[Comp703] Assignment 1 – HMM vs LCCRF

Compiled by: Sashen Moodley (219006946)

1 - Introduction

The aim of this report is to validate or invalidate the general view that Conditional Random Fields (CRF) perform better than Hidden Markov Models (HMM). The testing was performed as a Part of Speech (POS) sequence labelling task, using a provided "*isiZuluPOSData*" corpus. This report presents the data pre-processing activities and the steps taken in training and evaluating the respective HMM and CRF models. The remainder of this report goes through the comparison results, personal observations and deductions, and concluding remarks.

2- Data Pre-processing

Data cleaning in machine learning refers to the technique of preparing (cleaning and organizing), the raw data to make it more suitable for building and training machine learning models.[1]

There are five general steps taken in data pre-processing which I applied in conducting my research:

1. Acquire the data Set – An "*isiZuluPOSData*" file was conveniently provided to me by Mr. Edgar Jembere, which I formatted into a CSV format for easier manipulation. This file contained isiZulu words with their respective human-tagged Part of Speech and their morphological segmentation.
2. Import the crucial libraries – For the task of data pre-processing I utilized the commonly used libraries: NumPy and Pandas
3. Import the data set – The isiZulu corpus was loaded into a Pandas Data Frame for easy manipulation, where empty rows were conveniently omitted. I then extracted the sentences from the data frame in the form a list of tuples, with each tuple representing a (word, label) structure. This was the most appropriate format of the data to be fed into the models. For the CRF, I decided to omit the provided morphological segmentation, in favour of a custom feature set described later.
4. Identify and handle missing values – I noticed that certain rows, not originally treated as blank, were problematic in that they had no word nor POS tag yet had a morphological segmentation element. There were around 13 of these problematic rows in the corpus which were pruned. Failure to do so would lead to undesirable results as the blank word-label pairings would be represented as NaN values.
5. Split the data set – For both the HMM and CRF, an 80:20 split was made between the training data and test data respectively. The training and testing elements, in the form of sentences, were also randomized between executions and their results observed. The training and testing data will be used for model training and evaluation respectively.

3 - Model Training

Model training in machine language is the process of feeding an ML algorithm with data to help identify and learn good values for all attributes involved. There are several types of machine learning models, of which the most common ones are supervised and unsupervised learning.[2]

As I am using a labelled corpus, the leaning took place in a supervised manner (I.e., *Supervised Learning*). The Hidden Markov Model was created via a manual implementation, whereas the Conditional Random Field was created from the *sklearn* framework.

3.1 – Hidden Markov Model (HMM)

The labelled training data is used to define the components of the HMM, namely:

- States (Q) – All of the 16 possible labels {'PUNC', 'ADJ', 'CDEM', 'POSS', 'COP', 'PRO', 'NUM', 'CONJ', 'V', 'IDEO', 'N', 'REL', 'ADV', 'FOR', 'ABBR', 'INT'}
- Observations (O) – These come from the vocabulary of over 4500 unique words in our corpus

- Transition Probabilities (Matrix A) – A 2-D 16x16 matrix representing the probability of moving from one state to another
- Observation likelihoods (Matrix B) – A 2-D matrix representing the probability of a word being emitted from a given state
- Initial probability vector – List of initial probabilities assigned to a state, used within the Viterbi algorithm

As we can see, the HMM is trained based on counting principles where we calculate *transmission* and *emission* probabilities from our isiZulu training data.

With our training data passed in, with the appropriate tuple format, all these components are realized thus formulating a complete HMM which we can use to make predictions/inferences on unseen data.

3.2 – Conditional Random Field (CRF)

Utilizing the *sklearn* framework, the training data was split into two structures:

- Train_x – the words within our training sentences, represented as a set of features. More specifically, feature function with different weights on the features which we hope to optimise to create the best inferences.
- Train_y – the respective labels for the words (now represented as features)

The CRF is instantiated with the *Gradient Decent* using *L-BFGS* method. *Gradient decent* is an *optimization algorithm* for minimizing the *loss function*, and *BFGS* is a particular version of this convex optimization[3] routine. I set the maximum iterations for the *optimization algorithm* to 100, which is its recommended setting. After experimenting, I found setting the C1 and C2 coefficients to 0.2 and 0.3 respectively for the L1 and L2 regularization to work best when defining the CRF.

The training data is then fitted to this model where the parameters and weights are learned for future inferences.

4 – Model Evaluation

4.1 – Hidden Markov Model (HMM) Metrics and Results

For model evaluation using a manually implemented HMM, I passed through, for each sentence, the list of words without their labels into the Viterbi optimization algorithm. The Viterbi algorithm then given a prediction/inference on the most likely tag sequence.

From this, the basis of evaluation is established into two broad categories: Time and Accuracy

For time, a measure is made on the amount of time taken to run the Viterbi algorithm, that is to make a POS label prediction on a number of testing sentences. Due to the nature of the HMM, being a generative model, and having over 600 training sentences, it would take a long time to run the algorithm on all testing sentences. However, I have noted this time metric for various training subset lengths over 5 separate executions and averaging the times observed, the results of which have been summarised in the table below:

Number of Sentences	Number of Words	Average Time Taken (in seconds)
10	156	7.13
20	296	13.98
50	895	41.32
100	1614	73.54
300	4857	221.55
603 (all testing sentences)	9626	440.45

For accuracy, a flat score is made on the number of correctly predicted labels over the total number of words in the training sequence. Similarly, to the times metrics, an average accuracy over 10 separate executions on varying testing subset lengths were noted. The results have been summarised in the table below:

Number of Sentences	Number of Words	Average Accuracy (%)
10	156	74.34
20	296	71.28
50	895	80.56
100	1614	80.61
300	4857	79.93
603 (all testing sentences)	9626	80.43

4.2 - Conditional Random Field (CRF) Metrics and Results

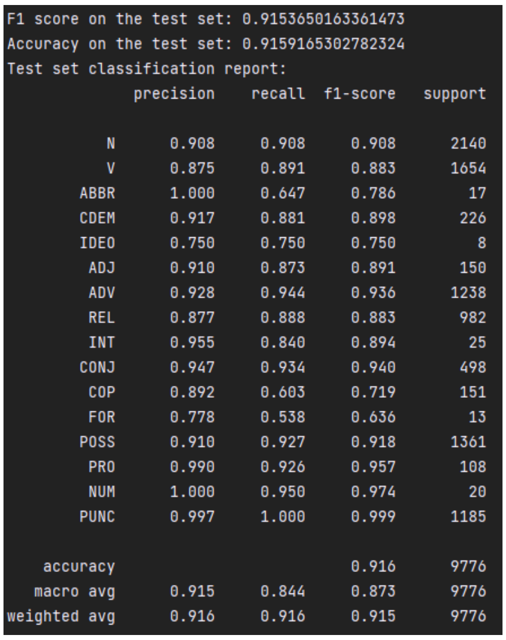
Similarly, two broad metric categories for CRF evaluation can be established: Time and Accuracy. Time and accuracy indicators were noted for both training and testing periods in the CRF.

For time metrics, the time taken to fit our training data to the model was observed as well as the time taken to make inferences/predictions on the testing data. Ten separate observations were made around these time metrics, and unlike the HMM, the testing was performed on all 603 testing sentences for each testing time observation. The results are summarised in this table below:

		Execution number									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Time metric	**Training Time (in seconds)**	15.4	14.99	14.9	14.96	15.16	14.85	15.1	14.8	14.97	15.2
	Testing Time (in seconds)	0.18	0.18	0.15	0.16	0.16	0.18	0.17	0.19	0.17	0.15

This returns an average training time of 15.03 seconds and an average testing time of 0.17 seconds.

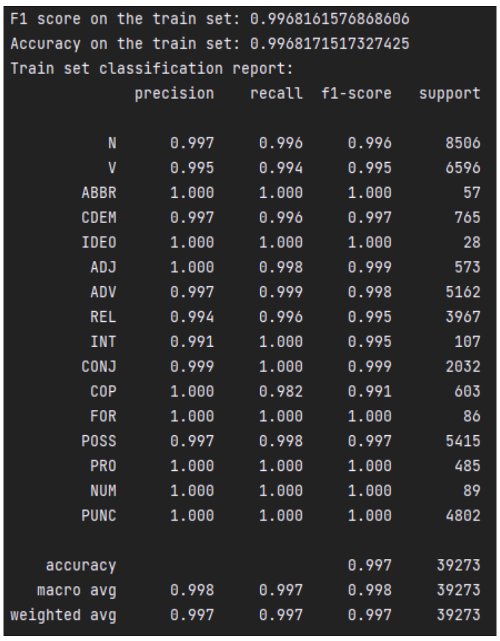
In the dimension of accuracy, I took note of the weighted average F1 and flat accuracy statistic. Additionally, *sklearn* provides a very convenient accuracy report detailing various label parameters. A snapshot of these accuracy metrics are presented below, which are followed by an accompanying description, particularly looking at the *sklearn* label accuracy report:



```
F1 score on the test set: 0.9153650163361473
Accuracy on the test set: 0.9159165302782324
Test set classification report:
              precision    recall  f1-score   support

           N      0.908     0.908     0.908      2140
           V      0.875     0.891     0.883      1654
        ABBR      1.000     0.647     0.786        17
        CDEM      0.917     0.881     0.898       226
        IDEO      0.750     0.750     0.750         8
         ADJ      0.910     0.873     0.891       150
         ADV      0.928     0.944     0.936      1238
         REL      0.877     0.888     0.883       982
         INT      0.955     0.840     0.894        25
        CONJ      0.947     0.934     0.940       498
         COP      0.892     0.603     0.719       151
         FOR      0.778     0.538     0.636        13
        POSS      0.910     0.927     0.918      1361
         PRO      0.990     0.926     0.957       108
         NUM      1.000     0.950     0.974        20
        PUNC      0.997     1.000     0.999      1185

    accuracy                          0.916      9776
   macro avg      0.915     0.844     0.873      9776
weighted avg      0.916     0.916     0.915      9776
```



```
F1 score on the train set: 0.9968161576868606
Accuracy on the train set: 0.9968171517327425
Train set classification report:
              precision    recall  f1-score   support

           N      0.997     0.996     0.996      8506
           V      0.995     0.994     0.995      6596
        ABBR      1.000     1.000     1.000        57
        CDEM      0.997     0.996     0.997       765
        IDEO      1.000     1.000     1.000        28
         ADJ      1.000     0.998     0.999       573
         ADV      0.997     0.999     0.998      5162
         REL      0.994     0.996     0.995      3967
         INT      0.991     1.000     0.995       107
        CONJ      0.999     1.000     0.999      2032
         COP      1.000     0.982     0.991       603
         FOR      1.000     1.000     1.000        86
        POSS      0.997     0.998     0.997      5415
         PRO      1.000     1.000     1.000       485
         NUM      1.000     1.000     1.000        89
        PUNC      1.000     1.000     1.000      4802

    accuracy                          0.997     39273
   macro avg      0.998     0.997     0.998     39273
weighted avg      0.997     0.997     0.997     39273
```

Precision – ratio between number of correctly recognised terms and all the recognised terms

Recall – Ratio between a number of correctly recognized terms and all the correct terms

F1-Score – weighted harmonic mean of recall and precision

As shown above a 99.7% accuracy has been achieved on the training data, and a 91.6% accuracy has been achieved on the testing data.

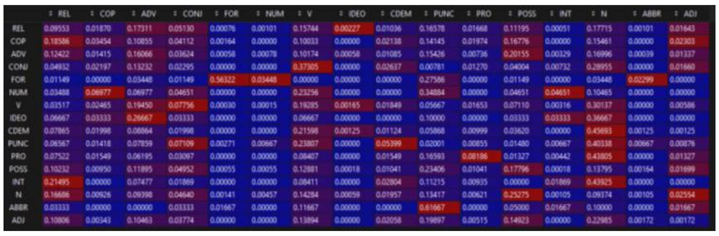
5 – Observations and Deductions

From the time and accuracy metrics, along with their results, we can note a few main observations around training time, testing time, and inference/prediction accuracy:

1. The HMM is significantly faster to train than the CRF
2. The CRF is significantly faster than the HMM in making inferences/predictions
3. The CRF produced more accurate inferences than the HMM

Diving into this deeper, we can try and gain an understanding of these observations. The underpinning differentiating factor is that the HMM is a *generative model* whilst the CRF is a *discriminative model.* The HMM is trying to model the joint distribution P(y,x), as such the model is being built on each possible class based on the training examples for each class. The CRF tries to model P(y|x) directly, in that it tries to learn which features from the training examples are most useful in discriminating between the different possible classes. From this we can start to gain an understanding of the main observations.

Firstly, from *point 1* listed above, in understanding why the HMM is faster than the CRF in training. With supervised learning, the HMM is trained based on counting principles. These counts are stored in its *emission likelihood* and *transitional* tables (such as in the alongside figure). Whereas Linear Chain CRFs are trained based on convex optimization[3] algorithms such as the *Gradient Decent* using *L-BFGS* method that was employed in my CRF, which yields calibrated probability estimates and good handling of imbalanced data.



	REL	COP	ADV	CONJ	FOR	NUM	V	IDEO	CDEM	PUNC	PRO	POSS	INT	N	ABBR	ADJ
REL	0.09553	0.01870	0.17311	0.05130	0.00076	0.00101	0.15744	0.00227	0.01036	0.16578	0.01668	0.11195	0.00051	0.17715	0.00101	0.01643
COP	0.18586	0.03454	0.10855	0.04112	0.00164	0.00000	0.10033	0.00000	0.02138	0.14145	0.01974	0.16776	0.00000	0.15461	0.00000	0.02303
ADV	0.12422	0.01415	0.16066	0.03624	0.00058	0.00078	0.10174	0.00058	0.01085	0.15426	0.00736	0.20155	0.00329	0.16996	0.00039	0.01337
CONJ	0.04932	0.02197	0.13232	0.02295	0.00000	0.00000	0.37305	0.00000	0.02637	0.00781	0.01270	0.04004	0.00732	0.28955	0.00000	0.01660
FOR	0.01149	0.00000	0.03448	0.01149	0.56322	0.03448	0.00000	0.00000	0.00000	0.27586	0.00000	0.01149	0.00000	0.03448	0.02299	0.00000
NUM	0.03488	0.06977	0.06977	0.04651	0.00000	0.00000	0.23256	0.00000	0.00000	0.34884	0.00000	0.04651	0.04651	0.10465	0.00000	0.00000
V	0.03517	0.02465	0.19450	0.07756	0.00030	0.00015	0.19285	0.00165	0.01849	0.05667	0.01653	0.07110	0.00316	0.30137	0.00000	0.00586
IDEO	0.06667	0.03333	0.26667	0.03333	0.00000	0.00000	0.06667	0.00000	0.00000	0.10000	0.00000	0.03333	0.03333	0.36667	0.00000	0.00000
CDEM	0.07865	0.01998	0.08864	0.01998	0.00000	0.00000	0.21598	0.00125	0.01124	0.05868	0.00999	0.03620	0.00000	0.45693	0.00125	0.00125
PUNC	0.06567	0.01418	0.07859	0.07109	0.00271	0.00667	0.23807	0.00000	0.05399	0.02001	0.00855	0.01480	0.00667	0.40338	0.00667	0.00876
PRO	0.07522	0.01549	0.06195	0.03097	0.00000	0.00000	0.08407	0.00000	0.01549	0.16593	0.08186	0.01327	0.00442	0.43805	0.00000	0.01327
POSS	0.10232	0.00950	0.11895	0.04952	0.00055	0.00055	0.12881	0.00018	0.01041	0.23406	0.01041	0.17796	0.00018	0.13795	0.00164	0.01699
INT	0.21495	0.00000	0.07477	0.01869	0.00000	0.00000	0.08411	0.00000	0.02804	0.11215	0.00935	0.00000	0.01869	0.43925	0.00000	0.00000
N	0.16686	0.00926	0.09398	0.04640	0.00141	0.00457	0.14284	0.00059	0.01957	0.13417	0.00621	0.25275	0.00105	0.09374	0.00105	0.02554
ABBR	0.03333	0.00000	0.00000	0.03333	0.01667	0.00000	0.11667	0.00000	0.00000	0.61667	0.00000	0.05000	0.01667	0.10000	0.00000	0.01667
ADJ	0.10806	0.00343	0.10463	0.03774	0.00000	0.00000	0.13894	0.00000	0.02058	0.19897	0.00515	0.14923	0.00000	0.22985	0.00172	0.00172

Snapshot of transition probability matrix used by HMM

Secondly, even though both the HMM and Linear Chain CRF use the Viterbi decoding algorithm, CRF achieves results faster because it only models the distribution over the labels and producing a prediction based on the most discriminative aspects (*discriminative model).* Contrast to the HMM which tries to model the joint distribution, trying to indicate how the data was generated (*generative model).*

Thirdly, our LCCRF seems to produce more accurate predictions due to its representation of words as weighted feature functions, thereby characterizing the data and allowing it to learn more about sequential dependencies. Whereas our HMM is somewhat more static in nature as it does not take into consideration aspects such as the positioning of the word, its prefix, suffix, etc or capturing interactions between observations.

6 – Conclusion

In conclusion, it is safe to say that my research validates the belief that Linear Chain CRFs are better than HMMs, especially when looking at sequence labelling tasks where an emphasis is placed on classification rather than generating samples. Apart from the lengthy training times, LCCRFs proved to be a better model for sequence labelling tasks.

References

1. Goyal K. Data Preprocessing in Machine Learning: 7 Easy Steps To Follow [Internet]. upGrad blog. 2021 [cited 2022 Apr 1]. Available from: https://www.upgrad.com/blog/data-preprocessing-in-machine-learning/

2. What is Model Training [Internet]. Oden Technologies. [cited 2022 Apr 1]. Available from: https://oden.io/glossary/model-training/

3. Kumar A. Convex optimization explained: Concepts & Examples [Internet]. Data Analytics. 2021 [cited 2022 Apr 1]. Available from: https://vitalflux.com/convex-optimization-explained-concepts-examples/