Evaluating Automated and Hybrid Neural Disambiguation for African Historical Named Entities

Jarryd Dunn

Dissertation presented for the degree of
Masters of Data Science

Computer Science
University of Cape Town
South Africa
February 2022

Abstract

Documents detailing South African history contain ambiguous names. Ambiguous names may be due to people having the same name or the same person being referred to by multiple different names. Thus when searching for or attempting to extract information about a particular person, the name used may affect the results. This problem may be alleviated by using a Named Entity Disambiguation (NED) system to disambiguate names by linking them to a knowledge base. In recent years, transformer-based language models have led to improvements in NED systems. Furthermore, multilingual language models have shown the ability to learn concepts across languages, reducing the amount of training data required in low-resource languages. Thus a multilingual language model-based NED system was developed to disambiguate people's names within a historical South African context using documents written in English and isiZulu from the 500 Year Archive (FHYA). The multilingual language model-based system substantially improved on a probability-based baseline and achieved a micro F1-score of 0.726. At the same time, the entity linking component was able to link 81.9% of the mentions to the correct entity. However, the system's performance on documents written in isiZulu was significantly lower than on the documents written in English. Thus the system was augmented with handcrafted rules to improve its performance. The addition of handcrafted rules resulted in a small but significant improvement in performance when compared to the unaugmented NED system.

Dedication

Firstly, all glory be to God, for this work could not have been done without Him.
I also dedicate this dissertation to my family for their support, patience and enduring many dissertation-related conversations. It is also dedicated to Kristen for her constant support and encouragement. Finally, to Ruby for the many mountain excursions spent pondering this work.

Declaration

I know the meaning of plagiarism and declare that all of the work in the dissertation, save for that which is properly acknowledged, is my own.
Jarryd Dunn

Acknowledgements

Thank you to my supervisor, Prof. Hussein Suleman, for his patience and guidance. I have really enjoyed and learnt a lot from our meetings.
Thank you to Dr. Carolyn Hamilton, Sizakele Gumede, Dr. Grant McNulty, Debra Pryor, Chole Rushovich and the rest of the 500 Year Archive members for their support and input.
Thank you to Mark Eccles for his input in the legal wording for the terms and conditions for the annotation site.
Thank you to Dr. Martin Puttkammer for his advice around using the CTexT API for natural language processing in South African languages.
Computations were performed using facilities provided by the University of Cape Town's ICTS High Performance Computing team: hpc.uct.ac.za

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Historical documents often contain many references to entities such as people, places and organisations. Within this project, entities are used to refer to people specifically. The span of text, which refers to a particular entity, is called a mention. More specifically, a mention is the longest span of text that refers to a specific person. For example, in the sentence "Shaka was a Zulu chief", "Shaka" is a mention that refers to the entity King Shaka kaSenzangakhona. Entity Linking (EL) is a Natural Language Processing (NLP) task that involves matching mentions of an entity in a document with a record for the entity in a knowledge base [1]. EL is a common upstream task that is often further utilised by other NLP techniques such as information retrieval, information extraction, and knowledge base population [2]. However, before entity linking occurs, mentions within a piece of text must be identified. Mentions may be found using a Named Entity Recognition (NER) system to identify spans of text that refer to an entity. For this document, a single system that both identifies mentions and links them to an entity in a knowledge base is referred to as a Named Entity Disambiguation (NED) system.

NED systems have previously been used to help facilitate research in digital humanities [3]. Thus, NED systems are likely to be beneficial for documents dealing with African history since the names they contain are often not uniquely identifiable. NED systems may also help with further tasks such as information retrieval and creating document collections based on the people mentioned within the documents. However, most previous work has focused on NED for documents in high-resource languages. In contrast, most of the official South African languages are considered to be low-resource languages as there is relatively little text available in these languages, which may be used to train systems for NLP tasks. For example, Wikipedia is often used as a text source for training models used for entity linking [4] or as a knowledge base for entity linking [5, 6, 7]. However, there are over 600 times as many Wikipedia articles written in English than in isiZulu [8] (the most spoken language in South Africa [9]).

This project explores the use of language model-based NED systems for disambiguating the names referring to people within the 500 Year Archive (FHYA) project. The FHYA contains artefacts from the last 500 years of South African history. The texts used in the project were sourced from scanned versions of documents contained within the FHYA and metadata records associated with each of the artefacts in the archive. The metadata records were written in English and are typically short and relatively structured. In contrast, the documents in the FHYA were written in one of several different languages, including English and isiZulu. Additionally, many of the documents in the FHYA are digitised books; thus, they contain thousands of words and have a far less rigid structure than the

metadata records.
The following section provides further background information. Then Section 1.2 presents the research questions. An outline of the experiments used to answer the research questions is given in Section 1.3. Finally, Section 1.4 contains an outline of the dissertation's structure.

1.1 Background

Language models are statistical representations of natural languages such as English or isiZulu. One of the challenges for language models is that the meaning of a particular word may depend on other words in the text to different degrees. One way to deal with this problem is by using attention mechanisms, such as the Transformer architecture proposed by Vaswani et al. [10], to determine how much a word depends on each of the other words in a piece of text. Recently, Transformer-based language models (such as BERT [11]) have been shown to produce state of the art performances or performances close to state of the art across a wide variety of NLP tasks, including Named Entity Recognition (NER) and question answering [11]. Furthermore, the multilingual versions of these language models, trained on datasets from multiple languages, may be used for low-resource languages with a lack of training data as they can learn concepts across languages[12]. Thus, the concepts learnt by training the language model on a high resource language may be applied to a low-resource language.
NED systems may be helpful in a historical South African context as ambiguous names can be particularly problematic in historical accounts. The problems arise as accounts of an event are subjective and may differ in various aspects depending on the source. In historical documents, names may be ambiguous if multiple people have the same name (homonyms); for example, the name Shaka could refer to a Zulu chief (Shaka kaSenzangakhona), film director (Shaka King) or basketball coach (Shaka Smart). Conversely, the same person could be referred to by multiple names (synonyms); for example, the Zulu chief Shaka was also known as Sigidi. The latter can be due to different historical accounts using different names for the same person. Thus, the information about a person or event may differ based on the name used to refer to them. NED systems solve this problem by linking the mentions of a person's name to a unique knowledge base record for the person.
However, there are several difficulties associated with the construction of NED systems in a historical South African context. Since these are historical documents, an Optical Character Recognition (OCR) system was used to extract the text from scanned versions of the original documents. OCR systems tend to introduce errors in the text, which can be difficult for NED systems [7] as well as language models to process correctly [13].
Secondly, within the FHYA collection, a few entities correspond to most mentions (head entities). In contrast, a large number of entities are seldom mentioned (tail entities). In general, it is easiest to link mentions that frequently appear within a data set and are unambiguous, while the hardest mentions to link are both infrequent and ambiguous [14, 15].
Another issue is the nature of the entities within the data set. Firstly, within a South African context, it is relatively common for both nouns and adjectives to be used as names [16, 17], which requires the model to use the context of a word to determine if it is an entity mention. These names are also likely to come from various languages and cultures, making it more difficult for the names to be detected by the system.

1.2 Research Questions

Based on the problems outlined above, the following research questions were investigated:

How accurate are transformer-based language models for NED applied to the text from the 500 Hundred Year Archive?
The FHYA consists of a collection of historical African artefacts and metadata related to the artefacts. Although transformer-based language models have been used successfully on many NLP tasks [11], it is not known how well they will perform NED within an African historical context. NED systems tend to be domain-specific [15]. Thus the NED models constructed using a language model will be compared to a minimalistic baseline. This baseline should cover the easiest cases of entity disambiguation.

What are the effects of handcrafted features on the accuracy of the NED model?
The more accurate the NED systems are, the more valuable they will be to the FHYA. The accuracy of the NED systems maybe be increased by incorporating handcrafted rules as they capture more information that the model can use for disambiguation. Thus the language model-based NED systems from the previous section were extended using handcrafted rules to explore their effect on the system's accuracy.

1.3 Experiment Overview

We developed two main systems to explore the use of NED systems within an African historical context. These systems followed a typical NED system architecture consisting of three main components: entity identification, candidate generation and entity selection (see Fig 1.1). Entity identification used a Named Entity Recognition (NER) system to identify named entities' mentions within the text. The second and third modules then performed entity linking to link the mentions found by the mention detection module to entries in the knowledge base. Firstly, the candidate generation module was used to generate a list of entities to which a mention is likely to refer. Finally, the entity selection module determined which entity the mention referred to by ranking the entities in the candidate list according to how likely it was that they were the entity mentioned in the text.
The first system made use of supervised learning on a labelled subset of the FHYA collection to train separate language models used for mention detection, candidate, generation and entity selection. Where there is a lack of training data, incorporating handcrafted rules may be used to improve a model's performance [18]. Thus the second system used a hybrid approach to improve the performance of the first system on the isiZulu documents by incorporating handcrafted rules. Both approaches resulted in systems that linked mentions of people within the FHYA data set records in a knowledge base. The results of linking people's names for a test set of documents were used to compare the performance between the models.
In addition to the two systems created this project also contributes an annotated data set including documents in isiZulu as well as a web-application for annotating mentions with the person to whom they refer.

1.4 Dissertation Outline

The next chapter reviews existing literature around NED systems, particularly within historical and low-resource contexts. This chapter also includes a review of some of the recent language models.
In Chapter 3 the construction and architecture of an annotation web application used to crowd-source labels for the documents from the FHYA project is described.
Chapter 4 details how the data used for training and evaluating the NED systems was collected using the annotation web application.
The first experiment is described in Chapter 5. This experiment deals with the construction and evaluation of the baseline systems used to benchmark the language model-based NED systems.
Chapter 6 then describes the second experiment, which makes use of supervised learning on the documents labelled using the annotation system. This required the construction and evaluation of the first language model-based NED system (the Automatic NED system).

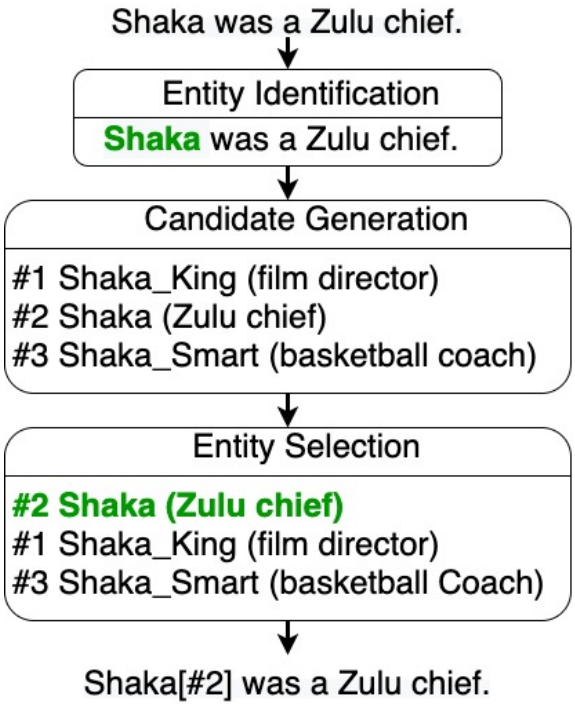

Figure 1.1: Typical modular architecture used for labelling named entities within the text. In this example, the input sentence is "Shaka was a Zulu chief". The output has linked the mention Shaka with the correct entity.

The third experiment is presented in Chapter 7. Here the Automatic NED system is augmented with handcrafted rules to improve its performance resulting in a hybrid NED system. This chapter also contains an evaluation of the hybrid system and a comparison with the Automatic NED system.
Chapter 8 presents the conclusions based on the results from the three experiments.
Finally, Chapter 9 details how this research may be extended in the future.

Chapter 2

Literature Review

2.1 Introduction

The previous chapter outlines some of the issues with ambiguous names in African historical documents and how they might be solved using NED systems to link mentions of people's names that appear within the text to entities from a knowledge base. These systems can also help create digital libraries (DLs), enabling more access to historical artefacts. The NED systems are particularly useful when dealing with large volumes of data where it is infeasible to annotate the data manually. Thus, this chapter reviews some of the existing literature relating to the NED systems. Due to the nature of the documents in the FHYA, NED systems designed for historical documents and documents in low-resource languages were reviewed. Similarly, NED systems are often context-specific but no previous research on NED within a South African context could be found. Thus previous research into the related tasks of NER within a South African context is explored.
Recently, transformer-based language models, such as BERT, have been successfully used to produce NED systems [19, 20]. These models can be pre-trained on large unlabelled corpora before being fine-tuned for specific tasks [11]. This approach requires less training data and time than training from scratch [11]. However, the FHYA contains documents written in multiple languages, primarily English and isiZulu. Hence, multilingual language models are likely to perform better than monolingual language models. Thus both monolingual and multilingual language models are reviewed.
The next section (Section 2.2) describes common architectures for NED systems. Section 2.3 explores several language models that could be used to construct the NED system. NED systems used within a historical context are examined in section 2.4 while section 2.5 looks at NED in low-resource environments. Section 2.6 looks at previous Natural Language Processing (NLP) systems developed within a South African context. Finally, section 2.7 provides a summary of the literature review.

2.2 NED Architecture

Many systems make use of three main components for Named Entity Disambiguation. These are mention detection, candidate generation and entity selection. These modules may be constructed individually or by grouping the candidate generation and entity selection and training them as a single Named Entity Linking (NEL) system with a separate mention detection system. Finally, some systems may also learn all three jointly [21].

2.2.1 Mention Detection

Mentions of entities (e.g. Shaka could be a mention for Shaka kaSenzangakhona) within the text must be identified before they can be linked. Some systems will also attempt to classify the mentions into pre-defined categories (such as a person or location)[22]. Mention detection may be treated as a completely separate task, and many systems exist for performing this task (Named Entity Recognition (NER) systems). Early systems used hand-crafted rules to identify entities. The rules took into account parts of speech, punctuation, suffixes and position in the document [23]. However, more recent systems tend to use these features as input for machine learning algorithms [5]. The use of machine learning rather than hand-crafted rules allows the systems to generalise to different domains [24].
A simple approach is to use dictionaries to compare word spans within the text to lists of entity names (gazetteers)[18]. However, this approach cannot identify mentions that do not occur in the gazetteers [22].
More recently, word embeddings have become prominent in mention detection as they allow for more generalisable systems. Static embeddings are often used, based on tools such as Flair [25]. However, static embeddings do not change depending on the context, meaning homonyms will have the same embedding. Thus dynamic embeddings have become more popular as they can adapt better to different contexts [24]. Dynamic embeddings are often created using bi-LSTMs [26] or BERT [7]. Once the embeddings have been generated they are typically used as inputs to classifier which tried to determine if they form part of a mention. For instance, by adding a conditional random field (CRF) layer on top of a bi-LSTM [26, 27].

2.2.2 Candidate Generation

Candidate generation creates a small but high recall list of candidates to link to a mention [28]. This stage aims to reduce the number of entities that must be evaluated as potential matches to a mention. This allows more accurate but also more computationally expensive techniques to be used for entity selection without making the runtime infeasibly long [20]. The choice of the number of candidates has a significant impact on the performance of the NED system as a higher number of candidates can increase recall [20]. However, too many candidates will reduce the accuracy of the entity selection module [20].
Similar to mention detection, a dictionary of potential entities for each mention can be stored and used to create a candidate list. This works well if there are no changes to the entities within the system and all entities are in the dictionary [29]. However, it is often infeasible to create these dictionaries, particularly for large corpora or corpora, that are frequently updated.
For more dynamic systems, conditional probabilities are often used to help determine candidates. These take the form of $p(e|m)$ where e is the candidate entity and m is the mention. These probabilities are often collected based on the links in knowledge bases such as Wikipedia [19, 30] and YAGO [31, 32]. Another common technique is to use similarity measures between a mention and an entity. Similarity measures might take the form of string similarity between the mention and entity name. The most basic form involves performing string matching. However, Levenshtein distance and other fuzzy matching techniques may be more useful as they can still provide helpful information if there is no exact match [33, 34]. More recently, similarity measures have been applied to contextualised embeddings rather than the surface form of the mention and entity [35, 36,

6]. Wu et al. [20] showed this approach could perform significantly better than a system using the BM25 information retrieval algorithm [37].

2.2.3 Entity Selection

Finally, an entity is selected from the candidate list and linked with the mention. If labelled data is available, this can be done using supervised techniques such as learning to rank algorithms [3]. Alternately, some models frame entity selection as a classification problem either by using a binary classifier to determine if a particular mention and entity should be linked [38] or by treating each of the candidate entities as possible labels for a mention [1]. As in the previous modules, similarity measures can be used either as the only factor or in combination with other features [30]. Similarity measures do not require labelled data; thus, they are popular in weakly supervised and zero-shot approaches [37, 20]. Other systems consider the global context based on co-reference probabilities between entities [32, 39]. However, since the problem of determining the optimum entity for each mention is NP-complete, some form of approximation is usually required [40]. Global techniques also require similar data to determine the prior probabilities of entities co-occurring. Since the prior probabilities are usually domain-specific, models tend not to perform well in other domains. Generally, the techniques used in entity selection may be similar to methods used in candidate generation but are too computationally expensive or sensitive to noise to be applied directly to all possible entities. Removing the candidate generation phase can reduce the complexity of the model while still achieving good results [21], however, Chen et al. [21] found that their model's performed significantly better when a candidate set was used. They found that the drop between the results on the validation and test sets was particularly large.

2.2.4 NIL Selections

NIL selection determines if the mention is not to be associated with any of the entries in the knowledge base. It is usually incorporated into the entity selection phase. A common approach uses a scoring mechanism, i.e. similarity, and assigns a NIL entity to the mention if the score is not above a set threshold [6, 35]. However, this technique may introduce a large number of false positives [35]. Alternatively, a classifier could be trained to classify NIL mentions [22].

2.3 Language Models

Language models try to learn a statistical representation of a natural language. Within a neural context is usually achieved by using a training objective that encourages the model to learn some form of language understanding. These objective functions include predicting the next word given a set of words or predicting a masked word in a passage [11]. Within NED Language Models (LMs) are typically used either to perform mention detection by training the models for classification [19] or to generate context-aware embeddings [20]. The context-aware embeddings are then used as inputs to further systems for performing NED, such as determining the similarity between mentions and entities.

2.3.1 LSTMs in NED

Long-Short Term Memory networks (LSTMs) capture information about the previous states as well as the current state, allowing long-range dependencies to be taken into account [41]. This means that the previous words that have appeared in a sequence can be taken into account when considering the current word. Thus LSTMs have been used to generate context-aware embeddings for NER [42, 43, 41]. The LSTM takes a sequence of tokens (or words) as input and maps it to a sequence of hidden states such that each state may depend on the previous state [10]. The mapped states encode some information about the earlier states. However, in languages, the meaning of a word may depend not only on previous words but also on subsequent words. A BiLSTM architecture may be used to try to capture both these relationships by running one LSTM from right to left and a second LSTM from left to right. A BiLSTM generates an embedding for each word by concatenating the outputs from the two LSTMs [44]. ELMo [45] makes use of pre-trained BiLSTM models to create contextualised word embeddings. These embeddings are based on the weighted contributions from both layers in the BiLSTM. Both LSTMs are pre-trained using next word prediction. Peters et al. [45] set the state-of-the-art performance for NLP tasks such as question answering, NER and sentiment analysis using ELMo to generate context-aware embeddings. However, LSTMs tend to put more emphasis on more recent states. There is also little room for parallelism when training an LSTM due to its sequential nature [10].

2.3.2 Transformers

To address these issues and improve LSTMs, Vaswani et al. [10] proposed the Transformer architecture (see Fig 2.1). The transformer consists of an encoder and decoder. The encoder and decoder are made up of a stack of N_x encoders and decoders, respectively. The attention blocks perform self-attention, which trains the model to look at other parts of a sequence to get information about a specific token [10]. Multi-head attention extends the self-attention mechanism by performing self-attention n times in parallel with the initial vectors corresponding to each token in the input initialised with different values. The n outputs are then concatenated to produce a single output for each token. Performing the calculations in parallel reduces the training time. The masked multi-head attention layer ensures that only information about tokens appearing to the left of the current token can be used. Transformer based architectures for producing context-aware embeddings (such as BERT and GPT) have been used to produce better results than LSTM based approaches (such as ELMo) for NLP tasks such as Natural Language Inference (NLI) [46], question answering and NER [11]. NLI requires a system to determine if two sentences encompass each other, contradict each other or are unrelated [47]. Thus NLI tasks are often used to compare different models ability to understand language.

2.3.3 Transformer-based Language Models

Architecture GPT One of the first language models that used the transformer architecture was the Generative Pre-Training (GPT) model developed by OpenAI. To construct GPT, Radford et al. [46] made use of a two-stage training technique consisting of an unsupervised pre-training phase on a large unlabelled corpus followed by supervised learning for a specific task, the target task. Similar two-stage approaches have been for many other language models including BERT [11] and XLM [48]. GPT is a multi-layer transformer

decoder (see Fig 2.1) where the transformer uses multi-head self-attention over the input tokens. The output from the transformer is fed into a set of feed-forward hidden layers. During the pre-training, Softmax activation is used to produce the final probability distribution for the next word. During the second stage, an additional layer is added to the pre-trained model, and the last hidden layer of the pre-trained model is used as its input. Subsequent GPT models (GPT2 and 3) made use of a similar architecture but with larger models. GPT2 contained twice as many training layers as GPT and 10 times as many training parameters. While GPT3 used twice as many layers as GPT2 as well as larger embeddings.

BERT Like GPT, the Bidirectional Encoder Representations from Transformers (BERT) model developed by Devlin et al. [11] uses a transformer-based architecture and is trained using a pre-training and fine-tuning phase. BERT also makes use of a similar multi-layer bidirectional Transformer [10]. However, the input consists of either a single or pair of spans of text. A special token separates the spans, and a learned embedding is applied, indicating to which span a token belongs. In addition to the special token separating spans, another special token, `[CLS]` is placed at the start of each sequence. The final hidden state of the `[CLS]` is used for classification as a representation of the entire sequence [11]. Two BERT models were produced: $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$. $\text{BERT}_{\text{BASE}}$ has the same number of parameters as GPT. In contrast, $\text{BERT}_{\text{LARGE}}$ has twice as many layers (transformer blocks) and around three times more trainable parameters.

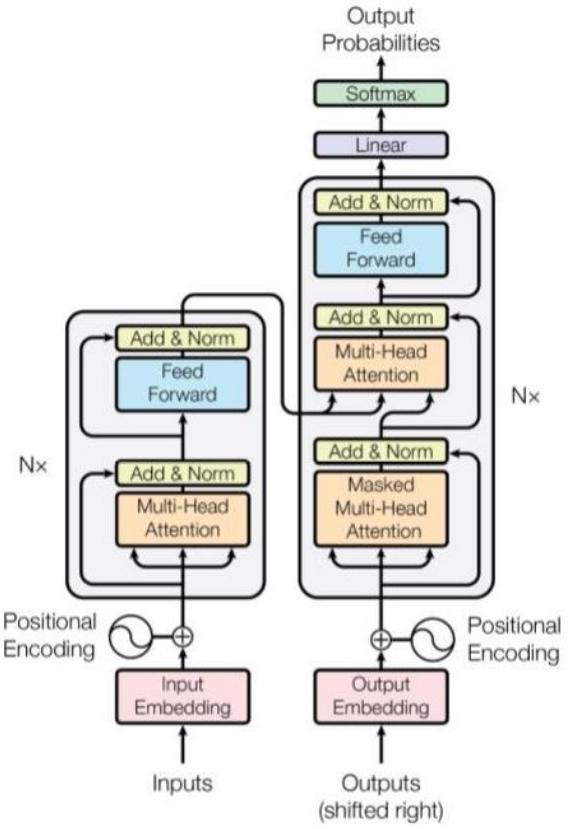

Figure 2.1: Transformer architecture with the N_x layer encoder on the left and decoder on the right. Source: Vaswani et al. [10]

Conneau et al. [49] also made use of a similar architecture to produce $\text{XLM-R}_{\text{BASE}}$ and $\text{XLM-R}_{\text{LARGE}}$ while build on their previous model – XLM [48]. These models have the same number of layers, attention heads and hidden units as $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$, respectively. However, due to a more extensive vocabulary (250K tokens compared to 30K used for the BERT models), the XLM models have more parameters.

Pre-training

One of the main differentiating factors between transformer-based language models is the objective function used during pre-training. GPT, GPT2 and GPT3 were pre-trained using next-word prediction. Next-word prediction requires a model to predict the probability of a particular word being the next word given the words preceding it (see Fig 2.2).

However, using next-word prediction means the model can only make use of the context

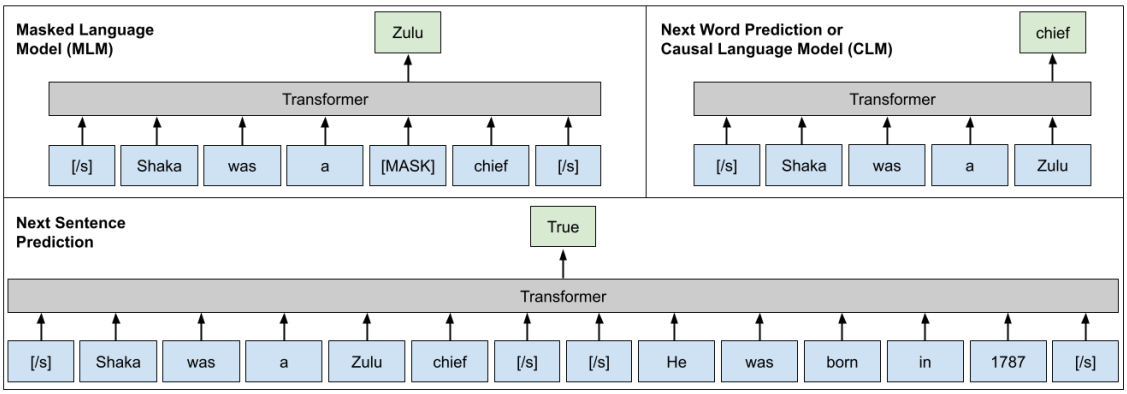

Figure 2.2: The diagram shows simplified examples of the tasks often used to train language models. The `[\s]` token indicates the start and end of a sentence. *Top Left*: Masked language modelling requires the masked token to be predicted [11]. *Top Right*: Next word prediction requires the next token to be predicted given the previous tokens in a sequence. *Bottom*: Next Sentence Prediction training objective used by BERT [11].

to the left of the current token. Since if a model is trained using next-word prediction, then to generate bi-directional embeddings (including the context from the left and right of a token) it would need to be trained from left to right (forward) and combined with another model trained from right to left (backward). However, this means that the forward model will have already seen all the tokens the backward model is trying to predict and vice versa.

Thus, rather than using next-word prediction, BERT is pre-trained using a Masked Language Model (MLM) to produce bidirectional embeddings [11]. In MLM, tokens are first randomly removed from the input. The learning objective is then for the model to predict the removed token (see Fig 2.2). MLM ensures that although the model can see the context to both the left and right for each token, it cannot see the token itself.

In addition to MLM, BERT also used Next Sentence Prediction (NSP) as a training task as MLM does not directly capture the relationship between sentences. NSP is performed by replacing the next sentence with a random sentence 50% of the time (see Fig 2.2). Including NSP as a pre-training task significantly improves the model's performance for tasks such as Question Answering (3.5% increase for the QNLI dataset) [11]. During pre-training, NSP is treated as a binary classification task that seeks to determine if a particular sentence follows another sentence.

Although BERT was able to produce better performances than GPT models of equivalent size [11], Liu et al. [50] found that the performance of BERT models can be improved by adjusting the pre-training regime. They propose four main changes: They train the model for longer, with more data in larger batches. Intuitively, this approach makes sense as most models perform better with access to more data and more training time, provided that the model does not start to overfit the training data. They also adjust the pre-training by removing the next sentence prediction objective. Furthermore, Liu et al. [50] make use of the maximum number of tokens for the input to the BERT model (512 tokens). Finally, they modify the MLM objective to mask tokens at runtime.

Conneau and Lample [48] investigate using next-word prediction, MLM and Translation Language Modeling (TLM) as objective functions for training a cross-lingual language model (XLM). TLM is a supervised task which extends MLM for cross-lingual language modelling by considering parallel texts in two different languages as opposed to a single

monolingual text (see Fig 2.3). Words from both texts are randomly masked such that the model can use either context in either text to determine the masked word. TLM encourages the model to learn to align texts from the two languages. The downside of this approach is that it relies on the availability of parallel text passages which may not always be available, especially for low-resourced languages. Subsequently, Conneau et al. [49] improved on the XLM language to produce XLM-R. XLM-R models are pre-trained using either CLM, MLM or a combination of MLM and TLM. For the TLM task, datasets containing parallel texts in text in English and one of 15 other languages are used. These include French, Arabic, Chinese, Hindi and Swahili. Tokenisers trained for the language were used where possible; otherwise, an English tokeniser was used.

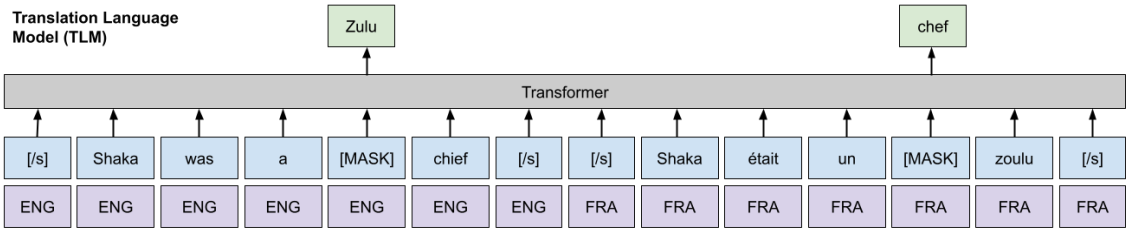

Figure 2.3: Simplified diagram showing an example of the TLM learning objective used by Conneau and Lample [48].

Datasets

GPT GPT and BERT were both pre-trained on the BookCorpus dataset [51]. However, BERT is also pre-trained on English Wikipedia. Unlike the 1B Word Benchmark [52] dataset used to train ELMo, these corpora keep all sentences in order. This allows the models to learn to utilise long-range information. GPT2 made use of the Webtext dataset which is ten times larger than the BookCorpus dataset. The WebText dataset was created using webpages gathered using CommonCraw which had appeared in Reddit posts that had more than three upvotes; this was done to try and ensure that high-quality pages were used. Pages from Wikipedia were excluded to avoid training the model on data that is likely to appear in the test sets of tasks on which it will be evaluated. However, for GPT3, the WebText dataset by adding English Wikipedia and additional high-quality passages from CommonCrawl. Thus GPT3 was pre-trained on 10 times as much data as GPT2.
In addition to the monolingual BERT model, a multilingual BERT model (mBERT) was also produced. mBERT has the same architecture as $\text{BERT}_{\text{LARGE}}$ and was pre-trained using the same objective, however, it was trained using the Wikipedia dumps for the 100 largest Wikipedias. This includes low-resource languages such as Swahili and Yoruba. Since there were large differences in the sizes of different Wikipedias, the proportions of samples from different languages are adjusted using exponential smoothing. This means that smaller Wikipedias are overrepresented, which should help to prevent larger languages from dominating during pre-training. Conneau et al. [49] take a similar approach when training XLM-R since texts in the data collection are sampled based on the inverse of the number of sentences.
However, in developing XLM-R, Conneau et al. [49] find there is a balance between the number of languages a model is trained on and its performance (curse of multilinguality). For a fixed-size model, increasing the number of languages decreases the capacity per language. However, the performance for LRLs increases with the addition of similar

languages. Using more languages also leads to vocabulary dilation since the number of words per language decreases for fixed vocabulary size. Therefore to train on a larger number of languages, both the model's size and vocabulary must increase. Thus, one of the biggest differences between XLM and XLM-R is the amount of data they were trained on (2.5 TB compared to 160GB), especially for LRLs [49]. XLM-R is trained on data in 100 languages, including isiXhosa, collected from CommonCrawl.
Similarly, RoBERTa improves on BERT by making use of a more extensive and diverse training corpus which allows the model to generalise better [50]. They created the CC-News corpus (a collection of English news articles). In addition to CC-News, the model is trained using BookCorpus, OpenWebText (similar to the dataset used to train GPT-2) [53] and STORIES (a CommonCrawl subset of data with a story-like style) [54]. The new training set contains around 160GB of uncompressed text compared to 13GB used by Devlin et al. [11] to pre-train BERT.

Results

	Question Answering		Natural Language Inference			Classification		Similarity		
Model	RACE	SQUAD (v1.1/v2.0)	MNLI-(m/mm)*	RTE*	QNLI*	CoLA*	SST-2*	STS-B*	QQP*	MRPC*
BiLSTM + ELMo + attn [11]	-	- / -	76.4 / 76.1	56.8	79.8	36	90.4	73.3	84.9	64.8
GPT [46]	59.0	- / -	82.1 / 81.4	56	88.1	45.4	91.3	82.0	70.3	82.3
$\text{BERT}_{\text{BASE}}$ [11]	-	88.5 / -	84.6 / 83.4	66.4	90.5	52.1	93.5	85.8	71.2	88.9
$\text{BERT}_{\text{LARGE}}$ [11]	72.0†	90.9 / 81.9	86.7 / 85.9	70.1	92.7	60.5	94.9	86.5	72.1	89.3
XLM-R [49]	-	- / -	88.9 / 89.0	-	93.8	-	95.0	91.2	92.3	89.5
RoBERTa [50]	**83.2**	**94.6 / 89.4**	**90.2 / 90.2**	**86.6**	**94.7**	**68.0**	**96.4**	**92.4**	**92.2**	**90.9**

Table 2.1: Comparison of language models for NLP tasks. The results for each model are taken from the original paper where the model was described unless otherwise stated.
* Tasks form part of the GLUE benchmark. † Evaluated using the model implemented by Liu et al. [50].

It is difficult to compare the performances of the language models for NER and NED tasks since their performance depends heavily on the dataset used. One way to compare the performances is by using the tasks in a standardised collection of NLP tasks. The results for the language models described above on a variety of NLP tasks mostly from the GLUE benchmark [55] are shown in Table 2.1. The classification tasks may be of particular interest for NER since it can be framed as a classification task. Similarly, the similarity tasks are important since entity linking often relies on measuring similarity between texts. Finally, question answering and natural language inference tasks are included since they offer some insight into a language model' language understanding.
GPT initially set the state-of-the-art performances for nine datasets, including question answering, classification, NLI and semantic similarity tasks. However, although it was not evaluated on the benchmarked tasks it was evaluated on eight language modelling datasets with no fine-tuning and set the state-of-the-art for zero-shot approaches for seven tasks. For each dataset, the full GPT-2 model significantly outperformed a model of the same size as GPT, showing that the adjustments made allowed the model to achieve better performances. Similarly, GPT-3 shows stronger performances than GPT-2. However, when GPT-3 is evaluated on the RTE data set its few-shot approach performs on par with fine-tuned $\text{BERT}_{\text{LARGE}}$ but significantly (17.6%) below RoBERTa. Across the whole SuperGLUE [55] suite of benchmarks, GPT-3 only requires 8 examples (K=8) to outperform $\text{BERT}_{\text{LARGE}}$. GPT-3 does not excel when evaluated on five comprehension tasks where it

typically performs well below the current state-of-the-art. For the SQUADV2.0 dataset, it is 11 points behind the fine-tuned $\text{BERT}_{\text{LARGE}}$ model.
In general $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$ perform well on the GLUE benchmarks with significant improvements over GPT. $\text{BERT}_{\text{LARGE}}$ performs significantly better than $\text{BERT}_{\text{BASE}}$ on all benchmarks, indicating that a larger model produces better performance than a smaller model.
Overall, RoBERTa does not seem to overfit the data despite the longer training time. By fine-tuning the parameters for specific tasks, RoBERTa sets state-of-the-art performances for all 9 of the tasks in the GLUE benchmark. By adjusting the learning rate schedule, they also produced the best performance for SQuAD V1.1 [56] and performed close to the best on SQuAD 2.0 [57]. These findings indicate that although the structure of the models is important, the training procedures and data can also significantly affect the performance of a model.
The tasks in Table 2.1 are monolingual thus mBERT was evaluated on XNLI - a cross-lingual NLI task that provides an English training set and then requires models to perform NLI in another language [47]. When evaluated mBERT outperforms the baselines for 15 different languages on XNLI. However, XLM improved on the performance of mBERT Conneau and Lample [48] and both the $\text{XLM-R}_{\text{BASE}}$ and $\text{XLM-R}_{\text{LARGE}}$ exceeded the performance of the XLM with an increase in average F1-score of 5.6% for $\text{XLM-R}_{\text{LARGE}}$ [49].
XLM-R is evaluated for NER using the CoNLL-2002 and CoNLL-2003 datasets. Conneau et al. [49] consider training the model in three different ways: separate models for each language, a single model trained only using English data and a single model trained on the data for each language. In the first scenario, XLM-R performs on par with a highly competitive FLAIR based approach [58]. When trained for NER using only the English training data, XLM-R performs on average just over 2% better than mBERT. Finally, when a single model is trained on the training data for each of the languages, the average F1 score for the XLM-R model increases by almost 9% compared to the model trained using only English data. These results indicate that XLM-R is competitive both for monolingual and cross-lingual NER.

2.3.4 Embeddings

Once the language model has been trained, it can be used to generate context-aware sentence embeddings for a span of text. These sentence embeddings are often generated by averaging the output layers of the language model [25]. In BERT-based models the `[CLS]` token has also been used [37, 20]. However, Reimers and Gurevych [59] found that their model SBERT could generate sentence embeddings which resulted in better results in a variety of tasks than those generated using the average of the output layers or the `[CLS]` token. SBERT was designed specifically for generating sentence embeddings for spans of text such that semantically similar spans produce similar embedding vectors. The sentence embeddings are generated using an additional layer on top of the final layer of the language model. The additional layer pools the output of the final layer to produce a fixed-length embedding (pooled embedding) either using the `[CLS]` token, the mean of all output vectors or the maximum value for each token in the output layers. The SBERT model is then trained to produce semantically meaningful sentence embeddings using one of three objective functions depending on the task. The first is a classification objective function that uses a weighted concatenation of the two input embeddings and a Softmax

layer with cross-entropy loss to adjust the weights. The second objective function is a regression function that uses the mean-squared error of the cosine similarity between two vectors. While the third takes a triplet containing an input sentence along with a positive and negative sentence. The positive sentence is more semantically similar to the input sentence than the negative sentence. A maximum margin loss function is then used based on the Euclidean distance between the embeddings generated for the input, positive and negative sentences. This loss function results in weights being adjusted such that the Euclidean distance between the input sentence and the positive sentence is smaller than the distance between the input and negative sentences.
Using this architecture, the sentence embeddings produced by SBERT outperformed those produced using the BERT `[CLS]` token or the aggregate of BERT hidden states all the tasks in the SentEval toolkit [60] testbeds as well as across several STS (Semantic Text Similarity) data sets. SBERT also produces the best performance of the systems it was compared with for five of the seven tasks for SentEval. The SentEval toolkit is designed to evaluate the quality of sentence embeddings based on tasks including sentiment prediction, subjectivity prediction, question type classification and paraphrase detection. Similarly, they use the 2012-2016 STS datasets as well as the STS benchmark. Each of these data sets is designed to test the ability of models to determine semantic similarity in either supervised or unsupervised settings. Again similarity score between SBERT sentence embeddings performed better than BERT-based models when evaluated on the Argument Facet Similarity (AFS) dataset. AFS contains argument pairs on controversial topics; thus, models are required to determine if two arguments are about the same topic and if they express the same point of view. However, when SBERT was evaluated using cross topic evaluation on the AFS dataset, it performed worse than a BERT-based model. Cross topic evaluation trained the model on two of the topics before testing on an unseen third topic. This indicates that the SBERT sentence embeddings cannot transfer knowledge as well as BERT-based models.

2.3.5 Determining Text Similarity

Reimers and Gurevych [59] use SBERT as a bi-encoder for determining the similarity between passages of text. In this setup, an embedding is generated independently for each section of text. The embeddings are then compared using cosine similarity. The SBERT bi-encoder is trained using a regression objective function based on the cosine similarity produced for two sentence embeddings. An alternative approach is to use a cross-encoder architecture. Cross-encoders are usually constructed by adding a classification layer on top of a language model. Cross-encoders take both pieces of text and input and classify them as matching or not [59]. This architecture allows the model to use full attention across both pieces of text which usually results in better performances compared to bi-encoders [59]. However, each comparison between two pieces of text is fairly computationally expensive making it infeasible to use for large numbers of comparisons. In contrast, since bi-encoders just compare the similarities between embeddings they are typically much faster. Furthermore, the embeddings can be pre-computed and then used for comparison at run time. Thus, a bi-encoder provides a feasible approach for computing the similarities between a large number of passages of text.

2.3.6 Discussion

Table 2.1 shows the results from evaluating the transformer-based LMs described above on various NLP tasks. ELMo, BERT and GPT all produce highly contextualised word embeddings [61] which help improve the performance in downstream NLP tasks. However, the LSTM-based LM ELMo [45] was outperformed by both GPT and BERT on every comparison between the models [46, 11]. The most significant difference between BERT and GPT is that BERT can utilise context to both the left and right of a token, while GPT can only access the context to the left of a token. The bi-directional embeddings appear to help BERT outperform GPT across all the NLP tasks for which it was evaluated despite the GPT and $\text{BERT}_{\text{BASE}}$ being approximately the same size, although $\text{BERT}_{\text{BASE}}$ is trained on significantly more data. More training data and longer training times should increase a model's performance [50]. However, the difference between the performance of $\text{BERT}_{\text{BASE}}$ and GPT is often relatively large. Thus, additional training data is unlikely to be the only reason for BERT's better performances. This indicates that BERT should be able to produce better context-aware embeddings than GPT. GPT has subsequently been extended to GPT-2 and GPT-3; the full version of these models has not been made publicly available, making it hard to use the models in future research. The models were also not tested on the GLUE benchmark, making comparisons difficult. Comparing the performance of $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$ shows that an increase in the model's size can significantly increase performance. RoBERTa manages to achieve further performance increases by adjusting the training for a $\text{BERT}_{\text{LARGE}}$ model. In addition to this, Liu et al. [50] also fine-tunes RoBERTa for each of the downstream tasks. This shows the importance of the training schedule for LMs as the differences between $\text{BERT}_{\text{LARGE}}$ and RoBERTa are typically larger than between $\text{BERT}_{\text{BASE}}$ and GPT, despite $\text{BERT}_{\text{LARGE}}$ and RoBERTa having very similar underlying architectures.
XLM-R also performs well on monolingual natural language understanding tasks where it outperforms $\text{BERT}_{\text{LARGE}}$ and is almost on par with RoBERTa. However, XLM-R also performs well for cross-lingual tasks where it outperforms XLM on all tasks for both the XLNI and Question Answering dataset. XLM-R is also unsupervised, whereas the best performing XLM model required some supervised training. Thus XLM-R offers competitive performances for both monolingual and cross-lingual LMs. However, this comes at the cost of a far larger LM with 55% more parameters than the $\text{BERT}_{\text{LARGE}}$ model. XLM-R also requires far more training data and requires significantly more training time.
For NED, LMs are typically used for mention detection by posing the problem as a sequence labelling task [19] or creating context-aware embeddings, which can be used to determine the similarity between mentions and entities. The models' ability to perform classification tasks like mention detection will be impacted by how well they capture information about each word. Thus, tasks such as NLI should indicate how well an LM may perform for mention detection. The results in Table 2.1 show that RoBERTa produces the best performances for both the natural language inference (NLI) and sentence similarity tasks. However, RoBERTa is a monolingual LM trained only on English data. This means that it struggles to deal with non-English texts. On the other hand, XLM-R shows strong performances on texts from multiple languages as well as English.
As part of the NED process, embeddings may need to be extracted for spans of text. There are several different ways this could be done for each of the language models discussed above. However, based on the work by Reimers and Gurevych [59] SBERT is likely to provide the best option for providing semantically meaningful embeddings.

2.4 NED in a Historical Context

There have been several previous studies looking at NED systems for disambiguating historical documents. Historical documents pose several additional challenges to NER on modern documents. Firstly, the data tends to be a lot noisier as the texts need to be digitised. Using OCR to extract the text tends to introduce errors for historical documents as they often have challenging layouts and fonts [62]. To reduce the effects of OCR errors, systems may use regular expressions [18], Levenshtein distance [19], or word recognition systems [13] to correct the errors before using the text for NED.
Another possible issue is the changes in language over time [18], which can decrease NED performance. For instance, Boros et al. [19] found that training their model using a multilingual transformer on historical text in French and German resulted in better performances than contemporary English documents for English Named Entity Recognition and Categorisation (NERC). NERC systems take NER one step further and attempt to classify the entity into one of a number of categories. Provatorova et al. [35] attempted to use the NATAS library for correcting historical spelling variation. However, taking this approach led to a significant increase in the number of false positives in their candidate generation phase.

2.4.1 HIPE CLEF

L3I

Boros et al. [19] (team L3I) developed a NED system for the Identifying People Places and Other Entities (HIPE) campaign run by CLEF. The HIPE campaign contains scanned newspapers from English, French, and German newspapers. They make use of separate NERC and Named Entity Linking (NEL) systems. The NERC system makes use of BERT models to generate context-aware embeddings. For the English document collection, two approaches were used. The first involved fine-tuning on the CoNLL-2003 dataset with a pre-trained $\text{BERT}_{\text{LARGE}}$ Cased model (trained on case-sensitive data). The second used the mBERT model and was fine-tuned on the training data from the French and German document collections, resulting in better performances. This is possibly due to the CoNLL dataset being more modern than the French and German training data. This further highlights the effects of the training data on a model's performance.
A classification layer consisting of six Conditional Random Fields (CRFs) was used to perform the classification. The classification layer allows for both coarse and fine-grained labels to be applied. An additional stack of transformers is placed between the BERT model and classification layer for the French and German documents.
The entity linking module consists of a BiLSTM to generate embeddings for each mention in the same vector space as the entity embeddings. The local score is then computed using a feed-forward network to combine the entity's probability of being linked to the mention, the similarity between the mention and entity embeddings and the long-range attention scores. The local score is combined with a global score based on the co-reference of the candidate entity with the previously disambiguated entities in the document. The co-reference and mention-entity probabilities were collected from Wikipedia.
The candidates generated were re-ranked based on whether they fit into the same category in DBpedia as was predicted by the NERC system. The most similar candidate would have the same type as the mention and the most similar surface form based on the Fuzzy

Wuzzy Weighted Ratio[1].
The model produced the top performance for 50 out of the 52 scoreboards in the competition. It obtained the best scores in all three languages for end-to-end linking and NERC.

SBB

Labusch and Neudecker [7] (team SBB) also produced a system for the CLEF HIPE campaign. The model makes use of BERT for NER. For the German documents, they make use of a BERT model pre-trained on historical and contemporary German documents [63]. They find that training on only historical documents leads to worse performance on contemporary data [7]. The NER systems for English and French makes use of a multilingual BERT model trained for NER on labelled Dutch, English, French and German data.
Labusch and Neudecker [7] perform NEL based on the similarity between texts containing references to entities in the KB. They make use of Wikipedia as a knowledge base as it provides large amounts of continuous data with entity mentions. The disambiguation is performed using a BERT model trained specifically for text comparison. The text collection for German is taken from Wikipedia pages, while for English and French, the Wikipedia IDs for the Germans articles were found and mapped to the English and French pages, respectively. However, this approach results in much smaller datasets for English and French as the Wikipedia IDs are not always aligned.
Candidate generation is performed based on the cosine similarity between a mention identified in the text and the page titles from Wikipedia articles. The embeddings for the article titles are precomputed and stored in an approximate nearest neighbour (ANN) index to increase the efficiency of the process.
Once the candidates have been retrieved from the ANN index, they are re-ranked using a BERT model trained as a binary classifier, which determines if two sentences refer to the same entity. The classifier is applied to the mention in context and the sentences for the candidate retrieved from the knowledge base. The final score for the candidate is calculated by using a random forest model over the sentence pair probabilities.
When evaluated on the HIPE test data, the system performs fairly well on the German and French data, coming second to the model produced by team L3I for both the NEL (mentions provided) and NED tasks. Their models score on par with the L3I model in terms of precision; however, their recall is far lower, resulting in a significantly lower F1 score. The low recall indicates that the candidate generation module needs to be improved. Their English models perform poorly, most likely due to inadequate coverage in their knowledge base.

2.4.2 Hybrid Historical Linking

For systems without labelled data, hand-crafted heuristics can be used to improve the performance of NED systems. Heino et al. [18] explore NED for WarSampo, a dataset containing a variety of types of data relating to the second world war (WWII). Specifically, they focus on disambiguating names for military units, places and people.
They highlight the issue of changing entities and territories throughout WWII. To deal with these changes, they implement an actor data model that takes temporal changes into account.

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

During the mention detection phase, they use a rule-based system that uses regular expressions to normalise names and remove OCR errors. Then a linguistic analyser is used to pass information to an n-gram generator that produces the mentions. Candidate generation is performed by querying a collection of reference datasets, including the actor model, for potential entities. Finally, entity selection is performed by re-ranking the entities based on information retrieved during candidate generation and the original text. For cases such as military units, they found that different ways of referring to a unit were unambiguous. However, there were multiple ways to refer to the same unit. Thus their actor ontology was used to normalise references to unit names to the same format. This approach was highly successful at disambiguating military units, although their knowledge base coverage limits it. For place names, they use a form of hierarchical surface form matching where names are first checked against historical names for populated areas in decreasing order of size, then against geographic regions within the region where WWII took place. Finally, they are matched against contemporary place names. This approach increases the probability of the correct place entity being assigned when there are several places with the same name. However, since it takes no other information into account, it means that incorrectly disambiguated names will always be assigned to the wrong entity. However, this approach still yields good results, in-KB F1 score of between 0.63 and 0.88 depending on the resource. Finally, for person names, their ranks are normalised to use the complete form rather than abbreviations. Lists are also expanded such that each name appears in full with its rank. Specific rules are also used to handle particular cases. Heuristics are used to filter candidates based on the lifespan of the candidate entities and the length of the match between the mention and entity name. Similarly to the approach to disambiguating place names, the rank and popularity of the entity contribute to the score as more popular or higher ranked entities are more likely to be mentioned. Finally, a threshold is used to determine if a name should be linked to the top candidate or NIL. This approach results in a high overall F1 score for the disambiguation of people names.

2.4.3 Discussion

The three systems described above take very different approaches to historical NER but were fairly successful. They all noted the difficulties due to OCR errors and changes in language use over time. Other than in terms of precision for the French and German datasets, the L3I model performs significantly better than the SBB model on the HIPE NED task. This highlights the importance of the knowledge base the NED system is linking against as this was largely responsible for the SBB teams low recall scores. The hybrid model was not tested on the same datasets, so it is impossible to compare it and the L3I or SBB models directly. However, the approach used by models for HIPE is likely to generalise far better to new datasets as they do not rely on hand-crafted rules. This also means that they can disambiguate mentions not seen during training while the Heino et al. [18] system relies on the names appearing in their reference collection. On the other hand, the system produced by Heino et al. [18] requires no additional labelled data.

2.5 Low-resource NED

Many NED systems require large amounts of data to be trained. For high-resource languages, such as English, this data is readily available. For example, Wikipedia articles

are often used to provide testing and training data as many of the entities within the articles are linked. However, there are typically very few purpose-built datasets for NED in LRLs, and there are far fewer Wikipedia articles. Thus different NED techniques are required. There are three common approaches: distant learning, zero-shot entity linking and transfer learning.

2.5.1 Distant Learning

Distant learning is a machine learning technique similar to supervised learning except that it generates noisy labels based on heuristics or external information such as a knowledge base [64] rather than having explicit labels. Hedderich et al. [12] makes use of gazetteers and hand-crafted rules to create pseudo labels. Since pseudo labels tend to be noisy, they use a confusion matrix based smoothing algorithm [65] to reduce the noise in the labels. Their models are constructed by extending mBERT. The models are incrementally given access to more labelled data to determine how they learn with different amounts of labelled data. Their distant learning model for NER in Yoruba with 100 labelled sentences produces similar results to using 400 manually labelled sentences. However, for small amounts of annotated data, the transfer learning approach described in Section 2.5.3 produces better performances.

To reduce the problem of noisy labels, Le and Titov [66] frame entity linking as binary multi-instance learning (MIL). MIL assigns a label to a class of observations indicating if the class contains at least one instance corresponding to the label. A positive and negative list was generated for each mention where the positive list should include the correct entity.

The positive list was constructed using a string matching heuristic to gather candidate entities containing all the words from the mention. This list was then filtered only to include entities with a relationship in the knowledge base with at least one of the candidate entities for another mention in the sentence. The size of the positive list was reduced to at most 100 entities, based on the order in which they appear in the knowledge base. In contrast, the entities in the negative list were randomly sampled from the knowledge base. However, this approach was very noisy such that the positive list often did not contain the correct entity. Thus a classifier was trained jointly with the MIL model to determine if the positive list contained the correct candidate.

Entity selection was performed using the similarity between the context aware-embeddings generated for the mention and an entity. A BiLSTM produces the mention embeddings based on the mention and a fixed-size context window. The entity embeddings were created based on the types associated with the entity in the knowledge base. The final score was produced by a single layer feed-forward neural network acting on the mention and entity embeddings. The training objective for the model is to score at least one entity from the positive list higher than any of the entities from the negative list. The addition of the noise classifier was used to reduce the weighting of noisy lists. The models are trained on unlabelled texts from the New York Times. Mention detection was performed using CoreNLPs NER system [67]. Finally, they used AIDA-A and AIDA-B as the development and test sets to evaluate the models. Since the AIDA dataset contains news articles from Reuters, there is no overlap between the test and training datasets. However, they do appear within the same domain. They found that their model has an F1 score over 20 points higher than a simple name matching benchmark. However, it still performs substantially worse than a fully supervised version of their model. As in the experiments by

Hedderich et al. [12], the addition of a noise detection mechanism significantly improved the results.

2.5.2 Zero-shot Entity Linking

Zero-shot linking involves classifying previously unseen classes at test time [28]. Logeswaran et al. [37] created a zero-shot NED system as well as a new dataset specifically for evaluating zero-shot NED systems. The dataset was constructed from several Wikias (community-written encyclopedias). Wikias tend to be highly domain-specific, with each Wikia containing a large number of unique entities [37]. The links are created automatically based on hyperlinks in the articles. The dataset is constructed using eight Wikias as the training dataset and four for both the validation and test datasets. Thus, the model will be required to link entities in domains it had not previously encountered when using the validation and test data.
The NEL model created by Logeswaran et al. [37] only expects a collection of entities and descriptions for the entities. It makes use of two stages: candidate generation and entity selection. The candidate generation is performed using an information retrieval approach where BM25 is used to measure the similarity between the mention and the candidate documents. The 64 most similar entities are selected as the candidates. The second phase is candidate ranking; this involves creating a joint representation of the mention and a candidate entity. The mention is represented by 128 tokens, including the mention and its context. Similarly, the candidate entity is defined as 128 tokens from the entity description. A BERT-based model takes a concatenation of the mention and entity embeddings separated by a special token as input. The output from the `[CLS]` token produced by the BERT-based model produces is used as the embedding for the mention-entity embedding pair. The final score is produced by feeding the input vector into a single layer feed-forward neural network. Logeswaran et al. [37] experimented with different approaches for training BERT for entity selection. Following the work by Devlin et al. [11], they use an open-corpus approach where the model is pre-trained on an unlabelled corpus before fine-tuning the model for entity selection. Logeswaran et al. [37] also use task-adaptive pre-training [68] by training the model on unlabeled data from the source and target domain after the training on the open corpus is complete. Training on the unlabelled data is done using MLM. The unsupervised training will allow the LM to learn the general structure and rules of the language. The task-adaptive approach then helps the model learn to generalise between the source and target domains. However, Logeswaran et al. [37] introduce an additional training step following the open-corpus and task-adaptive pre-training - Domain-adaptive pre-training(DAP). In DAP, the model is trained in an unsupervised manner on documents from the target domain. The authors hope that this will increase the quality of the representations for the target domain.
The model using all three pre-training forms produced the best performance with an accuracy of just over 2% more than the model that only used the open-corpus pre-training. The best performing model had an accuracy of 77.05% on the test set when the mention was correctly identified (normalised accuracy). However, they also report an unnormalised accuracy of 56.58% when bound by the recall of the candidate generation stage (68%).
Wu et al. [20] constructed a Zero-shot entity linker that improves on the performance of the model used by Logeswaran et al. [37]. Their model is based on BERT, with Wikipedia as the knowledge base. Following Logeswaran et al. [37], the model consists of two stages. First, a bi-encoder is used to generate a candidate list for each mention. Then

the cross-encoder is used to re-rank the entities and select the entity to be linked with the mention. The model is pre-trained on Wikipedia data.
The bi-encoder uses two separate instances of BERT to create embeddings for the mention and description. The mention embedding is produced based on the context surrounding the mention, while the entity description is based on the entity's title and description in the knowledge base. The similarity between the mention and an entity is calculated based on the dot-product of their embeddings. The candidate list consists of the k most similar entities gathered using a nearest neighbours search. The bi-encoder achieved a top-64 recall of almost 13% more than the BM25 algorithm used by Logeswaran et al. [37]. The candidate list and the mention are used as inputs for the cross-encoder, which works similarly to the second stage used by Logeswaran et al. [37]. The cross encoder uses a concatenation of the mention-context and candidate entity description sequences as an input for a BERT model. The output from the BERT model was combined with a linear layer to produce the final score for the mention-entity pair.
The system was tested on data from the TACKBP-2010, WikilinksNED [69] and Zero-shot EL [37] datasets. For the Zero-shot EL dataset, the model was trained and tuned using the testing and validation sets provided. The tuned model was evaluated on the test set, containing data from a different domain to the training and validation datasets. On this dataset, the system achieved a 6 point increase in unnormalised accuracy compared to Logeswaran et al. [37]. However, the recall for the Wu et al. [20] model's candidate generation is 13 points lower. This indicates that the biggest difference between the two systems is for the candidate generation phase. The efficacy of the second stage can be evaluated by looking at the accuracy when the gold candidate is in the candidate list. In this case, the normalised accuracy of the Wu et al. [20] model is only 1.5 points higher than the zero-shot system developed by Logeswaran et al. [37] and trained using the unlabelled corpus pre-training and is 0.5 points lower than the Logeswaran et al. [37] model trained using the DAP approach. These results suggest that the models may be improved by replacing the candidate generation module used by Logeswaran et al. [37] with the bi-encoder technique. For the TACKBP-2010 and WikilinksNED datasets, the Wu et al. [20] system was fine-tuned on data from the respective corpora. The model achieved state-of-the-art performances on both the TACKBP-2010 and WikilinksNED datasets.

2.5.3 Transfer Learning

Transfer learning is similar to zero-shot learning; it trains a model in a source language to be used on a different target language. Recently, multilingual language models such as mBERT and XLM have been used to learn representations that can be used across languages. Hedderich et al. [12] explores the effectiveness of mBERT and XLM-R for NER and topic modelling in three African languages, including isiXhosa. In their experiments for NER, they use a zero-shot approach trained on only the English CoNLL dataset. To evaluate the model, they incrementally add sentences from the African languages. This approach produces mixed results depending on the language. For the Hausa dataset, the zero-shot XLM-R model achieves a test F1-score of around 0.6, while the model that receives ten labelled Hausa sentences has an F1 score of just below 0.8. In contrast, when 1000 labelled sentences are used, the system achieves an F1 score of just over 0.8. This shows how effective XLM-R can be for transferring knowledge. However, on the isiXhosa dataset, the models are far less effective. The zero-shot system only achieves an F1

score of around 0.1; even when trained on the entire dataset, the F1 score was significantly less (approximately 0.07) than the score achieved by Eiselen [17] using a CRF model with hand-crafted features.

Cross-lingual Entity Linking (XEL)

Cross-lingual entity linking (XEL) is a transfer learning technique for entity linking. XEL systems link entities from text in a source language to text in a target language [27]. It is common for XEL systems to use Wikipedia as it provides a convenient source of multilingual data, including links between entities in text and pages for each entity [27]. In addition to this, Wikipedia also provides inter-language links for entities that can be used to construct bi-lingual entity maps. However, the reliance on Wikipedia means that these systems may not perform well on LRLs, which often have far fewer articles on Wikipedia [27]. Zhou, Rijhwani, and Neubig [27] proposed an XEL system for low-resource languages (LRLs), which performs candidate generation based on the similarity between entity embeddings. Since the resources to create the necessary embeddings may not be available for LRLs, a high-resource pivot language that is linguistically similar to the source language is used instead of the source language. For entity selection, Zhou, Rijhwani, and Neubig [27] use language-agnostic features based on either the entity itself or the relationships between a pair of entities (e.g. co-occurrence). These changes lead to improved performance for LRL source languages. However, the candidate generation phase performs significantly worse than using Wikipedia-based bi-lingual entity maps for high-resource languages (HRLs). Thus, they use a linear combination of the pivoting and entity maps for the final model, resulting in strong performances for both high and low resource languages. However, this model is dependent on the existence of a suitable pivot language, which is not always available.

2.5.4 Discussion

Each of the three methods described above can be used to overcome different issues associated with performing NED in LRLs. If no training data is available, zero-shot learning is the best technique. Zero-shot learning may be done with the least resources as it only requires some way to determine the similarity between a mention and candidate entities. Transfer learning can be particularly effective if there is a high-resource language that is linguistically similar to the low-resource language, which can be used to train the NED model. However, finding resources in a similar high-resource language is not always possible. Unlike zero-shot learning, distant learning requires some in-domain data. However, the data does not need to be labelled, and since data labelling is usually expensive, it is often much easier to access unlabelled data. Regardless of which technique is used, the addition of small amounts of labelled in-domain data can significantly affect the performance of the models[12].

2.6 South African NLP

There has not been much previous research focusing on NED within a South African context. However, work has been done on collecting corpora for NLP, such as creating NER systems. There are several challenges for NED Systems within a South African context.

Firstly, South Africa has eleven official languages, most of which are LRLs. These languages can be split into two main groups: Germanic (English and Afrikaans) and South African Bantu [70]. The Bantu languages can further be split into the Nguni (including isiZulu, isiXhosa) and Sotho languages, with Xitsonga and Tshivenda belonging to neither group [70]. The Nguni languages are conjunctive (a phrase is often represented as a single word), while the other Bantu languages are disjunctive [71].
Even within English documents, there are challenges as the system needs to deal with names coming from multiple other languages, and it is common for nouns and adjectives to be used as names [11]. One of the for systems for NER in a South African context was developed by Louis, De Waal, and Venter [16]. They found that capitalisation, word length and gazetteers of people and place names were helpful features. However, the gazetteers did not have very high coverage of South African names, and many names could not be added as they were nouns or adjectives, such as Blessing, which would have led to false positives.
More recently, Eiselen [17] created NER systems for each of the official South African languages except English. The data used are mainly from government sources such as websites or legislation. These datasets only contain 15 thousand tokens for each language. Eiselen [17] used a CRF model with L2 regularisation to perform NER for each of the languages. The NER models used several features, including internal capitalisation and gazetteers. The internal capitalisation was particularly helpful for conjunctive languages. However, Afrikaans was the only language with comprehensive gazetteers for people and place names. This issue is partly mitigated as loan words are common among the other languages, although a prefix is often added. Thus string matching with English gazetteers could be used as well as gazetteers for the specific language. The NER systems were evaluated based on the F1 score using 10-fold cross-validation. The results for each language are relatively similar, with all but one of the languages achieving F1 scores between 0.70 and 0.78. There are also no clear patterns between the score for disjunctive or conjunctive languages.

2.6.1 Discussion

Performing NED within a South African context is likely to pose several challenges beyond the fact that many languages are LRLs. These include names that come from multiple languages and cultures. It is also common for nouns and adjectives to be used as names. Gazetteers and heuristics have been used effectively to overcome some of these challenges. The NED systems produced by Eiselen [17] show good performance, although there is room for improvement. For instance, the NED system developed by Eiselen [17] significantly outperforms the transfer and distance learning systems designed by Hedderich et al. [12] for isiXhosa NED.

2.7 Summary

NED systems find mentions of entities within documents then link the mentions with records for the entities in a knowledge base (KB). NED is often performed as an upstream task that can help with other tasks such as information retrieval and data mining. NED is typically done over three stages: mention detection, candidate generation and entity selection. In the mention detection stage, the possible mentions of entities are identified. Then during candidate generation, a list of entities to which a mention could refer is

retrieved. Finally, entity selection is performed by selecting one of the candidate entities linking it to the mention.
Recently, language models such as BERT have been used effectively for NED. Several language models have subsequently been constructed optimising and improving the performance of BERT, such as RoBERTa. Performing NED within a historical South African context poses several additional challenges compared to performing NED on a typical dataset such as CoNLL. Firstly, since the documents need to be digitised, OCR errors are often introduced, which can cause significant problems for language models. There may also be differences in the style and grammar used during different periods. Within a South African context, names can be hard to identify as it is common for nouns or adjectives to be used as names. A further difficulty is that names can come from various languages and cultures since South Africa has 11 official languages. Most of these languages can be considered low-resource languages, meaning that few resources are available to train NED models. This problem may be mitigated by using multilingual language models such as XLM-R, which have been shown to be effective in many languages.

Chapter 3

Annotation System

3.1 Introduction

The previous chapter looked at previous research into NED systems. One of the findings that is apparent in previous research is that the more data available for training a system, the better it performs, particularly if the training data comes from the same domain as the test data. Additionally, labelled documents from the FHYA collection are required to evaluate the NED models. The labelled documents need labels for each mention of a person indicating the entity with which it is associated.
As there was no such labelled dataset, one had to be created. Although many systems, such as Amazon's Mechanical Turk[1] and BRAT[2], enable entities to be labelled using crowd-sourcing, at the time of writing, these systems do not allow the entities to be linked to entries in a custom database. Additionally, creating a custom web app would ensure that the system would generate all the required data and use a custom database, and it would not be an extremely complex website to build. Thus we built a custom web app (the Annotator) to annotate documents from the FHYA with the necessary data using crowd-sourcing. The system allowed users to log in, identify names from within passages of text from the FHYA documents, and label each name with the identifier of the person to whom the name referred.
This chapter contains a description of the annotation system we developed to label the documents from the FHYA. An overview of the required functionality is provided in the next section (Section 3.2). Subsequently, the development strategies and underlying architecture are discussed. Finally, the implementation of the required functionality is described.

3.2 Required Functionality

This section provides a brief overview of the functionality required for the annotation system. The subsequent sections will then give more details on how these features were implemented in the final application.

- *Create user profiles*. User profiles are required in order to determine which links each user created. Thus users can be paid for their contributions. User profiles also

[1] `https://www.mturk.com/`
[2] `https://brat.nlplab.org/`

allow users' work to be saved so they can return to continue with the annotation process.

- *Assign documents to users*. When users first sign up for the system, they need to be assigned a collection of documents to be annotated. This must be done such that the maximum number of documents are annotated as soon as possible. At the same time, the system must also ensure that the documents will not be annotated by more people than required and that a user does not annotate a document more than once.
- *Identify mentions of peoples' names from within a passage of text*. The system's primary purpose is to collect labels for mentions of peoples' names from within text documents. The first stage of this process allows users to select a text span that refers to a particular person. These spans of text are referred to as mentions.
- *Link the mentions with the entry in the database for the person to whom it refers*. Once a mention has been selected, the user needs to link the mention to a particular person. These links are used to create the labels required to train and evaluate the NED systems.
- *View links created*. Once a link has been created, the user needs to see which mentions have already been linked. Additionally, the user needs to check to whom a mention refers. Thus the users can verify that they have made the correct links.
- *Delete incorrect links*. If a link is incorrect either because the mention contains incorrect characters or it is linked to the wrong person, the user needs to remove the link.
- *Allow users to navigate between documents*. Each user will have multiple documents to annotate. Thus, the user must be able to move between documents to check on the links created for previous documents and move on to new documents once they have annotated a document.
- *Allow users to opt out of annotating a particular document*. The documents from the FHYA can contain words and sentiments that may be offensive or upsetting for some users. Thus an opt-out feature is required, so users are not forced to annotate a document where they are uncomfortable with the content.
- *Export the results of the document annotation*. Once the documents have been annotated, the labels must be exported so they can be used to train the NED models.
- *Export payment details for each of the users*. The participants are paid based on the number of correct links they create. A link is deemed to be correct if at least two users created the same link, thus the system needs to be able to check the links each user created to determine how much they should be paid.

3.3 Framework

The web app was written in Python using the Django[3] framework. Django was chosen as it provides an open-source, robust and simple framework for developing web applica-

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

tions. Django is based on the model-view-controller (MVC) architecture. This architecture separates the user interface, application logic and data management, making it easier to manage the application as a whole. The data management is facilitated using an Object-relational Mapper (ORM) layer, which allows for easy integration of different databases based on objects created for each of the tables in the database. Views are constructed as Python classes or functions, which access and process data from the database and user input. Finally, the user interface is created using templates that accept data from a view before being rendered to HTML and displayed using a web browser.
For this project, a Postgres[4] database was used as it can provide a reliable and fast database that can scale to match the requirements of this project. Postgres is an open-source relational database that is recommended for use in production Django systems.
The front-end for the web application was developed using the Bootstrap[5] Javascript library for aesthetics, while Vue[6] and JQuery[7] were used to provide dynamic elements of the web application. These are popular JavaScript libraries that have been well documented. Vue was primarily used to manage the dynamic elements required for document annotation, while JQuery managed calls to the database without reloading the page.
Finally, the web application was hosted using an AWS LightSail[8] instance. AWS LightSail provides a cheap and reliable virtual private server designed for hosting and serving web applications. LightSail bundles together a set of AWS services, making it easier to develop, deploy, configure and host a web application. Additionally, it contains predefined configurations for hosting Django applications. Following the recommendations from AWS LightSail, the Apache webserver was used as it is more reliable in production than the default Django webserver.

3.4 Development Process

The Annotator system was developed over several iterations. First, the requirements were gathered based on the nature of the labels required to train the NED models and input from members of the FHYA. An initial system was developed based on these requirements and refined over several iterations with feedback provided by experts and members of the FHYA. Once the system had the required functionality, a small pilot study was used where two users who had never seen the system before used the system to annotate a small subset of documents. The documents included scanned English and Zulu documents and metadata records. However, it transpired that the Zulu documents were misidentified and were written in siSwati. Another system was added to the document loading process to prevent this error from occurring in the final system, which performed language identification using the system from CTexT[9] to ensure that the documents were written in either English or isiZulu. One of the pilot users had some difficulties understanding the task; thus, another page was added to allow superusers to view a document as if they were a particular user. This makes it easier to determine if users are having an issue because of a technical problem with the system or if they do not understand the task correctly. Once these corrections were implemented, the system was used to annotate the final set

[4] https://www.postgresql.org/
[5] https://getbootstrap.com/
[6] https://vuejs.org/
[7] https://jquery.com/
[8] https://aws.amazon.com/lightsail/
[9] https://hlt.nwu.ac.za/

of documents.

3.5 Data Organisation

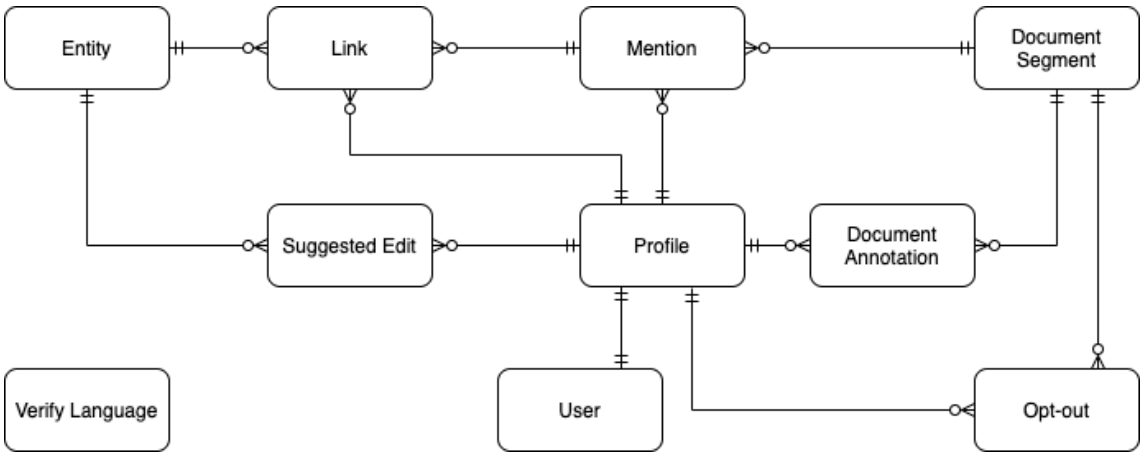

Figure 3.1: Simplified Entity Relationship (ER) diagram for the tables in the Annotator system's database.

The Annotator system makes use of a relational database to store both the annotation and user information. A relational database was used as the Annotator system consists of a relatively small amount of structured data. Another benefit of using a relational database is that they integrate well with the Django framework. Figure 3.1 shows the relationships between the tables in the database while the full Entity-Relationship diagram is available in Appendix A.1. The tables were designed to ensure that the data was mostly in third normal form (3NF). Adhering to 3NF reduces redundancy and increases data integrity within the database. To further reduce the complexity of the database, each of the tables uses an automatically incrementing integer as its primary key. This allows for a high degree of flexibility as composite keys made up of multiple fields would otherwise be required. However, strictly adhering to 3NF can increase computation time; thus, in a few cases, some information was pre-computed and stored, such as the number of words in a document, rather than being computed at runtime.
The DocumentSegment table contains the information for the passages of text annotated, including the text, title and number of words. However, the actual annotations were recorded using the Document Annotation table. The Document Annotation table ensured that documents would appear in a consistent order and tracked each user's progress when annotating the document. A new record was created in the Document Annotation table when a user was assigned a document. The record was then updated as the user annotated the document to capture the number of rows the user had completed. Finally, a boolean field was used to indicate if the user had completed the annotation of a particular document. The actual labels are stored using the Mention and Link tables. The Mention table corresponds with the Mention Detection module of the NED system. It stores the text and location that users consider to be referring to a person. When a user creates a new link, the system first determines if the mention already exists. If there is no matching mention, a new record is created in the Mention table. Once the correct mention has been retrieved, the Link table is used to associate mentions with entities. The Link table is necessary as each entity can be associated with many mentions. Since multiple users will

annotate each document, each mention may be associated with many entities. The Entity table contains a record for each of the people who have been linked to a mention. Each record contains a name and description, and optionally the date of birth and date of death of the person. If mistakes are found in these records, users can suggest corrections. These suggestions were tracked using the Suggested Edit table. The table stores details of the particular change as well as if the change has been reviewed and implemented.
The user information is stored in the Profile table, which extends the default User table provided by Django. The Profile table is discussed in greater detail in the next section. However, it links to many of the other tables to capture information about the creator of these tables' records. For instance, in the Link table, the creator must be known to be paid, while for the Document Annotation table, the users are tracked to manage document assignments. The Profile table also records the user's consent to participate in the project and their progress with the training tasks.
The Opt-out table records instances where a user has declined annotating a particular document. This prevents the system from re-assigning a document to a user who has previously declined to annotate it.
Finally, the Verify Language table stored the questions to verify if a prospective user can read and understand isiZulu. This process is explained in greater detail in Section 3.8.1

3.6 Uploading Documents and Entities

Before users can start annotating documents, the documents have to be uploaded to the system. Django's ORM allows JSON objects to be loaded into a database from a JSON file. This feature is used to load the documents to be annotated into the database. A Python script was written, which read in each of the documents, calculated the values for each field in the database then wrote out the results as a JSON file in the appropriate format. The documents were loaded into the Document Segment table. For each document, the title was based on the filename from which the text was extracted. The document text was split into sections containing around 350 words. The sections are not exactly 350 words as they were rounded to ensure that each section contained full sentences. A section could also be slightly larger than 350 words if splitting it would result in a very short section. For each segment, the total number of segments the original document was split into and the segment's position (e.g. the segment corresponding to the first 350 words of a document would have position 0) are stored in the database. Thus the original documents can be reconstructed. Additionally, the database stored the number of times each segment had been annotated and assigned.
Similarly, a Python script was used to format data for the Entity table as a JSON object to be loaded into the system. The entity table was bootstrapped based on the authoritative records maintained by the FHYA. The authoritative records were stored as a CSV file with several fields, including the authoritative name, date of birth, date of death, and the entity's description. The script extracted the relevant fields from the authoritative records then wrote them to a JSON file in the appropriate format.
Once the JSON objects were created, they were uploaded to an AWS Lightsail instance and loaded into the database for both the documents and entities. This provided a quick and convenient way to deal with uploading the data as well as making it easy to reset the database to its original state when testing the annotator system.

3.7 Document Assignment

The document segments described in Section 4.2.1 are assigned to users when they first signup to use the system. The segments were assigned to users based on the number of assignments and annotations for the segment. Each segment is annotated N times, where N is a configurable parameter (users per segment). Thus the first N users would all receive the same segments to annotate. First, segments that have been assigned fewer than N times are assigned. These documents are assigned in descending order of the number of times they were annotated. Once all segments have been assigned N times, any segments annotated fewer than N times may be assigned to the user. This assignment procedure was used to ensure that each segment was annotated by N users as soon as possible. Since there is considerable variation in segment length, the users are assigned segments totalling a preset number of words rather than a set number of segments.
However, users may not annotate all the documents they are assigned. Thus documents are deallocated from users after a period of time. Document deallocation is configured using two parameters that specify the time before a document was unassigned. The first deals with documents that are only partially completed, while the second is used for documents where the user has done no annotating. Documents are unassigned from a user after the pre-configured amount of time has passed and if the user has been inactive for more than an hour. When documents are unassigned from a user, a flag is set in the user profile indicating that documents should be reallocated the next time the user accesses the site. The documents are allocated using the same procedure as for the original documents.

3.8 User Accounts

The Django framework provides built-in authentication services; these include managing accounts, groups, privileges and cookie-based sessions. The Annotator system makes use of the default system for user verification and managing sessions. Participants annotating the documents are given the most basic set of privileges allowing them to annotate documents. Django provides a default superuser, which is also given read and write access to the database. In addition to this, only superusers have access to pages to export the payment and result information. However, the default user does not capture all the information required by the system. Thus it was extended by creating a one-to-one relationship with a Profile table that captured the additional information.

3.8.1 Registration

Users are required to sign up to the annotation web application before they can annotate any documents. The signup process is mostly managed using Django's built-in facilities for managing user accounts. This includes managing passwords and a basic login page to verify that a user's username and password match before accessing the web application. In addition to a username and password, users also have to enter a South African cellphone number. This number is used to pay participants for their contributions. When users enter the signup page, they are presented with a Language Disclaimer, which warns them that there may be offensive content within the documents from the FHYA. Users must acknowledge that they have read the disclaimer before they can proceed to create an account. Users must also tick boxes indicating that they have read the consent form and consent to participate in the survey. Similarly, they were required to confirm that they had

read and agreed to the privacy policy and the terms and conditions for the website. The consent form was created using a standard template used for research at the University of Cape Town (UCT) and has been read and approved by UCT's Science Faculty Ethics Board (see Appendix A.2). Finally, the signup page includes a pair of multiple-choice questions to test if the participant can read and understand isiZulu using a translation task (see Fig 3.2). These questions consisted of a phrase in isiZulu with three options for the English translation. The user's account is only created if they select the correct translation for both questions.

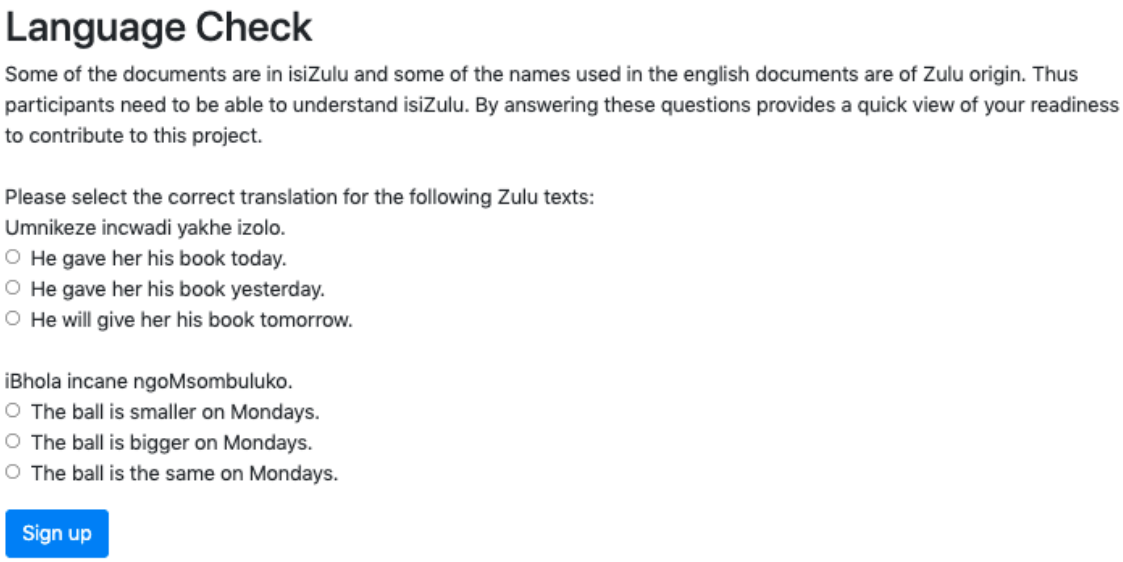

Figure 3.2: The last portion of the signup form verifies that the user can read and understand isiZulu based on a pair of multiple-choice questions.

Multiple-choice questions are used to remove issues around users creating semantically equivalent translations but using different words to the answer the system was expecting. Different combinations of questions are generated each time the signup page is accessed to make it difficult to answer the questions correctly by guessing repeatedly. Following the technique used by Sean Packham [72], a set of four groups of phrases was constructed. Each group consisted of three similar phrases in both isiZulu and English (see Appendix A.3). The questions displayed on the signup page were generated by selecting one of the groups at random using a uniform distribution. A phrase was then selected at random as the question phrase. The form would display the isiZulu for the question phrase as the question. The possible answers were the English translations for all of the phrases in the group. The process was repeated for the second question, except that the group selection could not be the same as in the first question. Thus a user had a 1 in 9 chance of guessing the correct translations for both questions. However, 54 unique combinations of questions made it unlikely that a user would guess the correct translation. The users were also given no information about which translations were correct if one was incorrect. The isiZulu translations for each of the phrases was provided independently by two people who were isiZulu literate. Using independent translators ensured that the translations were of reasonable quality.

3.9 Training

When users create an account, they need to be trained to use the system. The training takes place in two stages. First, the users are given instructions on how to annotate the documents. Once this is completed, the user has to complete a tutorial to ensure that they can perform the task before annotating any documents.
The initial instructions are given using a 7-minute video[10], which demonstrates the actions that the user needs to perform. The video consists of a screen recording showing the actions performed while a voice-over describes what steps are being taken. The screen recording has been uploaded to YouTube and is embedded into a page on the Annotator system. Text headings at the bottom of the screen indicate the broad action being performed, making it easier for viewers to find the section relating to a particular action. The instructional video describes how to: create links, link multiple names, suggest edits to person entries, mark a row as complete, create a new person entry, view created links, delete link, finish annotating a document, get help, opt-out of annotating a document and navigate between documents. Users are prevented from skipping the video as the page displaying the video would only allow users to move on to the tutorial once the video had finished playing. This should ensure that the users have a reasonable understanding of how to use the system. The Annotator system keeps a record of which users have watched the video to ensure users are not forced to re-watch it every time they log in.

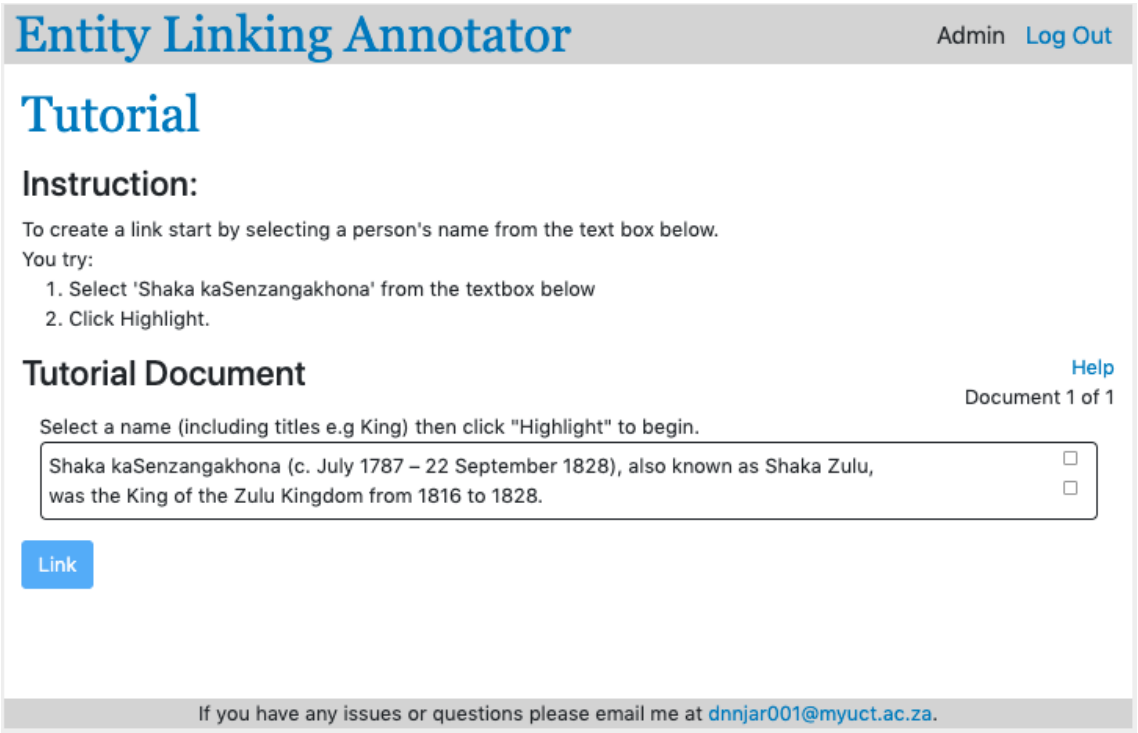

Figure 3.3: Screenshot of the document annotation tutorial for the Annotator system.

Finally, a tutorial was used to test that the users could correctly use the annotation system. Thus, the users must link two entities that appear in the text to an entity in the database. The tutorial page takes the same form as the page for annotating documents, except that it includes instructions for the user (see Fig 3.3). These instructions prompt the user to take specific steps rather than show the user exactly what to do. If the user fails to perform one

[10] https://youtu.be/74LLnY_p-lM

of the tasks or performs it incorrectly, they are given feedback indicating what mistake they made (see Fig 3.3). The tutorial page does not have the full functionality of the document annotation pages but can only respond to the set of steps the user is instructed to take. This prevents users from accidentally performing actions that could affect the database, such as creating new entities. Once they have completed the tutorial, the user will be able to proceed and start annotating documents. As with the instruction video, the system keeps a record of which users have completed the tutorial to ensure that only users who have not completed the tutorial do so before annotating any documents. Once a user has completed the tutorial, a button appears, allowing them to skip the tutorial if they revisit the page.

3.10 Document Annotation

3.10.1 Mention Selection

Mentions within the text are linked in two phases. First, the text making up the mention must be identified. Initially, the system used a search feature to search for the text the user wanted to select as a mention. This had the advantage of selecting all of the matching spans of text. However, when demonstrating the system to experts, it was noted that it would be more intuitive and user friendly to allow spans of text to be selected using the cursor. Thus the system was changed to allow the users to select spans of text using the cursor. In earlier iterations, the user selection would automatically expand to contain complete words. However, there were cases where this resulted in incorrect spans being selected. Thus in the final system, the span of text consists of the exact characters the users selected. Selecting a span of text produces a pop-up menu that displays the selected text and a button prompting the user to link an entity to the mention (see Fig 3.4). The selected text is shown to allow the user to verify that they have selected the correct text. If there is a mistake, the user can correct their selection by selecting a new span of text. Clicking the Highlight button brings up a pop-up menu (the Person Selection Menu) that allows the user to link text spans to an entity from the database. Once the user has clicked Highlight, the span of text is highlighted yellow.

3.10.2 Entity Selection

The second phase in linking the mentions requires the mention to be associated with an entry for a person from the knowledge base. This is done using the Person Selection menu. The Person Selection Menu contains two main sections seen in Figure 3.5. In the first section, the user can remove any mistaken mentions that they do not want to link. The second section deals with selecting the entity to link to the mention. This section comprises a search bar and a list of entities retrieved from the database. Users can use the search bar to search for names from the database. A partial search is used to retrieve entities from the database. An entity is returned if the query term is a substring of its name or a substring of a mention linked to the entity. Linking based on mentions that have been linked to an entity should mean that the search results get better as more documents are annotated. The correct entity is then selected using a radio button. When the page is first opened, the list is pre-populated with entities by searching for the first three characters of the most recently selected mention. The users can view the entity name and aliases to help them decide which entity is correct. This may be especially useful in cases where

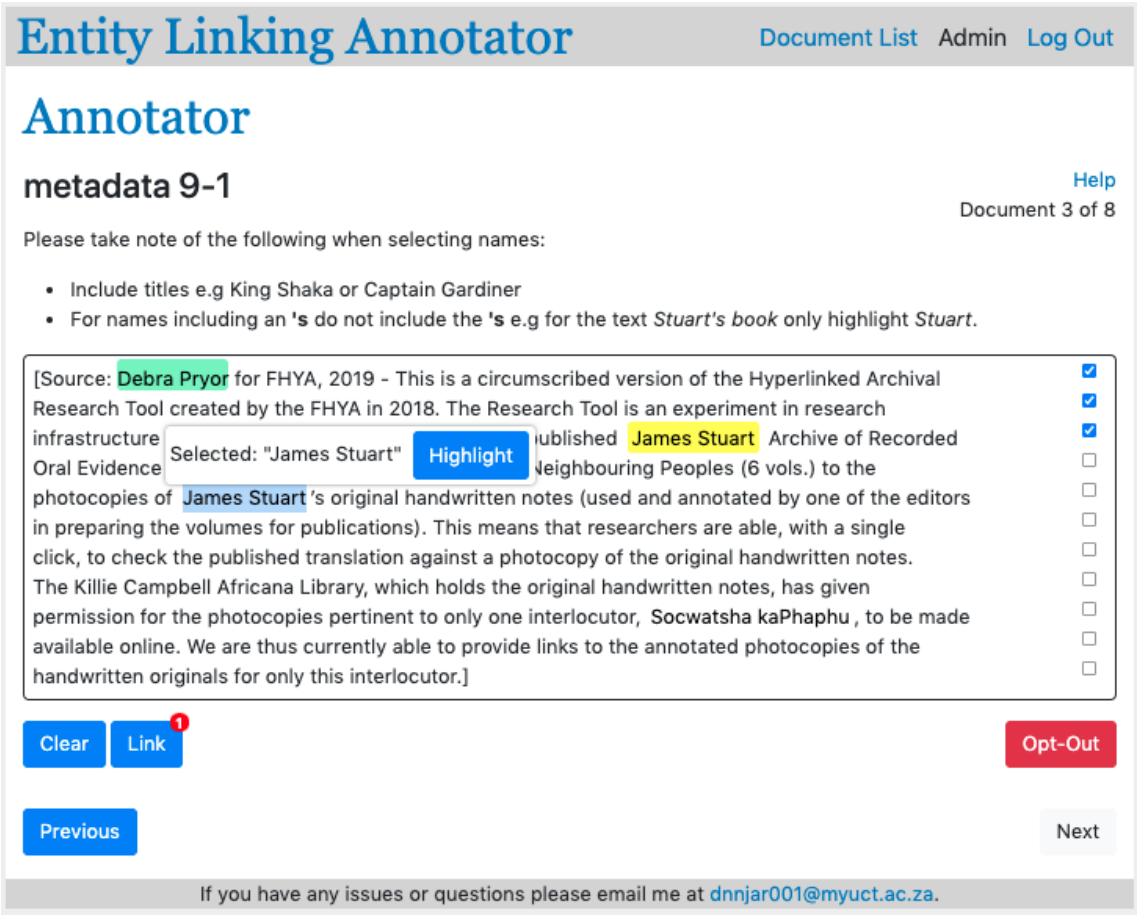

Figure 3.4: Document annotation using the Annotator system. The text highlighted in green indicates mentions that have been linked to a entity while the text highlighted in yellow indicates mentions that are yet to be linked.

the mention is not the same as the correct entity name. The description for the entity is retrieved from the database. The aliases show a list of each specific mention to which an entity has been linked.

3.10.3 Create a New Person Entry

If the mention refers to none of the entities in the database, the user can create a new entity record by selecting the Create Person item (see Fig 3.5). Each new entity record must have a name and description. These fields are automatically populated with the mention and a context window, respectively. The context window consists of approximately 250 words before and after the mention. The context window is then rounded to contain complete sentences. The create entity form also has fields for adding a date of birth and date of death; however, these fields are optional as this information is seldom available.
When the user clicks the link button, the selected entity is linked to the highlighted mentions. If a new entity needs to be created, it will be created then linked to the mentions. These mentions will now be highlighted green to show that they have been linked. Based on feedback from members of the FHYA, this functionality was extended to allow nested highlighting for the same mention. This feature may be required for cases where a mention for one entity contains a mention that may refer to another; for example, the name "*Shaka kaSenzengakhona*" refers to one person, but "*Senzengakhona*" refers to another.

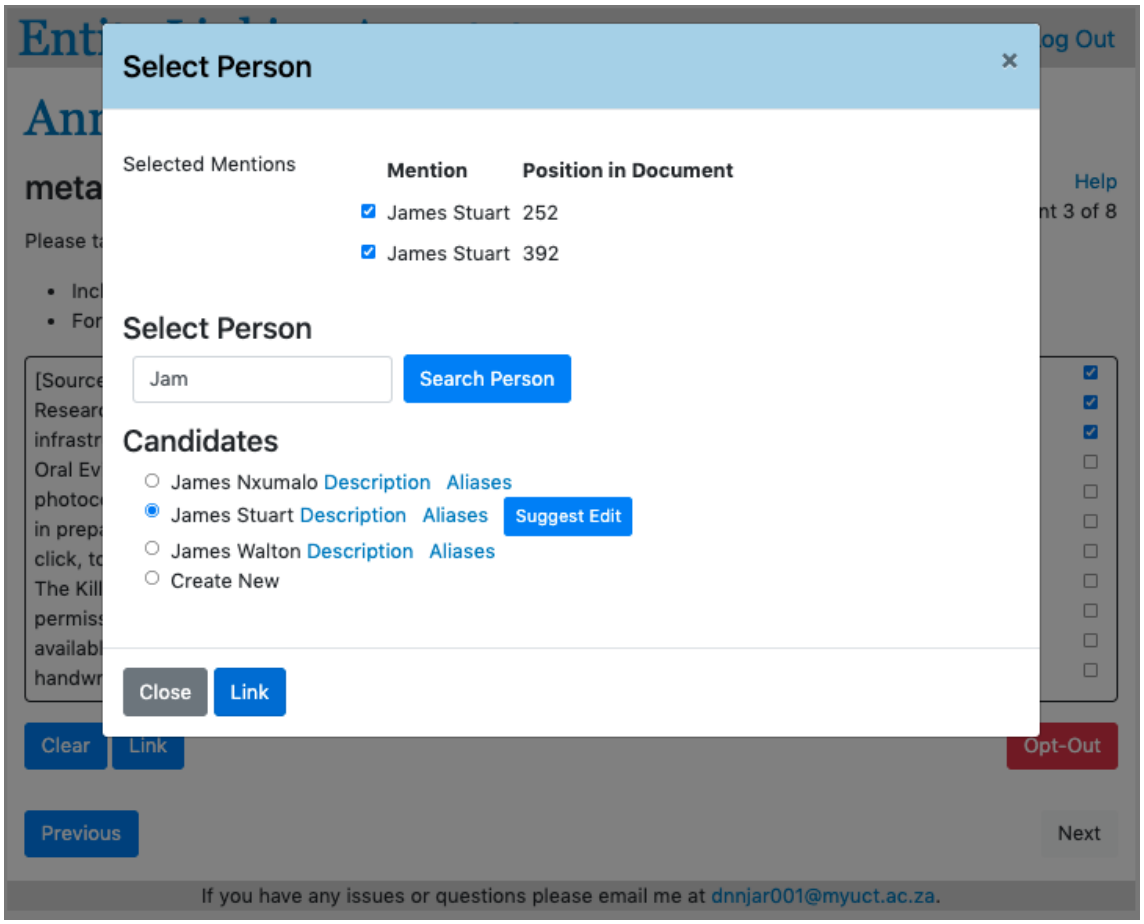

Figure 3.5: Select Person menu with the James Stuart entity selected as the entity to be linked to two mentions both with text James Stuart.

Thus the links may be nested. The Annotator supports the nesting of up to three separate entities.

3.10.4 Selecting Multiple Entities

Multiple names may be selected from the text and linked to the same entity. This allows users to speed up the annotation process as they only need to find and select the correct person record once. In order to link multiple entities, the user would close the Select Person menu and select another span of text. Selected mentions that were not linked to an entity were highlighted yellow, making it easy for users to see which mentions they would link. If the users made a mistake, they could either clear all the highlighted but unlinked spans by pressing the clear button or uncheck the box for certain mentions when selecting the entity.

3.10.5 Suggesting a Change to a Person Entry

A suggested edit feature was added to the annotator system to allow users to suggest corrections or improvements to either the name or description of entities in the database. This feature allows users to suggest changes to entities. However, these changes are not implemented but saved in the database. A site administrator can then access the suggested changes and determine if the changes should be implemented with the help of experts.

Only allowing administrators to edit the entities should prevent malicious or mistaken changes.

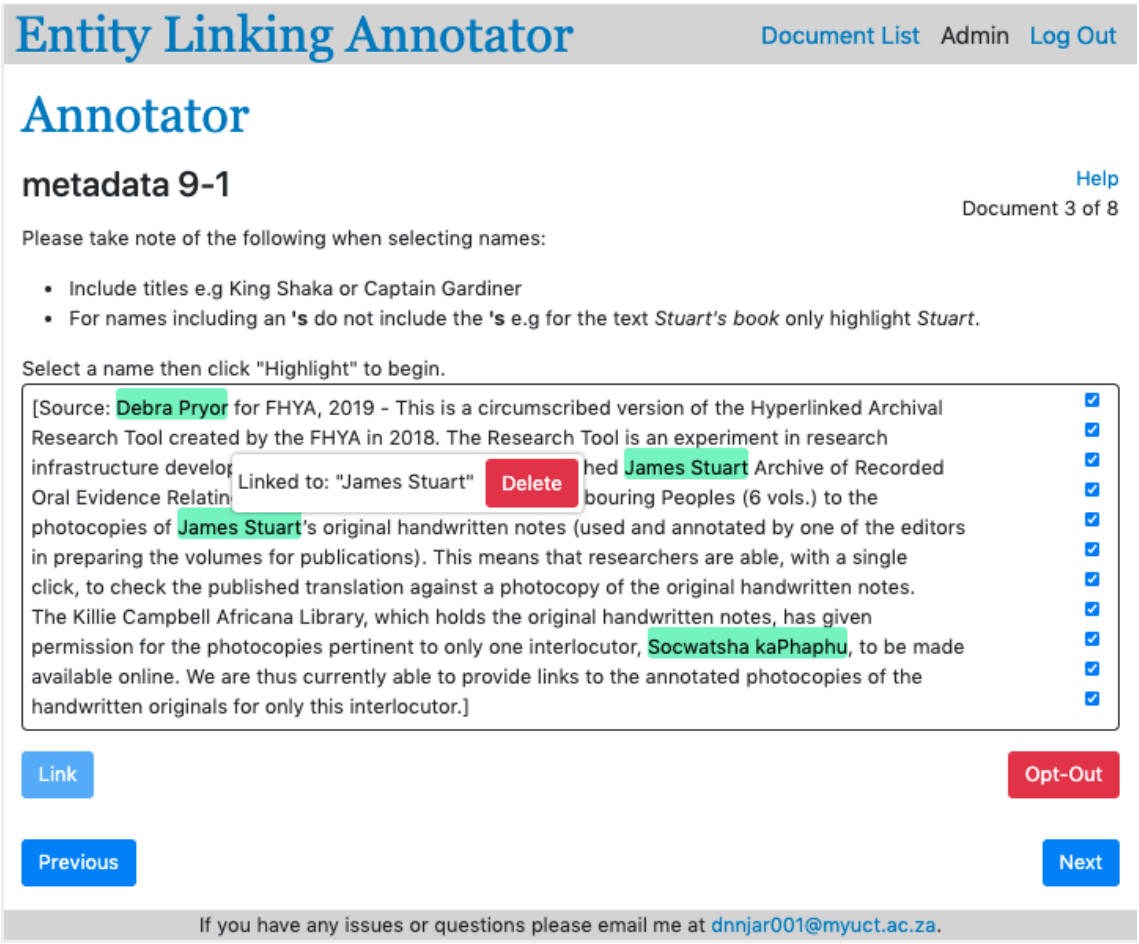

Figure 3.6: Completed document annotation where a user has clicked on the *James Stuart* mention to display the person it was linked to.

3.10.6 Viewing and Deleting Links

Linked mentions are highlighted in green. The user can check which entity a mention is linked to by clicking on the highlighted text; this brings up a pop-up menu (see Fig 3.6) that displays the name of the linked entity and a delete button. If the link is incorrect, the user could delete it and recreate it with the correct entity. Clicking the delete button removes the link and the highlighted text from the database.

3.10.7 Marking a Row as Complete

A checkbox at the end of each row indicates whether a user has checked the row for mentions. During the training, users were encouraged to check the boxes as they went through the document. These checkboxes were designed to ensure the users did a thorough job of checking each line for mentions. Users can only move to the next document if they have checked all the boxes on the current document. The next button is only activated if all the checkboxes have been ticked, as seen in Fig 3.6.

3.10.8 Opting Out of Annotating a Document

The documents from the FHYA contain content that some users may find highly offensive. Where highly offensive words appeared in the text, they were censored by replacing all but the first and last letters with an asterisk. This technique should allow researchers to reconstruct the censored words if necessary. However, in some cases, phrases or sentences contained offensive content. Longer passages could not be censored or removed as the text should be changed as little as possible as the changes affect future research. At the same time, users should not be forced to read and annotate content found offensive. Thus the user could avoid annotating a particular document by clicking the opt-out button. Opting out of a document un-assigns the document from the user and replaces it with another document. If someone has opted out of annotating a document, the database is adjusted to reflect that it has been unassigned, and it will be assigned to another user. However, the database keeps track of documents each user has opted out of annotating to ensure they are not re-assigned to the user.

3.10.9 Document Navigation

Users can navigate between documents using two methods. Firstly, at the bottom of each page, there are buttons to take the user to the previous and next pages, respectively. However, users can also move between documents by clicking the document list link. This redirects the user to a table of contents showing all the documents they were assigned. The following information is displayed for each document: the title, if the document had been annotated, and how many lines the user still needs to annotate. This allows users to get an overview of the documents they still need to complete.

3.10.10 Annotating Additional Documents

Once the user has annotated all of the documents assigned, they are directed to a review page. The review page allowed users to sign up to receive information about the study results, finish the survey, or annotate another document. If the user selects to annotate another document, they are assigned a new document and redirected to the page for annotating the document. The document is assigned to the user using the same algorithm described in Section 3.7.

3.10.11 Completing the Annotation Process

If the user selects the option to finish annotating documents from the review page, they are redirected to a page thanking them for their contribution. The page also displays the number of documents and words they processed as well as the number of links they created. Users are then prompted to log out of the system.

3.10.12 Email Notifications

One of the problems identified during the pilot study was that administrators of the Annotator system could not easily tell when users had completed the survey. Email notifications were added to remedy this. Whenever a user was directed to the completed survey page, an email was sent to the system administrator to notify them that someone had completed the task, the number of documents they had processed, and the links they had

created. These notifications help give feedback on the progress being made in annotating the documents. Similarly, email notifications were set up to notify the administrators of any errors that occurred. The error message and a stack trace were emailed to the administrators when an error occurred. This allowed problems to be quickly identified and resolved.

3.11 Exporting Results

Once the documents are annotated, the results can be exported as a JSON object containing tags for both mention boundaries and entities. The mentions boundaries are demarcated using the IOB tag format (see Fig 3.7).

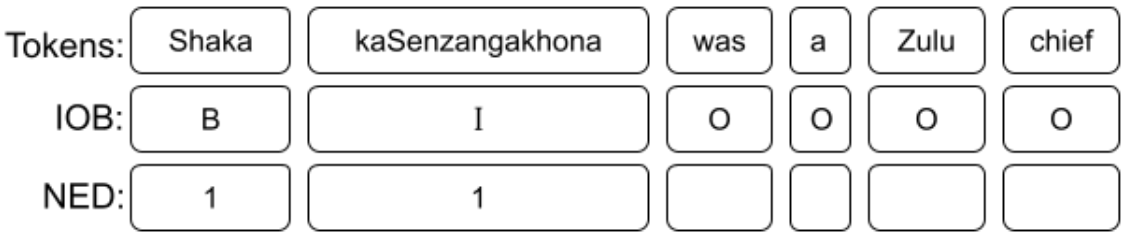

Figure 3.7: Example of the tag formats that are exported from the Annotation System.

The IOB categorises each token using either an I, O or B tag. The first token in a mention is labelled with a B tag; otherwise, subsequent tokens forming part of a mention or single token mentions are labelled with an I tag. All other tokens are labelled with an O tag. The entity labels make use of the ID of the entity to which a mention refers. Thus for each word in every document, the system would output two tags, one indicating if it was part of a mention and another with the corresponding entity ID if applicable.
Each JSON object contains five fields: `tokens`, `ner_tags`, `el_tags`, `all_ner_tags` and `all_el_tags` (see Fig 3.8). The `tokens` field contains a list of all the tokens that make up the text; these tokens are the words split on white space. Once the document has been split into tokens, the IOB and entity tags are generated for the links created by each user. Thus, there were N sets of ner and el tags for each document as N users annotated each document. Each of the lists was stored in the `all_ner_tags` and `all_el_tags` fields. The N sets of tags are then aggregated by selecting the mode tag for each token and stored in the `ner_tags` and `el_tags` fields, respectively. For both the ner and el tags, the mode tag is selected across the sets created by each user. In the case of multiple tags appearing the same number of times, all tags are returned. Results could be exported for an individual document as a single JSON object or all the documents in the system as a list of JSON objects. Access to the pages exporting results was restricted to superusers (the author) as there was no need for participants in the annotation process to access the results.
The payment details can also be exported from the Annotator system. Again this information is exported as a JSON object. The JSON object contains a nested JSON object for each user identified by the user ID. The nested JSON objects contain the information required to pay users, such as the users' cellphone number, the total number of links they created and the number of links they created that agreed with the majority of links for a

particular mention. The user object also contains a list of entities corresponding to the links they created and the most selected entity for the link. This allows the users' results to be quickly and easily verified. Since the information generated by this page contains private information, only superusers were able to access this page.


```
[
  {
    "id": 3,
    "tokens": ["[Source", "-", "Benathi", "Marufu", ...],
    "ner_tags": [["O"], ["O"], ["B"], ["I"], ... ],
    "all_ner_tags": [[["O"], ["O"], ["B"], ["I"], ... ],
                     [["O"], ["O"], ["O"], ["O"], ... ],
                     [["O"], ["O"], ["B"], ["I"], ... ]],
    "el_tags": [["O"], ["O"], ["B"], ["I"], ... ],
    "all_el_tags": [[[], [], [397], [397], ... ],
                    [[], [], [], [], ... ],
                    [[], [], [397], [397], ... ]],
  },
]
```


Figure 3.8: Mention and Entity tags exported from Document 3.

3.12 Summary

The NED systems required labelled data for training. However, there was no pre-existing labelled dataset; thus, a new dataset needed to be created. A custom web app, the Annotator, was created to crowd-source labels for documents from the FHYA. The Annotator allows users to create profiles, including giving consent for their work to be used for research and a simple test to verify that they can read and understand isiZulu. Before users can start creating labels, they must watch a video on how to use the system and complete a tutorial where they can practice linking names occurring in a document. Once their training has been completed, they can identify names occurring within the text and link them to the records for people from a database. Where there are no records for a particular person in the database, users can create a new person record. Having multiple users annotate each passage of text increases the quality of the labels. Thus the system manages the allocation of text passages to ensure that a pre-configured number of users annotates each passage. The assignment process gives users a number of documents when they first signup. Once these documents have been labelled, they may request additional documents. The labels for each document or for all the documents at once can be exported as a JSON file. Finally, the information required to make payments to the users, including the number of correct links for each user, can also be exported as a JSON file.

Chapter 4

Data Collection and Preparation

4.1 Introduction

The previous chapter describes a system used to label the documents from the FHYA collection, referred to as the Annotator. Once the Annotator had been constructed and tested, documents were loaded into the Annotator to be labelled. This resulted in a set of labelled documents and a knowledge base. This chapter details the processes used to prepare the documents from the FHYA collection such that they could be used to train the NED systems.
The following section (Section 4.2) describes how the documents that were labelled were selected from the FHYA collection. Section 4.3 then describes how the knowledge base was constructed. The configuration parameters for the Annotator system used to produce the labelled data and knowledge base is discussed in Section 4.4. Once the labelled data and knowledge base had been produced, they were split into folds for training and evaluating the NED systems as described in Section 4.6.

4.2 Document Collection

The documents used in the following experiments came from a snapshot of the FHYA collection taken on the 1st of April 2021. This snapshot includes artefacts organised into several collections. The artefacts include photographs, audio recordings and PDF scans of historical documents and books. Plain text versions of the PDF scans were required for training and evaluating the models. This text was extracted from a subset of the documents from the snapshot. Members of the FHYA were asked for advice on collections that contain a wide variety of the people names that might appear in the FHYA collection. Based on the recommendations from members of the FHYA, PDF documents were taken from the uMgungundlovu and FHYA Depot collections and converted to plain text. However, for several of the PDFs, the extracted text was unreadable; these documents were removed.
Before the text was extracted, the PDFs were edited to ensure that each page only had a single column of text and that all the pages only contained text. If a PDF was found to have two columns of text, either because it was a scan of a book showing two pages at a time or because each page had two columns, a script was used to split each page in half to form two new pages in the correct order. Optical character recognition (OCR) had already been performed to make the plain text available for most of the PDFs. However, if the

text was not available, OCR was performed using Tesseract[1] for predominantly English PDFs and the CTexT isiZulu OCR engine[2] for isiZulu documents. Once OCR had been performed, the text was then extracted from the PDFs using a simple Python script that used the pdftotext library[3]. The extracted text was then cleaned by removing unnecessary spaces and correcting simple OCR errors, such as numbers containing a lower case letter l instead of one. The extracted text also contained a large number of Unicode characters. As part of the cleaning process, these were converted into their ASCII equivalent where possible using a lookup table; otherwise, they were removed. A spell checker was used for the English documents to correct some of the remaining OCR errors. Finally, offensive terms that appeared within the text were censored. The statistics for the final collection of plain text documents are shown in Table 4.1.

Collection	Language	# Documents
uMgungundlovu	English	27
FHYA Depot	English	91
	isiZulu	13
	English and isiZulu	3
	Afrikaans	2
	French	2
	isiXhosa	1
	seSotho	1
Total	English	118
	isiZulu	13
	Other	9

Table 4.1: Number of documents in each language from the uMgungundlovo and FHYA Depot collections of the FHYA.

The FHYA also includes metadata for each of the artefacts in the collection. Each metadata record contains several fields; however, only the contents of the `scopeAndContent` field is used as it includes the source of the artefact and a brief description. As each metadata field only contains a small amount of text, all of the metadata files from the FHYA were included in the document collection. The resulting set of text from scanned PDFs and metadata is referred to as the plain-text dataset.

4.2.1 Labelled Document Preparation

A subset of the plain-text set containing approximately a quarter of a million words – the annotation dataset – was annotated with labels. The labels indicated to which entity each mention in the text referred.

We built an application with a supporting database to facilitate labelling the documents (see Chapter 3). However, some of the documents within the FHYA archive are too long to expect a single person to annotate the whole document. Thus, once the annotation collection documents had been selected, they were split into segments. Since different documents have different frequencies of mentions, the document segments were shuffled before being loaded into a database. This increases the likelihood that the documents for

[1] `https://opensource.google/projects/tesseract`
[2] `https://hlt.nwu.ac.za`
[3] `https://pypi.org/project/pdftotext/`

each user will contain a similar number of mentions compared to the documents assigned to another user. The database also stored the number of assignments and annotations for each segment of text.
The documents and metadata files were randomly sampled to form the annotation dataset. The documents were classed into three types: metadata, English or isiZulu. Most of the documents in the plain-text dataset are written in English, with approximately 10% of the documents written in isiZulu. If the number of words in the annotation dataset had been taken in these proportions, it would have been possible for the NED systems to achieve good performances overall, even if they performed poorly for isiZulu texts. However, that could lead to biases in downstream tasks. Thus the annotation collection contained all of the metadata files in the FHYA collection and an equal amount of isiZulu and English text. This resulted in approximately 47.5% of words coming from English documents, 47.5% from isiZulu documents and the remaining 3% from metadata.
The language detection API made available by the North-West University's (NWU) Centre for Text Technology (CTexT) was used to ensure that the documents were in the required language. Language detection was applied to the first 1000 characters of each document in the annotation collection to ensure it was in the desired language.

4.2.2 isiZulu Documents

Of the 13 isiZulu documents, two were not considered for annotation as they were largely made up of lists of bird and plant names. Two more documents were excluded as the plain text extracted was extremely noisy. The isiZulu documents were selected by randomly sampling the remaining nine documents using a uniform distribution until the required collection had the required number of isiZulu words. For some of the isiZulu documents, the text within the PDFs was not available. The isiZulu Optical Character Recognition (OCR) system from CTexT was used to extract the text for these documents.

4.2.3 English Documents

The English documents contributed a total of 118 992 words to the annotation collection. 77% of the English documents occurred in the FHYA Depot, while the remaining 23% occurred in the uMgungundlovu collection. This resulted in 27240 words coming from the uMgungundlovu collection and the remainder from the FHYA Depot. Documents were selected randomly using a uniform distribution. If the document selected was judged to produce a poor quality scan, it was removed and replaced with another document from the same collection. The final composition of the documents in the annotation dataset is shown in Table 4.2.
Many of the documents from the FHYA Depot collection were very long. Thus a maximum of 10000 words was taken from each document. These passages were selected by splitting the document into sections of up to 1000 words, then selecting one section at random using a uniform distribution. Limiting the number of words per document increases the number of documents represented in the annotation dataset.

4.3 Knowledge Base

The knowledge base (KB) is a simple database containing the name and a description of each entity as well as the date of birth and date of death, if available. The KB was

Type	Collection	# Documents	# Words
isiZulu	FHYA Depot	9	118694
		9	**118694**
English	FHYA Depot	10	91752
	uMgungundlovu	4	27240
		14	**118992**
Metadata	FHYA Depot	117	5281
	uMgungundlovu	5	1573
	other	58	5758
		180	**12612**
Total		**193**	**250108**

Table 4.2: The composition of the annotation dataset. Totals for subsets and the overall set are shown in bold.

initially bootstrapped using a list of authoritative records maintained by the FHYA. The authoritative list contained the name, date of birth, date of death, type of entity and description, among other fields (see Appendix B.1), for some of the entities that appear in the FHYA collection. These records were often for people who collected material in the FHYA collection or primary sources used in documents within the FHYA collection. The original list also contained organisations such as publishers; however, only the records for people were used to populate the knowledge base. This resulted in 376 entities – the canonical entities - being loaded in the knowledge base. Other entities are added to the knowledge base during the annotation process to form the final knowledge base. This knowledge base contains records for all the people identified in the labelled documents. During the evaluation of the NED system, the canonical entities in the KB were extended by including mentions from the training data that referred to the entity. For each mention, the mention text was used as the name and the context as the description. Thus each unique entity could have multiple name-description pairs in the KB. Extending the data should increase the likelihood of the correct entity being selected as the candidate generation and entity selection module may have access to multiple examples for each entity. This reduces a risk of an entity performing poorly because the single entry has a noisy or misleading description.

4.4 Crowsdsourced Labels

We used a custom-built web app, the Annotator (described in Chapter 3), to label the document segments described in Section 4.2.1. Figure 4.1 gives an overview of how the Annotator was used to produce the labelled documents. The Annotator had three main functions. Firstly, documents had to be assigned to users. The users then used it to label mentions that appeared within the text, including adding new entities which did not appear in the authoritative list. As a measure to improve the labels' quality, each label was assigned based on the majority vote of multiple annotators. Finally, the Annotator was used to create a knowledge base, labelled document set and export the information required to pay the users.

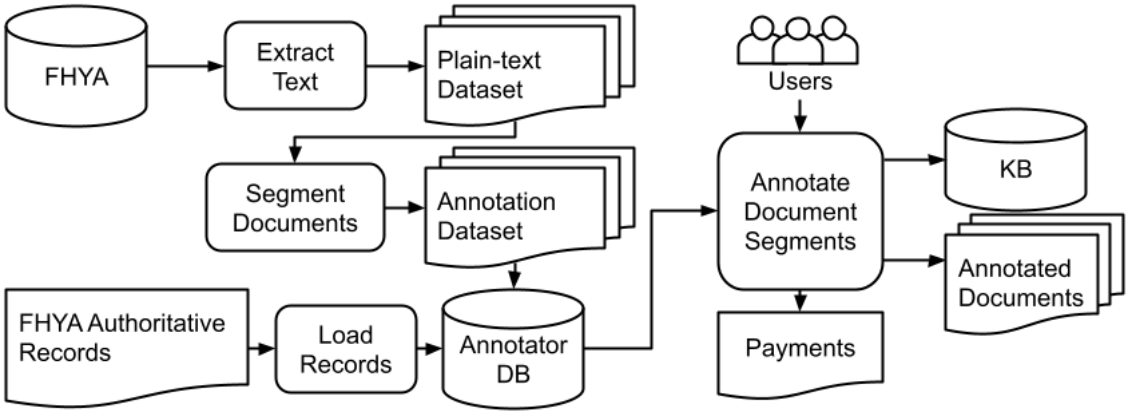

Figure 4.1: An overview of the system used to annotate documents and populate the knowledge base (KB).

4.4.1 Configuration

The Annotator allowed for the configuration of various parameters. For this experiment, the following parameters were used.

Users per segment: 3

The tags for each document segment were based on the majority vote of the tags assigned by three different users during the annotation process. Using multiple users increases the quality of the labels as it means an incorrect label should not agree with the labels created by other annotators. Thus using more users to label a document segment should result in more accurate labels. However, having redundant labels also increases the time required and the cost of annotating the documents. Three users were used as a compromise between the increased accuracy and cost of using multiple users.

Words Assigned: 2500 words

When a user signs up to the Annotator, they are assigned document segments. Since there is a considerable amount of variation in document length, the users are assigned documents until the total number of words assigned is larger than a preset number of words rather than a set number of documents to ensure a fair distribution of work among users. For this project, the number of words was set to 2500. The value for this parameter was selected to increase the chances of a user creating enough links for their payment to be above the minimum transaction size for the payment mechanism while not requiring too much time that people are dissuaded from volunteering. The minimum transaction size is ZAR 20, and based on a small sample of annotated data, there are approximately 30 words per link. Thus, since the users were paid ZAR 0.60 per link created, they should receive a payment of at least ZAR 50.00.

Unassigning Documents

As described in Chapter 3 documents were unassigned from inactive users after a period of time. For this experiment, users were considered inactive if they had not performed any activities for more than an hour. Documents with no annotations were unassigned after three hours. However, if the document has been partially assigned but not completed, it would only be unassigned after five days.

4.4.2 Payment

The participants were paid ZAR 0.60 per link. Assuming that users could annotate 1.5 mentions per minute (based on annotating a sample of the data) the user would earn around ZAR 55 per hour. However, they were only paid for a link if at least one other user made the same link between the mention and entity. This should encourage users to create as many links as possible without rewarding superfluous links. It was hoped that this approach would result in more links being created than if users were paid per word or document while resulting in more accurate links than if users were paid per link created. The payments can be viewed on the system to determine how much to pay each user. As with the annotations, a JSON object is used to export the payment information. Each user has a separate field containing the unique user id, the number of links created that agree with the majority of annotators and a field showing the entity links the user created.
Users were paid using the e-wallet service provided by First National Bank (FNB) to send voucher codes via SMS. This payment method was used because the users' bank details did not need to be collected and stored. Once the annotation process had been closed, the payment details were exported as described in Chapter 3. The exported data included the number of correct links that the user had created (where the link was to the same entity as the majority of links) and the user's cellphone number. The amount to pay the user was calculated based on the number of correct links created. Users were then sent vouchers for the amount due via FNB's e-wallet service. The vouchers allowed the money to be collected from an FNB ATM or one of several retail outlets making it easy for participants to collect their payment.

4.5 Final Annotated Data

The annotators were recruited using an invitation sent via email to a mailing list for recruiting participants in research projects from students at the University of Cape Town. The email required that participants were literate in isiZulu and had a valid South African cellphone number so they could be paid. A total of 180 people signed up to The Annotator; however, only 55 users completed the minimum number of documents or more.
The Annotator was open for approximately two and half months. However, not all of the documents had been annotated at the end of this period. Thus the final labelled dataset exported from the Annotator consisted of 675 document segments out of the original 858 document segments from the annotation dataset. The distribution of documents is shown in Table 4.3. 18.2% of the documents were from metadata records, 41.8% were from English documents, and 40.0% were from documents written in isiZulu. Thus, although the proportions are not the same as in the documents loaded into The Annotator, they were similar enough that the documents were not removed to balance the sets since this would have further reduced the amount of training data.

Type	# Documents	# words
isiZulu	270	92434
English	282	97136
Metadata	123	9766
Total	675	199336

Table 4.3: The composition documents labelled during the annotation process.

4.6 Fold Creation

We evaluated the Automatic, Hybrid and Baseline systems using 10-fold cross-validation. The document segments and labels exported from the Annotation system were split into ten folds such that each fold contained a similar number of mentions. This was achieved by using the labels to determine the number of mentions in each document. The total number of mentions was also calculated and divided by 10 to get the ideal number of mentions per fold. The document ids and the number of mentions they contained were stored as a collection of key-value pairs (`number_of_mentions: document_id`). The collection was sorted in descending order based on the number of mentions. Documents were then assigned to folds by iterating through the keys (number of mentions) and the values for each key (document_id). Document_ids were assigned to the fold with the lowest number of mentions. An example with five documents and two folds is shown in Fig 4.2. Additionally, some entities in each fold were hidden to mimic entities appearing in the text but not in the knowledge base (KB). For each fold, entities appearing in the fold were randomly selected and hidden using a uniform distribution until the entities corresponding to 10% of the mentions were hidden.

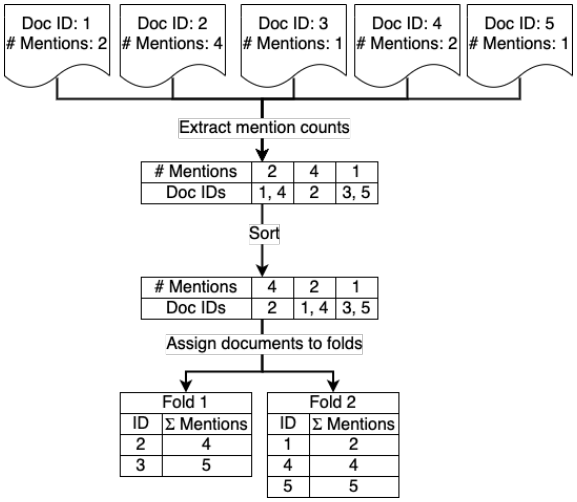

Figure 4.2: Example of splitting 5 documents into 2 folds. # Mentions denotes the number of mentions and Σ Mentions indicates the cumulative number of mentions in the fold.

4.7 Summary

Document segments of up to 350 words were selected from the FHYA collection to be labelled using the Annotator system. The Segments were selected such that all the metadata records were selected – contributing approximately 3% of the total number of words labelled. The remaining documents were selected such that there were approximately the same number of words from English (excluding metadata) and isiZulu documents. Once the document segments had been selected they were labelled based on the consensus of three isiZulu speaking users using the Annotator to produce a labelled data set and a

knowledge base. Before the labelled data set and knowledge base could be used to train the NED systems they were split into 10 folds such that each fold had a similar number of mentions. For each fold, the knowledge base record for the correct entity was hidden for approximately 10% of the mentions.

Chapter 5

Experiment 1: Baseline NED

5.1 Introduction

Once, the data from the FHYA collection had been labelled as described in the previous chapter. It was used to train and evaluate NED systems. However, a NED system's performance depends on the nature of the mentions within the dataset upon which it was trained and evaluated. Thus it is difficult to compare the performances of NED systems that make use of different datasets. This chapter describes the creation and evaluation of a baseline NED system, which was then used to benchmark the language model-based NED systems.
The most problematic entities for NED systems are infrequent and ambiguous [14]. Therefore a dataset with more infrequent and ambiguous mentions, a NED system should produce worse results than a dataset with few unambiguous names. Thus, following the proposal of Ilievski, Vossen, and Schlobach [14], we produced a probability-based baseline. The probability-based baseline was also used to determine the difficulty of performing NED on the documents from the FHYA collection in addition to benchmarking the language model-based NED systems.
The next section describes the construction of the probability-based baseline system. This is followed by a description of the different configurations for the baseline system that were created and how they were evaluated in Section 5.3. Section 5.4 contains the results of the evaluation of the baseline NED system. Lastly, a summary of the chapter is provided in Section 5.5.

5.2 Baseline Architecture

The baseline NED system used a probability-based approach by linking the entity to the mention it was most often linked to in the labelled dataset created by human annotators, as shown in Figure 5.1. Thus a particular mention will always be linked to the same entity regardless of its context. Similar systems have been used by Eshel et al. [73] and Ilievski, Vossen, and Schlobach [14] as they can deal with the simplest form of NED - common and unambiguous men-

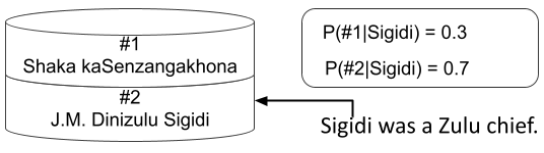

Figure 5.1: Overview of the baseline system. In this case the baseline system assigns the mention Sigidi to the wrong entity.

tions. Thus a comparison with the baseline should indicate how well a NED system can disambiguate more complex cases, while the results from the baseline give an indication of how much ambiguity exists in the dataset. The baseline system consists of two modules: mention detection and entity selection. The mention detection module identifies mentions within a passage of text. Once the mentions have been identified, the entity selection module links the mentions with entity records from the knowledge base (see Fig 5.2).

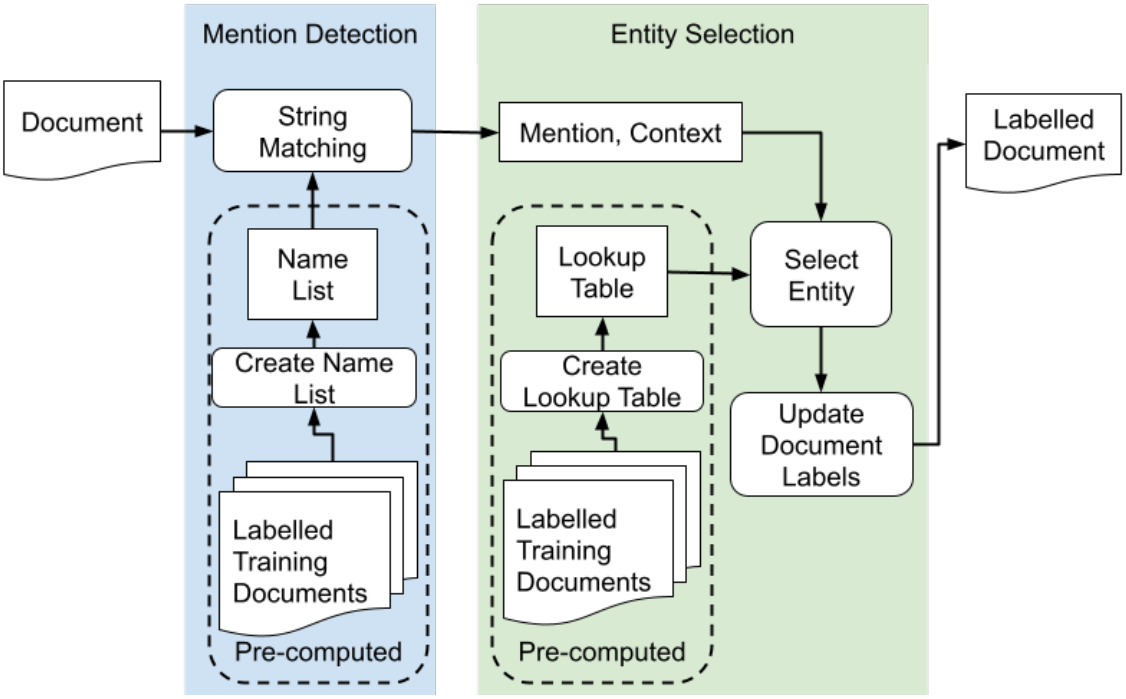

Figure 5.2: High-level overview of the design for the baseline systems.

5.2.1 Mention Detection

Mention detection is performed based on string similarity between the tokens in the input text split on white space and a list of names. The list of names is produced based on the entities labelled in the training data. Initially, all the mentions from the training data are added to the names list. The list is then expanded by splitting each name based on white space and adding the resulting parts to the list. Thus if the names list was `["James Stuart"]` it would become `["James Stuart", "James", "Stuart"]`. Finally, each of the names is converted to lower case and duplicates are removed.
The similarity between each token and name is calculated using the Levenshtein ratio[1]. The Levenshtein ratio was used rather than Levenshtein distance as it also takes the length of the strings into account. The Levenshtein ratio returns a value between 0 and 1, where a Levenshtein ratio of 1 means the strings are identical.
Mentions are identified by finding the longest span that matches one of the names in the names list. Initially, a span consists of a single token converted to lower case. The similarity between the span and each of the names in the names list is then calculated. A span is considered to match a name if the similarity is greater than 0.9. If a match occurs,

[1] `https://pypi.org/project/python-Levenshtein/`

the next token is converted to lower case, added to the sequence, and checked against the names list. This process is repeated until the sequence no longer matches one of the names in the names list. The mention is then the span excluding the final token. Once the mentions have been identified in all documents, they are passed on to the Entity Selection module.

5.2.2 Entity Selection

Entity selection is performed using a simple lookup table with entries for each mention, and its corresponding entity. During training, the baseline system is given access to the labels for the training portion of the labelled dataset. These labels are used to determine the conditional probability of an entity being referred to by a particular mention. This is calculated as $m \cap e$. Where $m \cap e$ is the number m of times the mention m refers to entity e. The final table is then constructed by pairing each mention with the entity id to which it was most likely to refer.

5.3 Method

Several baseline systems were created by varying the parameters for the string matching algorithm used by the mention detection module. For each module, the names list could either be used directly from the knowledge base or expanded as described in Section 5.2.1. Additionally, the threshold for a span of tokens to be considered a match was set to either 0.80, 0.90 or 0.95. Two additional baseline systems were produced to explore the effects of the mentioned detection module on the system's performance and gauge the level of ambiguity of the mentions within the dataset. The first additional baseline system (BL-S) used a pre-trained commercial English NER system – Spacy[2]. The Spacy model uses Spacy's en_core_web_sm[3] model for mention detection. Thus the effectiveness of using string matching could be evaluated. The second additional baseline system (BL-L) used the labels for the dataset for mention detection. This allowed the performance of the entity selection module to be measured independently of the mention detection module. All of the baseline systems described above used the same entity selection module. Thus, any differences between the baseline systems were due to the mention detection module.
Each baseline system was evaluated using 10-fold cross-validation over the dataset using the folds created in Section 4.6. For each iteration of the cross-validation process, a single fold was selected as the hold-out fold. The probability lookup table was derived from the nine remaining folds. The baseline was then evaluated on the hold-out fold. Additionally, to mimic out-of-KB entities, entities in the knowledge base (KB) were masked in each iteration such that the entity for approximately 10% of the mentions was not in the knowledge base. The final metrics were calculated as the average of the metrics for each of the 10-folds.

5.3.1 Result Evaluation

For each mention, the following information was recorded and used to evaluate the system's performance: the mention text, its context, its position within a document, the doc-

[2] https://spacy.io/
[3] https://spacy.io/models/en

ument id, the candidate list and the final candidate. This information was also used to evaluate the performance of each of the modules. The metrics and settings used to evaluate the systems are described below.

Metrics

The precision, recall and F1 score were calculated for each fold then aggregated to produce the final score for each metric. High precision was required as the system is only helpful if the mentions are linked to the correct entity; otherwise, they would be harmful to downstream tasks. A high recall was required as the system is most beneficial to downstream tasks when all the entities are identified within the text. Thus the F1-score was used as the primary metric for measuring the system's performance as it is the harmonic mean of the precision and recall. Another reason for using the F1 score was that it has often been used to evaluate NED systems in previous studies.
The accuracy of the entities predicted by the baseline for the mentions was calculated in two modes – All and Entity. The All accuracy was calculated as the number of mentions which were assigned to the correct entity divided by the total number of mentions. In contrast, the Entity accuracy is calculated as the number of mentions detected by the system where the predicted entity and label were the same, divided by the total number of mentions detected by the mention detection module. Thus, the Entity accuracy mode does not consider mentions that the mention detection system failed to detect. The Entity accuracy was used as it gives a simple and easily interpretable metric for the performance of the candidate generation and entity selection modules.

Strict Evaluation

In the CLEF HIPE campaign[74], the precision, recall and F1-score were calculated in strict and inexact modes. The strict metrics require the labels to be correct, and the mention boundaries must line up perfectly with the labelled data. In contrast, the inexact metrics, referred to as fuzzy in the CLEF HIPE campaign, only require some overlap between the predicted and actual mention boundary to be considered a match. For evaluating the NED models, the strict mode was used for calculating the micro, macro and F1 scores as these were calculated at a token level where the results should match the labelled data as closely as possible. In contrast, the inexact mode was used for the accuracy measures as this gave a better reflection of the performances of the candidate generation and entity linking models when given a mention rather than the correctness of the mention boundaries.

Micro and Macro Settings

Each of the metrics were calculated in both micro and macro settings. The macro metrics were determined by averaging the scores of each of the documents in the test dataset. In contrast, the micro metrics were calculated by treating the test dataset as a single document. Thus the micro metrics tended to be more stable than the macro metrics, which can be heavily influenced by a few documents with very high or low scores. Comparing the differences between the metrics in a micro and macro setting indicates the distribution of the ease of disambiguation. The primary comparison between models was based on the micro metrics. Thus the main metric for comparing the models was the micro F1 score.

5.4 Results

Model	Mention Detection		Macro-P	Macro-R	Macro-F1	Micro-P	Micro-R	Micro-F1
	Expand	Threshold						
BL1		0.95	0.007	0.043	0.012	0.009	0.046	0.015
BL2	✓	0.95	0.360	**0.285**	**0.318**	0.395	0.292	0.335
BL3		0.90	0.056	0.139	0.079	0.090	0.237	0.130
BL4	✓	0.90	0.434	0.249	0.316	0.504	0.267	0.348
BL5		0.80	0.316	0.265	0.287	0.409	**0.308**	**0.351**
BL6	✓	0.80	**0.578**	0.204	0.301	**0.628**	0.193	0.295
BL-S	English Spacy		0.213	0.268	0.237	0.184	0.158	0.170
BL-L	Label		0.695	0.726	0.709	0.702	0.758	0.729

Table 5.1: Micro and macro precision, recall and F1 scores averaged across the 10-folds. The micro statistics treat each fold as a single document when calculating each metric while the macro metrics are calculated as the mean of the metric for each document. The *Expand* column indicates if the plain list of names was used or if the list was expanded by including all sub-parts of each name as separate entries. The *threshold* is the minimum similarity required between a token and a name for the token to be considered to match the name.

The Baseline system was run on each of the ten folds described in Section 4.6. The final results are taken as the mean of the metrics from each fold. The micro and macro precision, recall and F1 scores are shown in Table 5.1. The results for models BL1-6 correspond to non-neural baseline systems, which used string matching for mention detection.

Based on the results in Table 5.1 the best instances of the baseline system were BL-2 and BL-5 as these systems had the highest recall and F1-scores in the macro and micro settings, respectively. Thus these two systems were used to benchmark the performance of the language model-based systems.

The results depend on both the mention detection and entity selection modules for each system. Since each of the systems used the same entity selection module, any differences in performance are due to the mention detection module. For the Baseline instances which use string matching for mention detection (BL1-6), both the micro and macro F1 scores are low. A comparison between these systems and BL-L system indicates that the low scores are primarily due to the mention detection module, as when the mentions are detected correctly (by using the labels), the scores increase significantly for all metrics. However, several of the string-based baselines outperformed the BL-S system. This indicates that detecting mentions from within the documents is challenging, yet the string-based method does a reasonable job for a simple system. The accuracy of the entity ids predicted for each of the mentions was calculated to explore how the entity selection module performed. The BL-1 model had the highest accuracy at 40.59%. However, where the mention corresponded to an entity (correctly detected mentions), BL-3 system had the highest accuracy of 92.76%. This indicates that entity selection in the baseline is accurate for the mentions that the mention detection module can find. As can be seen from Fig 5.3, almost all of the models (except BL-1) were more accurate than the baseline using labels for mention detection (BL-L) when only correctly detected mentions are taken into consideration. This suggests that the correct mentions that the string-based mention detection system found tended to be easier to disambiguate.

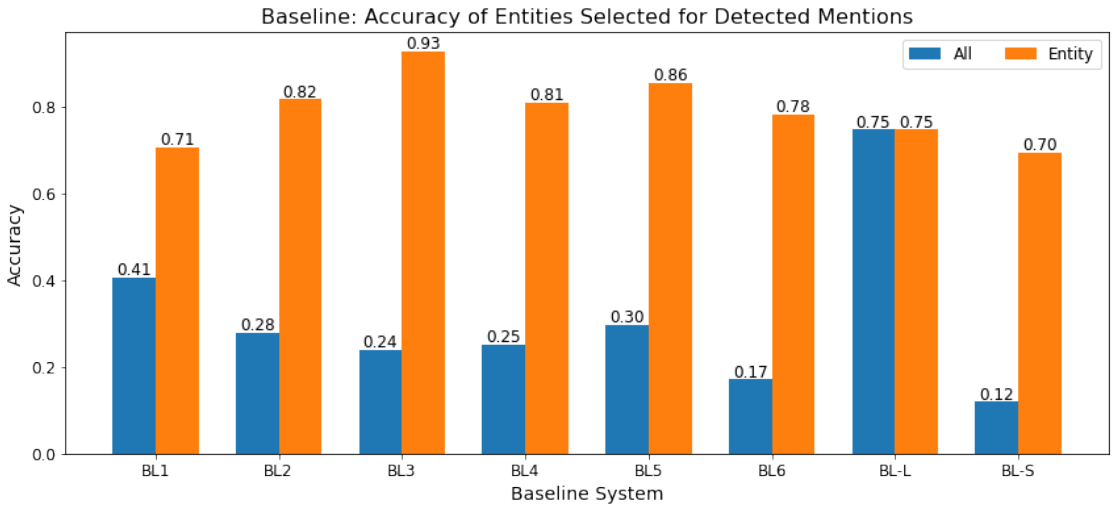

Figure 5.3: Accuracy for entity labels assigned to mentions extracted by the mention detection module. *All* includes all the mentions while *Entity* only includes mentions which were labelled with an entity id.

5.5 Summary

A simple baseline system was constructed because NED had not been performed on documents from the FHYA. The baseline NED system used string matching with a list of names for mention detection and a lookup table for entity selection. People's names were detected by comparing text spans to a list of names extracted from the knowledge base using the Levenshtein ratio as a similarity measure.

For entity selection, a lookup table was constructed from the labelled training data for each fold such that the key was a mention and the value was the entity most frequently associated with the mention in the training documents. Using a lookup table meant that the baseline system would always assign the same entity to a particular mention. Thus, the system would not be able to deal with ambiguous names and could indicate the difficulty of disambiguating named entities from the text.

The baseline was evaluated based on the micro and macro F1-scores using 10-fold cross-validation. The models performed poorly, largely due to the string-based matching system. When evaluated using the labels for mention detection, the system performed better but still only achieved a micro F1-score of 0.729. In this setting, all errors were due to the entity selection system. Thus the F1-score indicates the presence of a significant number of ambiguous names within the text. Baseline instances BL-2 and BL-5 were selected to be used to benchmark the language model-based NED systems as they produced the highest macro and micro F1-scores, respectively.

Chapter 6

Experiment 2: Automatic NED

6.1 Introduction

Language models have been successfully used in a wide variety of Natural Language Processing (NLP) tasks [11] including Named Entity Disambiguation (NED) [19]. However, much of this research is done on high resource languages. This experiment seeks to answer the first research question from Section 1.2 – "*How accurate are transformer-based language models for NED applied to text from the 500 Year Archive?*". Thus, we developed the Automatic NED system and benchmarked its performance against the performance of the baseline NED systems described in the previous chapter. While the baseline systems dealt well with common and unambiguous entities, they cannot deal with ambiguity. In a baseline system, if a mention could refer to two entities, the same entity will be chosen every time. In order to improve NED with more ambiguous names, the Automatic NED system used a language model-based approach such that the entity name and the context within which the name appears can be used for disambiguation.
The following section describes the algorithms used to produce the Automatic NED system. Section 6.3 describes the methods used to train and evaluate the Automatic NED system. The results are presented in Section 6.4. Finally, the second experiment is summarised in Section 6.5.

6.2 Automatic NED Architecture

The Automatic NED system makes use of three modules: mention detection, candidate generation and entity selection, as shown in Figure 6.1. As in the baseline system, the mention detection module identifies mentions within the text. These mentions are then passed to the candidate generation module, which creates a high recall list of entities to which the mentions could refer. Finally, the entity selection module links each mention with an entity from the mention's candidate list.

6.2.1 Mention Detection

Mention detection was performed using an XLM-R based NER model (CoNLL-Zu-NER) for both English and isiZulu documents. The model was first trained for token classification on the CoNLL03 training dataset [75]. The CoNLL03 dataset contains data from Reuters news articles using the IOB format to indicate entities [75]. The model was then

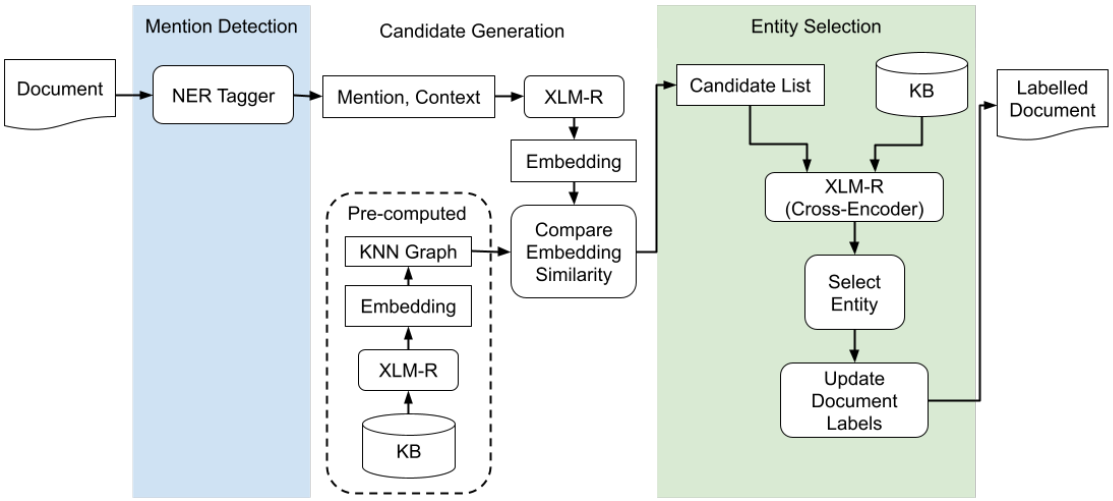

Figure 6.1: An overview of the design of the Automatic system.

further trained on the government domain isiZulu dataset to increase its performance on South African names. Finally, it was fine-tuned on the training data from the FHYA data set.

The CoNLL-Zu model was fine-tuned rather than XLM-R as the training dataset was relatively small (approximately 750 documents). Thus the models can benefit from general concepts learnt in the larger isiZulu and English corpora while picking up patterns specific to the documents in the FHYA collection.

Since the NER models were based on XLM-R, they could only take 512 tokens as an input. However, many of the passages in the FHYA collection contained more than 512 tokens since XLM-R uses a word-piece tokeniser. The word-piece tokeniser could tokenise each word into multiple tokens. To prevent more than 512 tokens from being used as input to the NER models, the text was split into chunks of 128 words. Each chunk had an overlap of ten words with the chunks to its left and right to ensure that there was a context window of at least ten words for each token the NER system processed. A ten-word context window for consistency with the bi-encoder and cross-encoders discussed below. A word-level context window was used rather than a token-level context window as the tokeniser was only trained on English data; thus, it is unclear how closely the isiZulu tokens align with semantically meaningful linguistic tokens for isiZulu texts.

6.2.2 Candidate Generation

Once the mentions had been identified, they were passed to the candidate generation module along with the context within which they appeared. The context consisted of the text within N words to the left and right of the mention; for this experiment $N = 10$. The Automatic model then used a bi-encoder architecture to compare the similarity between the mention and entities in the knowledge base. This architecture was used as Wu et al. [20] found it to be highly effective for making similarity comparisons between texts while performing candidate generation. The bi-encoder architecture used the SBERT [59] architecture to independently generate a vector representation (embedding) for the mention and entity. The similarity between the two embeddings was then measured using cosine similarity. The mention embedding was produced based on the mention text and its context in the form: *mention* `[ENT]` *context* where `[ENT]` is a special token separating a

mention from its description. Similarly, the same model was used to generate an embedding for each entity in the knowledge base. Since the text within the FHYA comes from multiple languages, the multilingual language model XLM-RoBERTa base (XLM-R) [49] was used as the language model for the SBERT model. XLM-R was used as previous research has shown that it can learn embeddings across languages [49]. SBERT provides an architecture for producing sentence level embeddings from language models. Language models such as BERT provide an output layer consisting of token embeddings and a `[CLS]` embedding (used for classification tasks). Previous research has made use of either the `CLS` embedding or an aggregate of the token embeddings as an embedding for the input text. However, SBERT was used as it has been shown to produce more semantically meaningful outputs [59]. Thus using the embeddings generated by SBERT should mean that the similarity between the embeddings corresponds better with the similarity between the actual text.
Since the entity embeddings did not change, they were pre-computed to reduce the runtime for the system. The candidates for each mention were retrieved using a k-nearest neighbour (KNN) search based (implemented by scikit-learn[1]) on cosine similarity using the mention embedding as the query item. The KNN search returned the 30 most similar entity embeddings as possible candidates. The candidate list is then filtered to remove candidates where the cosine similarity was less than 0.5 or until only the top six candidates remain. Thus the final candidate list always contained between 6 and 30 candidates. A value of 0.5 was chosen as, based on earlier tests, correct candidates usually have high cosine similarity scores (> 0.9). Thus using a threshold of 0.5 should ensure a high recall list while reducing the number of candidates. However, at least the top six candidates were always passed on to the candidate generation module even if their similarity was below 0.5. This was done to reduce the likelihood of the system incorrectly assigning a NIL entity as the cross-encoder should be better at discerning the correct entity than the bi-encoder. The entity selection model then determined if any of the candidates were suitable or if the mention refers to none of the entities in the knowledge base. Six candidates were used as in initial tests the actual entity appeared in the top 6 candidates for 99% of the mentions. Mentions which refer to entities that do not appear in the knowledge base (out-of-KB entities), are dealt with by maintaining a separate list of entities that have been created by the entity selection module. Once the initial candidate list has been retrieved, the cosine similarity between the mention embedding and each added entity embeddings is calculated. If the similarity is higher than that of the least similar candidate, the entity is added to the candidate list.

6.2.3 Entity Selection

Finally, entity selection was performed. The entity selection module used a cross-encoder to compare the entities from the candidate list with the mention. A cross-encoder takes two pieces of text as input; the language model then tries to determine the extent to which they are similar. Similar to the bi-encoder used for candidate generation, the cross-encoder was implemented using SBERT with the language model XLM-R. Previous research has shown that cross-encoders are more accurate than bi-encoders [20]. However, unlike bi-encoders, the embeddings cannot be pre-computed, so the cross encoder tends to be a lot slower [20]. Thus the cross encoder is only used for entity selection as the number of entities it needs to compare with the mention is much smaller than all of the

[1] `https://scikit-learn.org/stable/modules/neighbors.html`

entities in the knowledge base. Once a suitable entity has been found, the mention text is labelled with the entity's ID from the knowledge base. The system dealt with out-of-KB entities using a threshold-based approach; if the similarity between a mention and the most similar entity was below 0.03, a new entity was created and added to the knowledge base.

6.3 Method

The Automatic NED system was evaluated using 10-fold cross valuation. The same folds were used as in the baseline experiments in the previous chapter. Separate models were trained for each fold for each of the modules comprising the Automatic system to ensure that the test data used to evaluate the model had not been seen by any part of the system beforehand.

6.3.1 Training

Each of the modules making up the Automatic NED System were trained using labelled documents from the FHYA collection in a supervised manner. In order to effectively utilise the GPU provided by the UCT HPC cluster[2], a separate NER, bi-encoder and cross-encoder model were trained for each fold using the training documents for the fold. Models were trained separately as the GPU would run out of space if it was used for multiple tasks (either training or evaluating models). This resulted in 10 separate NER, bi-encoder and cross-encoder models being trained. During the evaluation of the NED system on the test documents in each fold, the corresponding models were used for mention detection, candidate generation and entity selection.

Mention Detection

The mention detection module used an XLM-R based Named Entity Recognition (NER) model (CoNLL-Zu-NER) pre-trained on the CoNLL03 data set, and an isiZulu dataset trained as described in Section 6.2.1. The CoNLL-ZU-NER was used as a base NER model, which was then fine-tuned on the training partition of the particular fold for token classification. Each of the models were implemented as instances of the XLMRobertaForTokenClassification[3] objects provided by the HuggingFace library. Thus facilities provided by the Huggingface library were used to train the models.

Candidate Generation and Entity Selection

The bi-encoder and cross-encoder were trained using triples, which contained two mentions in context and an integer indicating if they referred to the same person. The triples took the following format (*mention1* `[ENT]` *context1*, *mention2* `[ENT]` *context2*, 1). These triples were generated from mention-context pairs extracted from the training data. The context consisted of the span text within ten words to the left and right of the mention. The triples are classified as positive if both mention-context pairs refer to the same person; otherwise, they are classed as negative. The triples were then created by selecting an

[2] `http://hpc.uct.ac.za/`

[3] `https://huggingface.co/docs/transformers/model_doc/xlmroberta#transformers.XLMRobertaForTokenClassification`

entity from the knowledge base using a uniform random distribution. A random number was then generated to determine if a positive or negative set of triples should be created based on a pre-configured threshold; the threshold was set at 0.5 for this experiment. If a positive set of triples was required, up to 100 triples were selected where both mention-context pairs refer to the entity. Additionally, for each positive triple generated, a hard negative triple was created with a probability of 0.4, with an equal probability of the hard negative being created for the first or second mention in the triple. The hard negative examples were generated by finding mentions that refer to another entity but have a similar name to the corresponding mention based on string similarity measured using the Levenshtein ratio. Incorporating hard negatives should prevent the models from relying only on the entity name and mention similarity. For negative examples, a mention-context pair for another entity was selected at random. This process was repeated until the required number of triples were generated.

Reduced Training Data

The effect of the amount of training data was investigated by training each of the components of the Automatic NED system using considerably less data. The mention detection module was trained for each fold using half of the documents from the FHYA collection in the training partition for the fold. The documents used were randomly sampled using a uniform distribution from all the documents available for training the model. Once the documents had been selected, the mention detection system was trained in the same way as described in Section 6.2.1.
Similarly, both the bi-encoder used for candidate generation and the cross-encoder used for entity selection were trained on half the triples (4000 compared to 8000). The triplets were generated using the method mentioned above for the 8000 triples.

6.3.2 Evaluation

Once the models had been trained, the NED systems were evaluated using 10-fold cross-validation on the folds created in Section 4.6. The performance of the Automatic system was evaluated using the same techniques and metrics as in the baseline. The system was evaluated based on the recall, precision, and F1 scores in micro and macro settings. The final score for each metric was calculated as the average of each of the 10-folds.

6.4 Results

Model	MD	CG+ES	Macro-P	Macro-R	Macro-F1	Micro-P	Micro-R	Micro-F1
BL2	F	F	0.361	0.285	0.318	0.395	0.291	0.334
BL5	F	F	0.316	0.264	0.287	0.410	0.307	0.350
BL-N	F	F	0.655	0.652	0.658	0.678	0.668	0.672
AT	F	F	**0.744**	**0.732**	**0.737**	**0.731**	**0.722**	**0.726**
AT-R	R	R	0.705	0.671	0.687	0.698	0.674	0.685
AT-R-CG+ES	F	R	**0.707**	**0.696**	**0.701**	**0.702**	**0.693**	**0.696**
BL-Label	L	F	0.695	0.726	0.709	0.702	0.758	0.729
AT-Label	L	F	**0.797**	**0.823**	**0.810**	**0.773**	**0.821**	**0.796**

Table 6.1: A comparison of micro and macro precision, recall and F1 scores between the Automatic System (AT) and the best performing baseline systems (see Table 5.1). The BL-N system uses the same mention detection module as the automatic model (AT). The BL-Label shows the results for the baseline system using labels for mention detection. The MD column indicates whether the mention detection module was trained on the full dataset (F), trained on the reduced dataset (R) or used the data labels (L). Similarly, the CG+ES column shows the data used to train the candidate generation and entity selection modules. The highest values are shown in bold.

6.4.1 Comparison with the Baseline Models

The difficulty of performing NED varies across document collections depending on the level of ambiguity in the mentions. Thus the Automatic system was compared with a baseline system to determine how well it performed compared to simple non-neural techniques. The results of this comparison are presented in Table 6.1. The Automatic NED system was compared with the best performing baseline systems as discussed in Section 5.4. BL-N was also included as it used the same Mention Detection module as the Automatic NED system. Thus any differences between the two models are due to the Entity Linking modules (Candidate Generation and Entity Selection).

The Automatic NED system performs substantially better than all of the baseline models except the model that uses labels for mention detection (BL-Label) where the micro F1-score is lower, but the macro f1-score is higher. The Automatic NED system also performs better than the BL-N model across all metrics, indicating that the entity linking modules used by the Automatic system can utilise the context around mentions to make more accurate links to entities.

Similarly to the baseline models, there was a considerable improvement across all metrics when the labels were used for mention detection. This shows that improvements to the mention detection module as well as the candidate generation and entity selection modules could lead to substantial improvements in the overall system.

The accuracy of the predictions made by the Automatic system is far better than those made by the baseline systems as can be seen from Figure 6.2. As in the baseline models, labels were used for mention detection, the accuracy of the entity labels for *Entity* mentions decreases. This supports the findings from the baseline system evaluation that mentions that are harder for the mention detection module to detect are often more challenging for the Automatic NED system to disambiguate.

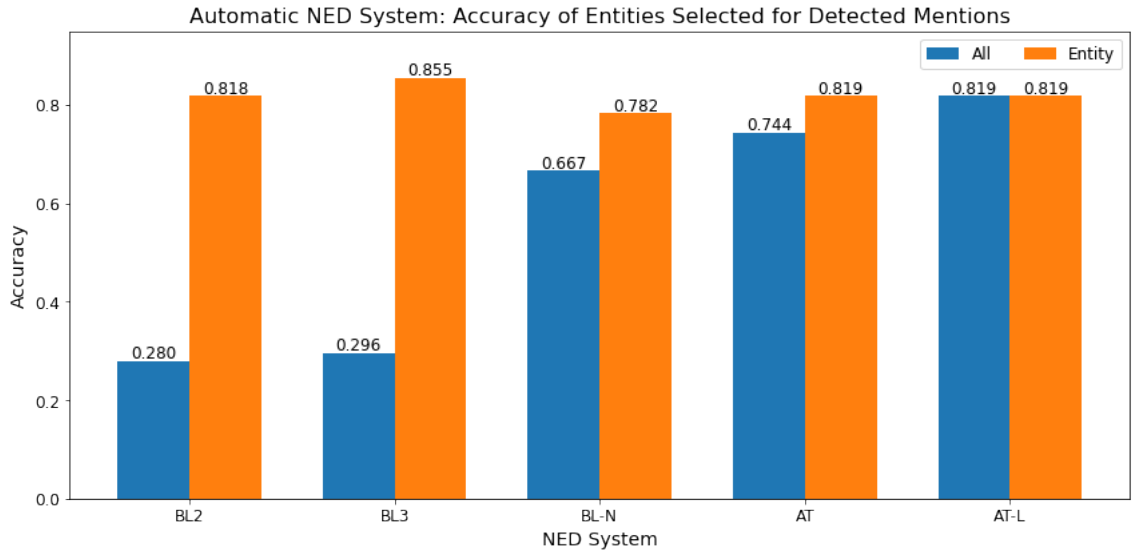

Figure 6.2: Comparison between the accuracy of labels assigned by the baseline or automatic systems. *All* includes all the mentions while *Entity* only includes mentions which were labelled with an entity ID.

6.4.2 Performance by Document Type

The document collection could be split into three broad categories: English, Zulu, and metadata, where the Zulu documents are written in isiZulu. In contrast, the English and metadata documents are written in English. In general, the metadata documents are shorter and more structured than the English documents as they follow a set template and all give descriptions of some item from the FHYA. From Figure 6.3 it can be seen that in general, the English and Metadata documents tend to have higher F1 scores than the Zulu documents.

The accuracy of the labels assigned to the detected mentions shows the same trends as the F1 scores, with the mentions found in metadata documents having the highest accuracy. Similar to the F1 scores, the accuracy for mentions in English documents was slightly lower but still higher than those in Zulu documents (see Fig 6.4).

It is interesting to note the differences in the relative performance of the candidate generation and entity selection systems for documents from the different categories (see Fig. 6.4). If the top candidate from the candidate generation module were selected as the final entity for the English and metadata documents, it would have performed close to or slightly better than the Entity Selection module. However, for the Zulu documents, the top candidate was the correct candidate less often than the entity selected by the Entity Selection module. This shows that the bi-encoder architecture is not as good at determining the similarity between mentions as the cross-encoder for isiZulu texts but has similar performances for English texts. The bi-encoder and cross-encoder were both based on the same language model (XLM-R) and trained on the same data. In both cases, the language model was pre-trained in English then trained using SBERT on mentioned pairs extracted from the training partition for each fold. However, the bi-encoder is used to produce embeddings that can be compared using cosine similarity, while the cross-encoder acts as a classifier for a pair of mention-context pairs. Thus the cross-encoder has access to each token from both mention-context pairs, enabling better performances, with restricted access to training data.

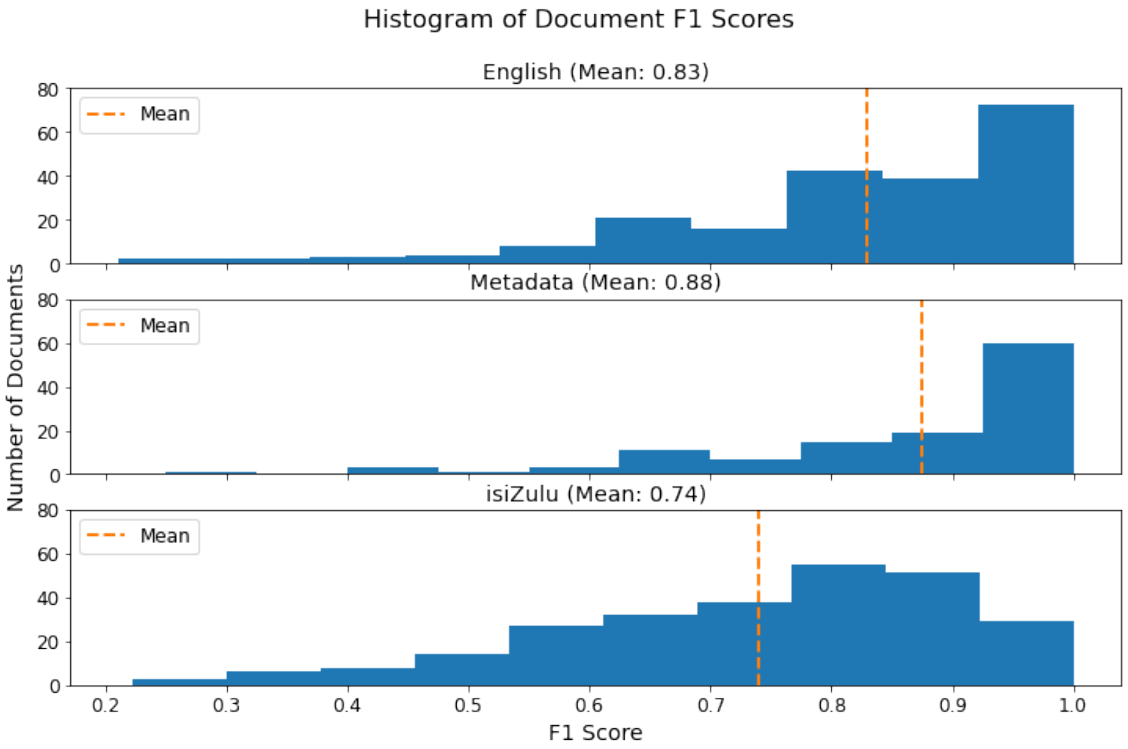

Figure 6.3: Histograms of the F1 scores for the Automatic NED system on the English, metadata and Zulu documents.

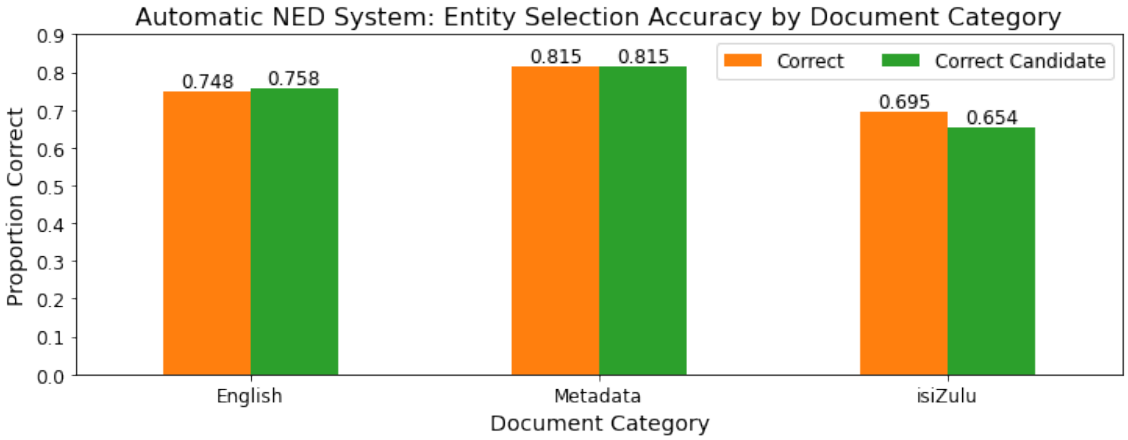

Figure 6.4: Accuracy of entity labels assigned to mentions detected by the Automatic NED system split by language. *Correct* shows the proportion of mentions where the correct label was predicted while *Correct Candidate* shows the proportion of labels where the top candidate was the same as the label for the mention.

6.4.3 Module Evaluation

The metrics in the previous sections evaluate the system's performance as a whole. However, the system is comprised of three separate modules, such that the performance of one module limits the performance of subsequent modules. The Mention Detection module essentially sets an upper bound on the performance of entity linking modules (Candidate Generation and Entity Selection) because the modules cannot correctly link mentions that are not detected. Similarly, the Entity Selection module cannot identify the correct en-

tity unless it appears in the candidate list produced by the Candidate Generation module. Thus it is vital to evaluate each module as part of evaluating the system as a whole.

Mention Detection

Document Type	Precision	Recall	F1
English	0.948	0.941	0.945
Metadata	0.969	0.972	0.970
Zulu	0.813	0.929	0.867
All	0.893	0.942	0.906

Table 6.2: Macro Precision, Recall and F1 Scores for the mention detection module.

The main component of the mention detection module was a Named Entity Recognition (NER) system that tagged each token in the input text using the IOB format. The average precision, recall, and F1 scores across each type of document were calculated based on the IOB tags; the results are shown in Table 6.2. The mention detection module has reasonably high precision and recall at 0.893 and 0.943, respectively, resulting in an F1 score of 0.906. This shows that the mention detection system was effective at extracting mentions within the text. The high recall means that most of the mentions within the text were extracted. However, the lower precision means that a number of spans of text that the system detected as mentions were not mentions. Evaluating the system on the different types of documents (see Table 6.2) shows similar patterns to the overall results with the best performance on metadata documents followed by English then Zulu documents. The precision for Zulu documents is particularly low relative to the precision for the English and metadata documents (see Table 6.2). This indicates that there may be forms of isiZulu words where the NER system struggled to determine if they referred to a person.

Candidate Generation

Document Type	Contained (%)	Mean Position
English	77.82	1.146
Metadata	83.54	1.133
Zulu	71.95	1.567
All	74.07	1.433

Table 6.3: Percentage of mentions for which the candidate list contained the correct candidate and the mean position in the candidate list of the correct candidate when it was present.

The primary purpose of the Candidate Generation module is to provide a high recall list of entities for the Entity Selection module. Table 6.3 shows the proportion of mentions for which the Candidate Generation module included the correct entity as one of the candidates. Typically, the candidate generation module had a fairly high recall with the candidate lists for approximately three-quarters of the mentions containing the correct candidate. As in previous sections, the system performs best on the metadata documents and worst on the Zulu documents; however, differences in the performances on the different types of documents for both the percentage contained and mean position were smaller than the differences for the Mention Detection module.

The candidate list was ordered by the similarity scores between the mention and candidate entity. For all three document types, the correct candidate usually appeared very close to the front of the list, indicating that the candidate generation module was quite confident where the correct entity was found. This also indicates that using a larger candidate list is unlikely to have much impact on the Automatic NED system's performance.

Entity Selection

Document Type	Accuracy
English	0.961
Metadata	0.975
Zulu	0.965
All	0.965

Table 6.4: Accuracy of the predictions made by the Entity Selection module where the correct entity was present in the candidate list.

The Entity Selection module's performance was evaluated using the accuracy metric described in Sections 5.3.1. However, the mentions were filtered beforehand only to include mentions where the correct entity appeared in the candidate list. The results in Table 6.4 show a similar pattern to the previous section, except the differences between the different document types have become even smaller. This indicates that the Entity Selection module is performing well. However, these high scores are likely due in part to some of the more difficult cases being filtered out by the Candidate Generation module. Since the metrics do not show how well the Entity Selection module would perform on the mentions where the Candidate Generation module did not include the correct entity in the candidate list, these mentions are likely to have been more difficult to link. However, the Entity Selection module is far slower than the Candidate Generation module, making it infeasible to run the Automatic NED system without the Candidate Generation module. This is also likely to be the reason why the entity selection was more accurate on Zulu documents than English documents. Since the candidate generation module performed better on English documents, the more challenging mentions for the entity selection system to deal with were not filtered out to the same extent.

Reduced Training Data

The results for the AT-Reduced and AT-R-CG+ES models (see Table 6.1) show that reducing the amount of training data available to the Automatic NED system has a substantial negative impact on the system's performance. However, both models trained on the reduced dataset still performed better than the baseline models. (Bl-2, Bl-5 and Bl-N).
The differences between the performances of the BL-N and AT-R-CG+ES systems indicate that even when trained on a much smaller data set, the bi-encoder and cross-encoder are able to utilise a mention's context when linking the mentions. Both systems use the same mention detection module; thus, the differences in results are likely because the BL-N module relies entirely on the surface form of the mention for disambiguation while the bi-encoder and cross-encoder use the context as well.
A comparison between the AT-Reduced and AT-R-CG+ES systems results indicates that reducing the amount of training data had a greater impact on the entity linking mod-

ules (candidate generation and entity selection) than on the mention detection module. Switching the mention detection module trained on half the data for the model trained on the entire training set increases the macro and micro F1-scores by 0.014 and 0.009, compared to increases of 0.036 and 0.030 when the entity linking modules trained on the entire training set were used. This is likely because the mention detection module was pre-trained on the CoNLL and isiZulu government domain datasets (see Section 6.2.1). Thus, it may have already been exposed to many of the linguistic concepts found in the FHYA collection. Therefore, fine-tuning the model with less data would have a smaller impact on the model. In contrast, the training data for the bi-encoder and cross-encoder came from the FHYA dataset.

Document Type	Precision	Recall	F1
English	+0.050	-0.050	+0.024
Metadata	+0.042	+0.013	+0.028
Zulu	+0.030	+0.26	+0.029
All	+0.035	+0.017	+0.027

Table 6.5: Change in Micro Precision, Recall and F1 Scores between the mention detection module trained on the reduced and full data sets.

The additional training of the mention detection module resulted in improvements in the F1 score for all three document types (see Table 6.5). However, for the English and metadata documents, the improvements in the F1-score were primarily due to improvements in precision. The increase in recall was much smaller than the increase in precision for the metadata documents, and the recall decreased for English documents. This indicates that the additional training improved the models' ability to distinguish words that were previously mistaken as names. The increases in precision and recall for the Zulu documents were reasonably similar, showing that the additional training improved the systems' ability to detect names and distinguish words that are not people names within Zulu text.

6.4.4 Hidden Entities

During the entity selection phase if the maximum similarity between a mention and entity was below 0.03 a new entry, representing a new entity, was created in the knowledge base. Thus, future mentions could be linked to the new entity. This approach was used to deal with mentions of entities which did not appear in the knowledge base. During training and evaluation entities for approximately 10% of the mentions were hidden for each fold. Ideally, the NED system would be able to identify where the entity corresponding to a mention did not appear in the knowledge base and create a new entry for it. However, for the automatic NED system, new entities were only assigned to 10.4% of the mentions whose entities were hidden. Similarly, in an ideal system, every hidden entity would result in precisely one created entity that would be linked to all the mentions that refer to the hidden entity. However, on average, mentions corresponding to a hidden entity were assigned to 2.82 different entities. While for new entities created, only 40.7% were for mentions whose entities were hidden. On average, each created entity was linked to mentions for 1.71 actual entities. These results indicate that the Automatic NED system struggled to deal with hidden entities and tended to generate too many new entities.

6.5 Summary

The Automatic NED system used three separate multilingual language models (XLM-R). Mention detection was treated as a token classification task where the IOB format was used to classify tokens as being part of a person's name or not. A language model was pre-trained for the token classification task on the CoNLL corpus and an isiZulu corpus derived from South African Government documents for the mention detection module. The language model was then fine-tuned on documents from the FHYA collection. The candidate generation module used a language model in a bi-encoder architecture to compare the similarity between mentions and entries in the knowledge base. The bi-encoder was trained on triples containing a mention and entity and a label indicating if the mention referred to the entity. Finally, the entity selection module used a language model in a cross-encoder architecture to determine if a mention in context referred to an entity with its description. The cross-encoder performed the same task as the bi-encoder; however, it took far longer but was more accurate in the previous research [20]. Thus, the bi-encoder created a high recall candidate list of potential entities that the cross-encoder used to select the final entity.
The Automatic model was evaluated based on the micro and macro F1-score using 10-fold cross-validation. Separate token classifiers, bi-encoders and cross-encoders were trained for each fold using documents from the other nine folds before being evaluated on the documents within the fold.
Compared to the best baseline models, the Automatic system performed substantially better, increasing the micro and macro F1-score by 0.237 and 0.342, respectively. However, the Automatic system did not perform as well on documents written in isiZulu compared to English documents. The recall for the mention detection for Zulu documents was particularly poor (0.135 less than for the English documents and 0.156 less than for metadata documents). The percentage of mentions where the correct entity appeared in the candidate list was also substantially lower for the Zulu documents compared to the English and metadata documents.

Chapter 7

Experiment 3: Hybrid NED

7.1 Introduction

The previous chapter evaluated the Automatic NED system, which only had access to the text from the FHYA documents and the knowledge base. Although the Automatic NED system performed reasonably well, the performance on documents written in isiZulu was substantially poorer than on documents written in English. Without access to additional isiZulu training data, the model's performance may be increased by using additional resources such as handcrafted rules. Previous research such as by Hedderich et al. [12] and Wu, Liu, and Cohn [76] has shown that the inclusion of additional resources such as handcrafted rules can improve the performance of Named Entity Recognition (NER) systems. This chapter explores how additional resources might be used to improve the performance of the Automatic NED system. The results from this chapter will also be used to answer the second research question from 1.2 – "*What are the effects of handcrafted features on the accuracy of the NED model?*".
The next section describes how hybrid rules were combined with the Automatic NED system to produce hybrid NED systems. This is followed by a description of how the hybrid systems were trained and evaluated in Section 7.3. Section 7.4 contains the results from the evaluation of the hybrid system. Finally, Section 7.5 provides a summary of the third experiment.

7.2 Hybrid NED Architecture

In order to assess how the incorporation of additional resources affected the Automatic NED system's performance, it was extended by incorporating additional resources in the form of hybrid rules (see Fig 7.1) to form a new hybrid NED system (referred to as the Hybrid NED system). Additional resources were selected to address cases where the Automatic NED system performed poorly (Section 6.4); the system typically performed worst on the Zulu documents.

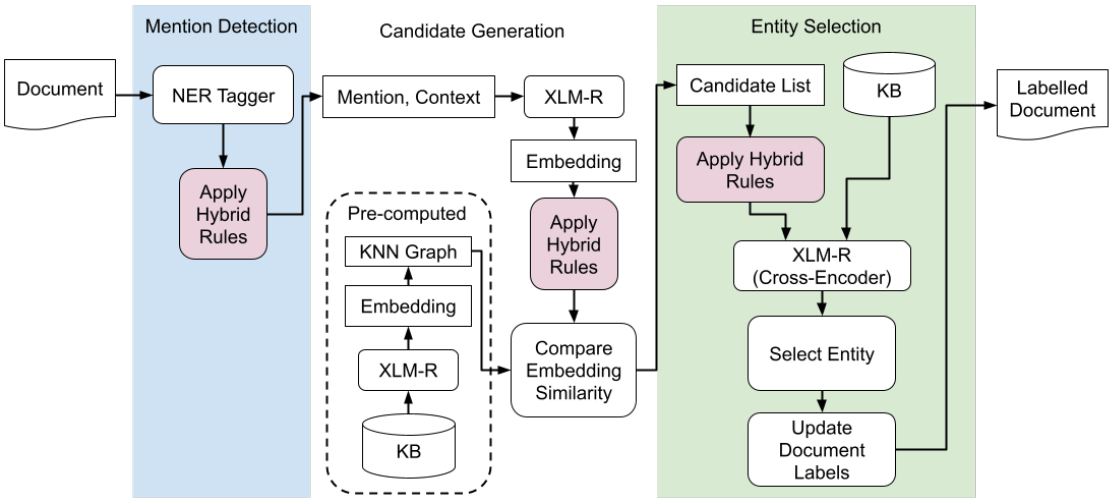

Figure 7.1: Overview of the hybrid NED systems, showing where hybrid rules were incorporated into the Automatic NED system.

7.2.1 Mention Detection

Prefix	Frequency	Proportion
u	23	4.64%
ama	23	4.64%
ku	9	1.81%
kwa	7	1.41%
o	7	1.4%

Table 7.1: Frequency of the five most common prefixes among mentions that were not detected by the Automatic NED system's mention detection module and the proportion of the total mentions which they make up.

The mention detection module performed reasonably well on all three document types. The largest difference was between the precision for the Zulu and metadata documents, with the precision for the Zulu documents 0.156 points lower. This indicates that the mention detection module identified spans of text that did not refer to a person as being a mention more often in the Zulu documents than the English and metadata. An analysis of the missed mentions showed that approximately 51% of the names that the system failed to detect had prefixes appearing before the name (e.g. uShaka). The most commonly occurring prefixes among mentions missed by the Automatic NED system are shown in Table 7.1. It should be noted that for this experiment, prefixes refer to the sequence of lower-case characters appearing immediately before a capitalised letter rather than Zulu prefixes in the linguistic sense. It is not entirely surprising that the system struggled to deal with the prefixes; since most of the data used in pre-training and training the models was not in isiZulu but in English or other languages that do not tend to attach prefixes to people's names.

In consultation with experts from the FHYA, the prefixes were analysed to determine which could be used as good indicators of whether a token was a person's name or not. Some prefixes such as u- are almost always used before a person's name. However, prefixes such as kwa- show that something comes from a place; thus, they may be used

before a person's name but also in other circumstances. Thus since the prefix cannot be used as an absolute indicator of whether the word refers to a person, the prefixes were used to adjust the confidence scores from the NER model used by the mention detection modules. The tokens were then relabelled based on the adjusted scores. Four methods were attempted to adjust the confidence scores from the NER model. These made use of a neural network, handcrafted rules, a lookup table, and a combination of the handcrafted rules and lookup table, respectively.

Reclassification Neural Network

A simple feed-forward network was constructed to relabel the tokens assigned by the token classification model. The input for the reclassification model consisted of:

- The label which the token classification model assigned.
- The confidence score for the label assigned by the token classification model.
- A boolean variable that indicated the presence of a prefix.
- A one-hot encoding for the presence of a particular prefix.
- The length of the prefix.
- The maximum similarity between the token with the prefix removed and a name from a list of names in the training data. The similarity was calculated using the Levenshtein ratio.

The reclassification model contained two hidden layers that used the sigmoid activation function. The final output was produced using softmax activation to assign a probability for the label being either an `I`, `O` or `B` tag. This approach differs from the others as the generated probabilities were not used to adjust the confidence scores of the tags assigned by the NER system. Rather the tag with the highest probability was the then assigned as the new label.

Handcrafted rules

The handcrafted rules manually adjusted the confidence score for each token where a prefix was present. The adjustments were made by adding or subtracting an amount determined based on consultations with the experts from the FHYA. If the adjusted score was less than 0.5, then the label for the token was changed. For labels indicating an entity (`I` or `B`), the label was changed to `O`, while for `O` labels, the labels would be changed to either an `I` or `B` depending on the surrounding labels.

Lookup Table

The lookup table worked in a similar way to the handcrafted rules. However, the amount by which the score was adjusted was based on the probability that a prefix preceded a person's name in the training data set. The lookup table was constructed by checking each token in each document in the training dataset for a prefix. Where a prefix was found, it was added to the lookup table, or the lookup table was updated such that the number of times each prefix appeared and the number of times it appeared before a person's

name was recorded. Using this information, the probability that the token containing the word was a person was associated with each prefix. A separate lookup table was built for each fold using the training data for the fold. Only prefixes that occurred more than ten times within the fold were added to the table to increase the accuracy of their predicted probabilities. At run time, if the token had a prefix and the prefix was in the lookup table, the score was calculated by interpolating the confidence score (C) from the mention detection NED model and the probability from the lookup table (P). Thus the new score is calculated as $\lambda C + (1 - \lambda)P$ where $\lambda \in [0, 1]$ is a configurable parameter. The models were evaluated with λ = 0.5 and $\lambda^\star$ – calculated by optimising the F1-score for the mention detection module.

Handcrafted Rules - Lookup Table Hybrid

The handcrafted rules covered relatively few cases. In contrast, the lookup table has high coverage of the prefixes, but the probabilities associated with the prefixes are approximates generated from the training data. By combining the handcrafted rules and lookup table, a few prefixes may be dealt with explicitly while the rest use the approximates from the lookup table. The handcrafted rules were combined with the lookup table that if the handcrafted rules did not contain a rule to handle the prefix for a token, the lookup table was used to relabel the token.

7.2.2 Entity Linking

An analysis of the results from the Automatic system showed that the system seemed to struggle when long prefixes were added to a name —for example, predicting Zaka instead of zikaDingiwe. In this case, the surface form of the predicted name is very similar to the prefix (zika), indicating that the system struggled to differentiate between names and prefixes. Thus a simple rule was added to remove prefixes from the mention text before it was concatenated with its context used by the bi-encoder or cross encoder. However, not all prefixes could be removed as some may change the person to whom the name refers. For instance, kaSenzengakona refers to a descendant of Senzenaghona and not Senzenaghona. In consultation with experts from the FHYA, a list of prefixes that would not be removed was created. The prefix-removal rule was applied in both the candidate generation and mention detection phases.

7.3 Method

7.3.1 Training

The hybrid system builds on the Automatic NED system. Thus, the NER, bi-encoder and cross-encoder models used in the Automatic NED system were reused with no additional training so that the only differences in performance are due to the handcrafted rules. However, the Reclassifier neural network used in the mention detection model required training.

Reclassifier Neural Network

Separate Reclassifier models were trained for each fold. The datasets for training each Reclassifier model were constructed by labelling the tokens in the training data for the fold using the mention detection NER model for the fold. The dataset for each fold included the tokens, predicted labels, confidence scores for the predicted labels, and the actual labels. Each model was trained for 100 epochs using the cross-entropy loss function. The majority of tokens in the training data for each fold have an O label since they are not part of a person's name. However, this meant that the Reclassifier model could achieve a high degree of accuracy by labelling all tokens with an O tag. Thus the data used to train the Reclassifer was balanced such that approximately 50% of the tokens had O labels.

Lookup Table

The lookup table used linear interpolation to combine the confidence score for a label produced by the NER model with the probability that the token's prefixes preceded an entity name. The interpolation was controlled using the λ parameter as described in Section 7.2.1. An optimisation algorithm was used to calculate the value of λ ($\lambda^{\star}$) such that the F1-Score was maximised. Values for $\lambda^{\star}$ were calculated for each fold based on the fold's training data. For each fold, the mention detection module augmented with a lookup table was used to generate labels for the tokens in each document in the training set. The value for λ was adjusted to maximise the F1-score of the resulting labels using the Brent optimisation algorithm implemented using the optimise module from the Scipy library[1].

7.3.2 Evaluation

Once the models had been trained, the hybrid NED systems were evaluated using 10-fold cross-validation on the folds created in Section 4.6. The performance of the hybrid systems was evaluated using the same techniques and metrics as in the Automatic NED system. The system was evaluated based on the recall, precision, and F1-scores in micro and macro settings. The final score for each metric was calculated as the average of each of the 10-folds.

Statistical Significance

A one-sided Wilcoxon signed-rank test was used to calculate the significance for the F1-scores for the hybrid NED systems compared to the Automatic NED system. The Wilcoxon signed-rank test was used as both the macro and micro f1-Scores did not follow a normal distribution and were dependent as they were generated for the same sets of documents. For the macro F1-score, the significance was calculated using the differences between the F1-scores for a particular Hybrid system and the Automatic NED system at a document level. In contrast, the significance for the micro F1-score used the differences in the F1-scores at a Fold level. In both cases, a p-value of less than 0.05 was classified as being statistically significant.

[1] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimise.minimize_scalar.html`

7.4 Results

7.4.1 Mention Detection

Technique	λ	Macro-P	Macro-R	Macro-F1	Micro-P	Micro-R	Micro-F1
Neural Network		**0.748**	0.678	0.710	**0.739**	0.615	0.664
Handcrafted		0.743	0.733	0.737	0.729	0.725	0.726
Lookup Table	0.5	0.741	**0.735**	**0.737**	0.728	**0.728**	**0.727**
	$\lambda^\star$	0.743	0.733	0.737	0.731	0.724	0.727
Handcrafted + Lookup	0.5	0.742	0.733	0.737	0.730	0.726	0.727
	$\lambda^\star$	0.744	0.731	0.737	0.732	0.722	0.727
Split Lists		0.711	0.691	0.700	0.708	0.685	0.696
None		0.744	0.732	0.737	0.731	0.722	0.726

Table 7.2: A comparison of micro and macro precision, recall and F1-scores for different techniques for incorporating hybrid rules into the mention detection module for the Automatic NED system. The None technique indicates the Automatic NED system with no hybrid rules.

Table 7.2 shows that none of the hybrid techniques explored to improve the performance of the mention detection module resulted in statistically significant improvements to the micro or macro F1-scores based on a Wilcoxon signed-rank test. The model with the highest micro F1-score only increases the F1-score by 0.001 (w=9712, p=0.120). The models which incorporated handcrafted rules resulted in small increases in the recall compared to the Automatic NED system; however, this came at the expense of the precision, which decreased compared to the Automatic NED system. This indicates that adding handcrafted rules made the model less conservative in identifying mentions. This seems reasonable since words matching the rules were likely to be identified as mentions regardless of the context. The exception was the reclassification neural network which significantly increased the precision but at the cost of the recall resulting in low F1-scores. This indicates that despite balancing the Reclassifer training data, the Reclassifier network tended to incorrectly label tokens as not belonging to an entity.
The small performance increases when the hybrid rules are applied, maybe because the NED system has learnt to deal with the most frequent use of prefixes, but it struggles with more complex cases. For more complex cases, the hybrid rules used are likely to have a minimal effect as they targeted more general cases, particularly where they were applied directly as handcrafted rules or using the lookup table.

7.4.2 Entity Linking

System	CG	ES	Macro-P	Macro-R	Macro-F1	Micro-P	Micro-R	Micro-F1
AT			0.744	0.732	0.737	0.731	0.722	0.726
AT-CG	✓		**0.749**	**0.737**	**0.742**	0.740	0.731	0.735
AT-ES		✓	0.740	0.729	0.734	0.727	0.718	0.722
AT-CG+ES	✓	✓	0.747	0.736	0.741	**0.741**	**0.732**	**0.736**

Table 7.3: A comparison of micro and macro precision, recall and F1-scores between the Automatic System augmented with hybrid candidate generation (CG) and hybrid entity selection (ES). Thus the first row shows the results for the Automatic NED system from the previous chapter.

The results of applying the rule in one or both of the modules are shown in Table 7.3. Applying the rule only in the candidate generation module (AT-CG) leads to the highest macro F1-score, while the highest micro F1-score was achieved when both modules applied the rule (AT-CG+ES). Although, in both cases, the differences were slight. However, both systems performed better than the system that only applied the rule to the entity selection module (AT-ES), which performs worse than the Automatic NED system (AT). The decrease in F1-score when the prefix removal rule was applied may be because the candidate generation module generates an embedding for the mention context pair. In contrast, the entity selection module takes two mentioned context pairs as an input and determines if they refer to the same entity. The mention text always appeared in the context in the mention-context pairs used to train the cross-encoder. However, when the prefix removal rule was applied, the prefix was only removed from the mention while the context was unchanged. The mismatch between the surface forms of the mention and the mention in context may have disrupted concepts the cross-encoder had learnt. Removing the prefix from the mention in context may also disrupt the cross-encoder by changing the meaning of the context. When the prefix-removal rule was applied to both the candidate generation and entity selection modules, fewer of the candidates generated included a prefix. Thus, fewer cases occurred where there was a mismatch between the surface form of the mention and mention in context, resulting in better performances.
Figure 7.2 shows the accuracy of the entity linking modules when the hidden entities are ignored. This means that the entity for each mention was present in the knowledge base. The mentions were obtained using the labels produced by the volunteers' annotations. This meant all mentions from the FHYA collections were included, and each mention correctly referred to an entity. Using this setting, the accuracy of the Automatic and hybrid systems was high, with over 90% of the mentions linked to the correct entity. There was a considerable improvement in the performance of the entity linking modules between the baseline and Automatic NED system. However, in line with previous results discussed above, the hybrid rules slightly increased the entity linking accuracy.

Candidate Generation

The addition of hybrid rules to the candidate generation was most effective for the Zulu documents resulting in the correct entity being contained in the candidate list for almost 2% more of the mentions. There was also a corresponding decrease in the mean position of the correct entity within the candidate list. However, the hybrid rules had little to no effect on the metadata and English documents. This was expected as most names

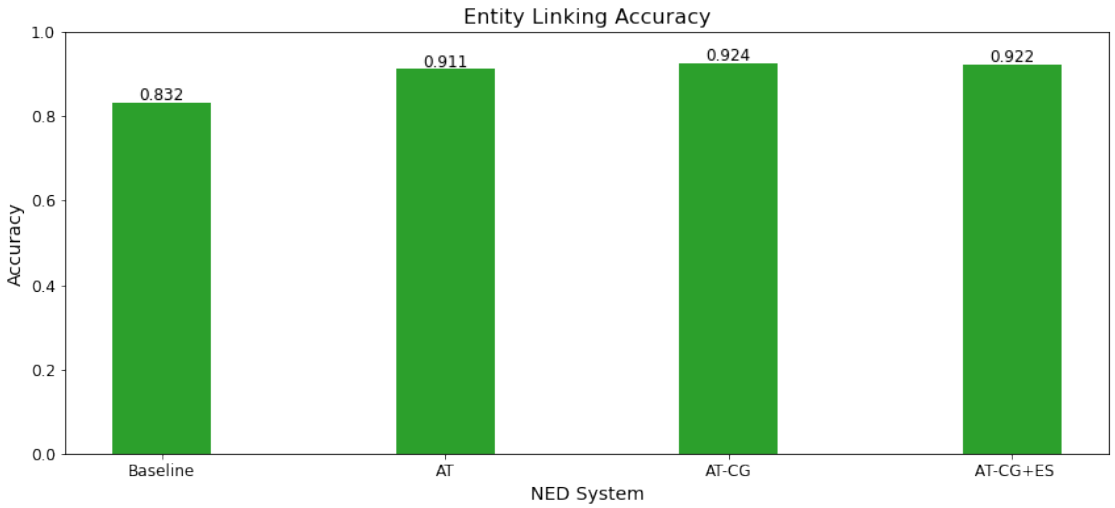

Figure 7.2: Accuracy of the entity linking module for the Baseline, Automatic and hybrid systems when the mention boundaries are given by the document labels and hidden entities are excluded.

Document Type	Contained (%)	Mean Position
English	77.82 (0.00)	1.130 (-0.017)
Metadata	83.54 (0.00)	1.133 (0.000)
Zulu	73.91 (+1.96)	1.361 (-0.206)
All	75.44 (+1.38)	1.290 (-0.143)

Table 7.4: Percentage of mentions for which the candidate list contained the correct candidate and the mean position in the candidate list of the correct candidate when it was present. The brackets show the change compared to the Automatic NED system.

that included a prefix occurred in the Zulu documents. However, despite the increase in the number of correct candidates in the Zulu documents, there is still a considerable gap between the percentage of correct candidates in the Zulu documents compared to the metadata and English documents.

Entity Selection

Using the hybrid entity selection module without the hybrid candidate generation module decreased the overall performance of the NED system (see Table 7.5a). However, when used in conjunction with the hybrid candidate module, the resulting system achieved the highest micro F1-Score. However, based on the results shown in Tables 7.5a and 7.5b, the increases are primarily due to the hybrid candidate generation module as the proportion of mentions for which the hybrid entity selection module identified the correct candidate decreased in relative terms. However, since the decrease in the performance of the entity selection module is smaller than in the increase in the performance of the mention detection module, the overall system's performance increases. Hence, this system achieves the highest micro F1-score. The decrease in performance is not entirely unexpected as it is measured only for mentions where the correct candidate was predicted. Thus, when the candidate generation module's performance increases, the number of mentions on which the entity selection module is evaluated increases. It is also likely that many of the new mentions are harder to disambiguate; this explains why they were missed by the mention

Document Type	Accuracy
English	0.961 (0.000)
Metadata	0.970 (-0.005)
Zulu	0.957 (-0.008)
All	0.957 (-0.006)

(a) Hybrid rules applied to the candidate generation and entity selection modules (model AT-CG+ES from Table 7.3)

Document Type	Accuracy
English	0.961 (0.000)
Metadata	0.975 (0.000)
Zulu	0.956 (-0.009)
All	0.958 (-0.007)

(b) Hybrid rules applied only to the candidate generation module (model AT-CG from Table 7.3)

Table 7.5: Accuracy of the entity selection module for mentions where the correct entity was present in the candidate list. The table on the left uses the hybrid entity selection module while the table on the right uses the entity selection module from the Automatic Hybrid NED system. The difference between the hybrid NED system and the Automatic NED systems performance is shown in brackets.

detection module used by the Automatic NED system.
A comparison between Table 7.5a and Table 7.5b shows that the hybrid entity selection module provides a slight advantage over the Automatic entity selection module for Zulu documents. However, the hybrid entity selection module was slightly less accurate on the metadata documents, but since there are far more Zulu mentions than metadata mentions, the hybrid system produced a slightly better performance overall.

7.4.3 Comparison with Automatic NED System

Mention Detection	λ	CG	ES	Macro-P	Macro-R	Macro-F1	Micro-P	Micro-R	Micro-F1
Lookup	0.5	✓		0.749	**0.734**	**0.741**†	0.743	0.727	0.734†
	0.5	✓	✓	0.747	0.733	0.739	0.744	**0.728**	0.735
Handcrafted + Lookup	$\lambda^\star$	✓		**0.752**	0.731	0.741†	0.748	0.722	0.734†
	$\lambda^\star$	✓	✓	0.751	0.729	0.739	**0.749**	0.724	**0.735**†
None		✓		**0.749**	**0.737**	**0.742**†	0.740	0.731	0.735†
		✓	✓	0.747	0.736	0.741†	**0.741**	**0.732**	**0.736**†
Automatic NED				0.744	0.732	0.737	0.731	0.722	0.726
Labels				0.797	0.823	0.810	0.773	0.821	0.796

Table 7.6: Micro and macro precision, recall and F1-scores for the the Hybrid and Automatic NED systems. Automatic NED and AT-Label show the metrics for the Automatic NED. Labels use the labels generated by the annotator to find the mentions.
$\lambda^\star$ is the value for λ determined by maximising the f1-score for the training data.
†: Significant at $p < 0.05$ using a one-sided Wilcoxon signed-rank test.

The results presented in Table 7.6 show that the hybrid NED systems were able to produce a small but significant increase in the macro and micro F1-scores. All but one of the systems produced a significant increase in Micro F1-score and only two systems failed to produce a significant increase in macro F1-score. Unsurprisingly, given the results in the Sections 7.4.1 and 7.4.2, the most significant increases were achieved by only adding rules to the entity linking modules. The systems that applied rules to the mention detection module had the same difficulties as the mention detection model in isolation; signifi-

cant increases in precision were countered by decreases in the recall, leading to marginal increases in the F1-score. This pattern held for both the micro and macro metrics.

7.5 Summary

The Automatic NED system performed reasonably well; however, there was a large discrepancy between its performances on the metadata and English documents compared to the Zulu documents. Thus, hybrid rules were used to try to improve the performance of the Automatic NED system. Based on the results from the Automatic NED system, mentions including prefixes before the name were particularly problematic. Thus a set of hybrid rules was created to deal with the prefixes. Separate rules were created for each module (mention detection, candidate generation and entity selection). For the mention detection module, rules were created to reclassify tokens based on their prefixes. Four methods were used to incorporate these rules with the mention detection module: a neural network, handcrafted rules, a lookup table and a combination of the handcrafted rules and lookup table. Of these techniques, the lookup table and the combination of handcrafted rules and lookup table performed the best. However, none of these approaches resulted in statistically significant improvements to the performance of the mention detection module. The hybrid candidate generation module was created by adding a rule removing the prefix from a mention when generating candidates. The hybrid candidate generation resulted in the most significant performance boost, resulting in a 2% increase in the number of candidate lists that contained the correct mention. The highest macro F1-score was achieved using the Automatic NED system with the hybrid candidate generation module. However, when a similar rule was applied to the entity selection module, the NED system's performance deteriorated. Although used in combination with the hybrid candidate generation module, the best performance in terms of micro F1-score was achieved. When the hybrid mention detection, hybrid candidate generation and hybrid entity selection modules were combined, the resulting NED systems performed better than the Automatic NED model. However, they performed worse than the systems only using hybrid candidate generation or hybrid candidate generation with hybrid entity selection for all metrics except the micro and macro precision. This suggests that hybrid rules can be used to improve the performance of the Automatic NED system. However, more specific rules may be needed to see a larger increase in performance.

Chapter 8

Conclusions

In the previous chapters, three experiments were run to explore the use of language models for Named Entity Disambiguation (NED) within a historical South African context. The historical South African documents were collected from the FHYA project and included documents in English and isiZulu. Thus, the NED systems had to deal with documents written in a low-resource language (isiZulu) and documents using archaic language due to their age.
This chapter draws conclusions from the results of the three experiments. The next section provides an overview of the conclusions from the experiments. Section 8.2 provides answers to the main research questions. A brief comparison between the results from the experiments and previous research is made in Section 8.3. Finally, Section 8.4 details the main contributions from the experiment results.

8.1 Overview

Experiments 1 and 2 showed that language models were reasonably effective for NED within a historical South African context compared to a simple baseline. Furthermore, this showed that the language models could attain some level of language understanding and utilise the context surrounding a mention in both English and isiZulu (a low-resourced language). Although, as expected, the performances were better when the documents were written in English (considered a high-resource language), compared to isiZulu. However, experiment three showed that by incorporating hybrid rules, the language-model based NED system's performances could be further improved, especially for documents written in low-resource languages (isiZulu).
In the first experiment, a baseline system was created. The baseline served two purposes. Firstly, it was used to benchmark the performance of the language model-based system, but it also indicated how difficult it was to disambiguate the documents in the FHYA collection. The baseline system did not perform particularly well due to the string-matching mention detection module. This simple mention detection module still produced better results than when the module was based on a commercial English NER system. However, the entity selection module performed reasonably well. Since the entity selection system could not deal with ambiguity, its performance, when given the mentions, indicated the degree of ambiguity in the mentions from the FHYA collection. When given the mention boundaries, the baseline could link 75% of the mentions to the correct entity, indicating that although most of the mentions were unambiguous, a large number of mentions were difficult to disambiguate.

The Automatic NED system (a language model-based NED system) was created and evaluated in the second experiment. The Automatic NED system performed well on the documents within the FHYA collection, showing substantial improvements over the baseline systems. The Automatic NED system also performed well on documents in both English and isiZulu. This indicates that multilingual language models such as XLM-R can learn concepts across multiple languages even when trained on relatively small amounts of data from low-resourced languages. However, the Automatic NED system performed far better on the English documents compared to the Zulu documents. Thus the language models used also performed better for English documents, indicating that although the multilingual language models can transfer concepts between languages, they still perform better for high resource languages. Thus although multilingual language models are effective, it seems that they are not a perfect substitute for a monolingual language model trained on a large corpus.
The third experiment looked at how hybrid rules could be used to decrease the difference between the performance of the Automatic NED system on the English and Zulu documents. Several hybrid models were created by incorporating various rules into the Automatic NED system. However, many of these rules only resulted in minor improvements. The mention detection module, in particular, barely improved compared to the mention detection module used by the Automatic NED system. The most effective rule focused on removing the prefixes attached to isiZulu names in the candidate generation module. This resulted in a substantial increase in the number of mentions where the correct entity appeared in the candidate list. Overall, the hybrid models were able to provide significant increases in the micro and macro F1-scores compared to the Automatic NED system. There was also a slight increase in the accuracy of the entities linked to the mentions detected. Although these increases were relatively minor, they are still useful as document collections such as the FHYA collection contain thousands of mentions.

8.2 Research Questions

How accurate are transformer-based language models for NED applied to the text from the 500 Hundred Year Archive?
The Automatic NED system described in experiment 2 made use of the three modules, each based on a XLM-RoBERTa (XLM-R) language model. The Automatic NED system showed considerable improvements over the baseline system and achieved micro and macro F1-scores of 0.737 and 0.726, respectively. The mention detection and entity linking modules both performed far better than those used in the baseline systems. The improvement in the entity linking modules of the Automatic NED system compared to the baseline shows that the system could utilise the context within which the mentions appeared when linking them to an entity. These results indicate that transformer-based language models are effective for performing NED on documents from the FHYA

What are the effects of handcrafted features on the accuracy of the NED model?
The third experiment augmented the Automatic NED system with hybrid rules to create hybrid NED systems. The results from experiment three indicated that hybrid rules could be used to improve the performance of NED systems since the hybrid systems achieved a small but statistically significant increase in the performance compared to the Automatic NED model based on the micro and macro F1-scores. This indicates that hybrid rules can be used to improve the accuracy of NED models. However, it seems likely that more

complex rules targeting specific cases will likely lead to larger improvements in the NED system than the relatively general rules applied to the hybrid models in experiment three.

8.3 Comparisons with Previous Research

The Automatic and hybrid NED systems were based on the architecture used by the zero-shot NED system created by Wu et al. [20]. However, it is impossible to compare the two models as they were evaluated on different datasets. For example, the system developed by Wu et al. [20] had an accuracy of 0.766 on the Zero-shot EL dataset [37] compared to 0.945 on the TACKBP-2010 dataset [20]. Additionally, the Wu et al. [20] model was evaluated in a zero-shot setting while the Automatic and hybrid NED models had access to in-domain training data. However, the Automatic and hybrid NED systems' accuracy of 0.812 and 0.83 (ignoring mentions associated with hidden entities), respectively, indicates that these models perform at a similar level to the model developed by Wu et al. [20].
Previous research has looked at NER within a South African context. Most recently, Eiselen [17] developed CRF-based NER systems for 10 of the official South African languages. The NER system used in the mention detection module also outperforms the isiZulu NER system developed by Eiselen [17], resulting in increases of 0.07, 0.26 and 0.15 for the precision, recall and f1 score. This may be due to differences in the datasets. Additionally, the NER system used in the mention detection module was only trained to detect person names rather than other entities such as organisations or places. However, this still provides strong evidence that using multilingual language models can provide substantial performance increases compared to the CRF-based NER models.

8.4 Contributions

The results from the previous experiments indicate that multilingual transformer-based language models can be used effectively in NED systems for historical South African documents written in English and isiZulu. However, it seems likely that the language model-based NED systems would be able to generalise to other South African languages, requiring relatively few additional documents for training. This seems especially likely as previous research has found that many of the South African languages are similar enough that the same NLP models can be used for documents in different South African languages [71].
The language model-based NED systems also provide further evidence that multilingual language models can work effectively with multiple languages. Each of the language model-based NED systems used the same NER, bi-encoder and cross-encoder models, which all produced good results for both English and isiZulu text. Although, the models did tend to perform better on English data.
The third experiment shows that hybrid techniques that incorporate handcrafted heuristics can be effective in improving NED system performances. However, the heuristics used were fairly general and provided evidence that more complex rules targeting specific linguistic features may lead to further improvements. This suggests that the multilingual language models learnt the core isiZulu concepts using a small amount of data but struggled to learn more nuanced linguistic features. Similar effects would likely be observed in other low-resourced languages due to the lack of datasets available for training the language models.

In addition to the contributions from the experiments, there is also the dataset that was created. The dataset consists of the labelled documents from the FHYA and the entity records from the knowledge base. This dataset may interest future researchers as it contains annotated isiZulu documents. The dataset is publically available[1] in the format described in Section 3.11.
Finally, from a software perspective, the source code for both The Annotator[2] and NED systems[3] have been placed in publicly available repositories allowing them to be used for future research.

[1] https://doi.org/10.25375/uct.19029692
[2] https://gitlab.com/jwdunn/mastersannotator
[3] https://gitlab.com/jwdunn/mastersned

Chapter 9

Future Work

9.1 Further Development of Hybrid Rules

After the analysis for the hybrid rules had been done, an improvement made to the mention detection module meant that the initial weaknesses of the Automatic system were reduced. Before the changes were made, the recall for Zulu documents was 0.45 points lower for the metadata documents. Thus, most of the hybrid rules focused on how the mention detection module's recall could be improved. However, after the changes had been made, the recall for the Zulu documents was 0.05 and 0.012 less than the English and metadata documents, respectively. Thus, much of the work on hybrid rules was done for a section that offered less room for improvement than initially thought. The results of the third experiment show that although the hybrid rules were able to significantly improve the performance of the Automatic NED system, they did not significantly improve the performance of the mention detection module. This showed that the language model could deal with common isiZulu linguistic concepts. Thus the development of rules that target more complex cases will likely yield better results.

9.2 Development of OCR systems for Texts in African Languages

The scans for the isiZulu texts tended to be noisier than for the English and metadata texts. There was almost no noise in the metadata documents as they were already in plain text. In contrast, the English and Zulu documents were extracted from PDFs using Optical Character Recognition which often introduced additional noise. Future NED systems could improve on the results by adding components to deal with the noise in the data explicitly or through the development of OCR systems explicitly for African languages, which would provide cleaner text.

9.3 Extending the Types of Entities Identified

The NED systems developed only targeted names of people. However, there are other named entity types such as locations and organisations. This would add some additional complications; for example, places may be named after people. However likely, disambiguating other entity types would only require some modifications to the mention

detection module to be achieved.

9.4 Further Exploration of Language Models for African Languages

This project used the multi-lingual language model XLM-R in all three of the modules for the Automatic NED system and the hybrid NED systems. However, the development and training of transformer-based language models is still an active area of research. Thus, subsequently to the development of the NED systems, language models trained specifically for low-resource African languages such as AfriBERTa [77] have been produced. The use of these language models may lead to improved performances.

9.5 Exploration of the Effects of the Language of the Documents used for Fine-tuning

The models used in the Automatic and Hybrid NED systems were fine-tuned using English and Zulu documents from the FHYA. However, it would be intersting to compare these results with results from models that were fine tuned using only English or only Zulu documents. These experiments could show if there is a benefit to training on a mixutre of the two languages. The results from the model trained on only the English documents could give an idication of the models ability to generalise to unseen languages.

9.6 Further Investigation of Hyper-parameters

Liu et al. [50] showed that by adjusting the training for BERT resulted in significant improvements to the models performance. Similarly further experiments with the hyper-parameters for the models used for each of the stages may yield better results.

9.7 Further Exploration of NED for South African Languages

The documents used for training and evaluating the NED systems were written in either English or isiZulu. However, these are only two of the 11 official South African languages. The FHYA contains a number of documents in siSwati. However, due to the complexities of finding volunteers to annotate documents in siSwati and time constraints, they were not included in the labelled collection used for training and evaluating the NED systems.

Appendix A

Annotation System

A.1 Annotator System ER Diagram

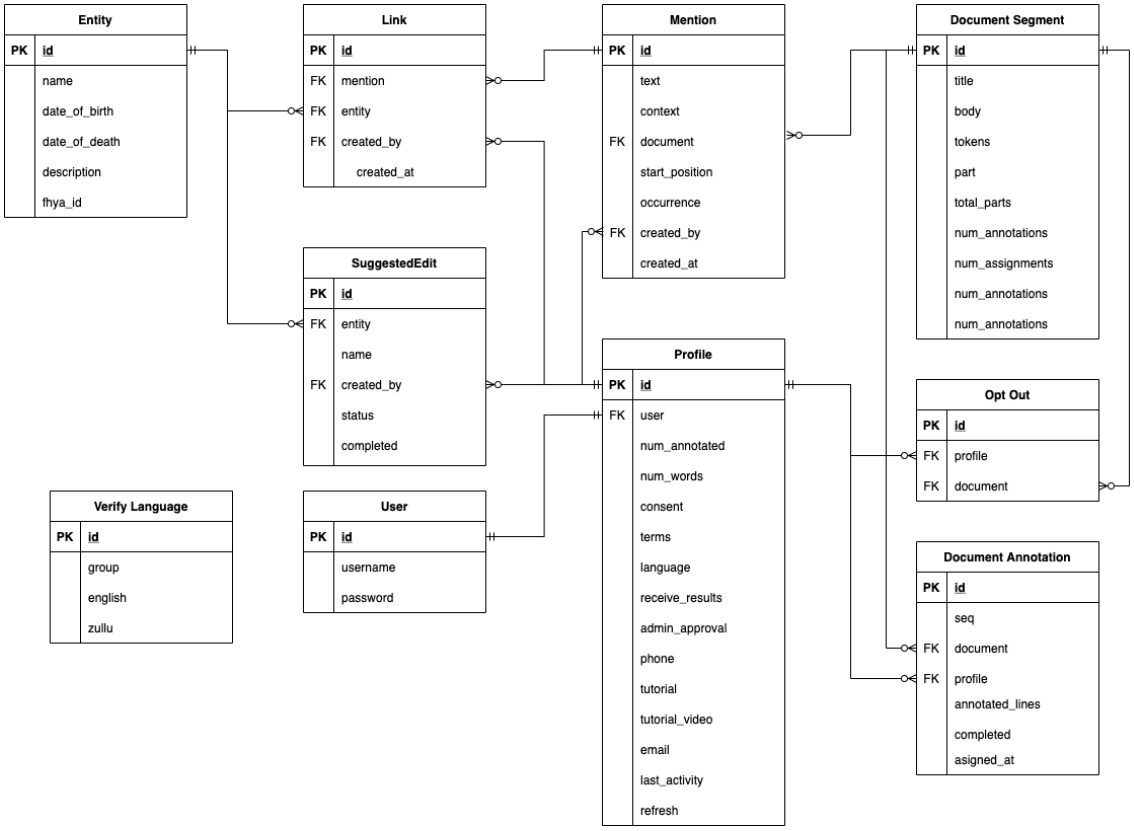

Figure A.1: Full entity-relationship diagram showing the organisation of the database for the Annotator system.

A.2 Ethics Approval Letter

UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Faculty of Science
University of Cape Town
Rondebosch
South Africa 7701

E-mail: melissa.densmore@uct.ac.za
Tel: 021 650-9111

14 May 2021

Mr Jarryd Dunn
Department of Computer Science

Evaluating Automated and Hybrid Neural Disambiguation for African Historical Named Entities

Dear Mr Jarryd Dunn

I am pleased to inform you that the Faculty of Science Research Ethics Committee has approved the above-named application for research ethics clearance, subject to the conditions listed below.

- Restrictions on involving human participants in research must be adhered to, given current concerns about the spread of Covid-19. Please ensure that you are aware of and comply with UCT policy on this, as communicated by management.
- Implement the measures described in your application to ensure that the process of your research is ethically sound; and
- Uphold ethical principles throughout all stages of the research, responding appropriately to unanticipated issues: please contact me if you need advice on ethical issues that arise.

Your approval code is: **FSREC 047 – 2021**

I wish you success in your research.

Yours sincerely

Dr Melissa Densmore
Acting Chair: Faculty of Science Research Ethics Committee

Cc: **Prof Hussein Suleman (supervisor)**

Figure A.2: Ethics approval letter for using human participants to annotate documents from the 500 Year Archive.

A.3 isiZulu Literacy Questions

Group	isiZulu	English
1	Ngiyakuthanda	I like you.
	Angiyakuthanda	I don't like you.
	Zange ngakuthanda.	I didn't like you.
2	iBhola incane ngoMsombuluko.	The ball is smaller on Mondays.
	iBhola inkulu ngoMsombuluko.	The ball is bigger on Mondays.
	iBhola iyafana ngoMsombuluko.	The ball is the same on Mondays.
3	Umnikeze incwadi yakhe namhlanje.	He gave her his book today.
	Umnikeze incwadi yakhe izolo.	He gave her his book yesterday.
	Uzomnikeza incwadi yakhe kusasa.	He will give her his book tomrrow.
4	Uhlaza ngumbala awuthandayo.	Green is her favourite colour.
	Uhlaza wayengumbala awuthandayo.	Green was her favourite colour.
	Uhlaza asingumbala awuthandayo.	Green is not her favourite colour.

Table A.1: Phrases used to generate questions to verify that participants are literate in isiZulu when they sign up. The questions are based on those used by [72]

.

Appendix B

Data Collection

B.1 FHYA Authoritative Records

The fields for each record are shown in Table B.1. Only the records where typeOfEntity was ”Person” were used. The name was then taken to be the value of the authorizedFormOfName field while the description was taken from the description field. The date of birth or date of death was only used where a complete date was available.
Before using the data some cleaning was required. Initially the authorizedFormOfName field had a mix of names formatted as `surname`, `first_names` and `first_names surname`. These were standardised to the `first_names surname` format.

Field	Description
culture	The entities culture.
typeOfEntity	What is the entity. Values can be one of Corporate, Family, Family?, Government Body, Museum, Person, Publisher, Research or University.
authorizedFormOfName	Entity Name
datesOfExistance	Date of birth and death formatted separated by a hyphen e.g 7 May 1905 - 26 August 1986.
history	A list of the descriptions that have been used for the entity and the source of those descriptions.
retrievedBy	Name of the person who created the entry for the record.
retrievedYear	Year when the entity record was added.
retrievedFrom	Source of the entity and its description.
description	A description of the entity.

Table B.1: Fields for entity records in the authoritative records maintained by the FHYA. For conciseness fields that were always blank have been excluded.

Bibliography

[1] Samuel Broscheit. "Investigating Entity Knowledge in BERT with Simple Neural End-To-End Entity Linking". In: *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 677–685. DOI: 10.18653/v1/K19-1063. URL: https://www.aclweb.org/anthology/K19-1063.

[2] W. Shen, J. Wang, and J. Han. "Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions". In: *IEEE Transactions on Knowledge and Data Engineering* 27.2 (2015), pp. 443–460. DOI: 10.1109/TKDE.2014.2327028.

[3] Jan-Christoph Klie, Richard Eckart de Castilho, and Iryna Gurevych. "From Zero to Hero: Human-In-The-Loop Entity Linking in Low Resource Domains". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 6982–6993. DOI: 10.18653/v1/2020.acl-main.624. URL: https://www.aclweb.org/anthology/2020.acl-main.624.

[4] Shuyan Zhou, Shruti Rijhwani, John Wieting, Jaime Carbonell, and Graham Neubig. "Improving Candidate Generation for Low-resource Cross-lingual Entity Linking". In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 109–124. DOI: 10.1162/tacl_a_00303. URL: https://aclanthology.org/2020.tacl-1.8.

[5] Cı'cero dos Santos and Victor Guimarães. "Boosting Named Entity Recognition with Neural Character Embeddings". In: *Proceedings of the Fifth Named Entity Workshop*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 25–33. DOI: 10.18653/v1/W15-3904. URL: https://www.aclweb.org/anthology/W15-3904.

[6] Eneko Agirre, Ander Barrena, Oier Lopez de Lacalle, Aitor Soroa, Samuel Fernando, and Mark Stevenson. "Matching Cultural Heritage items to Wikipedia". In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*. Istanbul, Turkey: European Language Resources Association (ELRA), May 2012, pp. 1729–1735. URL: http://www.lrec-conf.org/proceedings/lrec2012/pdf/1021_Paper.pdf.

[7] Kai Labusch and Clemens Neudecker. "Named Entity Disambiguation and Linking Historic Newspaper OCR with BERT". In: *Working Notes of Conference and Labs of the Evaluation Forum (CLEF 2020)*. Ed. by Linda Cappellato, Carsten Eickhoff, Nicola Ferro, and Aurélie Névéol. Vol. 2696. CEUR Workshop Proceedings. Thessaloniki, Greece: CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696/paper%5C_163.pdf.

[8] *List of Wikipedias*. https://meta.wikimedia.org/wiki/List_of_Wikipedias. Accessed: 2022-01-18.

[9] Statistics South Africa. *Census 2011 Census in Brief*. https://www.statssa.gov.za/census/census_2011/census_products/Census_2011_Census_in_brief.pdf. 2012.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://www.aclweb.org/anthology/N19-1423.

[12] Michael A. Hedderich, David Adelani, Dawei Zhu, Jesujoba Alabi, Udia Markus, and Dietrich Klakow. "Transfer Learning and Distant Supervision for Multilingual Transformer Models: A Study on African Languages". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 2580–2591. DOI: 10.18653/v1/2020.emnlp-main.204. URL: https://www.aclweb.org/anthology/2020.emnlp-main.204.

[13] Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. "Combating Adversarial Misspellings with Robust Word Recognition". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 5582–5591. DOI: 10.18653/v1/P19-1561. URL: https://www.aclweb.org/anthology/P19-1561.

[14] Filip Ilievski, Piek Vossen, and Stefan Schlobach. "Systematic Study of Long Tail Phenomena in Entity Linking". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 664–674. URL: https://www.aclweb.org/anthology/C18-1056.

[15] Alex Olieman, Kaspar Beelen, Milan van Lange, Jaap Kamps, and Maarten Marx. "Good Applications for Crummy Entity Linkers? The Case of Corpus Selection in Digital Humanities". In: *Proceedings of the 13th International Conference on Semantic Systems*. Semantics2017. Amsterdam, Netherlands: Association for Computing Machinery, 2017, pp. 81–88. ISBN: 9781450352963. DOI: 10.1145/3132218.3132237. URL: https://doi.org/10.1145/3132218.3132237.

[16] Anita Louis, Alta De Waal, and Cobus Venter. “Named Entity Recognition in a South African Context”. In: *Proceedings of the 2006 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries*. SAICSIT ’06. Somerset West, South Africa: South African Institute for Computer Scientists and Information Technologists, 2006, pp. 170–179. ISBN: 1595935673. DOI: 10.1145/1216262.1216281. URL: https://doi.org/10.1145/1216262.1216281.

[17] Roald Eiselen. “Government Domain Named Entity Recognition for South African Languages”. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*. Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, pp. 3344–3348. URL: https://www.aclweb.org/anthology/L16-1533.

[18] Erkki Heino, Minna Tamper, Eetu Mäkelä, Petri Leskinen, Esko Ikkala, Jouni Tuominen, Mikko Koho, and Eero Hyvönen. “Named entity linking in a complex domain: Case second world war history”. In: *International Conference on Language, Data and Knowledge*. Cham: Springer International Publishing, 2017, pp. 120–133. ISBN: 978-3-319-59888-8.

[19] Emanuela Boros, Elvys Linhares Pontes, Luis Adrián Cabrera-Diego, Ahmed Hamdi, José G. Moreno, Nicolas Sidère, and Antoine Doucet. “Robust Named Entity Recognition and Linking on Historical Multilingual Documents”. In: *Conference and Labs of the Evaluation Forum (CLEF 2020)*. Vol. 2696. Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum. Thessaloniki, Greece: CEUR-WS Working Notes, Sept. 2020, pp. 1–17. URL: https://hal.archives-ouvertes.fr/hal-03026969.

[20] Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. “Scalable Zero-shot Entity Linking with Dense Entity Retrieval”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 6397–6407. DOI: 10.18653/v1/2020.emnlp-main.519. URL: https://www.aclweb.org/anthology/2020.emnlp-main.519.

[21] Haotian Chen, Xi Li, Andrej Zukov Gregoric, and Sahil Wadhwa. “Contextualized End-to-End Neural Entity Linking”. In: *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*. Suzhou, China: Association for Computational Linguistics, Dec. 2020, pp. 637–642. URL: https://www.aclweb.org/anthology/2020.aacl-main.64.

[22] Julien Plu, Giuseppe Rizzo, and Raphaël Troncy. “A Hybrid Approach for Entity Recognition and Linking”. In: *Semantic Web Evaluation Challenges*. Ed. by Fabien Gandon, Elena Cabrio, Milan Stankovic, and Antoine Zimmermann. Cham: Springer International Publishing, 2015, pp. 28–39. ISBN: 978-3-319-25518-7.

[23] Nina Wacholder, Yael Ravin, and Misook Choi. “Disambiguation of Proper Names in Text”. In: *Fifth Conference on Applied Natural Language Processing*. Washington, DC, USA: Association for Computational Linguistics, Mar. 1997, pp. 202–208. DOI: 10.3115/974557.974587. URL: https://www.aclweb.org/anthology/A97-1030.

[24] Vikas Yadav and Steven Bethard. "A Survey on Recent Advances in Named Entity Recognition from Deep Learning models". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 2145–2158. URL: https://www.aclweb.org/anthology/C18-1182.

[25] Stefan Schweter and Luisa März. "Triple E-Effective ensembling of embeddings and language models for NER of historical German". In: *Conference and Labs of the Evaluation Forum (CLEF 2020)*. 2020.

[26] Konstantin Todorov and Giovanni Colavizza. "Transfer Learning for Named Entity Recognition in Historical Corpora". In: *Conference and Labs of the Evaluation Forum (CLEF 2020) (Working Notes)*. 2020. URL: http://ceur-ws.org/Vol-2696/paper_168.pdf.

[27] Shuyan Zhou, Shruti Rijhwani, and Graham Neubig. "Towards Zero-resource Cross-lingual Entity Linking". In: *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 243–252. DOI: 10.18653/v1/D19-6127. URL: https://www.aclweb.org/anthology/D19-6127.

[28] Yogarshi Vyas and Miguel Ballesteros. *Linking Entities to Unseen Knowledge Bases with Arbitrary Schemas*. 2020. arXiv: 2010.11333 [cs.CL].

[29] G. Wu, Y. He, and X. Hu. "Entity Linking: An Issue to Extract Corresponding Entity With Knowledge Base". In: *IEEE Access* 6 (2018), pp. 6220–6231. DOI: 10.1109/ACCESS.2017.2787787.

[30] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. "Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation". In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 250–259. DOI: 10.18653/v1/K16-1025. URL: https://www.aclweb.org/anthology/K16-1025.

[31] Xiyuan Yang, Xiaotao Gu, Sheng Lin, Siliang Tang, Yueting Zhuang, Fei Wu, Zhigang Chen, Guoping Hu, and Xiang Ren. *Learning Dynamic Context Augmentation for Global Entity Linking*. 2019. arXiv: 1909.02117 [cs.CL].

[32] Ikuya Yamada, Koki Washio, Hiroyuki Shindo, and Yuji Matsumoto. "Global Entity Disambiguation with Pretrained Contextualized Embeddings of Words and Entities". In: (2020). arXiv: 1909.00426 [cs.LG].

[33] Elvys Linhares Pontes, Luis Adrián Cabrera-Diego, José Moreno, Emanuela Boros, Ahmed Hamdi, Nicolas Sidère, Mickaël Coustaty, and Antoine Doucet. "Entity Linking for Historical Documents: Challenges and Solutions". In: *Digital Libraries at Times of Massive Societal Transition: 22nd International Conference on Asia-Pacific Digital Libraries, ICADL 2020, Kyoto, Japan, November 30 – December 1, 2020, Proceedings*. Kyoto, Japan: Springer-Verlag, Nov. 2020, pp. 215–231. ISBN: 978-3-030-64451-2. DOI: 10.1007/978-3-030-64452-9_19.

[34] Anderson A Ferreira, Marcos André Gonçalves, and Alberto HF Laender. "A brief survey of automatic methods for author name disambiguation". In: *Acm Sigmod Record* 41.2 (2012), pp. 15–26.

[35] Vera Provatorova, Svitlana Vakulenko, E. Kanoulas, K. Dercksen, and Johannes M. van Hulst. "Named Entity Recognition and Linking on Historical Newspapers: UvA.ILPS & REL at CLEF HIPE 2020". In: *Conference and Labs of the Evaluation Forum (CLEF 2020)*. 2020.

[36] Marcel R Ackermann and Florian Reitz. "Homonym Detection in Curated Bibliographies: Learning from dblp's Experience". In: *International Conference on Theory and Practice of Digital Libraries*. Springer. 2018, pp. 59–65.

[37] Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. "Zero-Shot Entity Linking by Reading Entity Descriptions". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 3449–3460. DOI: 10.18653/v1/P19-1335. URL: https://aclanthology.org/P19-1335.

[38] Hien T Nguyen and Tru H Cao. "Named entity disambiguation: A hybrid statistical and rule-based incremental approach". In: *Asian Semantic Web Conference*. Springer. 2008, pp. 420–433.

[39] Phong Le and Ivan Titov. "Boosting Entity Linking Performance by Leveraging Unlabeled Documents". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 1935–1945. DOI: 10.18653/v1/P19-1187. URL: https://www.aclweb.org/anthology/P19-1187.

[40] Phong Le and Ivan Titov. "Improving Entity Linking by Modeling Latent Relations between Mentions". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 1595–1604. DOI: 10.18653/v1/P18-1148. URL: https://www.aclweb.org/anthology/P18-1148.

[41] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. *Neural Architectures for Named Entity Recognition*. 2016. arXiv: 1603.01360 [cs.CL].

[42] John M Giorgi and Gary D Bader. "Transfer learning for biomedical named entity recognition with neural networks". In: *Bioinformatics* 34.23 (June 2018), pp. 4087–4094. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bty449. URL: https://doi.org/10.1093/bioinformatics/bty449.

[43] Simone Magnolini, Valerio Piccioni, Vevake Balaraman, Marco Guerini, and Bernardo Magnini. "How to Use Gazetteers for Entity Recognition with Neural Models". In: *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*. Macau, China: Association for Computational Linguistics, Aug. 2019, pp. 40–49. URL: https://www.aclweb.org/anthology/W19-5807.

[44] Alex Graves and Jürgen Schmidhuber. "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural Networks* 18.5 (2005). IJCNN 2005, pp. 602–610. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2005.06.042. URL: https://www.sciencedirect.com/science/article/pii/S0893608005001206.

[45] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 2227–2237. DOI: 10.18653/v1/N18-1202. URL: https://www.aclweb.org/anthology/N18-1202.

[46] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. *Improving language understanding by generative pre-training*. 2018.

[47] Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. *XNLI: Evaluating Cross-lingual Sentence Representations*. 2018. arXiv: 1809.05053 [cs.CL].

[48] Alexis Conneau and Guillaume Lample. "Cross-lingual Language Model Pretraining". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/c04c19c2c2474dbf5f7ac4372c5b9af1-Paper.pdf.

[49] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. *Unsupervised Cross-lingual Representation Learning at Scale*. 2020. arXiv: 1911.02116 [cs.CL].

[50] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL].

[51] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. "Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 19–27. DOI: 10.1109/ICCV.2015.11.

[52] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. "One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling". In: *INTERSPEECH-2014*. 2014.

[53] Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. *OpenWebText Corpus*. http://Skylion007.github.io/OpenWebTextCorpus. 2019.

[54] Trieu H. Trinh and Quoc V. Le. *A Simple Method for Commonsense Reasoning*. 2019. arXiv: 1806.02847 [cs.AI].

[55] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: In the Proceedings of ICLR. 2019.

[56] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. "SQuAD: 100,000+ Questions for Machine Comprehension of Text". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. DOI: 10.18653/v1/D16-1264. URL: https://www.aclweb.org/anthology/D16-1264.

[57] Pranav Rajpurkar, Robin Jia, and Percy Liang. "Know What You Don't Know: Unanswerable Questions for SQuAD". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 784–789. DOI: 10.18653/v1/P18-2124. URL: https://www.aclweb.org/anthology/P18-2124.

[58] Alan Akbik, Duncan Blythe, and Roland Vollgraf. "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, Aug. 2018, pp. 1638–1649. URL: https://www.aclweb.org/anthology/C18-1139.

[59] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3982–3992. DOI: 10.18653/v1/D19-1410. URL: https://aclanthology.org/D19-1410.

[60] Alexis Conneau and Douwe Kiela. "SentEval: An Evaluation Toolkit for Universal Sentence Representations". In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), May 2018. URL: https://aclanthology.org/L18-1269.

[61] Kawin Ethayarajh. *How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings*. 2019. arXiv: 1909.00512 [cs.CL].

[62] Guillaume Chiron, Antoine Doucet, Mickaël Coustaty, Muriel Visani, and Jean-Philippe Moreux. "Impact of OCR Errors on the Use of Digital Libraries: Towards a Better Access to Information". In: *Proceedings of the 17th ACM/IEEE Joint Conference on Digital Libraries*. JCDL '17. Toronto, Ontario, Canada: IEEE Press, 2017, pp. 249–252. ISBN: 9781538638613.

[63] Kai Labusch, Preußischer Kulturbesitz, Clemens Neudecker, and David Zellhöfer. "BERT for Named Entity Recognition in Contemporary and Historical German". In: *Proceedings of the 15th Conference on Natural Language Processing, Erlangen, Germany*. 2019, pp. 8–11.

[64] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. "Distant supervision for relation extraction without labeled data". In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Asso-

ciation for Computational Linguistics, Aug. 2009, pp. 1003–1011. URL: https://www.aclweb.org/anthology/P09-1113.

[65] Michael A. Hedderich and Dietrich Klakow. "Training a Neural Network in a Low-Resource Setting on Automatically Annotated Noisy Data". In: *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*. Melbourne: Association for Computational Linguistics, July 2018, pp. 12–18. DOI: 10.18653/v1/W18-3402. URL: https://www.aclweb.org/anthology/W18-3402.

[66] Phong Le and Ivan Titov. "Distant Learning for Entity Linking with Automatic Noise Detection". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 4081–4090. DOI: 10.18653/v1/P19-1400. URL: https://www.aclweb.org/anthology/P19-1400.

[67] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. "The Stanford CoreNLP Natural Language Processing Toolkit". In: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 55–60. DOI: 10.3115/v1/P14-5010. URL: https://www.aclweb.org/anthology/P14-5010.

[68] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach". In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. ICML'11. Bellevue, Washington, USA: Omnipress, 2011, pp. 513–520. ISBN: 9781450306195.

[69] Yasumasa Onoe and Greg Durrett. *Fine-Grained Entity Typing for Domain Independent Entity Linking*. 2020. arXiv: 1909.05780 [cs.CL].

[70] P.n. Zulu, G. Botha, and E. Barnard. "Orthographic measures of language distances between the official South African languages". In: *Literator* 29.1 (2008), pp. 185–204. DOI: 10.4102/lit.v29i1.106.

[71] Roald Eiselen and Martin Puttkammer. "Developing Text Resources for Ten South African Languages". In: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, pp. 3698–3703. URL: http://www.lrec-conf.org/proceedings/lrec2014/pdf/1151_Paper.pdf.

[72] Sean Packham. "Crowdsourcing a text corpus for a low resource language". MA thesis. University of Cape Town, 2016.

[73] Yotam Eshel, Noam Cohen, Kira Radinsky, Shaul Markovitch, Ikuya Yamada, and Omer Levy. "Named Entity Disambiguation for Noisy Text". In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 58–68. DOI: 10.18653/v1/K17-1008. URL: https://www.aclweb.org/anthology/K17-1008.

[74] Maud Ehrmann, Matteo Romanello, Alex Flückiger, and Simon Clematide. "Overview of CLEF HIPE 2020: Named entity recognition and linking on historical newspapers". In: *International Conference of the Cross-Language Evaluation Forum for European Languages*. Cham: Springer International Publishing, 2020, pp. 288–310.

[75] Erik F. Tjong Kim Sang and Fien De Meulder. "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. 2003, pp. 142–147. URL: https://www.aclweb.org/anthology/W03-0419.

[76] Minghao Wu, Fei Liu, and Trevor Cohn. "Evaluating the Utility of Hand-crafted Features in Sequence Labelling". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 2850–2856. DOI: 10.18653/v1/D18-1310. URL: https://www.aclweb.org/anthology/D18-1310.

[77] Kelechi Ogueji, Yuxin Zhu, and Jimmy Lin. "Small Data? No Problem! Exploring the Viability of Pretrained Multilingual Language Models for Low-resourced Languages". In: *Proceedings of the 1st Workshop on Multilingual Representation Learning*. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 116–126. DOI: 10.18653/v1/2021.mrl-1.11. URL: https://aclanthology.org/2021.mrl-1.11.