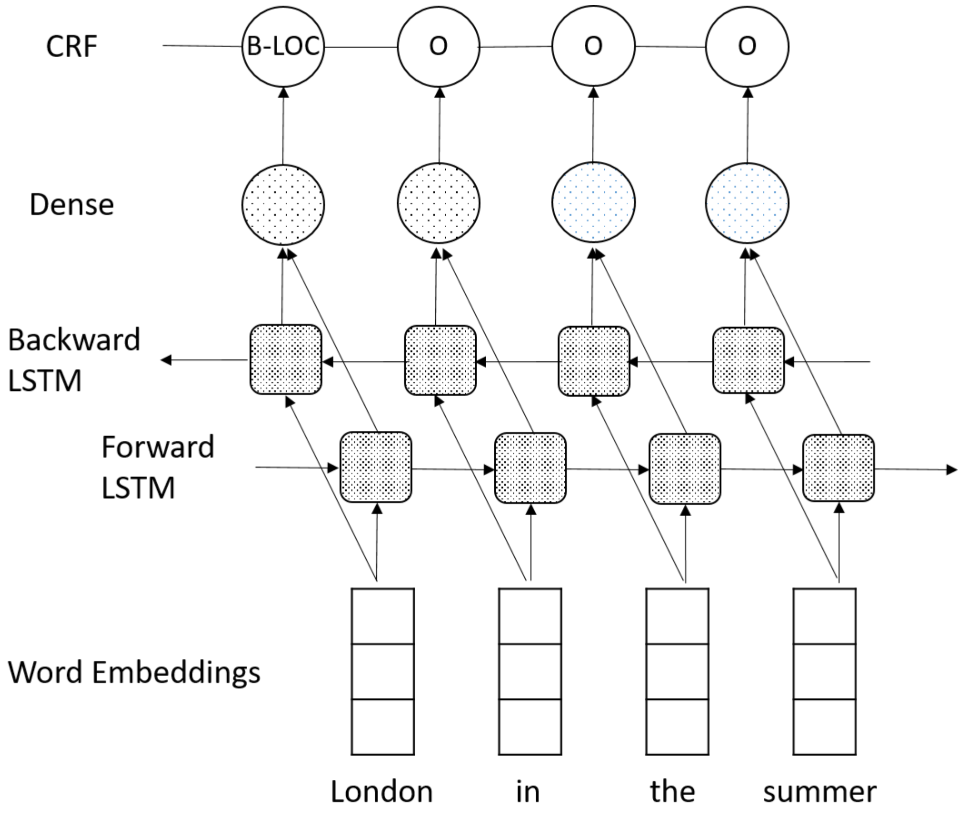


Figure 4.4: BiLSTM-CRF network architecture.

Input Layer

The input layer for the neural network is designed to process vector representations of tokens looked up within the embeddings. The input layer size needs to be able to handle the longest sentence in the CoNLL-2003 dataset, which is approximately 124 tokens. The Keras *models*

API is used to instantiate a 2D tensor of shape (None, 124). The input layer is capable of processing 124-dimensional vector, keeping in mind the sentences have all been padded to the same length during data preprocessing. The first dimension of the input layer is batch size, which is assigned as "None". This "None" dimension caters for the shape changing during the development process of this model, since this parameter can be tuned.

The embedding layer [4] also forms part of the input layer where each token is assigned a 50-dimensional vector from the pre-trained word embedding matrix. The embedding layer acts as a big lookup table, converting the 2D tensor (batch_size, 124) into a 3D tensor (batch_size, 124, 50). It is important to use masking, which allows the embedding layer to ignore the padding tokens from the lookup process. Since this layer is using pre-trained word embeddings, there is no need to further train this layer and instead the weights are fixed.

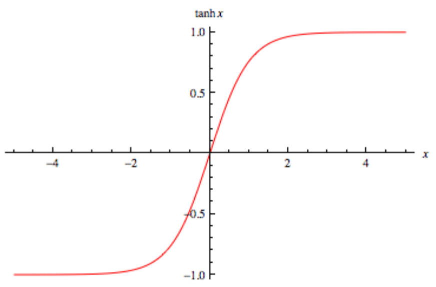
BiLSTM Layer

The bidirectional LSTM layer[5] is responsible for encoding the context by processing token sequences from both directions. The LSTM cells (Hochreiter and Jürgen Schmidhuber, 1997) contain a complex network of gates that allows information to be retained whilst discarding the irrelevant information. The LSTM cells are capable of recognising the long-range dependencies between the tokens, which is then later used by the network to predict a token entity tag. Within this layer, non-linearity is introduced in the form of activation functions as well as well as dropout, a regularization technique that randomly removes neurons in the network during training (Srivastava et al., 2014).

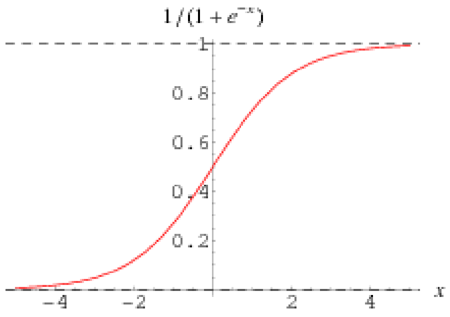
An activation function allows the network to detect non-linear patterns within the data. These functions are then used to update the network weights during backpropagation. These functions are particularly useful when trying to classify complex data in which the decision boundary may be non-linear in nature. Without these functions the network would be performing simple linear regression. An activation function is used for the entire layer as well as another activa-

[4]https://keras.io/api/layers/core_layers/embedding/
[5]https://keras.io/api/layers/recurrent_layers/bidirectional/

tion function for each recurrent time step. The Keras default activation functions of tanh and sigmoid (Figure 4.5) are used for layer activation and recurrent activation respectively. Lastly, the bidirectional LSTM layer can be of variable size. The number of LSTM cells is one of the hyperparameters along with dropout that can be tuned. The length of the resulting encoding tensor is doubled, since the forward and backward LSTM cells are concatenated into one single tensor.



(a) Tanh Function.



(b) Sigmoid Function

Figure 4.5: LSTM activation functions.

Output Layer

The output layer is comprised of a fully-connected hidden dense layer[6] which is then connected to a final conditional random fields layer. The single hidden dense layer allows for a deeper level of learning (additional weights) and another opportunity to detect non-linearity with additional activation functions. The chosen activation function for this layer is a Rectified Linear Unit, also known as ReLU. ReLU is a commonly used activation function due to its favourable properties. The function is more computationally efficient, and a consistent gradient of 1 assists a faster convergence as apposed to gradually plateaued gradients found with exponential functions such as tanh and sigmoid functions (Baheti, 2021). A value of zero is returned if the input is less than zero, resulting in sparse representations across neurons in the dense layer. This sparsity produces a subset of neurons that are active, which has been shown to have a number of advantages for tasks involving text data (Glorot, Bordes, and Bengio, 2011).

[6]https://keras.io/api/layers/core_layers/dense/

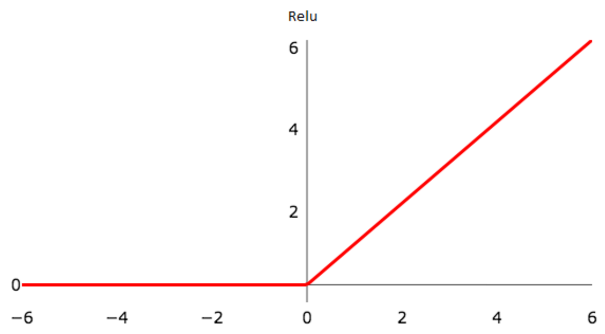


Figure 4.6: ReLU activation function.

A TimeDistributed [7] wrapper layer is used around the dense layer so that only one entity tag is predicted per timestamp, given the full sequence of tokens as an input. This is aligned with the previous BiLSTM layer that returns a sequence of values for each timestamp. Lastly, the dense layer is fully-connected to a CRF layer [8] of which the dimensionality must equal the number of possible entity labels, in this case 9. The CRF layer uses the default linear activation function.

4.2.4 Training

Training a neural network is an iterative process in which the weights for each neuron are slowly updated through forward and backpropagation (see Figure 4.7). The weights are randomly initialised before the first batch of input data is passed through the entire network. Each neuron has an activation function (mentioned in the previous section) that controls the output value and usually introduces some non-linearity into the network. The data is passed through each neuron from one layer to the next, finally resulting in a predicted label. A loss function is then used to calculate the error between the predicted value and the true value. The error is then passed backwards through each layer and the weights are adjusted in an effort to minimise the loss and reduce the error closer to zero. The updating of the weights is done by means of gradient descent. Gradient descent is a method of updating the weights in small increments by calculating the derivative of the loss function after every backpropagation. By calculating the

[7]https://keras.io/api/layers/recurrent_layers/time_distributed/
[8]https://github.com/keras-team/keras-contrib/blob/master/keras_contrib/layers/crf.py

gradient of the loss function it is possible to determine in which direction the global minimum is and what changes need to be made to the weights in order to reach it.

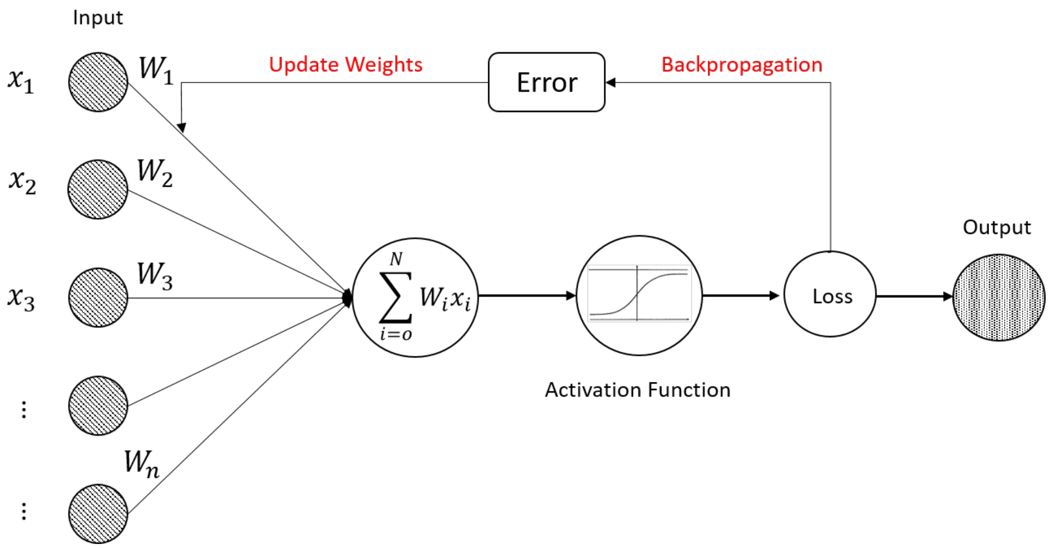


Figure 4.7: A visual illustration of backpropagation.

The gradient descent algorithm used for training is referred to as an optimiser within the framework of Keras. A number of different algorithms were explored during hyperparameter tuning, including Adam (Kingma and Ba, 2014), Nadam (Dozat, 2015) and stochastic gradient descent (SGD). All algorithms use a fixed learning rate of 0.01 and gradient clipping is explored to enhance performance.

The output layer of the network is a CRF layer which allows the network to compute the negative log-likelihood for a linear chain conditional random field. This loss function is coupled with a metric function which uses the Viterbi algorithm (Viterbi, 1967) to assess the model's performance during training. The metric functions are similar to loss functions, however only the loss function is used to adjust the weights. In order to ensure adequate training, the network will be trained to a total of 20 epochs and early stopping is utilised. Early stopping monitors the validation loss after each epoch. The goal of training is to minimise the loss and therefore if the validation loss does not reduce for two consecutive epochs, the training is terminated.

In the next section, the framework for hyperparameter tuning will be outlined.

Hyperparameter Optimisation

Similar to the hyperparameter tuning conducted for conditional random fields, a grid search is needed to test all possible hyperparameters. Traditionally a KerasClassifier[9] wrapper is used in conjunction with $scikit-learn$'s GridSearchCV to finely test all the variables within the neural network. Due to the one-hot-encoding of the target variable, the target variable is converted to a three-dimensional vector. Unfortunately KerasClassifier is unable to handle data larger than 2 dimensions and thus a custom approach is needed to conduct a grid search.

A for-loop is utilised with a model selection package known as $Kfold$ (Pedregosa et al., 2011), allowing us to split the training data into 3 random folds. Each fold is then used once as a validation while k-1 remaining folds form the training set on which the BiLSTM-CRF model is built and trained on. Table 4.5 displays which parameters were tuned during this process.

Symbol	Description	Type	Range
$batch\_size$	The number of batches	Discrete	(32, 256)
$lstm\_units$	Number of LSTM cells	Discrete	(100, 300)
$layer\_dropout$	The dropout rate in the lstm layer	Continuous	(0, 1)
$recurrent\_dropout$	The dropout of recurrent connections	Continuous	(0, 1)
$dense\_units$	Number of nodes in hidden layer	Discrete	(10, 100)
$clipvalue$	gradient clipping	Continuous	(0, 5)

Table 4.5: Summary of tunable hyperparameters for BiLSTM-CRF network.

In deep learning, batch size refers to the number of training samples that are passed through the network in one iteration. For example, a batch size of 32 means a total of 32 sentences are processed in each iteration until all 14041 sentences have been passed. One epoch is completed when all the training sentences have been passed.

One of the many methods used to prevent the overfitting of a neural network is the use of dropout between layers in the network (Srivastava et al., 2014). In the grid search, both recurrent dropout and layer-level dropout are explored. Layer dropout randomly deactivates LSTM

[9]https://www.tensorflow.org/api_docs/python/tf/keras/wrappers/scikit_learn/KerasClassifier

cells, and therefore prevents a subset of information to be passed onto the next layer of the network. Recurrent dropout is when recurrent connections between the LSTM cells are dropped. A combination of both recurrent dropout and layer dropout (known as variational dropout) has been shown to significantly reduce the test error on natural language tasks, like sentiment analysis (Gal and Ghahramani, 2016).

The remaining two discrete hyperparameters are the number of LSTM cells and the number of units in the hidden layer. The number of LSTM cells is the total number of recurrent units allocated for detecting and retaining long-term dependencies between tokens within sentences, whilst the number of hidden units allow for the construction of a deep network which is able to detect more complex non-linearity within the training data. Lastly, gradient clipping is explored to combat the vanishing and exploding gradient issues that arise when using LSTM networks, as discussed in section 2.2.2. Here, the gradient of the loss function is clipped if the gradient should go above or below a specified threshold. The standard grid search for BiLSTM-CRF hyperparameter tuning is as shown in the order of operations below:

1. For every given combination of hyperparameters:
    (a) Create 3 sequential CV folds.
    (b) Create and train a BiLSTM-CRF model on each CV fold.
    (c) Record performance as average validation accuracy across all folds.
2. Choose best performing combination of hyperparameters.

There are various steps taken to ensure the results are reproducible. The Python hash-seed environment variable is set to a constant integer (3) which ensures that certain standard operations within Python are constant and reproducible. The random seed is also set to a fixed constant for packages such as $Numpy$, $Random$ and $Tensorflow$ (Abadi et al., 2015). Fixing the seed for Tensorflow is very important as the model architecture contains many layers with weights that are initialised randomly. Fixing these initial random weights has a significant impact on the reproducibility of results when training neural networks. Lastly, the configuration for the Tensorflow session is set to force Tensorflow to use a single thread in order to reduce parallel operations (Keras, 2020).

4.3 Evaluation Framework

After the hyperparameters are tuned as outlined in the previous section, the finely tuned models are trained and tested on the unseen test dataset. It is this unseen dataset on which the models' overall performances are evaluated and compared. All the models are evaluated using the key metrics, namely precision, recall and F1 score (Rijsbergen, 1979). Tools such as classification reports and confusion matrices allow for in-depth analysis of model performance to adequately see the shortfalls and advantages of each model architecture. In addition, error summaries containing a variety of tagging error types such as no extraction, no annotation, and wrong tag errors are also utilised (B. A. Nguyen, K. V. Nguyen, and N. L.-T. Nguyen, 2019).

4.3.1 Conditional Random Fields

Conditional Random Field models are tested across all window sizes explored in the hyperparameter tuning and the key evaluating metrics are attained for each. For this final testing the maximum number of iterations is increased to 50. Initial testing indicated no significant improvement when running more than 50 iterations, and the algorithm converges within fewer iterations. The CRF model relies on the L1 and L2 regularisation attained via cross-validation to prevent overfitting.

4.3.2 BiLSTM-CRF

Unlike the CRF model, the BiLSTM-CRF model is trained on the entire training dataset with an early stopping criterion to ensure convergence, while reducing the risk of over-training.

In this section a model is created and tested for each word embedding using the finely tuned hyperparameters attained via individual grid searches. Whilst a variety of measures have been implemented to ensure the results are reproducible, there is an element of uncontrollable variation due to working on rented GPU's. The BiLSTM-CRF models are run on single GPU's, which are known to produce non-deterministic outputs. GPU's tend to run many operations in parallel and the order in which these are executed can have an impact on the performance and results. According to the Keras team, complete avoidance of non-deterministic nature may not be possible and it is for this reason that an average score is taken over 3 runs in order to compare the mean performance of the models (Keras, 2020). The mean score is attained for precision, recall, F1 score as well as the classification report.

Chapter V

Results

This chapter outlines the results attained during hyperparameter tuning and holdout testing. Firstly, the hyperparameter tuning results for conditional random fields and BiLSTM-CRF models are evaluated and summarised. Lastly, the holdout test results attained from the final models are evaluated.

5.1 Hyperparameter Grid Search

A variety of hyperparameters are tested on both models using standard 3-fold cross-validation. In each grid search, the best hyperparameters are selected based on the validation accuracy score. The random seed is fixed for the random folds in order to attain reproducible results.

5.1.1 Conditional Random Fields

For conditional random fields, only L1 and L2 regularisation are evaluated for different window sizes ranging from 1 (W_1) to 3 (W_3). For each window size, a total of 11 regularisation coefficients are explored for L1 and L2, resulting in a grid search of 121 different combinations. The results are represented in Figures 5.1 – 5.4 below, with the combination resulting in the highest average validation accuracy indicated by a red dot.

Tables 5.1 – 5.4 list the five highest average validation accuracies for each window size, together with the corresponding regularisation combinations.

Window Size: 0

L1	L2	Average Train Accuracy (%)	Average Val Accuracy (%)
0.1	0.0	45.674	44.721
0.1	0.1	45.666	44.718
0.1	0.2	45.666	44.704
0.1	0.3	45.666	44.699
0.2	0.0	45.645	44.697

Table 5.1: Summary of five highest average validation accuracies attained in conditional random field hyperparameter tuning: Window size 0.

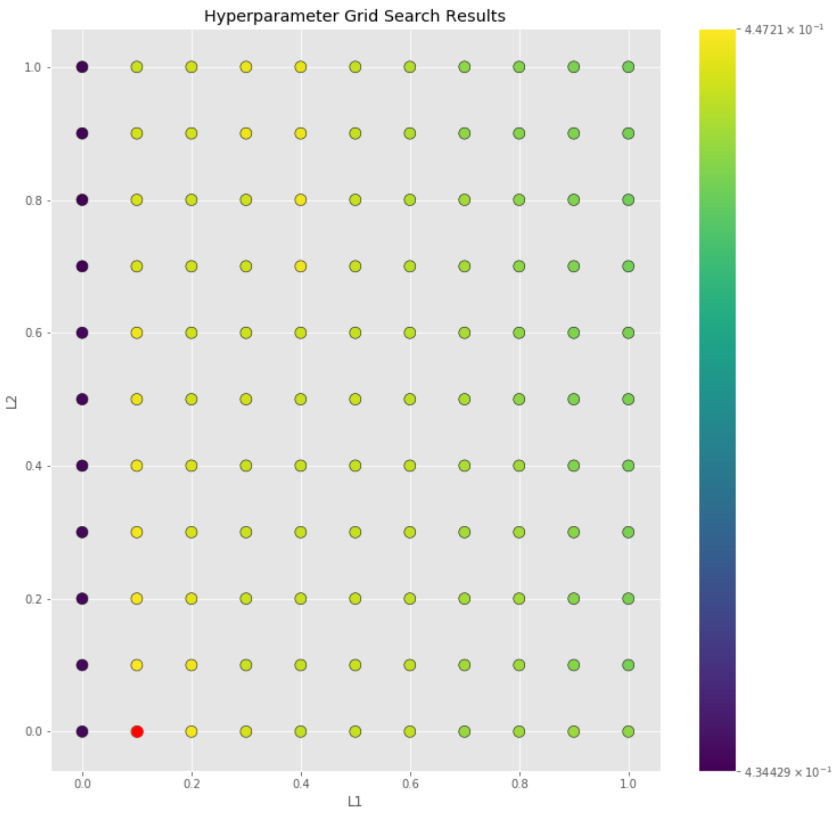


Figure 5.1: Grid Search for conditional random field: Window size 0

Window Size: 1

L1	L2	Average Train Accuracy (%)	Average Val Accuracy (%)
0.1	0.3	61.405	60.481
0.1	0.4	61.409	60.480
0.1	0.2	61.404	60.479
0.1	0.1	61.406	60.478
0.1	0.0	61.406	60.473

Table 5.2: Summary of five highest average validation accuracies attained in conditional random field hyperparameter tuning : Window size 1

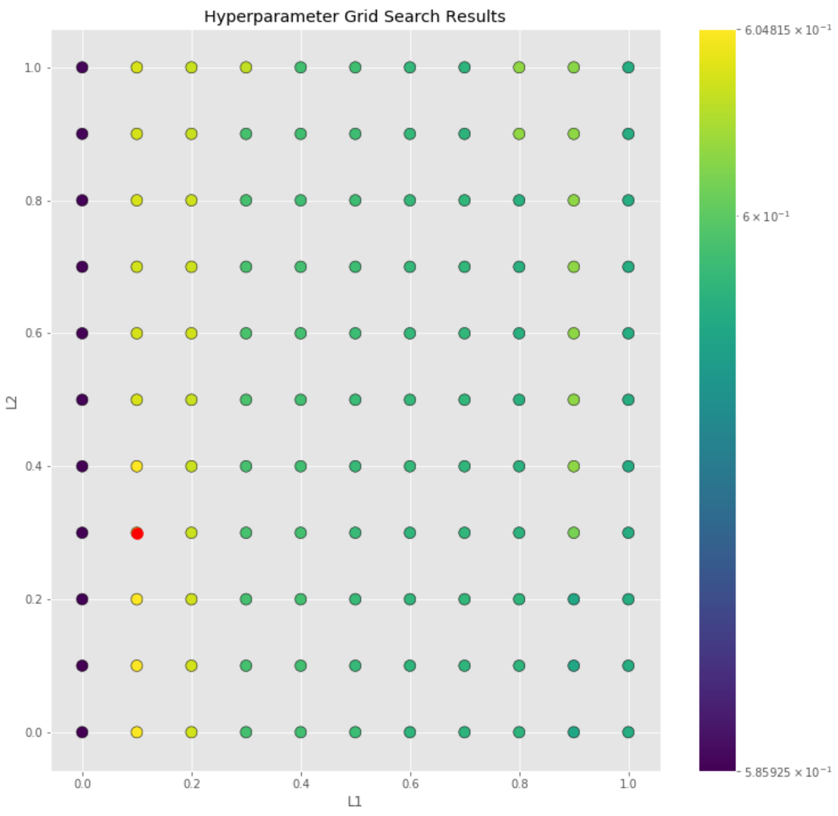


Figure 5.2: Grid search for conditional random field: Window size 1

Window Size: 2

L1	L2	Average Train Accuracy (%)	Average Val Accuracy (%)
0.4	0.7	62.802	61.279
0.4	0.5	62.804	61.278
0.4	0.6	62.802	61.275
0.1	0.9	62.893	61.272
0.4	0.4	62.802	61.270

Table 5.3: Summary of five highest average validation accuracies attained in conditional random field hyperparameter tuning: Window size 2

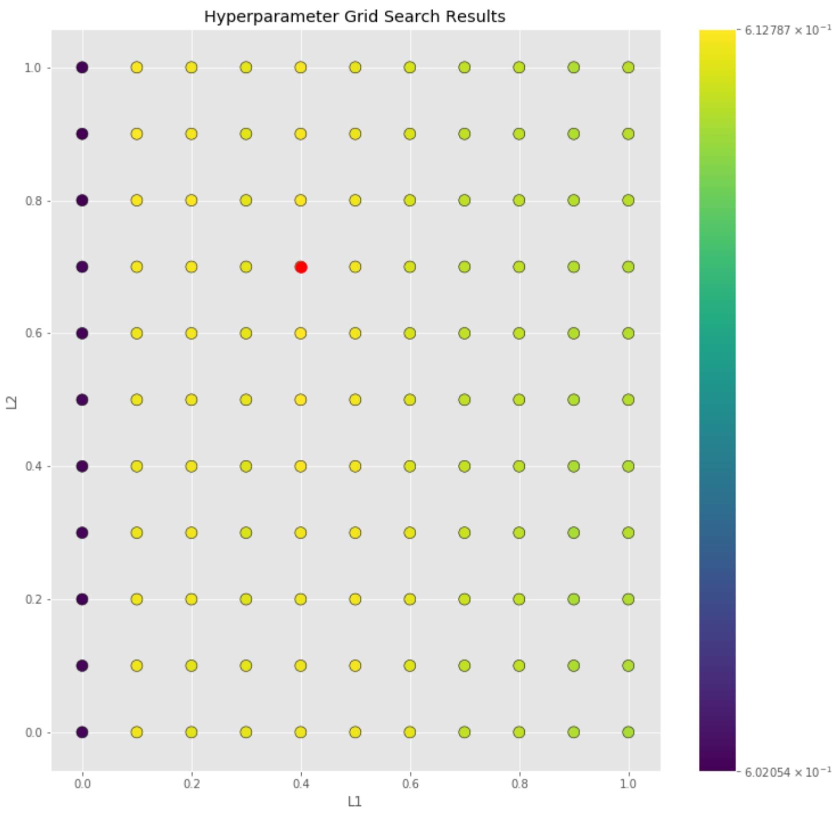


Figure 5.3: Grid Search for conditional random field: Window size 2

Window Size: 3

L1	L2	Average Train Accuracy (%)	Average Val Accuracy (%)
0.1	0.5	64.770	62.728
0.1	0.6	64.772	62.728
0.1	0.8	64.760	62.720
0.1	0.7	64.765	62.718
0.1	0.4	64.767	62.717

Table 5.4: Summary of five highest average validation accuracies attained in conditional random field hyperparameter tuning: Window size 3

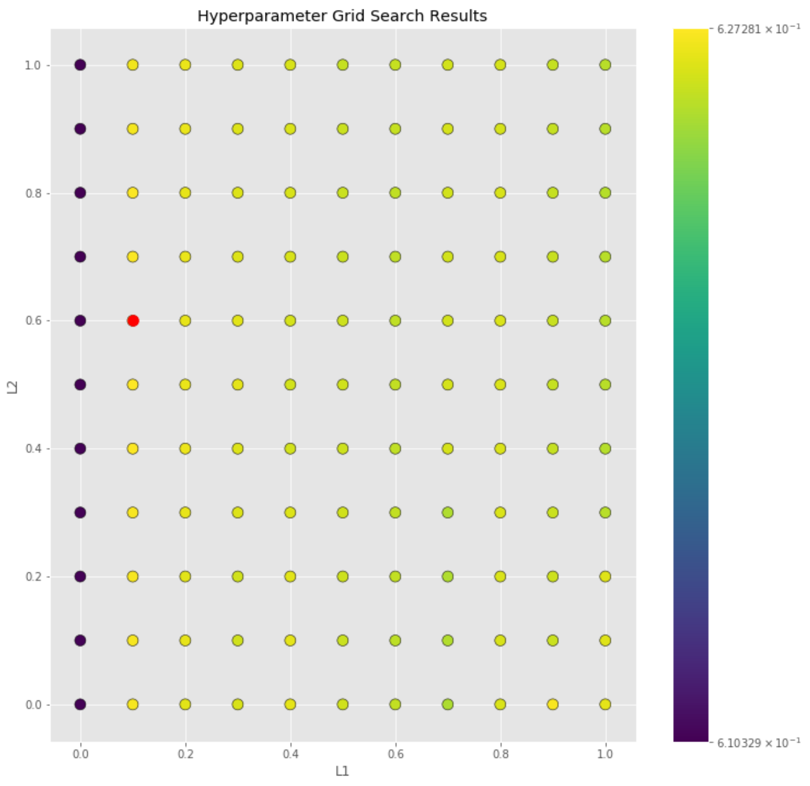


Figure 5.4: Grid Search for conditional random field: Window size 3

Results Summary

Across windows sizes 0, 1, and 3 the best parameters resulted in a relatively small implementation of L1 regularisation (0.1). For window sizes 1, 2, and 3 a strong L2 regularisation produced the best performance whilst a window size 0 resulted in no L2 regularisation. It is important to note that for 3 out of the 4 window sizes, a combination of both L1 and L2 regularisation produces the best model performance. As the window size increases, allowing for more contextual data, the run time required to perform the grid search gradually increases, with the final run taking just over of 7 hours to complete.

Window Size	Domain	Tuning Time	L1	L2
$[W]$	(0.0, 0.1, 0.2. . . .1)	69 min	0.1	0.0
$[W_{-1}, W, W_{+1}]$	(0.0, 0.1, 0.2. . . .1)	208 min	0.1	0.3
$[W_{-2}, W, W_{+2}]$	(0.0, 0.1, 0.2. . . .1)	326 min	0.4	0.7
$[W_{-3}, W, W_{+3}]$	(0.0, 0.1, 0.2. . . .1)	459 min	0.1	0.5

Table 5.5: Summary of hyperparameter tuning for conditional random fields.

5.1.2 Bidirectional-LSTM-CRF

For the BiLSTM-CRF model, a grid search is performed sequentially for each hyperparameter, starting with the batch size and ending with gradient clipping. For each grid search, the optimal value is determined by 3-fold cross-validation using the average validation accuracy. The optimal hyperparameter is then subsequently used in the next hyperparameter grid search, gradually resulting in a fine-tuned model. The partial epochs are a result of some folds converging earlier than other folds due to the early stopping criterion, thus the average is displayed.

Tables 5.6 – 5.11 list the results for each grid search when using both the GloVe and SENNA word embeddings, as discussed in section 4.2.2.

Batch Size

Embedding	Batch Size	Train loss	Train Acc	Val Loss	Val Acc	Epochs
GloVe	32	39.008	0.980	39.064	0.968	15.66
	64	39.017	0.977	39.066	0.966	18
	128	39.036	0.972	39.073	0.964	20
	256	39.061	0.965	39.088	0.960	20
SENNA	64	39.002	0.982	39.064	0.968	16
	32	39.002	0.982	39.065	0.967	12
	128	39.014	0.979	39.069	0.967	20
	256	39.037	0.972	39.077	0.964	20

Table 5.6: Summary of BiLSTM-CRF hyperparameter tuning: Batch size

Optimisation Algorithm

Embedding	Optimiser	Train loss	Train Acc	Val Loss	Val Acc	Epochs
GloVe	adam	39.010	0.979	39.060	0.969	14.66
	nadam	39.001	0.982	39.064	0.967	11.66
	sgd	39.242	0.912	39.240	0.918	20
SENNA	adam	39.008	0.980	39.068	0.966	14.33
	nadam	39.011	0.979	39.077	0.960	8
	sgd	39.267	0.916	39.288	0.908	20

Table 5.7: Summary of BiLSTM-CRF hyperparameter tuning: Optimiser

Number of LSTM Cells

Embedding	LSTM Cells	Train loss	Train Acc	Val Loss	Val Acc	Epochs
GloVe	200	38.999	0.983	39.063	0.968	13.33
	100	39.013	0.978	39.063	0.967	14.33
	300	39.005	0.981	39.065	0.967	10.66
SENNA	100	39.005	0.981	39.064	0.968	15
	300	38.993	0.987	39.068	0.967	12
	200	39.000	0.984	39.069	0.967	13

Table 5.8: Summary of BiLSTM-CRF hyperparameter tuning: LSTM cells

Dropout Rate

Embedding	Dropout	Train loss	Train Acc	Val Loss	Val Acc	Epochs
GloVe	0.10	39.002	0.982	39.066	0.969	12.66
	0.05	39.004	0.982	39.068	0.967	10.66
	0.50	39.039	0.965	39.060	0.965	20
	0.00	38.996	0.987	39.075	0.965	10
	0.25	39.017	0.975	39.064	0.965	15.33
SENNA	0.25	39.017	0.975	39.060	0.968	18.33
	0.05	39.009	0.981	39.073	0.965	12.33
	0.10	39.006	0.981	39.064	0.965	15
	0.00	38.995	0.987	39.075	0.965	12.66
	0.50	39.052	0.962	39.067	0.963	20

Table 5.9: Summary of BiLSTM-CRF hyperparameter tuning: Dropout

Number of Hidden Layer Nodes

Embedding	Hidden Nodes	Train loss	Train Acc	Val Loss	Val Acc	Epochs
GloVe	100	38.998	0.983	39.064	0.969	12
	50	39.008	0.980	39.067	0.968	10.66
	20	39.001	0.983	39.065	0.968	13.33
	10	39.007	0.981	39.065	0.967	13
SENNA	50	39.014	0.976	39.058	0.969	18
	100	39.017	0.976	39.060	0.968	16.66
	10	39.019	0.975	39.059	0.968	19.66
	20	39.016	0.975	39.061	0.967	19.33

Table 5.10: Summary of BiLSTM-CRF hyperparameter tuning: Hidden nodes

Gradient Clipping

Embedding	Clipping Value	Train loss	Train Acc	Val Loss	Val Acc	Epochs
GloVe	1	38.998	0.984	39.066	0.970	12.66
	2	39.004	0.981	39.064	0.969	10.66
	5	39.001	0.982	39.065	0.969	11.33
	0	39.002	0.982	39.075	0.968	11.33
SENNA	2	39.015	0.976	39.059	0.968	17.66
	1	39.010	0.978	39.058	0.968	20
	0	39.013	0.977	39.058	0.968	18.66
	5	39.022	0.974	39.063	0.967	15

Table 5.11: Summary of BiLSTM-CRF hyperparameter tuning: Gradient clipping

Results Summary

The initial grid search uses a set of default hyperparameters which are sequentially tuned as the grid search process is expanded. The first grid search already produces a divergence in model parameters between the different embeddings, with GloVe favouring smaller batch sizes (32 vs 64) than SENNA. In both cases a smaller batch size resulted in a faster convergence. The Adam optimiser with a fixed learning rate of 0.01 is selected as the training optimisation algorithm due to its superior performance and faster convergence across both embeddings. Both the Nadam and Adam optimisers resulted in faster convergence whilst stochastic gradient descent (sgd) required the full 20 epochs to converge.

The grid search for the LSTM cell count is the second grid search where the two embeddings diverge on hyperparameter selection. The embedding favours a greater number of cells (200) whilst the SENNA embedding favours fewer cells (100). This result may be due to the GloVe embedding containing fewer missing words (7.10%) than the SENNA embedding (9.05%), and therefore requires a larger BiLSTM layer to detect long-range dependencies between words. Both models seem to perform better with relatively small levels of dropout. GloVe and SENNA grid searches produced the best results with 10% and 25% for both recurrent and layer-level dropout respectively.

The last two grid searches see the embeddings differ once again, with GloVe embedding opting for more nodes (100 neurons) in the hidden layer and a smaller gradient clipping threshold of 1. For SENNA embedding a higher gradient clipping threshold of 2 was preferred. The final hyperparameters can be found in the Table 5.12 below.

Embedding	**Hyperparameter**	**Value**
GloVe	batch size	32
	optimiser	Adam
	LSTM Cells	200
	Dropout Rate	0.10
	Hidden Nodes	100
	Gradient Clipping	1.0
SENNA	batch size	64
	optimiser	Adam
	LSTM Cells	100
	Dropout Rate	0.25
	Hidden Nodes	50
	Gradient Clipping	2.0

Table 5.12: Final tuned BiLSTM-CRF hyperparameters.

5.2 Holdout Testing

After the hyperparameters have been tuned as outlined in the previous section, the finely tuned Conditional Random Fields and BiLSTM-CRF models are trained and tested on the CONLL-2003 test dataset. To ensure consistency in performance comparison, we will compare the same key metrics for both models.

5.2.1 Classification Metrics

The performance of each model is evaluated on three distinct metrics: precision, recall and F1 score. The results are displayed in classification reports as well as confusion matrices. The classification report displays the three metrics on an overall model level as well as on an entity level. The classification report displays what the precision, recall and F1 score will be for a single entity (e.g "I-PER"), highlighting a model's overall performance for a particular named entity. At the bottom of the report the macro, micro and weighted averages for each metric is displayed. Weighted average is calculated by weighting the individual entity scores by the number of times that entity occurs in the dataset. Macro average is computed by taking the average scores across all entities, hence weighting all entities equally. Micro on the other hand uses the global number of true positives, false positives, and false negatives to calculate the average score and therefore better suited to handle the class imbalance. The micro average is the selected metric on which the overall model performance is assessed.

Confusion matrices allow for an even deeper dive into a model's performance to see exactly which entities were incorrectly predicted and which entities the model may struggle to predict. Lastly, the error types summaries are utilised to allow for a different perspective on model performance, highlighting the common errors produced by individual models and overall error count.

5.2.2 Conditional Random Fields

The results, as detailed above, of the CRF model are presented for each window size in Tables 5.13 – 5.16. The "Support" column in these tables refers to the number of occurrences of the named entity on the test dataset.

Window Size 0

Entity	Precision	Recall	F1 score	Support
I-PER	85.67	79.99	82.73	2773
I-ORG	69.71	71.98	70.83	2491
I-LOC	78.85	84.52	81.59	1919
I-MISC	72.08	78.11	74.97	909
B-MISC	0.00	0.00	0.00	9
B-ORG	0.00	0.00	0.00	5
B-LOC	0.00	0.00	0.00	6
micro avg	77.27	78.19	77.73	8112
macro avg	43.76	44.94	44.30	8112
weighted avg	77.42	78.19	77.73	8112

Table 5.13: Conditional random fields test results: Window size 0

Window Size 1

Entity	Precision	Recall	F1 score	Support
I-PER	86.33	89.47	87.87	2773
I-ORG	79.21	78.48	78.84	2491
I-LOC	86.69	86.87	86.78	1919
I-MISC	76.86	77.12	76.99	909
B-MISC	0.00	0.00	0.00	9
B-ORG	0.00	0.00	0.00	5
B-LOC	0.00	0.00	0.00	6
micro avg	83.21	83.88	83.54	8112
macro avg	47.01	47.42	47.21	8112
weighted avg	82.95	83.88	83.40	8112

Table 5.14: Conditional random fields test results: Window size 1

Window Size 2

Entity	Precision	Recall	F1 score	Support
I-PER	86.96	89.97	88.44	2773
I-ORG	79.29	78.68	78.98	2491
I-LOC	86.51	85.57	86.04	1919
I-MISC	73.25	74.7	73.97	909
B-MISC	0.00	0.00	0.00	9
B-ORG	0.00	0.00	0.00	5
B-LOC	0.00	0.00	0.00	6
micro avg	82.98	83.53	83.25	8112
macro avg	46.57	46.99	46.78	8112
weighted avg	82.75	83.53	83.13	8112

Table 5.15: Conditional random fields test results: Window size 2

Window Size 3

Entity	Precision	Recall	F1 score	Support
I-PER	87.56	90.88	89.19	2773
I-ORG	81.29	78.68	79.96	2491
I-LOC	86.54	85.41	85.97	1919
I-MISC	73.59	76.35	74.94	909
B-MISC	0.00	0.00	0.00	9
B-ORG	0.00	0.00	0.00	5
B-LOC	0.00	0.00	0.00	6
micro avg	83.84	83.99	83.91	8112
macro avg	47.00	47.33	47.15	8112
weighted avg	83.61	83.99	83.78	8112

Table 5.16: Conditional random fields test results: Window size 3

Interpretation

In reviewing each of the classification reports, none of the four models were able to correctly classify the beginning ("B-") named entities. The reason for this may be due to the rarity of these labels in the training and test data. As one can see in the support column (far right), the beginning named entities support numbers are significantly less than the inside ("I-") named entities. The named entity with the highest F1 score across all four window sizes was "I-PER".

As can be seen in the Table 5.17 below, the model performance gradually improved with the increase in window size. The most significant improvement is seen with the introduction of the first window word (W_1), resulting in an additional 5.81% increase to F1 score. The subsequent additional windows resulted in minor improvements to the CRF model's performance. The CRF model with the largest window size (W_3) attains the highest performance across all three metrics, including an F1 score of 83.91.

Window Size	**Precision**	**Recall**	**F1 Score**	Δ F1 score
$[W]$	77.27	78.19	77.73	
$[W_{-1}, W, W_{+1}]$	83.21	83.88	83.54	+ 5.81
$[W_{-2}, W, W_{+2}]$	82.98	83.53	83.25	- 0.29
$[W_{-3}, W, W_{+3}]$	**83.84**	**83.99**	**83.91**	+ 0.66

Table 5.17: Summary of test results for conditional random fields.

5.2.3 Bidirectional-LSTM-CRF

Tables 5.18 and 5.19 summarise the results of the BiLSTM-CRF models when using GloVe and SENNA embeddings respectively.

GloVe

Entity	**Precision**	**Recall**	**F1 score**	**Support**
I-PER	94.67	87.05	90.70	2773
I-ORG	77.41	74.03	75.68	2491
I-LOC	88.27	84.69	86.44	1919
I-MISC	72.26	66.85	69.45	909
B-MISC	0.00	0.00	0.00	9
B-ORG	0.00	0.00	0.00	5
B-LOC	0.00	0.00	0.00	6
micro avg	85.25	80.01	82.55	8112
macro avg	47.52	44.66	46.04	8112
weighted avg	85.10	80.01	82.47	8112

Table 5.18: Average BiLSTM-CRF classification report using GloVe embedding.

SENNA

Entity	Precision	Recall	F1 score	Support
I-PER	87.70	94.12	90.80	2773
I-ORG	80.56	72.54	76.34	2491
I-LOC	88.03	83.91	85.92	1919
I-MISC	68.67	68.98	68.82	909
B-MISC	0.00	0.00	0.00	9
B-ORG	0.00	0.00	0.00	5
B-LOC	0.00	0.00	0.00	6
micro avg	83.58	82.02	82.79	8112
macro avg	46.42	45.65	45.98	8112
weighted avg	83.23	82.02	82.51	8112

Table 5.19: Average BiLSTM-CRF classification report using SENNA embedding.

Interpretation

Overall the SENNA embedding produced a marginally better performance compared to GloVe embedding with a +0.24 higher F1 score. This result is to be expected since SENNA was partly developed using the Reuters RCV1 corpus (Lewis et al., 2004) which makes up the CoNLL-2003 dataset. The GloVe embedding model attained a higher precision, scoring an average of 85.25% and a best run precision of 86.08% in the second run (see Table3.6 in Appendix C).

A higher precision often comes at a trade-off with recall. Indeed, the GloVe model falls short on the recall score with the majority of the run scores in achieving close to 70%. The SENNA model attained much higher recall scores with an average score of 82.02% and a best run of 82.80% in the third run. The classification reports echo what is seen in Table 5.20, with the GloVe model scoring higher on precision and lower on recall.

Both models fail to detect the beginning entities ("B-") as well as the I-MISC entity. The confusion matrices in Appendix C make it possible to calculate the number of named entities that were predicted as outside entities ("O") and thus failed to be recognised as named entities at all. This number is added to Table 5.20 and located under the "Missed NE" column. It is with this single metric that the models' performance significantly diverges. The GloVe embedding model failed to recognise 12.59% of the named entities in the test set whilst the SENNA embedding model missed as few as 9.64%.

Embedding	Run	Precision	Recall	F1 Score	Missed NE (%)
GloVe	1	85.03	80.94	82.93	11.55
	2	86.08	79.72	82.78	13.51
	3	84.68	79.39	81.95	12.72
	Avg	**85.25**	**80.01**	**82.55**	**12.59**
SENNA	1	83.83	81.18	82.48	10.55
	2	84.33	82.13	83.22	9.92
	3	82.68	82.80	82.74	8.46
	Avg	**83.58**	**82.02**	**82.79**	**9.64**

Table 5.20: Summary of BiLSTM-CRF test results.

5.2.4 Model Evaluation

Overall, both the CRF and BiLSTM-CRF models achieved relatively high scores across all three metrics on the test dataset. The CRF model with a window size of 0 (CRF-0) produced the lowest F1 score of 77.73% as can be seen in Table 5.21. As the window size increases, the CRF model's performance improves substantially as the model is fed more contextual features.

The model that achieved the highest F1 score is the CRF model with the largest window size of 3 (CRF-3), achieving an F1 score of 83.91%. With the exception of the CRF-0 model, all the CRF models outperformed the BiLSTM-CRF models in terms of the F1 score. Arguably one of the most important metrics is recall, as it is the total number of true entities tagged in the dataset. Once again the CRF-3 model outperformed the rest achieving a 83.99% recall score and tagging a total of 6813 (out of 8112) entities correctly. Whilst CRF models dominated the highest recall and F1 scores, the BiLSTM-CRF (GloVe) model achieved the highest precision with a score of 85.25%.

Model	Precision	Recall	F1 Score
CRF $[W_0]$	77.27	78.19	77.73
CRF $[W_{-+1}]$	83.21	83.88	83.54
CRF $[W_{-+2}]$	82.98	83.53	83.25
CRF $[W_{-+3}]$	83.84	**83.99**	**83.91**
BiLSTM-CRF (GloVe)	**85.25**	80.01	82.55
BiLSTM-CRF (SENNA)	83.58	82.02	82.79

Table 5.21: Summary of test results for all models.

Table 5.22 allows for a different perspective on individual model performance. As shown in this table, the BiLSTM-CRF models extracted fewer named entities on average when compared to CRF models. Approximately 47.65% (1022 errors) of the errors made by the BiLSTM-CRF (GloVe) model can be attributed to this error type. This means that the model is failing to identify entities and instead labels these entities as outside tokens ("O"). At the same time this same model achieved the smallest number of incorrect tags (584), further demonstrating the trade-off between recall and precision. This is reiterated in Table 5.21 which shows that the BiLSTM-CRF model achieves a higher precision at the cost of lower recall.

Model	No extraction	No annotation	Wrong Tag	Total Errors
CRF $[W]$	494	591	**1254**	**2360**
CRF $[W_{-+1}]$	375	440	916	1748
CRF $[W_{-+2}]$	367	421	952	1757
CRF $[W_{-+3}]$	380	394	902	1693
BiLSTM-CRF (GloVe)	**1022**	523	584	2145
BiLSTM-CRF (SENNA)	781	**630**	659	2089

Table 5.22: Summary of error types for all models.

On the contrary, the CRF-2 model extracted the most named entities from the dataset but achieved a very high wrong tag rate of 952 (54.18%). The model was able to recognise that these tokens are entities, but failed to tag the tokens to the correct named entity class. The model which produced the most errors is the CRF-0 model with a total of 2360 errors. However, expanding the window size of the CRF model greatly reduces the number of errors, resulting in the CRF-3 model attaining the lowest error count of 1693 errors.

From these results it can determined that the BiLSTM-CRF model's ability to recognise entities is of a lower standard when compared to CRF models. However, the precision at which the the BiLSTM-CRF model categorises the entities is superior. The total error count offers much insight into the performance of these models and is a strong metric on which to judge overall performance. It can be concluded that whilst BiLSTM-CRF models do result in a marginally higher error count, it is still less than the baseline conditional random field model (CRF-0). Once the window size is increased, the CRF model outperforms the BiLSTM model scoring on average 18.15% less errors.

Chapter VI

Conclusions and Recommendations

6.1 Summary and Conclusions

Upon review of the results it can be concluded that a BiLSTM-CRF model architecture is certainly a powerful alternative to conditional random field models. A BiLSTM model with Glove or SENNA pre-trained word embeddings can outperform a standard conditional random field model with a restricted window size of 0 (CRF-0). When the window size increases, allowing for more contextual data, the conditional random field models only marginally beat the BiLSTM-CRF on F1 score by an average of 0.90%.

BiLSTM-CRF models are more hesitant to extract entities, however are more precise at tagging the entities correctly. This means that BiLSTM-CRF models operate at a higher precision but a lower recall when compared to conditional random field models with large window sizes. The SENNA embeddings marginally outperformed the Glove embeddings with a +0.24% difference in F1 score. Overall BiLTSM-CRF models produces fewer errors than the baseline conditional random field model (CRF-0), but was unable to outperform models with larger window sizes.

6.2 Limitations and Improvements

Named entity recognition is a difficult natural language sub-task to model. There are many different variables and processes to consider when developing a deep learning neural network architecture, and this is an area of research that is in constant development.

The data itself can also be pre-processed in a variety of techniques, especially with the handling of digits. In this dissertation all numbers were reduced to a single digit in an effort to reduce the vocabulary and the dimensions of the embedding layer. This preprocessing approach risks losing information within the text, potentially making it harder to detect some hidden dependencies between tokens. Other data preprocessing techniques should be investigated in order to find the optimal method for preprocessing data for named entity recognition.

6.3 Further Research

The research into our natural language continues to be as dynamic and ever-evolving as the languages themselves, allowing for a wide range of approaches to better our current understanding. The following three ideas are presented as starting points when building on the research conducted in this study.

- Investigate how the use of other pre-trained word embeddings such as word2Vec (Google) and fastText (Facebook) have an effect on BiLSTM-CRF model performance. Contextual word embeddings such as ELMo can also be explored.
- Investigate how the use of pre-trained word embeddings with larger dimensions (100D, 300D etc.) might affect the performance of BiLSTM-CRF models.
- The CoNLL-2003 dataset is available in a variety of tagging schemes (BIO, IOB, IOBES etc.), which have an impact on the distribution of tags. Future research should include investigating all the possible tagging schemes to see what the effect on overall performance is.

Bibliography

Abadi, Martin et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: `http://tensorflow.org/`.

Abuleil, Saleem (2004). "Extracting Names From Arabic Text for Question-Answering Systems." In: pp. 638–647.

Akbik, Alan, Duncan Blythe, and Roland Vollgraf (2018). "Contextual String Embeddings for Sequence Labeling". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, pp. 1638–1649. URL: `https://www.aclweb.org/anthology/C18-1139`.

Andrew, Galen and Jianfeng Gao (2007). "Scalable Training of <i>L</i><sup>1</sup>-Regularized Log-Linear Models". In: *Proceedings of the 24th International Conference on Machine Learning*. ICML '07. Corvalis, Oregon, USA: Association for Computing Machinery, pp. 33–40. ISBN: 9781595937933. DOI: `10.1145/1273496.1273501`. URL: `https://doi.org/10.1145/1273496.1273501`.

Antonio Moreno Doaa Samy, Jose M Guirao (2005). "A Proposal For An Arabic Named Entity Tagger Leveraging a Parallel Corpus." In:

Baheti, Pragati (2021). *12 Types of Neural Network Activation Functions: How to Choose?* URL: `https://www.v7labs.com/blog/neural-networks-activation-functions` (visited on 01/05/2021).

Bender, Oliver, Franz Josef Och, and Hermann Ney (2003). "Maximum Entropy Models for Named Entity Recognition". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Edmonton, Canada: Association for

Computational Linguistics, pp. 148–151. DOI: 10.3115/1119176.1119196. URL: https://doi.org/10.3115/1119176.1119196.

Bengio, Yoshua, Patrice Y. Simard, and Paolo Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5 2, pp. 157–66.

Berger, Adam L., Vincent J. Della Pietra, and Stephen A. Della Pietra (1996). "A Maximum Entropy Approach to Natural Language Processing". In: *Comput. Linguist.* 22.1, pp. 39–71. ISSN: 0891-2017. URL: http://dl.acm.org/citation.cfm?id=234285.234289.

Bottou, Leon (1991). "Stochastic gradient learning in neural networks". In: *In Proceedings of Neuro-Nîmes. EC2*.

Carreras, Xavier, Lluis Marquez, and Lluis Padro (2003). "A Simple Named Entity Extractor Using AdaBoost". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Edmonton, Canada: Association for Computational Linguistics, pp. 152–155. DOI: 10.3115/1119176.1119197. URL: https://doi.org/10.3115/1119176.1119197.

Chieu, Hai Leong and Hwee Tou Ng (2003). "Named Entity Recognition with a Maximum Entropy Approach". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Edmonton, Canada: Association for Computational Linguistics, pp. 160–163. DOI: 10.3115/1119176.1119199. URL: https://doi.org/10.3115/1119176.1119199.

Chinchor, Nancy and Beth Sundheim (1993). "MUC-5 Evaluation Metrics". In: *Proceedings of the 5th Conference on Message Understanding*. MUC5 '93. Baltimore, Maryland: Association for Computational Linguistics, pp. 69–78. ISBN: 1-55860-336-0. DOI: 10.3115/1072017.1072026. URL: https://doi.org/10.3115/1072017.1072026.

Chiticariu, Laura, Yunyao Li, and Frederick R. Reiss (2013). "Rule-Based Information Extraction is Dead! Long Live Rule-Based Information Extraction Systems!" In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, pp. 827–832. URL: https://www.aclweb.org/anthology/D13-1079.

Chiu, Jason P. C. and Eric Nichols (2015). “Named Entity Recognition with Bidirectional LSTM-CNNs”. In: *CoRR* abs/1511.08308. arXiv: 1511.08308. URL: http://arxiv.org/abs/1511.08308.

Cho, Kyunghyun et al. (2014). *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. arXiv: 1406.1078 [cs.CL].

Chollet, Francois et al. (2015). *Keras*. https://keras.io.

Collobert, Ronan and Jason Weston (2008). “A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning”. In: *Proceedings of the 25th International Conference on Machine Learning*. ICML ’08. Helsinki, Finland: ACM, pp. 160–167. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390177. URL: http://doi.acm.org/10.1145/1390156.1390177.

Collobert, Ronan, Jason Weston, et al. (2011). “Natural Language Processing (Almost) from Scratch”. In: *J. Mach. Learn. Res.* 12, pp. 2493–2537. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1953048.2078186.

Devlin, Jacob et al. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv: 1810.04805 [cs.CL].

Dozat, Timothy (2015). “Incorporating Nesterov Momentum into”. In:

Eddy, Sean R (1996). “Hidden Markov models”. In: *Current Opinion in Structural Biology* 6.3, pp. 361–365. ISSN: 0959-440X. DOI: https://doi.org/10.1016/S0959-440X(96)80056-X. URL: http://www.sciencedirect.com/science/article/pii/S0959440X9680056X.

Etzioni, Oren et al. (2005). “Unsupervised named-entity extraction from the Web: An experimental study”. In: *Artificial Intelligence* 165.1, pp. 91–134. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2005.03.001. URL: http://www.sciencedirect.com/science/article/pii/S0004370205000366.

Florian, Radu et al. (2003). “Named Entity Recognition Through Classifier Combination”. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*

- *Volume 4*. CONLL '03. Edmonton, Canada: Association for Computational Linguistics, pp. 168–171. DOI: 10.3115/1119176.1119201. URL: https://doi.org/10.3115/1119176.1119201.

Freund, Yoav and Robert E Schapire (1997). "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: *J. Comput. Syst. Sci.* 55.1, pp. 119–139. ISSN: 0022-0000. DOI: 10.1006/jcss.1997.1504. URL: http://dx.doi.org/10.1006/jcss.1997.1504.

F.R.S., Karl Pearson (1901). "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11, pp. 559–572. DOI: 10.1080/14786440109462720.

Gal, Yarin and Zoubin Ghahramani (2016). "A Theoretically Grounded Application of Dropout in Recurrent Neural Networks". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain: Curran Associates Inc., pp. 1027–1035. ISBN: 9781510838819.

Garofolo, John S. et al. (1993). "DARPA TIMIT:: acoustic-phonetic continuous speech corpus CD-ROM, NIST speech disc 1-1.1". In:

Ghaddar, Abbas and Philippe Langlais (2018). "Robust Lexical Features for Improved Neural Network Named-Entity Recognition". In: *CoRR* abs/1806.03489. arXiv: 1806.03489. URL: http://arxiv.org/abs/1806.03489.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio (2011). "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, pp. 315–323. URL: http://proceedings.mlr.press/v15/glorot11a.html.

Goyal, Archana, Manish Kumar, and Vishal Gupta (2017). "Named Entity Recognition: Applications, Approaches and Challenges". In:

Graves, Alex and Jurgen Schmidhuber (2005). "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural networks : the official* *journal of the International Neural Network Society* 18, pp. 602–10. DOI: 10.1016/j.neunet.2005.06.042.

Grishman, Ralph and Beth Sundheim (1996). “Message Understanding Conference-6: A Brief History”. In: *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*. COLING ’96. Copenhagen, Denmark: Association for Computational Linguistics, pp. 466–471. DOI: 10.3115/992628.992709. URL: https://doi.org/10.3115/992628.992709.

Hearst, M. A. et al. (1998). “Support vector machines”. In: *IEEE Intelligent Systems and their Applications* 13.4, pp. 18–28. DOI: 10.1109/5254.708428.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long Short-term Memory”. In: *Neural computation* 9, pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

Huang, Zhiheng, Wei Xu, and Kai Yu (2015). “Bidirectional LSTM-CRF Models for Sequence Tagging”. In: *CoRR* abs/1508.01991. arXiv: 1508.01991. URL: http://arxiv.org/abs/1508.01991.

Ivakhnenko, Aleksei Grigor’evich and Valentin Grigorevich Lapa (1965). *Cybernetic Predicting Devices*. CCM Information Corporation.

Jain, L. C. and L. R. Medsker (1999). *Recurrent Neural Networks: Design and Applications*. 1st. USA: CRC Press, Inc. ISBN: 0849371813.

Joulin, Armand et al. (2016). “Bag of Tricks for Efficient Text Classification”. In: *CoRR* abs/1607.01759. arXiv: 1607.01759. URL: http://arxiv.org/abs/1607.01759.

Jurafsky, Daniel and James H. Martin (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 1st. USA: Prentice Hall PTR. Chap. 8. ISBN: 0130950696.

– (2009). *Speech and Language Processing (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc. Chap. 6. ISBN: 0131873210.

Keras (2020). *Keras documentation: Keras FAQ*. URL: https://keras.io/getting_started/faq/#how-can-i-obtain-reproducible-results-using-keras-during-development (visited on 07/02/2020).

Kingma, Diederik P. and Jimmy Ba (2014). *Adam: A Method for Stochastic Optimization*. arXiv: 1412.6980 [cs.LG].

Lafferty, John D., Andrew McCallum, and Fernando C. N. Pereira (2001). "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 282–289. ISBN: 1-55860-778-1. URL: http://dl.acm.org/citation.cfm?id=645530.655813.

Lample, Guillaume et al. (2016). "Neural Architectures for Named Entity Recognition". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, pp. 260–270. DOI: 10.18653/v1/N16-1030. URL: https://www.aclweb.org/anthology/N16-1030.

Lewis, David D. et al. (2004). "RCV1: A New Benchmark Collection for Text Categorization Research". In: *J. Mach. Learn. Res.* 5, pp. 361–397. ISSN: 1532-4435.

Li, Jing et al. (2018). "A Survey on Deep Learning for Named Entity Recognition". In: *CoRR* abs/1812.09449. arXiv: 1812.09449. URL: http://arxiv.org/abs/1812.09449.

Liu, Dong C. and Jorge Nocedal (1989). "On the Limited Memory BFGS Method for Large Scale Optimization". In: *Math. Program.* 45.1-3, pp. 503–528. ISSN: 0025-5610. DOI: 10.1007/BF01589116. URL: https://doi.org/10.1007/BF01589116.

Malouf, Robert (2002). "A Comparison of Algorithms for Maximum Entropy Parameter Estimation". In: *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*. COLING-02. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 1–7. DOI: 10.3115/1118853.1118871. URL: https://doi.org/10.3115/1118853.1118871.

Mayfield, James, Paul McNamee, and Christine Piatko (2003). "Named Entity Recognition Using Hundreds of Thousands of Features". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Edmonton, Canada:

Association for Computational Linguistics, pp. 184–187. DOI: 10.3115/1119176.1119205. URL: https://doi.org/10.3115/1119176.1119205.

McCallum, Andrew, Dayne Freitag, and Fernando C. N. Pereira (2000). “Maximum Entropy Markov Models for Information Extraction and Segmentation”. In: *Proceedings of the Seventeenth International Conference on Machine Learning*. ICML ’00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 591–598. ISBN: 1558607072.

McCallum, Andrew and Wei Li (2003). “Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-enhanced Lexicons”. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL ’03. Edmonton, Canada: Association for Computational Linguistics, pp. 188–191. DOI: 10.3115/1119176.1119206. URL: https://doi.org/10.3115/1119176.1119206.

Mikolov, Tomas (2012). “Statistical language models based on neural networks”. PhD thesis. Brno University of Technology.

Mikolov, Tomas et al. (2013). *Efficient Estimation of Word Representations in Vector Space*. URL: http://arxiv.org/abs/1301.3781.

Nadeau, David and Satoshi Sekine (2007). “A Survey of Named Entity Recognition and Classification”. In: *Lingvisticae Investigationes* 30. DOI: 10.1075/li.30.1.03nad.

Nair, Vinod and Geoffrey E. Hinton (2010). “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML’10. Haifa, Israel: Omnipress, pp. 807–814. ISBN: 9781605589077.

Nguyen, Binh An, Kiet Van Nguyen, and Ngan Luu-Thuy Nguyen (2019). *Error Analysis for Vietnamese Named Entity Recognition on Deep Neural Network Models*. arXiv: 1911.07228 [cs.CL].

Panchendrarajan, Rrubaa and Aravindh Amaresan (2018). “Bidirectional LSTM-CRF for Named Entity Recognition”. In: *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*. Hong Kong: Association for Computational Linguistics. URL: https://www.aclweb.org/anthology/Y18-1061.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2012). “Understanding the exploding gradient problem”. In: *CoRR* abs/1211.5063. arXiv: 1211.5063. URL: http://arxiv.org/abs/1211.5063.

Pedregosa, F. et al. (2011). “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pennington, Jeffrey, Richard Socher, and Christopher Manning (2014). “Glove: Global Vectors for Word Representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://www.aclweb.org/anthology/D14-1162.

Quinlan, J. R. (1986). “Induction of Decision Trees”. In: *Mach. Learn.* 1.1, pp. 81–106. ISSN: 0885-6125. DOI: 10.1023/A:1022643204877. URL: https://doi.org/10.1023/A:1022643204877.

Ramshaw, Lance and Mitch Marcus (1995). “Text Chunking using Transformation-Based Learning”. In: *Third Workshop on Very Large Corpora*. URL: https://www.aclweb.org/anthology/W95-0107.

Rijsbergen, C. J. Van (1979). *Information Retrieval*. 2nd. Newton, MA, USA: Butterworth-Heinemann. ISBN: 0408709294. URL: http://www.dcs.gla.ac.uk/Keith/Preface.html.

Sasaki, Yutaka (2007). “The truth of the F-measure”. In: *Teaching, Tutorial materials*, pp. 1–5.

Sha, Fei and Fernando Pereira (2003). “Shallow Parsing with Conditional Random Fields”. In: *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 213–220. URL: https://www.aclweb.org/anthology/N03-1028.

Srivastava, Nitish et al. (2014). “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *J. Mach. Learn. Res.* 15.1, pp. 1929–1958. ISSN: 1532-4435.

Sutton, Charles and Andrew McCallum (2012). “An Introduction to Conditional Random Fields”. In: *Found. Trends Mach. Learn.* 4.4, pp. 267–373. ISSN: 1935-8237. DOI: 10.1561/2200000013. URL: http://dx.doi.org/10.1561/2200000013.

Tjong Kim Sang, Erik F. (2002). "Introduction to the CoNLL-2002 Shared Task: Language-independent Named Entity Recognition". In: *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*. COLING-02. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 1–4. DOI: 10.3115/1118853.1118877. URL: https://doi.org/10.3115/1118853.1118877.

Tjong Kim Sang, Erik F. and Fien De Meulder (2003). "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*, pp. 142–147. URL: https://www.aclweb.org/anthology/W03-0419.

Vail, Douglas L., Manuela M. Veloso, and John D. Lafferty (2007). "Conditional Random Fields for Activity Recognition". In: *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*. AAMAS '07. Honolulu, Hawaii: ACM, 235:1–235:8. ISBN: 978-81-904262-7-5. DOI: 10.1145/1329125.1329409. URL: http://doi.acm.org/10.1145/1329125.1329409.

Viterbi, A. (1967). "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". In: *IEEE Transactions on Information Theory* 13.2, pp. 260–269.

Zaghouani, Wajdi (2012). "A Rule-Based Arabic Named Entity Recognition System". In: *ACM Transactions on Asian Language Information Processing - Volume 11*.

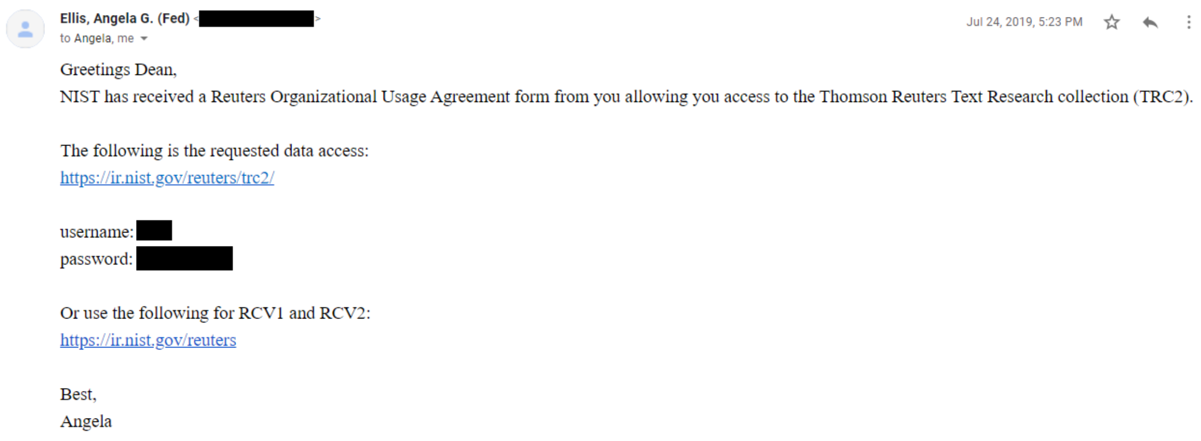
Zhou, GuoDong and Jian Su (2002). "Named Entity Recognition Using an HMM-based Chunk Tagger". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, pp. 473–480. DOI: 10.3115/1073083.1073163. URL: https://doi.org/10.3115/1073083.1073163.

Zou, Hui and Trevor Hastie (2005). "Regularization and variable selection via the Elastic Net". In: *Journal of the Royal Statistical Society, Series B* 67, pp. 301–320.

Appendices

Appendices A

Data Licence Agreement



Ellis, Angela G. (Fed) <[redacted]>
to Angela, me

Jul 24, 2019, 5:23 PM

Greetings Dean,
NIST has received a Reuters Organizational Usage Agreement form from you allowing you access to the Thomson Reuters Text Research collection (TRC2).

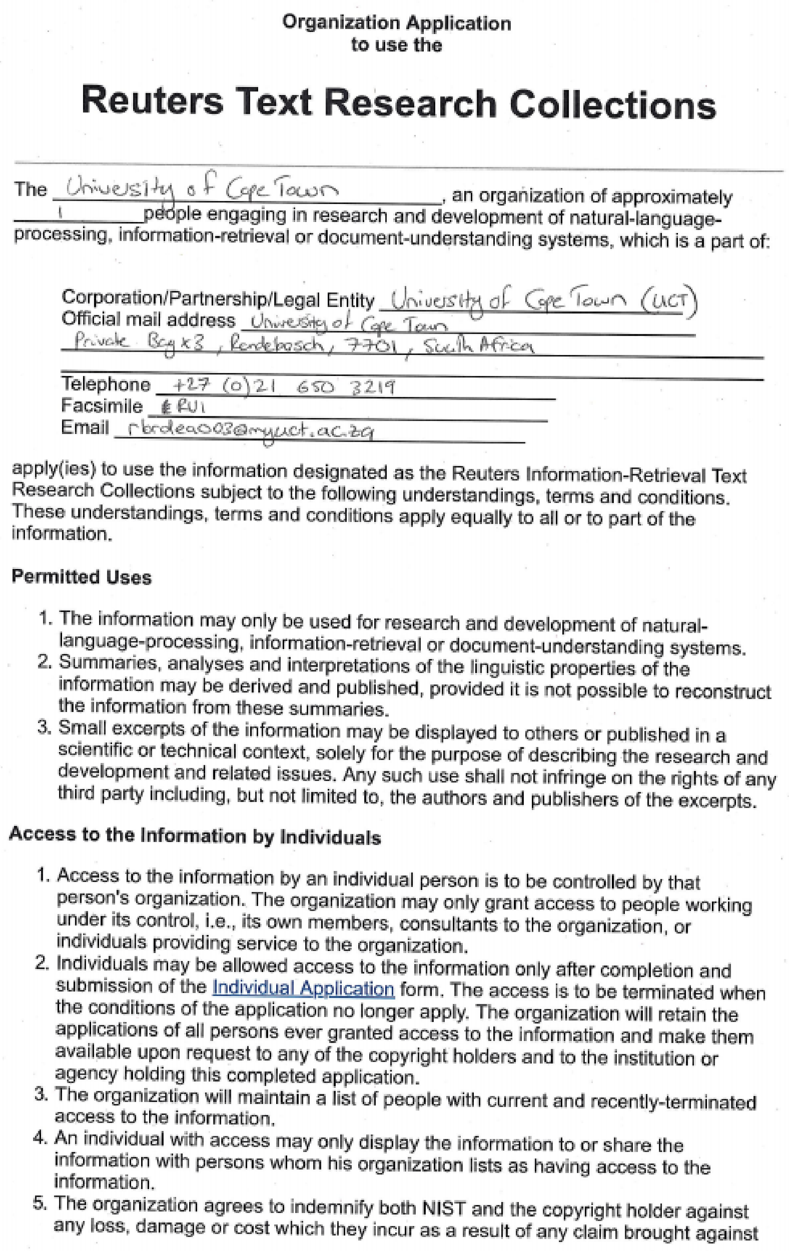
The following is the requested data access:
https://ir.nist.gov/reuters/trc2/

username: [redacted]
password: [redacted]

Or use the following for RCV1 and RCV2:
https://ir.nist.gov/reuters

Best,
Angela

Figure 1.1: Confirmation email from Reuters.



Organization Application
to use the

Reuters Text Research Collections

The University of Cape Town, an organization of approximately 1 people engaging in research and development of natural-language-processing, information-retrieval or document-understanding systems, which is a part of:

Corporation/Partnership/Legal Entity University of Cape Town (UCT)
Official mail address University of Cape Town
Private Bag x3, Rondebosch, 7701, South Africa

Telephone +27 (0)21 650 3219
Facsimile NUL
Email rbrdea003@myuct.ac.za

apply(ies) to use the information designated as the Reuters Information-Retrieval Text Research Collections subject to the following understandings, terms and conditions. These understandings, terms and conditions apply equally to all or to part of the information.

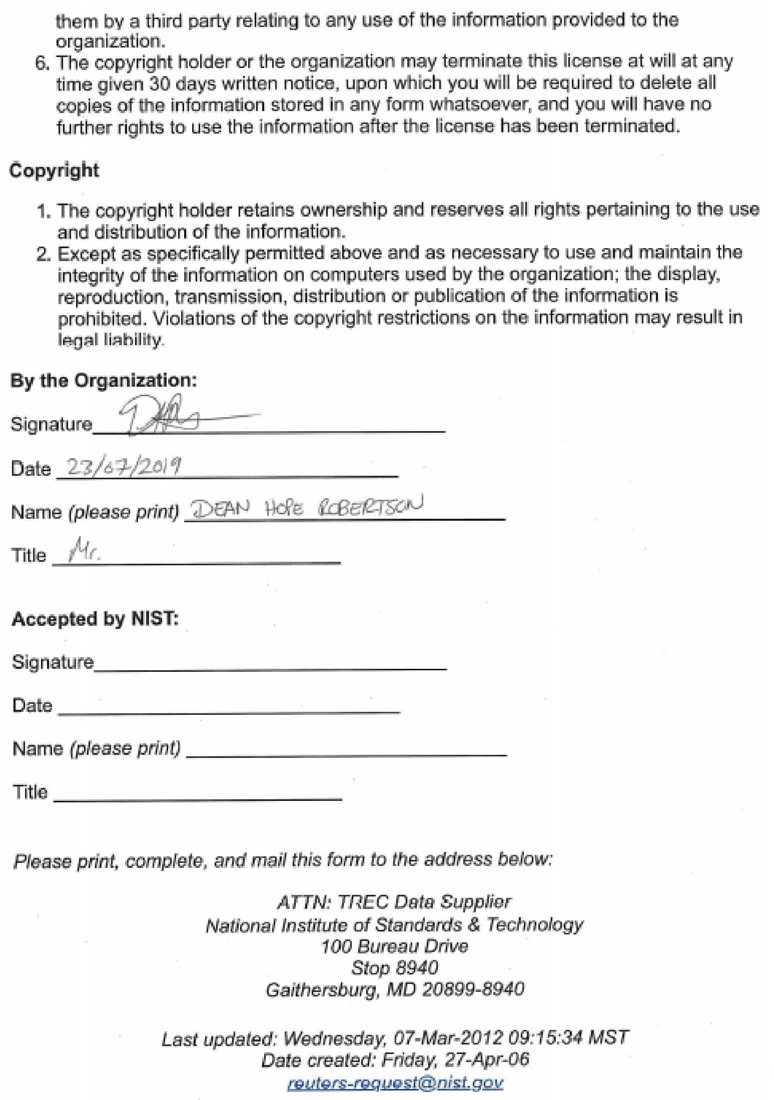
Permitted Uses

1. The information may only be used for research and development of natural-language-processing, information-retrieval or document-understanding systems.
2. Summaries, analyses and interpretations of the linguistic properties of the information may be derived and published, provided it is not possible to reconstruct the information from these summaries.
3. Small excerpts of the information may be displayed to others or published in a scientific or technical context, solely for the purpose of describing the research and development and related issues. Any such use shall not infringe on the rights of any third party including, but not limited to, the authors and publishers of the excerpts.

Access to the Information by Individuals

1. Access to the information by an individual person is to be controlled by that person's organization. The organization may only grant access to people working under its control, i.e., its own members, consultants to the organization, or individuals providing service to the organization.
2. Individuals may be allowed access to the information only after completion and submission of the Individual Application form. The access is to be terminated when the conditions of the application no longer apply. The organization will retain the applications of all persons ever granted access to the information and make them available upon request to any of the copyright holders and to the institution or agency holding this completed application.
3. The organization will maintain a list of people with current and recently-terminated access to the information.
4. An individual with access may only display the information to or share the information with persons whom his organization lists as having access to the information.
5. The organization agrees to indemnify both NIST and the copyright holder against any loss, damage or cost which they incur as a result of any claim brought against

Figure 1.2: Page 1 of Reuters Organizational Usage Agreement.



them by a third party relating to any use of the information provided to the organization.

6. The copyright holder or the organization may terminate this license at will at any time given 30 days written notice, upon which you will be required to delete all copies of the information stored in any form whatsoever, and you will have no further rights to use the information after the license has been terminated.

Copyright

1. The copyright holder retains ownership and reserves all rights pertaining to the use and distribution of the information.
2. Except as specifically permitted above and as necessary to use and maintain the integrity of the information on computers used by the organization; the display, reproduction, transmission, distribution or publication of the information is prohibited. Violations of the copyright restrictions on the information may result in legal liability.

By the Organization:

Signature [illegible]

Date 23/07/2019

Name *(please print)* DEAN HOPE ROBERTSON

Title Mr.

Accepted by NIST:

Signature ______

Date ______

Name *(please print)* ______

Title ______

Please print, complete, and mail this form to the address below:

ATTN: TREC Data Supplier
National Institute of Standards & Technology
100 Bureau Drive
Stop 8940
Gaithersburg, MD 20899-8940

Last updated: Wednesday, 07-Mar-2012 09:15:34 MST
Date created: Friday, 27-Apr-06
reuters-request@nist.gov

Figure 1.3: Page 2 of Reuters Organizational Usage Agreement.

Appendices B

Data Exploration

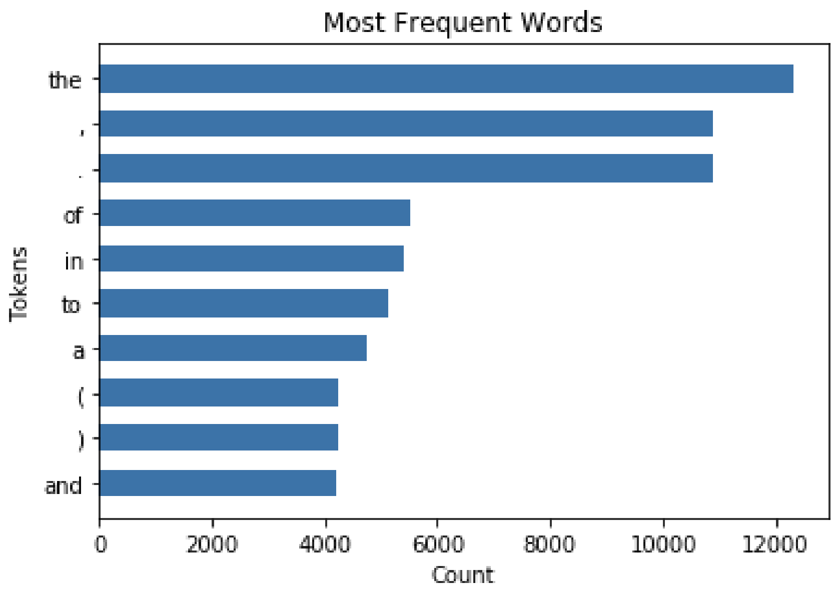


Figure 2.1: The top 10 most frequent words in CoNLL-2003 English dataset.

Appendices C

Results

3.1 Holdout Testing

3.1.1 Classification Reports

Conditional Random Field: Window Size 0

Entity	Precision	Recall	F1 score	Support
I-PER	85.67	79.99	82.73	2773
I-ORG	69.71	71.98	70.83	2491
I-LOC	78.85	84.52	81.59	1919
I-MISC	72.08	78.11	74.97	909
B-MISC	0.00	0.00	0.00	9
B-ORG	0.00	0.00	0.00	5
B-LOC	0.00	0.00	0.00	6
Micro avg	77.27	78.19	77.73	8112
Macro avg	43.76	44.94	44.30	8112
Weighted avg	77.42	78.19	77.73	8112

Table 3.1: Classification Report: Conditional Random Field (Window 0).

Conditional Random Field: Window Size 1

Entity	Precision	Recall	F1 score	Support
I-PER	86.33	89.47	87.87	2773
I-ORG	79.21	78.48	78.84	2491
I-LOC	86.69	86.87	86.78	1919
I-MISC	76.86	77.12	76.99	909
B-MISC	0.00	0.00	0.00	9
B-ORG	0.00	0.00	0.00	5
B-LOC	0.00	0.00	0.00	6
Micro avg	83.21	83.88	83.54	8112
Macro avg	47.01	47.42	47.21	8112
Weighted avg	82.95	83.88	83.40	8112

Table 3.2: Classification Report: Conditional Random Field (Window 1).

Conditional Random Field: Window Size 2

Entity	Precision	Recall	F1 score	Support
I-PER	86.96	89.97	88.44	2773
I-ORG	79.29	78.68	78.98	2491
I-LOC	86.51	85.57	86.04	1919
I-MISC	73.25	74.7	73.97	909
B-MISC	0.00	0.00	0.00	9
B-ORG	0.00	0.00	0.00	5
B-LOC	0.00	0.00	0.00	6
Micro avg	82.98	83.53	83.25	8112
Macro avg	46.57	46.99	46.78	8112
Weighted avg	82.75	83.53	83.13	8112

Table 3.3: Classification Report: Conditional Random Field (Window 2).

Conditional Random Field: Window Size 3

Entity	Precision	Recall	F1 score	Support
I-PER	87.56	90.88	89.19	2773
I-ORG	81.29	78.68	79.96	2491
I-LOC	86.54	85.41	85.97	1919
I-MISC	73.59	76.35	74.94	909
B-MISC	0.00	0.00	0.00	9
B-ORG	0.00	0.00	0.00	5
B-LOC	0.00	0.00	0.00	6
Micro avg	83.84	83.99	83.91	8112
Macro avg	47.00	47.33	47.15	8112
Weighted avg	83.61	83.99	83.78	8112

Table 3.4: Classification Report: Conditional Random Field (Window 3).

BiLSTM-CRF (Glove): Run 1

Entity	Precision	Recall	F1 score	Support
I-PER	93.25	89.22	91.19	2773
I-ORG	76.79	74.63	75.69	2491
I-LOC	88.57	84.84	86.66	1919
I-MISC	74.69	66.56	70.39	909
B-MISC	0.00	0.00	0.00	9
B-ORG	0.00	0.00	0.00	5
B-LOC	0.00	0.00	0.00	6
Micro avg	85.03	80.94	82.93	8112
Macro avg	47.61	45.04	46.28	8112
Weighted avg	84.78	80.94	82.8	8112

Table 3.5: Classification Report: BiLSTM-CRF (Glove) - Run 1.

BiLSTM-CRF (Glove): Run 2

Entity	Precision	Recall	F1 score	Support
I-PER	95.14	86.15	90.42	2773
I-ORG	79.76	73.38	76.44	2491
I-LOC	88.17	85.04	86.58	1919
I-MISC	71.94	67.99	69.91	909
B-MISC	0.00	0.00	0.00	9
B-ORG	0.00	0.00	0.00	5
B-LOC	0.00	0.00	0.00	6
Micro avg	86.08	79.72	82.78	8112
Macro avg	47.86	44.65	46.19	8112
Weighted avg	85.93	79.72	82.7	8112

Table 3.6: Classification Report: BiLSTM-CRF (Glove) - Run 2.

BiLSTM-CRF (Glove): Run 3

Entity	Precision	Recall	F1 score	Support
I-PER	95.73	85.79	90.49	2773
I-ORG	75.79	74.03	74.9	2491
I-LOC	88.03	84.31	86.13	1919
I-MISC	70.55	65.9	68.15	909
B-MISC	0.00	0.00	0.00	9
B-ORG	0.00	0.00	0.00	5
B-LOC	0.00	0.00	0.00	6
Micro avg	84.68	79.39	81.95	8112
Macro avg	47.16	44.29	45.67	8112
Weighted avg	84.73	79.39	81.94	8112

Table 3.7: Classification Report: BiLSTM-CRF (Glove) - Run 3.

BiLSTM-CRF (SENNA): Run 1

Entity	Precision	Recall	F1 score	Support
I-PER	89.43	92.72	91.05	2773
I-ORG	78.78	72.42	75.47	2491
I-LOC	89.58	82.44	85.86	1919
I-MISC	67.97	69.09	68.53	909
B-MISC	0.00	0.00	0.00	9
B-ORG	0.00	0.00	0.00	5
B-LOC	0.00	0.00	0.00	6
Micro avg	83.83	81.18	82.48	8112
Macro avg	46.54	45.24	45.84	8112
Weighted avg	83.57	81.18	82.29	8112

Table 3.8: Classification Report: BiLSTM-CRF (SENNA) - Run 1.

BiLSTM-CRF (SENNA): Run 2

Entity	Precision	Recall	F1 score	Support
I-PER	87.89	94.77	91.2	2773
I-ORG	82.74	71.98	76.99	2491
I-LOC	88.62	84.37	86.44	1919
I-MISC	67.9	68.43	68.16	909
B-MISC	0.00	0.00	0.00	9
B-ORG	0.00	0.00	0.00	5
B-LOC	0.00	0.00	0.00	6
Micro avg	84.33	82.13	83.22	8112
Macro avg	46.74	45.65	46.11	8112
Weighted avg	84.02	82.13	82.9	8112

Table 3.9: Classification Report: BiLSTM-CRF (SENNA) - Run 2.

BiLSTM-CRF (SENNA): Run 3

Entity	Precision	Recall	F1 score	Support
I-PER	85.92	94.84	90.16	2773
I-ORG	80.32	73.22	76.61	2491
I-LOC	86.08	85.04	85.56	1919
I-MISC	70.42	69.42	69.92	909
B-MISC	0.00	0.00	0.00	9
B-ORG	0.00	0.00	0.00	5
B-LOC	0.00	0.00	0.00	6
Micro avg	82.68	82.8	82.74	8112
Macro avg	46.11	46.07	46.04	8112
Weighted avg	82.29	82.8	82.42	8112

Table 3.10: Classification Report: BiLSTM-CRF (SENNA) - Run 3.

3.1.2 Confusion Matrices

Conditional Random Field: Window Size 0

	I-PER	I-ORG	I-MISC	I-LOC	B-ORG	B-MISC	B-LOC	O
I-PER	2218	136	48	66	0	0	0	121
I-ORG	286	1793	44	139	5	0	3	302
I-MISC	46	82	710	22	0	9	0	116
I-LOC	102	255	28	1622	0	0	3	47
B-ORG	0	1	0	0	0	0	0	0
B-MISC	0	0	0	0	0	0	0	5
B-LOC	0	0	0	0	0	0	0	0
O	121	224	79	70	0	0	0	37732

Table 3.11: Confusion Matrix: Conditional Random Field (Window 0).

Error Type	Number	%
No Extraction	494	20.93
No Annotation	591	25.04
Wrong Tag	1254	53.14
Wrong Range	18	0.76
Wrong Tag + Range	3	0.13
Total	2360	100

Table 3.12: Summary Errors: Conditional Random Field (Window 0).

Conditional Random Field: Window Size 1

	I-PER	I-ORG	I-MISC	I-LOC	B-ORG	B-MISC	B-LOC	O
I-PER	2481	189	48	59	2	0	3	92
I-ORG	134	1955	51	94	3	0	0	231
I-MISC	18	69	701	34	0	8	0	82
I-LOC	57	136	27	1667	0	0	1	35
B-ORG	0	0	0	0	0	0	0	0
B-MISC	0	0	0	0	0	0	0	0
B-LOC	0	0	0	0	0	0	0	0
O	83	142	82	65	0	1	2	37883

Table 3.13: Confusion Matrix: Conditional Random Field (Window 1).

Error Type	**Number**	**%**
No Extraction	375	21.45
No Annotation	440	25.17
Wrong Tag	916	52.4
Wrong Range	12	0.69
Wrong Tag + Range	5	0.29
Total	1748	100

Table 3.14: Summary Errors: Conditional Random Field (Window 1).

Conditional Random Field: Window Size 2

	I-PER	**I-ORG**	**I-MISC**	**I-LOC**	**B-ORG**	**B-MISC**	**B-LOC**	**O**
I-PER	2495	203	38	60	1	0	1	71
I-ORG	129	1960	71	110	4	1	3	194
I-MISC	26	60	679	45	0	7	0	110
I-LOC	54	124	32	1642	0	0	0	46
B-ORG	0	0	0	0	0	0	0	0
B-MISC	0	0	0	0	0	0	0	0
B-LOC	0	0	0	0	0	0	0	0
O	69	144	89	62	0	1	2	37902

Table 3.15: Confusion Matrix: Conditional Random Field (Window 2).

Error Type	**Number**	**%**
No Extraction	367	20.89
No Annotation	421	23.96
Wrong Tag	952	54.18
Wrong Range	11	0.63
Wrong Tag + Range	6	0.34
Total	1757	100

Table 3.16: Summary Errors: Conditional Random Field (Window 2).

Conditional Random Field: Window Size 3

	I-PER	**I-ORG**	**I-MISC**	**I-LOC**	**B-ORG**	**B-MISC**	**B-LOC**	**O**
I-PER	2520	198	39	56	0	0	0	65
I-ORG	101	1960	64	107	5	1	3	170
I-MISC	22	77	694	44	0	7	0	99
I-LOC	55	109	30	1639	0	0	1	60
B-ORG	0	0	0	0	0	0	0	0
B-MISC	0	0	0	0	0	0	0	0
B-LOC	0	0	0	0	0	0	0	0
O	75	147	82	73	0	1	2	37929

Table 3.17: Confusion Matrix: Conditional Random Field (Window 3).

Error Type	**Number**	**%**
No Extraction	380	22.45
No Annotation	394	23.27
Wrong Tag	902	53.28
Wrong Range	13	0.77
Wrong Tag + Range	4	0.24
Total	1693	100

Table 3.18: Summary Errors: Conditional Random Field (Window 3).

BiLSTM-CRF (Glove): Run 1

	I-PER	**I-ORG**	**I-MISC**	**I-LOC**	**B-ORG**	**B-MISC**	**B-LOC**	**O**
I-PER	2474	63	21	6	0	0	0	89
I-ORG	110	1859	114	68	1	5	2	262
I-MISC	21	108	1628	20	1	0	1	59
I-LOC	2	36	23	605	5	0	2	137
B-ORG	0	0	0	0	0	0	0	0
B-MISC	0	0	0	0	0	0	0	0
B-LOC	0	0	0	0	0	0	0	0
O	166	425	133	210	2	0	1	37776

Table 3.19: Confusion Matrix: BiLSTM-CRF (GloVe) : Run 1.

Error Type	**Number**	**%**
No Extraction	937	44.77
No Annotation	547	26.13
Wrong Tag	592	28.28
Wrong Range	3	0.14
Wrong Tag + Range	14	0.67
Total	2093	100

Table 3.20: Summary Errors: BiLSTM-CRF (GloVe) - Run 1.

BiLSTM-CRF (Glove): Run 2

	I-PER	**I-ORG**	**I-MISC**	**I-LOC**	**B-ORG**	**B-MISC**	**B-LOC**	**O**
I-PER	2389	54	13	5	0	0	0	50
I-ORG	91	1828	94	46	0	4	2	227
I-MISC	29	118	1632	15	1	0	1	55
I-LOC	6	33	32	618	5	0	0	165
B-ORG	0	0	0	0	0	0	0	0
B-MISC	0	0	0	0	0	0	0	0
B-LOC	0	0	0	0	0	0	0	0
O	258	458	148	225	3	1	3	37826

Table 3.21: Confusion Matrix: BiLSTM-CRF (GloVe) : Run 2.

Error Type	**Number**	**%**
No Extraction	1096	51.17
No Annotation	497	23.2
Wrong Tag	536	25.02
Wrong Range	0	0
Wrong Tag + Range	13	0.61
Total	2142	100

Table 3.22: Summary Errors: BiLSTM-CRF (GloVe) - Run 2.

BiLSTM-CRF (Glove): Run 3

	I-PER	**I-ORG**	**I-MISC**	**I-LOC**	**B-ORG**	**B-MISC**	**B-LOC**	**O**
I-PER	2379	39	16	8	0	0	0	43
I-ORG	151	1844	83	64	1	5	3	282
I-MISC	28	128	1618	13	1	0	1	49
I-LOC	12	46	37	599	4	0	0	151
B-ORG	0	0	0	0	0	0	0	0
B-MISC	0	0	0	0	0	0	0	0
B-LOC	0	0	0	0	0	0	0	0
O	203	434	165	225	3	0	2	37798

Table 3.23: Confusion Matrix: BiLSTM-CRF (GloVe) : Run 3.

Error Type	**Number**	**%**
No Extraction	1032	46.97
No Annotation	525	23.9
Wrong Tag	625	28.45
Wrong Range	1	0.05
Wrong Tag + Range	14	0.64
Total	2197	100

Table 3.24: Summary Errors: BiLSTM-CRF (GloVe) - Run 3.

BiLSTM-CRF (Glove): Average

	I-PER	**I-ORG**	**I-MISC**	**I-LOC**	**B-ORG**	**B-MISC**	**B-LOC**	**O**
I-PER	2414	52	17	6	0	0	0	61
I-ORG	117	1844	97	59	1	5	2	257
I-MISC	26	118	1626	16	1	0	1	54
I-LOC	7	38	31	607	5	0	1	151
B-ORG	0	0	0	0	0	0	0	0
B-MISC	0	0	0	0	0	0	0	0
B-LOC	0	0	0	0	0	0	0	0
O	209	439	149	220	3	0	2	37800

Table 3.25: Average Confusion Matrix: BiLSTM-CRF (GloVe).

Error Type	**Number**	**%**
No Extraction	1022	47.65
No Annotation	523	24.38
Wrong Tag	584	27.23
Wrong Range	2	0.09
Wrong Tag + Range	14	0.65
Total	2145	100

Table 3.26: Average Summary Errors: BiLSTM-CRF (GloVe).

BiLSTM-CRF (SENNA): Run 1

	I-PER	**I-ORG**	**I-MISC**	**I-LOC**	**B-ORG**	**B-MISC**	**B-LOC**	**O**
I-PER	2571	116	34	20	0	0	0	134
I-ORG	64	1804	138	46	0	5	5	228
I-MISC	11	94	1582	13	1	0	1	64
I-LOC	3	62	52	628	6	0	0	173
B-ORG	0	0	0	0	0	0	0	0
B-MISC	0	0	0	0	0	0	0	0
B-LOC	0	0	0	0	0	0	0	0
O	124	415	113	202	2	0	0	37724

Table 3.27: Confusion Matrix: BiLSTM-CRF (SENNA) : Run 1.

Error Type	**Number**	**%**
No Extraction	856	40.26
No Annotation	599	28.17
Wrong Tag	653	30.71
Wrong Range	0	0
Wrong Tag + Range	18	0.85
Total	2126	100

Table 3.28: Summary Errors: BiLSTM-CRF (SENNA) - Run 1.

BiLSTM-CRF (SENNA): Run 2

	I-PER	**I-ORG**	**I-MISC**	**I-LOC**	**B-ORG**	**B-MISC**	**B-LOC**	**O**
I-PER	2628	139	42	20	0	0	0	161
I-ORG	19	1793	103	46	0	5	4	197
I-MISC	12	99	1619	25	1	0	1	70
I-LOC	10	72	41	622	6	0	0	165
B-ORG	0	0	0	0	0	0	0	0
B-MISC	0	0	0	0	0	0	0	0
B-LOC	0	0	0	0	0	0	0	0
O	104	388	114	196	2	0	1	37730

Table 3.29: Confusion Matrix: BiLSTM-CRF (SENNA) : Run 2.

Error Type	**Number**	**%**
No Extraction	805	39.4
No Annotation	593	29.03
Wrong Tag	628	30.74
Wrong Range	0	0
Wrong Tag + Range	17	0.83
Total	2043	100

Table 3.30: Summary Errors: BiLSTM-CRF (SENNA) - Run 2.

BiLSTM-CRF (SENNA): Run 3

	I-PER	**I-ORG**	**I-MISC**	**I-LOC**	**B-ORG**	**B-MISC**	**B-LOC**	**O**
I-PER	2630	160	40	26	0	0	0	205
I-ORG	41	1824	113	51	0	5	2	235
I-MISC	15	124	1632	29	1	0	3	92
I-LOC	2	51	41	631	5	0	0	166
B-ORG	0	0	0	0	0	0	0	0
B-MISC	0	0	0	0	0	0	0	0
B-LOC	0	0	0	0	0	0	0	0
O	85	332	93	172	3	0	1	37625

Table 3.31: Confusion Matrix: BiLSTM-CRF (SENNA) : Run 3.

Error Type	**Number**	**%**
No Extraction	686	32.78
No Annotation	698	33.35
Wrong Tag	693	33.11
Wrong Range	0	0
Wrong Tag + Range	16	0.76
Total	2093	100

Table 3.32: Summary Errors: BiLSTM-CRF (SENNA) - Run 3.

BiLSTM-CRF (SENNA): Average

	I-PER	**I-ORG**	**I-MISC**	**I-LOC**	**B-ORG**	**B-MISC**	**B-LOC**	**O**
I-PER	2610	138	39	22	0	0	0	167
I-ORG	41	1807	118	48	0	5	4	220
I-MISC	13	106	1611	22	1	0	2	75
I-LOC	5	62	45	627	6	0	0	168
B-ORG	0	0	0	0	0	0	0	0
B-MISC	0	0	0	0	0	0	0	0
B-LOC	0	0	0	0	0	0	0	0
O	104	378	107	190	2	0	1	37693

Table 3.33: Average Confusion Matrix: BiLSTM-CRF (SENNA).

Error Type	**Number**	**%**
No Extraction	782	37.43
No Annotation	630	30.16
Wrong Tag	659	31.55
Wrong Range	0	0
Wrong Tag + Range	18	0.86
Total	2089	100

Table 3.34: Average Summary Errors: BiLSTM-CRF (SENNA).