HG2051 Group 3

Alyssa Lim (U2330523D), Adoncia Ho (U2330644K), Lim Zhe Xun (U2121481J)

Introduction

A core task in Natural Language Processing (NLP) is part of speech tagging, which refers to the task of identifying the word class of a particular lexical item in a sentence. Part of speech tagging serves as an important link between language and downstream tasks such as machine translation, named entity recognition, and information extraction. This project aims to develop an automatic part of speech tagger for the Pnar language, a low-resource language originating from India.

Objectives

This project aims to achieve the following objectives:

1. Process and perform simple analysis on the Pnar dataset
2. Develop and evaluate part of speech taggers
3. Implement rules/features to improve on the baseline tagger

Repository Structure

This section describes the key files and directories in the repository.
utils.py - A python file containing utility functions for loading and preprocessing data
project2.py - Python code to implement and evaluate our best performing tagger
project2.ipynb - Notebook containing implementations, results, and analysis of different taggers

Data

The format of data in this project is similar to project 1. Instead of 5 levels of interlinear search, this project focuses on the “\mb” and “\ps” tags. In addition to the data files provided at the start of the project, we incorporated data from the original dataset (Hiram, 2017). This data incorporates 5895 sentences and 89868 (mb, pos) pairs. In total, there are 50 unique pos tags and 2595 unique Mbs.

Preprocessing

To extract the words and their corresponding part of speech from the data file, we make use of the fact that each line starts with their respective label. To split the text into individual clips, we used Python’s split function to separate text based on the “\ref” string. Iterating through each chunk of text generated from the split, we used Pythons’s startswith function to extract lines starting with “mb” and “ps”. Finally, we used list comprehension to combine the features together as tuples. This step is essential for implementing taggers with the nltk library. Our final data structure is a list of lists of tuples. Each tuple represents (mb, ps) pairs; the inner list represents sentences; and the outer list represents the whole data file.

Exploratory Data Analysis

Before developing a part of speech tagger, we first explore the train data and visualize the frequency distribution of Mb and part of speech. Figure 1 illustrates the distribution of part of speech tags, showing that verbs and clitics are the most commonly occurring parts of speech in the Pnar language. On the other hand, figure 2 shows the distribution of the top 30 most frequent Mb. We observe that Mbs ending with "=" occur very often in the dataset, and thus hold higher statistical significance.

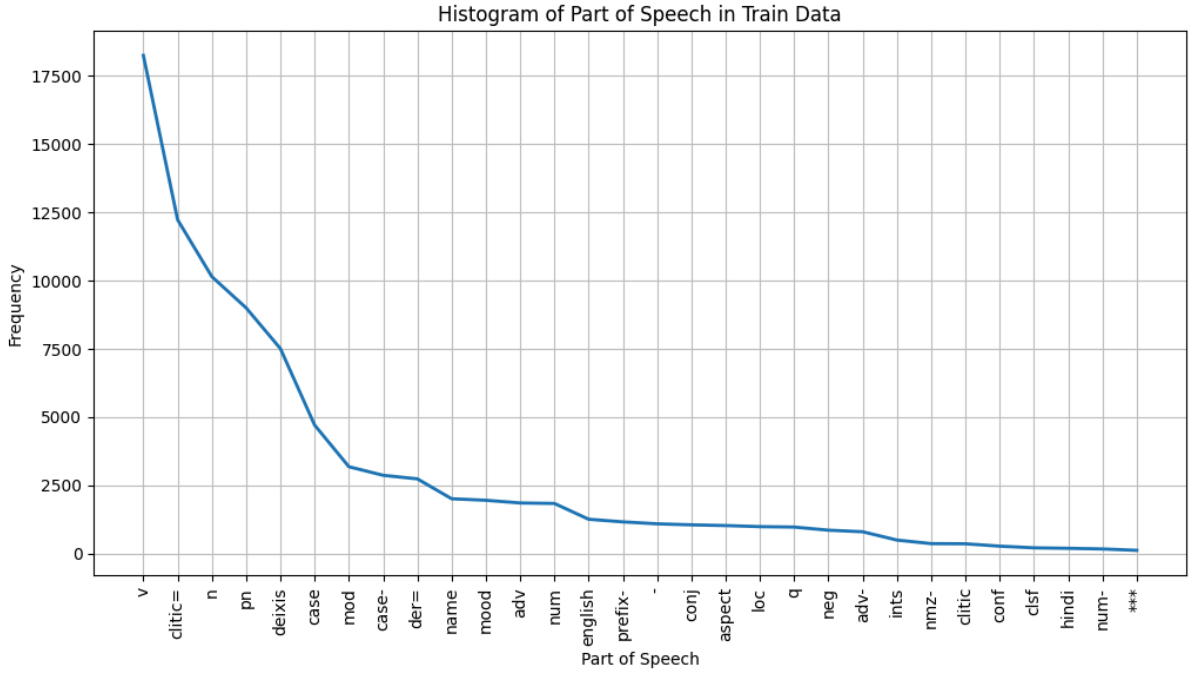

Fig. 1: Histogram of part of speech tags in train data

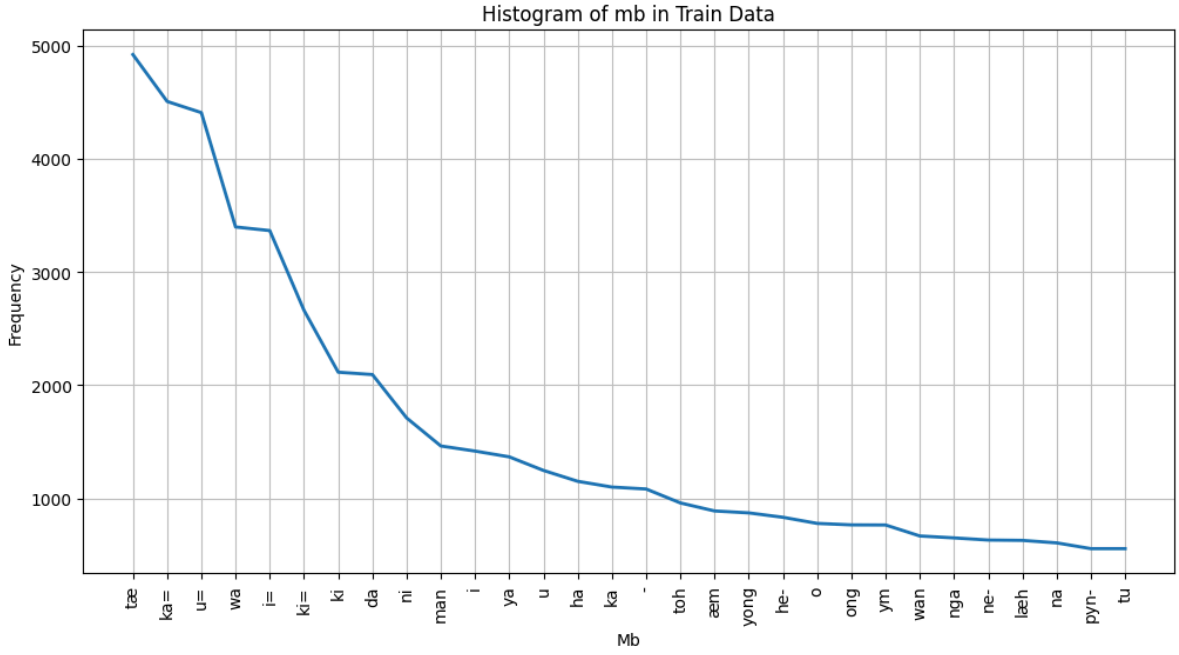

Fig. 2: Histogram of "mb" in train data

Experiments

Evaluation Metrics

To evaluate our taggers, we use the accuracy metric. In the context of part of speech tagging, this metric measures the proportion of parts of speech correctly labeled by the tagger.

$$Accuracy = Correctly\ labeled\ POS\ /\ Total\ POS$$

Volume of Training Data

We first study how the volume of training data affects tagging accuracy. Using nltk's unigram tagger, we incrementally add training data and observe how the accuracy on the test data is affected. Starting from the first 500 lines, we continually add more lines in intervals of 500 until the full training data is used. Figure 3 shows a plot of test accuracy against the number of lines used for training. The graph shows a rapid initial increase followed by a steady decrease in growth rate, reaching an upper limit of around 90% accuracy. From this, we can infer that having a large volume of training data will significantly improve tagging performance, and allows us to reach a high baseline accuracy. However, it also shows that simply adding data has a threshold on performance, and further analysis needs to be done to raise the tagging accuracy. For the rest of the project, we continue our experiments using the full training data.

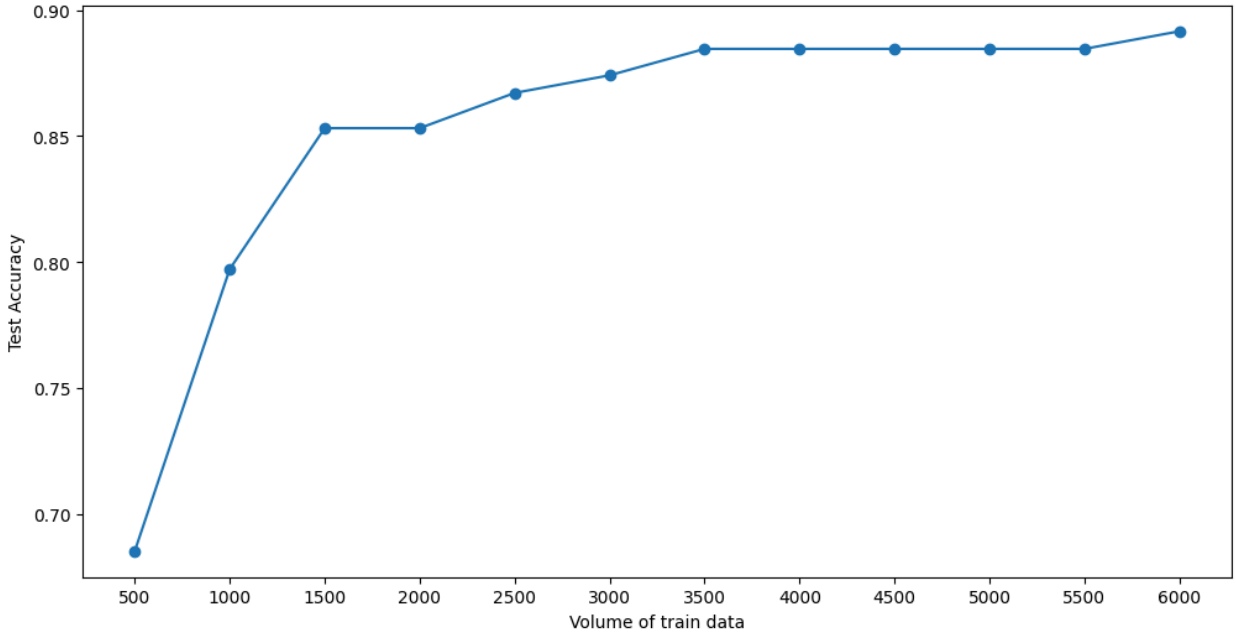

Fig. 3: Plot of test accuracy against volume of training data

N-gram taggers

Nltk's N-grams taggers function by looking at the current word together with the n-1 preceding words, and finding the same subsequence of words that occurs in the training data. It then picks the pos tag that occurs most frequently among the same subsequence. Here, we implement the unigram, bigram, and trigram taggers independently and evaluate their test accuracy. We observe that as n increases, the

tagger accuracy decreases. This is because when n increases, the subsequence of n words becomes longer and more unlikely to have occurred in the training data. Hence, it becomes more likely for the tagger to have never seen the n-gram before, resulting in it predicting a “None” label.

	Unigram Tagger	Bigram Tagger	Trigram Tagger
Test Accuracy	0.892	0.416	0.276

Table 1: Test accuracies of n-gram taggers

Default tagger

The default tagger simply assigns the same pos tag to all words. While it has limited functionality by itself, it is effective at labeling words that do not occur anywhere in the training data. This can be achieved by analyzing the words that the unigram tagger failed to label, and implementing a default tagger to assign the most statistically frequent pos tag. The table below shows all words that were predicted “None” by the unigram tagger. We can observe that among them, all words have the pos of “english”. As such, we created a default tagger assigned to the “english” pos, and integrated it as the backoff of the unigram tagger. This results in a test accuracy of 0.913, improving the score about ~2%.

Mb	Predicted pos	Actual pos	Count
land	None	english	1
holding	None	english	1
revision	None	english	1
phorward	None	english	1
issue	None	english	1
rebenu	None	english	1

Table 2: Frequency of words not labeled by unigram tagger

Ensemble taggers

In conjunction with the default tagger, we implement the various n-grams in a sequential manner using the backoff feature. From the table below, we observe that adding the bigram only improves accuracy by 0.7%, while adding trigram does not affect accuracy at all. This suggests that in the Pnar language, only the previous Mb is a contributing factor to the pos of the current word.

	Unigram Tagger	Bigram with backoff	Trigram with backoff
Test Accuracy	0.913	0.920	0.920

Table 3: Test accuracies of ensemble taggers

Custom bigram tagger

Using the Bigram tagger with backoff that we have implemented so far, we once again analyze the wrongly predicted pos tags in the training data. Table 4 breaks down the top 10 Mbs that were wrongly labeled, in descending order of frequency. We observe that the majority of errors are derived from “der=” pos tags that are wrongly labeled as “clitic=”. This occurs in terms ending with “=”, including “u=”, “ka=”, “i=”, and “ki=”. Upon inspection, we discover that these Mbs are statistically likely to have the “clitic=” pos, but there are exceptions where they should be “der=” instead.

Mb	Predicted pos	Actual pos	Count
u=	clitic=	der=	1919
ka=	clitic=	der=	492
da	mood	case	281
wa	mod	conj	230
i=	clitic=	der=	195
ki=	clitic=	der=	121
laeh	adv	v	119
dooh	loc	v	98
jooh	aspect	v	83
dooh	loc	adv	68

Table 4: Top 10 incorrect labels from bigram tagger

Using the interlinear search function implemented from project 1, we observe that these terms are more likely to have “der=” when it is followed by a verb. However, this rule was not identified by nltk’s bigram tagger as it involves a forward, multilevel search which the base bigram tagger does not support. As such, we built on the base bigram tagger to implement custom rules to differentiate between the “der=” and “clitic=” pos tags, described by the following pseudo code:

```
For each Mb in sentence:

        If Mb is one of [ “u=”, “ka=”, “i=”, “ki=”] and the pos tag of the next Mb is “v”:

                Set the pos of Mb to “der=”
```

By implementing this custom bigram tagger, we observe that the test accuracy increased by 2.7% to 0.948. This shows that the implementation of custom rules was successful, and that the rule successfully distinguishes the pos tags of the various terms.

Results

Overall, we were able to significantly improve on the baseline score provided in the original repository. Our best performing tagger is the custom bigram tagger, with a test accuracy of 0.948. Figure 4 illustrates the train accuracy (blue) and test accuracy (orange) of the various taggers implemented in this project. Our methods were able to improve the accuracy on both the train and test data, closing the gap between the scores. This shows that our implementations did not simply try to overfit on the test data.

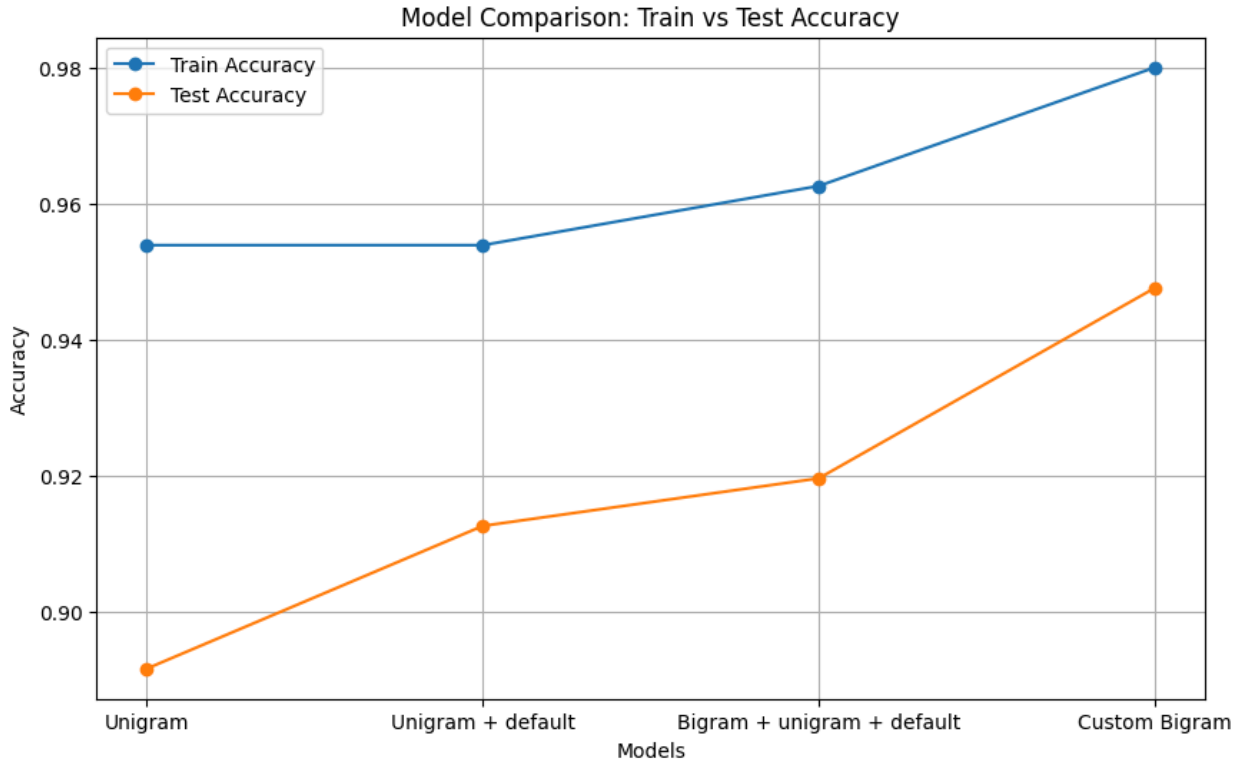

Fig. 4: Accuracy against different taggers

Discussion

While the custom bigram tagger performed well with a high test accuracy, there is still room for improvement. The table below shows the top 10 wrongly labeled pos tags from the custom bigram tagger. From this, we can still see that there are many instances of “u=”, “i=” and “ka=” wrongly labeled. This section discusses what further action could be taken to improve the labeling accuracies for the remaining wrong terms.

Mb	Predicted pos	Actual pos	Count
u=	der=	clitic=	450
i=	der=	clitic=	315

da	mood	case	267
wa	mod	conj	229
u=	clitic=	der=	166
bhah	v	n	47
laeh	adv	v	29
ka=	clitic=	der=	18
bait	adv	v	18

Table 5: Top 10 wrong labels from custom bigram tagger

“u=” and “i=”

We observe that unlike in table 4, the labeling of these two terms have been reversed. This suggests that the previous ruleset implemented is too general, and there are instances where the following Mb being a verb does not necessarily indicate that the pos tag is “der=”.

“u=” and “ka=”

There still remains some instances of “u=” and “ka=” being labeled as “clitic=”. This suggests that there are instances where they are “der=”m, but not followed by a verb. Once again suggesting that the ruleset implemented still needs finetuning and further analysis.

Others

The next most significant sources of error are the “da” and “wa” terms. By conducting further analysis on the occurrence of these terms, we could find additional rules to differentiate their pos tags. This will further reduce the error rate, and improve on the tagging accuracy.

References

Ring, Hiram, 2017, "Replication Data for: A grammar of Pnar", https://doi.org/10.21979/N9/KVFGBZ, DR-NTU (Data), V1