UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Final Paper 2020

Title: **Morphological Segmentation of Low-Resource Languages Using Conditional Random Fields**

Author: **Tumi Moeng (MNGTUM007)**

Project Abbreviation: **MORPH-SEGMENT**

Supervisor(s): **Dr. Jan Buys**

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	
Theoretical Analysis	*0*	*25*	
Experiment Design and Execution	*0*	*20*	
System Development and Implementation	*0*	*20*	
Results, Findings and Conclusions	*10*	*20*	
Aim Formulation and Background Work	*10*	*15*	
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	
Total marks		**80**	

Morphological Segmentation of Low-Resource Languages Using Conditional Random Fields

Tumi Moeng
University of Cape Town Computer Science Department
Cape Town, South Africa

ABSTRACT

The Nguni languages of Southern Africa have had two types of Language Processing Applications Developed for them: Rule-based applications which have high accuracy whilst only working for a small subset of data, and Data-driven applications that tend to be low accuracy. We aim to improve the data-driven approaches. How can we facilitate the development of more high accuracy applications? Within the field of Computational Linguistics there is a study called Morphological Segmentation. This study deals with the breaking down of individual words into morphemes, the smallest unit of language with meaning. The use of morphological segmentation may facilitate the creation of better language processing applications because with little data, words could be broken down to their smallest units making it easier to determine their meaning. Low resource languages are languages for which there is little access to text data, both annotated and unannotated. Thus a model is needed that can segment and label words with high accuracy, and do that with low amounts of text data.

This task can be accomplished through the use of several Machine Learning models but the model that will be the focus of this paper is known as Conditional Random Fields (CRF). CRFs are a class of discriminative probabilistic models used to label sequence data such as text corpora and images, using Supervised Machine Learning.

Two varieties of CRFs were implemented to this end: a traditional log-linear CRF and a Bidirectional Long Short Term Memory CRF. After the implementation of both models we were able to observe average F1 scores of 98% in the segmentation task across both models. Due to a high variance it would not be representative to give the average score for the other two tasks, however, it should be noted that the Baseline CRF consistently outperformed the implemented Bidirectional Long Short Term Memory CRF.

KEYWORDS

Natural Language Processing (NLP); Conditional Random Fields (CRF); Morphological Segmentation; Surface Segmentation; Tagging Problem; Supervised Machine Learning; Nguni Languages; low-resource Languages;

1 INTRODUCTION

Morphological Segmentation is a linguistic task that involves the separating of words into their composite *morphemes*. Morphemes are the smallest possible building blocks of a language that also have meaning when standing alone[18]. This task is most useful when being applied to languages that are *agglutinative*, which have words that are solely composed of aggregating morphemes, generally without making significant alterations to the spelling of the morphemes. Obtaining these morphemes is useful because it allows for the analysis of language for further Natural Language Processing (NLP) tasks. One such task is translation; given a word, the word can be broken down to its composite morphemes and then these individual morphemes can be translated to get an approximate translation for the word itself.

This segmentation can be done manually but for the sake of efficiency there has been a move towards performing this task using both Supervised and Unsupervised Machine Learning methods, and Rule based methods. Supervised Machine Learning entails the training of a model through the use of data that is correctly labelled and over time the model will learn to label unlabelled data based upon the the information it has gained during the training phase. Unsupervised Machine Learning involves the training of a model through the use of unlabelled data. The model then needs to find patterns in the training data, and it uses the patterns it is able to discern over time to label unlabelled data. Rule based methods include the use of boolean logic and hard-coded rules to determine how words could be segmented. This paper will focus on a class of supervised machine learning models known as Conditional Random Fields (CRFs). CRFs are a class of models that use probability to determine the best sequence of labels given an input sequence, such as a sentence[10].

Low-Resource Languages are defined by a distinct lack of written data in that language, and a lack of NLP applications such as translation and speech-to-text services. The lack of data can be exacerbated by a combination of a lack of research being conducted in language and few speakers of the language. Where NLP applications do exist, they tend to be low accuracy. This project will focus on a family of low-resource Southern African languages known as the

Nguni Languages with a specific focus on isiNdebele, isiXhosa, isiZulu and siSwati. Morphological Segmentation is particularly applicable to this family of languages because they are agglutinative. This means that despite the lack of written data, if the morphemes of a word can be uncovered, NLP applications can be developed using this knowledge in conjunction with Machine Learning techniques which would mitigate the need for large corpora of written data.
We implemented a traditional CRF and a Bidirectional Long Short Term Memory CRF. With the former of the two we were able to achieve an average f1 score of 97% in the segmentation task, 77% in the labelling task, and 59% in a pipeline operation that uses both models. With the latter of the two we were able to achieve an average f1 score of 94% in the segmentation task, 24% in the labelling task, and 8% in the pipeline operation.
It is our hope that this project and the work that comes from it will show that high performance that can be achieved by these models on these low-resource languages. Additionally, we hope that this high performance will facilitate a movement towards the development of more and superior NLP applications for these languages. These models could theoretically also be used for any agglutinative low-resource language in the development of tools that could benefit speakers and students of a language, and linguists that are attempting to do research in one of these languages. Additionally, languages are important, they facilitate communication between people and ultimately allow us to connect. Given enough time these low-resource languages may eventually die out being replaced by more commonly used languages, due to the decreasing number of speakers. The preservation of these languages can be aided by NLP applications and other forms of research into them.

2 BACKGROUND INFORMATION

In this section we will discuss some information crucial to the further understanding of this project.

Morphological Segmentation

Morphological Segmentation is achieved by separating a word into its composite morphemes such that the word can be studied at a granular level. In this brief section we will outline the ways in which words can be segmented and which we will focus on.
There are two main ways in which words can be segmented, *surface segmentation* and *canonical segmentation*.

Surface Segmentation. Using **surface segmentation**, the word w will be segmented into a sequence of substrings, which when added back together will form the word w again[4].

Canonical Segmentation. Using **canonical segmentation**, the word w will be analyzed as a sequence of canonical morphemes, based on word forms that will have been annotated for Supervised Machine Learning. Each canonical morpheme c will correspond to a surface morpheme s as its Orthographic Manifestation. This means that c will be the same as s after applying editing operations such as insertion, deletion and modification[4].

Given the isiZulu word "*ngezinkonzo*":

- **Surface Segmentation**: *nge-zin-konzo*
- **Canonical Segmentation**: *nga-i-zin-konzo*

The move from the surface form to the canonical form involves, amongst other edit operations, the insertion of a morpheme "i" that does not appear in the surface form. The segments generated by canonical segmentation correspond better to the underlying morphemes used by linguists to study a language, and as such, they tend to be a better analysis tool than the segments generated by surface segmentation[4].
Once a word has been segmented, they can be further subdivided based on whether or not they are labelled. If they are to be labelled, then labels will be added to each morpheme and these labels will detail the function of that morpheme to the word as a whole. These functions can vary from being a word root, a suffix for the word or even the noun class of the word.
Due to the fact that CRFs are unable to directly predict canonical morphemes without significant extensions, this project will focus on surface segmentation and the labelling of the identified morphemes.

3 CONDITIONAL RANDOM FIELDS

CRFs are a class of discriminative probabilistic supervised machine learning models that can be used for a variety of tasks[10, 21]. This project will involve the use of several variations of the CRF in the task of morphological segmentation and then in the labelling of the identified morphemes. Discriminative Models are used in contrast to Generative Models. The difference is that Discriminative Models are able to directly model the conditional probability $p(\mathbf{Y}|\mathbf{X})$, meaning the probability of $\mathbf{Y}$ given $\mathbf{X}$[21, 24]. Additionally, they are able to learn direct mappings from an input, $\mathbf{X}$, to a label, $\mathbf{Y}$[24]. Contrarily, Generative Models make use of joint probability models of $\mathbf{X}$ and $\mathbf{Y}$, of the form $p(\mathbf{X},\mathbf{Y})$, and then used Bayesian Rules to calculate the score of a given sequence of labels[21, 24]. It is generally agreed that Discriminative Models are superior because they are able to directly compute the classifier problem without an additional middle step[24]. Probabilistic refers to the use of probability maximization in determining the optimal output sequence of labels.

CRFs accept data sequences as inputs and output label sequences. All the labels in the label sequence need to be predefined within a label alphabet. Within both varieties of CRF one will be generated for each subtask. The first is a character-level CRF which will take in a word and analyse it character by character to determine where the morphemes are. Each character will be labelled according to whether it is the start of a new morpheme, a part of the previous morpheme, the end of a previous morpheme, or a single length morpheme. The second will be a word-level CRF which will take in the morphemes generated by the first CRF and they will be labelled according to their linguistic function to the word as a whole.

This project will involve the training of CRFs on annotated training data and being optimized on a development set of data. Following the training and optimization of the CRFs both will be used to predict the morphemes and then labels of words in the test set using probabilities that will have been learnt from the training stage.

CRFs are based upon Markov Models, however their performance is typically superior due to the fact that they are able to overcome a problem faced by Markov Models known as the *label bias problem*[10]. This is a problem whereby when faced with two similar words such as *pet* and *pat* exist in a training set, however one occurs more than the other in the training set, therefore when predicting labels the model favours the transition of the more commonly occurring word meaning that the other will always be labelled incorrectly[10]. This problem exists in other state based classifiers and is a contributor to CRFs out-performance of these types of models.

For this project we will be implementing two variants of the CRF which will be discussed in more detail below. The first is the traditional log-linear CRF which will be acting as the baseline for comparison of the other models. The second is a CRF with a neural net extension, specifically known as a Bi-directional Long Short Term Memory Network CRF (Bi-LSTM-CRF).

Traditional CRFs

Given the random variable $\mathbf{X}$, the input sequence for which the CRF needs to predict the output label sequence, $\mathbf{Y}$. Given a finite set of labels, the CRF needs to be able to determine the conditional probability, for each morpheme, $p(\mathbf{Y}_i|\mathbf{X}_i)$ here meaning the probability of label $\mathbf{Y}_i$ given the morpheme $\mathbf{X}_i$. In broader terms the CRF will need to maximize the probability $p(\mathbf{Y}|\mathbf{X})$. CRFs do this taking into account the current label, $\mathbf{Y}_1$, and a previous label ,$\mathbf{Y}_2$. Finally, in order to ensure the CRF is valid it also needs to obey the Markov Property that given $p(\mathbf{Y}_1|\mathbf{X},\mathbf{Y}_2)$, $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are neighbors[10].

Mathematically CRFs compute the conditional probability of a sequence as follows: Assuming the labels of $\mathbf{Y}$ conditioned on $\mathbf{X}$ form a chain,we define $\mathbf{Y}_0$, as the start state, and $\mathbf{Y}_{n+1}$, as the stop state. Given this chain structure, a Matrix, $\mathbf{M}$, can be defined for which the score of a label sequence is given[10]. These scores will later be normalized to obtain probabilities. For each position, i, in the sequence $\mathbf{X}$, we define a |y|*|y| matrix with the random variable $M_i(x) = [M_i(y',y|x)]$ by:

$$\begin{aligned} M_i(y', y|x) &= exp\left(\Lambda_i(y', y|x)\right) \\ \Lambda_i(y', y|x) &= \sum_k \lambda_k f_k(e_i, Y|_{e_i} = (y', y), x) \\ &\quad + \sum_k \mu_k g_k(v_i, Y|_{v_i} = y, x) \end{aligned} \tag{1}$$

Here, x represents the input sequence, and y represents a label output sequence. Additionally, e_i is the edge with the labels (y_{i-1},y_i) and v_i is the vertex with the label y[10]. Finally, f_k and g_k are features of input sequence, $\mathbf{X}$, and the labels, y and y', that are assumed to be given and fixed[10]. This equation creates a matrix representing the probability of a label for a given position of the input sequence.

When using CRFs there is the need for a function, known as a *normalization (partition)* function, $Z_\theta(x)$, that will ensure the model will define a valid distribution. This comes at the cost that the probability distribution over Y is not valid till the whole sequence $\mathbf{X}$ has been processed[14]. CRFs are globally normalized meaning conditional probability, after normalization, for position i does not need to sum to one[14]. This function is the product of all the matrices for the input sequence[10]. It is defined as follows:

$$\begin{aligned} Z_\theta(x) &= (M_1(x) * M_2(x) * \ldots * M_{n+1}(x)) \\ &= (M_1(y_0, y_1|x) * M_2(y_1, y_2|x) \ldots * M_{n+1}(y_{n+1}, y_{n+1}|x)) \\ &= \prod_{i=1}^{n+1} M_i(y_{i-1}, y_i|x) \end{aligned} \tag{2}$$

Therefore, for the input sequence, $\mathbf{X}$, the probability of a label sequence, $\mathbf{Y}$, being assigned to that input sequence is given by:

$$p_\theta(Y|X) = \left(\frac{1}{Z_\theta(x)}\right) * \prod_{i=1}^{n+1} M_i(y_{i-1}, y_i|x) \tag{3}$$

Finally, θ is a parameter that needs to be determined, and it represents $(\lambda_1,\lambda_2,\ldots,\lambda_n;\mu_1,\mu_2,\ldots,\mu_n)$[10]. θ will be estimated using the training data, $D = \{(x^i, y^i)\}_{i=1}^n$, with the distribution $p(x, y)$[10]. θ is calculated as follows, using a log-likelihood objective function[10]:

$$O(\theta) = \sum_{i=1}^{N} log p_\theta(y^i|x^i) \tag{4}$$

During testing, the model makes use of a dynamic programming algorithm known as the *Viterbi Algorithm* to calculate the highest scoring label sequence. At a high level a matrix is created with one column representing one part of the input sequence, and each row representing all the possible labels. Each entry in this matrix is the probability that the label sequence will be at that label, given the previous labels. Furthermore, this value is calculated recursively by using the most probable path to get to that label[8]. Due to the use of dynamic programming this model tends to be efficient, particularly with cases where there are many labels.

Bi-LSTM CRF

This represents an extension upon traditional CRFs achieved by combining them with a Neural Network (NN). Using NN as an extension to traditional CRFs is beneficial, because unlike traditional CRFs where features have to be manually designed, NN can generate the features for the CRF layer. This happens when the NN generates context of the input sequence, and then this context is fed to the CRF as the features.

The specific NN we will be implementing is known as a Long Short Term Memory Network (LSTM). LSTMs were developed to be an improvement upon Recurrent Neural Networks (RNN), which was another class of models that operates on data sequences as their inputs[11]. LSTMs were built to counteract a problem faced by RNNs known as the *vanishing problem* whereby they are biased to recent inputs in a sequence which leads to previously established patterns being overlooked in favour of more recent ones[11, 12]. To solve the vanishing problem, LSTMs were developed with a memory cell which allowed them to more easily capture long term dependencies[11]. Normally an LSTM generates the left context of a sequence at every position, however it would be useful to also get the right context of a sequence at every position. This was achieved by adding a second LSTM that reads the sequence in reverse, the first LSTM is known as the *forward LSTM* and the second is known as the *backward LSTM*[11]. The pair are combined to create what is known as a bidirectional LSTM (BLSTM) which can create a representation of words in context from both sides known as h_t[11].

To implement this with CRFs, the Bi-LSTM is used to generate a n^*k matrix, called P, where n is the number of words in the sequence and k is the number of labels in the label alphabet. Each position P_{ij} is the score of the j^{th} label of the i^{th} word in the sequence for a given sequence of predictions[11]. The score is given by:

$$s(X, y) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i} \tag{5}$$

In this equation A is a square matrix, of dimensions n+2 where n is equal to the number of labels in the label alphabet, and the 2 added represent the *start* label and *end* label of the sequence[11]. Each position $A_{i,j}$ represents the score of the transition from label i to label j[11]. Yx represents the set of all possible distributions of labels over the input sequence[11].

The final conditional probability equation is given by:

$$p(Y|X) = \frac{e^{s(x,y)}}{\sum_{\widetilde{y} \epsilon Yx} e^{s(x,\widetilde{y})}} \tag{6}$$

This calculates the score of this sequence divided by the score of all possible sequences to determine the probability of that exact sequence of labels. During Neural CRF Training, the model attempts to maximize the log probability of a supplied correct label sequence using the equation[11]:

$$log(p(Y|X)) = s(x, y) - log\left(\sum_{\widetilde{y} \epsilon Yx} e^{s(x,\widetilde{y})}\right) \tag{7}$$

This equation creates the probabilities that will be used by the model when operating on data, and we attempt to maximize y during the training stage.

Alternative Models. Morphological Segmentation can be performed using a variety of different models. Having looked at CRFs we will now give a high-level overview of two other models. The first is another supervised machine learning model known as Sequence to Sequence Models and the second is an unsupervised machine learning model based on Morfessor Models.

Sequence to Sequence Models are an improvement of Deep Neural Networks (DNN) which are a powerful class of Machine Learning Models that tend to perform well for tasks such as translation[20]. However, they suffer from the problem that they are unable to deal with sequence inputs because DNNs require that the dimensions of the input are known fixed, but it is not possible to know how long an input is before it is processed[20]. One way in which to approach the problem is through the implementation of LSTMs, using one to read the input sequence, one time step at a time, to obtain a fixed sized vector representation of the input, and then use another LSTM to extract the output sequence from that vector[2, 20]. This architecture is known as an *Encoder-Decoder*, with the first LSTM being the encoder and the second being the decoder[2]. The Encoder takes in the input sequence, x and converts it to a vector c, known as the context vector[2]. The Decoder is trained to predict the next label given the context vector and all previously predicted labels[2]. The Decoder functions as a RNN conditioned on the input sequence and it defines a probability over the label sequence by using the conditional probability with the current label, the context vector and the context vector of

previously predicted words[20].
Morfessor models are a probabilistic model family designed specifically for Morphology Learning[5]. The task needed to be done is to induce the creation of a model of language, from only an unannotated text corpus[5]. The aim is to find the optimal model that will be able to generate a Morph vocabulary, a lexicon of morphemes, and a grammar[5]. There are two models types to be considered known as *Maximum Likelihood Models* and *Maximum Entropy Models*. Maximum Likelihood models only consider the accuracy of the representation of the data with no regard for the model complexity, like model size[5]. Unless this model is modelled with some form of restrictive model search heuristics or model smoothing, then it tends to suffer from overlearning[5]. Maximum entropy provides a framework for the estimation of probability distributions based on a set of training data[3]. The entropy of a specific distribution is a measure of uncertainty and it is maximized when the distribution is close to being a uniform distribution[3].

4 DESIGN AND IMPLEMENTATION (METHODOLOGY)

Data Preprocessing

The data that we used was the Annotated Text Corpora from The National Center for Human Technology (NCHLT)[6, 16]. As previously stated we used the isiNdebele, isiXhosa, isiZulu and siSwati corpora, however there was much preprocessing that had to be done before they were ready for use. The corpora contained the following entries on one line with each entry separated by a tab:

ngezinkonzo
khonzo
P
(RelConc)-nga[NPre]-i(NPrePre)-zin[BPre]-konzo[NStem]

The first entry represents the word itself, the second entry represents the root morpheme, the third represents the part of speech of the whole word and the fourth is the labelled canonical form of the word. We removed all elements that were just a punctuation symbol, an integer or a float because we are looking at words individually they can be ignored where they are not a part of a word. We ensured that no words in the training set existed in the test set and where they did, we removed them from the training set. We normalized all the canonical labelled forms of the words which involved making sure all the words we took forward to our final dataset had the same amount of labels as morphemes, removing front labels, combining double labels and making sure all labels use the same type of bracket. Using the word itself and the canonical form of the word we generated the surface segmented form of the word, a process which will be expanded upon below. We generated the surface labelled form of the word using the surface segmented form of the word and the canonical labelled form of the word. Finally, we generated a development set using the training set and all the entries in the development set were removed from the training set. The following are the words included in the final corpora, all entries were separated by the " | " sequence of characters:

ngezinkonzo
nge-zin-konzo
nge[NPre]zin[BPre]konzo[NStem]
nga[NPre]i[NPrePre]zin[BPre]konzo[NStem]

Here the first entry is the word itself, the second is the surface segmented form off the word, the third is the labelled surface segmented form of the word, and finally, the fourth is the labelled canonical form of the word.

Generating Surface Segmented Form. To generate the surface segmentation form of the word we created a method that takes in a word and the canonical form of the word. The first thing the method does is check to see if the lowered and de-segmented form of the canonical form is the same as the lowered form of the word. This is the simplest check and if this proves true we then return the canonical form of the word.
Following this, we generate the Levenshtein Distance edit operations to get from the de-segmented canonical form to the word. Levenshtein Distance is a python library that allows one to determine which edit operations need to be used to change one word, the source word, to another word, the destination word as follows:

Segmented Canonical: "nga-i-zin-konzo"
De-segmented Canonical: "ngaizinkonzo"
Word: "ngezinkonzo"
Edit Operations: [('delete', 2, 2), ('replace', 3, 2)]

The structure of the edit operations is as follows, the first position in the three-tuple is the operation which can only be one of "insert", "replace", "delete". The second is the position in the source word, and the third is the position in the destination word. These letter at each position is set at the start so despite there being deletion operations, the n^{th} position will always belong to a certain letter.
We found that, on average, if all the operations were replacement operations, we were able to place the dashes directly from their positions in the canonical to their place in the word. This is because no characters would be lost so the segments stay in the same place and the same size.
The final step was to go through each edit in the list of

edit operations and do them on an editable copy of the desegmented canonical form of the word and the segmented copy of the word. This would allow us to detect the deletion of entire segments and other such important operations. After all these operations we determine where the left over dashes are in the segmented canonical form and place the dashes in the same place in our word to create the surface segmentation.
We generated several numbers from which we could determine the efficiency of the method responsible for generating the surface segmented form.

- Total Words, the total number of words we have available in that particular language
- Edited Words, the number of words we had to edit to generate the surface form
- Edit Operations, the number of total edit operation performed on the words that had to be edited
- Deletion Operations, of all the edit operations, which are deletion operations
- Replacement Operations, of all the edit operations, which are replacement operations
- Segments, total number of segments in canonical forms of words that have been edited
- Matches, total number of surface segments that align with canonical segments, in words that have been edited

In the appendix, a table can be found that details the exact numbers for each language. However on average over all the languages, 45% of the words had to have the words manually edited to generate the surface form. Of all the edit operations, 38.83% of the operations were replacement operations, and the remaining 61.17% were deletion operations. Finally, of all the segments generated in the surface form, on average, 60.26% of the segments align to a segment in the canonical form making them more likely to be correct.

Model Research and Adaptation

The initial stages of the project were spent looking for implementations of each of the types of CRFs upon which to base our implementations.
We found a python library implementation of a baseline CRF known as sklearn-crfsuite that was powerful enough allowing us to achieve high scores that will be further discussed[9]. Using this model we needed to generate the features to be used by the models and then feed the features and the words to the model for the model to train upon. We decided as features for the surface segmentation to use character N-gram with N being in the range zero to six where possible, to give the model context to the word it was looking at. We also used whether the character was a vowel or a consonant, and finally whether the character was uppercase or lowercase. This is in an attempt to help the model build patterns based on the characteristics of the character. For the segment labelling we used the length of the morphemes, the beginning and end characters, whether the length of the segment was odd or even, whether the first character was uppercase or lowercase and whether the first character was a vowel or a consonant. Finally, we were able to run the words from the test set through the model and output the model predicted morphemes and morpheme labels which could be compared to the actual morphemes and morpheme labels.
For the Bi-LSTM-CRF we were able to find a useful implementation[7]. This implementation was found through github and was well suited to the first task more than the second and third. Some minor editing had to be made to the source code to make it more suitable for the task, remove some out of place print statements and remove some out-of-place code blocks. With Bi-LSTM-CRFs the Bi-LSTM component generates the features that will be used by the CRF to make predictions so all we needed to be concerned with was the implementation. To implement it all that was necessary was for us to feed all the data, unique labels and unique morphemes from the training data into separate files for the model to read from and then the model was able to learn from those files. Following this the model was instantiated in the file and we were able to feed the words in one-by-one and get the predicted output to be compared to the actual values for use in calculating the metrics.

Hyper Parameter Optimization

Each model implemented has hyper-parameters that determine the way in which they train, and consequently, the way in which they are able to predict words and the results.
For each model we created several loops with possible values for each of the hyper-parameters and each time the predicted metrics were better than the previous run we saved that set-up which allowed us to get the best possible performance for each model. The metrics we used to determine performance were *recall*, *precision* and *F1-score*. *Recall* is the ratio per label of correctly predicted positive labels to the correct labels in the test data. *Precision* is the ratio of correctly predicted labels to total predicted positive labels. Finally, *F1-score* is the geometric average of both precision and recall.
For the Baseline Model the parameters we tuned were as follows: Epsilon, the condition of convergence for the CRF. Max Iterations which is the number of iterations for algorithm to run.
There are several hyper-parameters within the Bi-LSTM-CRF however we will only be tuning the few that over the course of our testing have made the biggest difference, and the rest will maintain their default values. Number of epochs which is how many training cycles it will go through. The learning

rate which is how well the model is able to learn but having this too high may lead to over-fitting. Weight Decay, which is the normalization parameter of the CRF.
The optimal hyper parameters for each model can be found in the appendix.

5 RESULTS AND DISCUSSION

Testing

This section will detail how we tested each model and how we decided which results to use as the final results.
Following the hyper-parameter optimization and using the hyper-parameters taken from this process we ran the models on each task 5 times and took the averages of our chosen metrics across all the runs as the final results. As previously stated our chosen metrics were precision, recall and f1-score. We calculated these using the metrics module within sci-kit learn which contain methods that, given a list of the predicted labels and the actual labels was able to give us a score for each of the metrics. Using this class also meant that each score was computed in the same manner making them more easily comparable. For the labelled segmentation we created a tool that would allow us to calculate these measures ourselves such that they could deal with the potential of more or less segments in one of the inputs. The reason for this is that the segmentation models are unlikely to perform perfectly and as such there is a high possibility that the model will predict more, or less, segments than are actually present in the word.

Results

Laid out below, are the results of each model in the task of morphological segmentation, labelling of the correct segments, and a pipeline task which involves the labelling of segments generated by the first CRF, and these labels are compared to the correct labels.

Model	Language	Precision	Recall	F1 Score
Baseline CRF	isiNdebele	97.94	96.62	97.27
	isiXhosa	97.16	97.13	97.14
	isiZulu	97.88	96.82	97.35
	siSwati	97.17	96.40	96.78
Bi-LSTM-CRF	isiNdebele	96.59	96.21	96.40
	isiXhosa	94.88	95.61	95.24
	isiZulu	96.64	96.64	96.64
	siSwati	90.59	91.48	91.03

Table 1: Average Results from Surface Segmentation Task

This table shows the results for the two models in the task of surface segmentation. As can be seen for each model, the scores are fairly similar.

Model	Language	Precision	Recall	F1 Score
Baseline CRF	isiNdebele	77.07	78.24	77.65
	isiXhosa	71.16	71.50	71.33
	isiZulu	71.82	72.11	71.96
	siSwati	84.69	90.05	87.29
Bi-LSTM-CRF	isiNdebele	20.99	20.99	20.99
	isiXhosa	22.82	22.13	22.47
	isiZulu	25.07	25.07	25.07
	siSwati	29.14	31.34	30.20

Table 2: Average Results from Surface Segment Labelling Task using Correct Segments

This table shows the results for the two models in the task of labelling. This is specifically labelling of the correct segments. For the baseline model the scores are more similar, however note the outliers of siSwati and isiNdebele. The Bi-LSTM scores were not as good as we had predicted, and this shall be further discussed in the following section.

Model	Language	Precision	Recall	F1 Score
Baseline CRF	isiNdebele	51.16	55.14	53.08
	isiXhosa	53.31	57.57	55.36
	isiZulu	52.01	55.31	53.61
	siSwati	76.29	73.31	74.77
Bi-LSTM-CRF	isiNdebele	5.38	4.65	4.99
	isiXhosa	7.13	6.43	6.76
	isiZulu	7.82	7.22	7.51
	siSwati	13.51	12.31	12.88

Table 3: Average Results from Pipeline Task

This table shows the results for the two models in the pipeline task. In regards to the baseline model, these scores are similar with the exception of the outlier that is siSwati which is likely for the same reason as in the labelling task. The Bi-LSTM-CRF delivered disappointingly low results here that will be expanded upon.

Discussion

In this section we will discuss the results of the varieties of CRFs and what the results mean in comparison to one another. Additionally, we will do a comparison to the two other methods mentioned above.
The baseline CRF yielded state-of-the-art performance in the surface segmentation task with an average F1 score of 97.13%. In the task of labelled segmentation it performed admirably, giving an average F1 score of 77.06%. It should be noted that siSwati was a positive outlier yielding an F1 score of 87.9% and we determined that this was caused by

the fact that whilst the other three languages had 275 labels, on average, in their training sets, siSwati had only 105. Ndebele was also a positive outlier here with an F1 score of 77.65%, however, as this language was very similar to the others the cause was indeterminable. Finally, in the pipeline task, the model yielded an average F1 score of 59.21%, with siSwati being an outlier again with an F1 score of 74.77%. We suspect that this is for the same reason as above because of the smaller pool of labels that can be assigned to each segment. All in all, these scores show the strengths of this model and how well it is able to perform, however, because the problems with the labelled model cascade to the pipeline task, the performance is heavily affected. These high results prove the efficiency of the baseline CRF, however as they are the baseline results they will be harder for the other CRF to overcome. These results could perhaps be further improved through the addition of more features.

The Bi-LSTM-CRF also yielded state of the art performance in the surface segmentation task yielding an average F1 score of 93.81%. In the task of labelled segmentation the performance left a lot to be desired. Despite significant time spent optimizing the hyper parameters, the model struggled to achieve an F1 score of above 30%. The average achieved F1 score in this task was 24% which was disappointing and we have determined that this must be due to some inefficiency in the implementation chosen that we were not able to overcome. Due to the difficulties in achieving highs score with the model, we resorted to optimise the hyper parameters per language and these will be found in the appendix. The results of siSwati were notably higher than the rest, and we suspect that this is caused by the lower amount of labels in this dataset. Unfortunately due to the nature of the pipeline task, a low score such as this cascades into it and affects performance. As such we were not surprised to see low scores in this section. The pipeline task achieved a disappointingly low average F1 score of 8.04%. siSwati was higher than the rest and this is likely caused by the same reason as above. The results of the Bi-LSTM-CRF were disappointingly low in the second and third tasks.

Laid out below are the F1 scores achieved by the creators of the NCHLT data set in the task of surface segmentation[6]:

- **isiNdebele:** 82.26%
- **isiXhosa:** 84.66%
- **isiZulu:** 85.19%
- **siSwati:** 83.42%

As can be seen, both the baseline CRF and the Bi-LSTM-CRF outperformed their model by a significant margin with the exception of the siSwati score for the Bi-LSTM-CRF which is only a 2% improvement. These models use a rule based approach to their segmentation and had more data than the CRF models implemented, however, despite this, the CRFs were able to represent a vast improvement on their scores. This goes to further demonstrate the strengths that CRFs can have when implemented in the correct task.

In the task of surface segmentation, the Unsupervised Machine Learning Morfessor based models scored as follows for each language in terms of F1 score:

- **isiNdebele:** 20.98%
- **isiXhosa:** 28.10%
- **isiZulu:** 21.70%
- **siSwati:** 40.02%

This represents an average F1 score of 27.70% which is lower than the average scores achieved by both the baseline and Bi-LSTM-CRF in this task. From this we can determine that the CRFs implemented are more powerful than the Morfessor models in this task, all other factors being the same.

We decided to put the Sequence to Sequence models to the task of Canonical Segmentation, so the scores achieved cannot be directly compared, however it should be noted that in the completion of this task the Sequence to Sequence models were able to achieve an average precision of 75.58%, and average recall of 69.76% and an average F1 score of 72.54%.

6 CONCLUSIONS AND FUTURE WORK

Future Work

In future, we would like to try alter the features of the labelling model in an attempt to get more performance out of it, as this would likely lead to the improvement of the pipeline model's performance as well. We would also like to find or create a suitable semi-markov crf implementation as a further comparison of results. Given the results for the secondary and tertiary tasks, it might be beneficial to find another Bi-LSTM-CRF model implementation in the hopes of getting higher scores. Finally, we may also want to find an alternate, perhaps larger, database. Many labels in the test set did not exist in training set and as such would be impossible for it to predict, and additionally, after pre-processing all data sets were made significantly smaller, so a larger set may be better.

Conclusions

At the beginning of the project timeline we set out to determine whether or not CRFs were an appropriate tool for morphological segmentation. As can be seen from the results above, not only are CRFs an appropriate tool for this task, but they are an exceptional tool that are able to deliver state of the art-performance. In the task of surface segmentation the baseline model is able to achieve an average F1 score of 97.13 percent, and the Bi-LSTM-CRF giving average scores of 93.81%. This performance shows the strengths of CRFs.

We also believed that the extension of the CRF through the addition Bi-LSTM would allow for the model to achieve higher scores, which was disproved. The baseline model consistently performed better than the Bi-LSTM-CRF.
As well as performing well in the task of surface segmentation, the baseline model is also able to achieve good results in the task of labelling and in the pipeline task which was impressive given how much more computationally complex this task is. Contrarily, the Bi-LSTM-CRF model performed worse than the baseline CRF which we expect may be caused by inefficiencies in the implementation. In the task of surface segmentation it performs only 3% average on worse which still represents good performance, but disappointingly, it was not able to outperform the baseline CRF. In the task of labelling it performs significantly worse than the baseline, and this performance extends to the pipeline operation.
In the task of surface segmentation, both models were able to outperform the Morfessor Based Models by a significant margin which speaks positively to their strengths.
Having shown the strengths of these models, we conclude that it would be appropriate to use them in the further development of language processing applications in the low resource languages mentioned, and perhaps even other agglutinative languages, given enough data. We say this with particular reference to the baseline CRF model

7 ACKNOWLEDGEMENTS

The author thanks his family for their support throughout his studies. He thanks his friends for helping him get through it all. He thanks Jessica for everything. He thanks his project partners, Aaron and Sheldon, for their contribution, all the interesting work related conversations, and the late night work sessions. He also thanks Dr Jan Buys for all the advice and help throughout the project timeline. He also thanks and congratulates his computer for surviving all of this. This work is based on research supported in part by the National Research Foundation of South Africa (Grant Number: MND190922478537).

REFERENCES

[1] Andrew, G. (2006). A hybrid Markov/semi-Markov conditional random field for sequence segmentation. COLING/ACL 2006 - EMNLP 2006: 2006 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, (July), 465–472. https://doi.org/10.3115/1610075.1610140

[2] Bahdanau, D. et al. 2015. Neural machine translation by jointly learning to align and translate. 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings. (2015), 1—15.

[3] Blei, D. M., Ng, A. Y., Jordan, M. I., Wallach, H. M., Hinton, G. E., Osindero, S., & Teh, Y.-W. (2004). Conditional random fields: An introduction. Neural Computation, 18(4–5), 1–9. https://doi.org/10.1162/jmlr.2003.3.4-5.993

[4] Cotterell, R., Vieira, T., & Schütze, H. (2016). A joint model of orthography and morphological segmentation. 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference, 664–669. https://doi.org/10.18653/v1/n16-1080

[5] Creutz, M. and Lagus, K. 2007. Unsupervised Models for Morpheme Segmentation and Morphology Learning. ACM Transactions on Speech and Language Processing. 4, 1 (2007), 1—34. DOI:https://doi.org/10.1145/1187415.1187418.

[6] Eiselen, R. and Puttkammer, M. 2013. Developing Text Resources for Ten South African Languages. 3698–3703.

[7] Jidasheng. bi-lstm-crf. Retrieved from https://github.com/jidasheng/bi-lstm-crf

[8] Jurafsky, D. and Martin, J. 2019. Ch. 8: Part-of-speech tagging. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.

[9] Korborov, M. 2015. sklearn-crfsuite. Retrieved from https://sklearn-crfsuite.readthedocs.io/en/latest/

[10] Lafferty, J., & Mccallum, A. (2014). Conditional Random Fields. Computer Vision, 2001(June), 146–146. https://doi.org/10.1007/978-0-387-31439-6-100233

[11] Lample, G. et al. 2016. Neural architectures for named entity recognition. 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference (2016), 260—270.

[12] Ma, X., & Hovy, E. (2016). End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers, 2, 1064–1074. https://doi.org/10.18653/v1/p16-1101

[13] Muller, T., Schmid, H., & Schutze, H. (2013). Efficient higher-order CRFs for morphological tagging. EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, (October), 322-332.

[14] Murphy, K.P. 2012. Undirected Graphical Models (Markov Random Fields). Machine Learning: A Probabilistic Perspective. 661—705.

[15] Mzamo, L. et al. 2019. Evaluation of combined bi-directional branching entropy language models for morphological segmentation of isiXhosa. CEUR Workshop Proceedings (2019), 77-–89.

[16] NCHLT Speech Corpus: https://sites.google.com/site/nchltspeechcorpus/

[17] Peng, F., Feng, F., & McCallum, A. (2004). Chinese segmentation and new word detection using conditional random fields. 562–568.https://doi.org/10.3115/1220355.1220436

[18] Ruokolainen, T., Kohonen, O., Virpioja, S., & Kurimo, M. (2013). Supervised morphological segmentation in a low-resource learning setting using conditional random fields. CoNLL 2013 - 17th Conference on Computational Natural Language Learning, Proceedings, (2008), 29–37.

[19] Sha, F., & Pereira, F. (2003). Shallow parsing with conditional random fields. (June), 134–141. https://doi.org/10.3115/1073445.1073473

[20] Sutskever, I. et al. 2014. Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems (2014), 3104—3112.

[21] Sutton, C., & McCallum, A. (2011). An introduction to conditional random fields. Foundations and Trends in Machine Learning, 4(4), 267–373. https://doi.org/10.1561/2200000013

[22] Sutton, C., McCallum, A., & Rohanimanesh, K. (2007). Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. Journal of Machine Learning Research, 8, 693–723.

[23] Tseng, H., Chang, P., Andrew, G., Jurafsky, D., & Manning, C. (2005). A Conditional Random Field Word Segmenter for Sighan Bakeoff 2005. Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing, 168–171. Retrieved from http://www.aclweb.org/anthology/I05-3027

[24] Xue, J.H. and Titterington, D.M. 2008. on discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. Neural Processing Letters. 28, 3 (2008), 169-–187. DOI:https://doi.org/10.1007/s11063-008-9088-7.

[25] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Torr, P. H. S. (2015). Conditional random fields as recurrent neural networks. Proceedings of the IEEE International Conference on Computer Vision, 2015 International Conference on Computer Vision, ICCV 2015, 1529–1537. https://doi.org/10.1109/ICCV.2015.179

A SURFACE SEGMENTATION METRICS

Language	Total Words	Edited Words	Edit Ops	Deletion Ops	Replacement Ops	Segments	Matches
isiNdebele	15200	7961	15868	9103	6764	32734	19348
isiXhosa	19282	11594	24072	15124	8948	47265	27219
isiZulu	20634	11424	23105	14054	9051	46520	26619
siSwati	16310	1978	2201	1401	800	4061	2726

Table 4: Average Results from Surface Segmentation Task

B HYPER-PARAMETERS

Baseline CRF Hyper Parameters

- **Epsilon:** 1e-07
- **Max Iterations:** 160

Bi-LSTM-CRF Hyper Parameters

isiNdebele and siSwati Best Performance.

- **Epochs:** 20
- **Learning Rate:** 4e-4
- **Weight Decay:** 0.0

isiXhosa and isiZulu Best Performance.

- **Epochs:** 20
- **Learning Rate:** 9e-4
- **Weight Decay:** 0.0