Unsupervised Chunking for Indian Languages

Prajneya Kumar - 2019114011
Jayant Panwar - 2019114013

NLP Project Final Report

Introduction

Chunking, or shallow-parsing, is a task that requires the identification of syntactic units which belong together, for example verbs and verbal auxiliaries are one chunk. The aim of this project is to create a phrase based chunking algorithm for Indian Languages. It is important to account for the free word order nature of most Indian languages.

In this project, we have tried to implement two chunking algorithms for the Hindi Language with an unsupervised approach.

Literature Review

Majority of the research work carried out so far has been emphasized more on very resource-poor languages. Indian Languages have been explored little because of the amount of information available in non-English languages is relatively less.

Not much literature exists for Chunking, especially using unsupervised algorithms and a general pattern observed is that if any, they are not transferable to Indian Languages. This is either because they are restricted to patterns observed in the language or due to lack of resources in IL. Some techniques we reviewed which could be helpful in some way or the other include:

1. **Shallow Parsing Pipeline for Hindi-English Code-Mixed Social Media Text:** This paper from IIIT, Hyderabad aims to accomplish data annotation along with the development of a language identifier, a normalizer, a part-of-speech tagger and a shallow parser on Code Mixed Social Media Text(CSMT). The point of focus for us here was the shallow parser. However, an unsupervised approach was not taken into account here, but the paper still gave us some insight on how to approach the problem of chunk labelling.

2. **Chunking in NLP Decoded:** This TowardsDataScience Article talks about the problem of chunking, syntax tree and grammar rules. The article uses the example of the NLTK Library to show usage of sentence chunking along with its implementation.

3. **K-Means Clustering:** This Towards-DataScience Article talks about the famous K Means clustering algorithm, which we implemented as one of our solutions to the chunking problem.

4. **Unsupervised Chunking Based on Graph Propagation from Bilingual Corpus:** This paper concerned itself with chunking Chinese sentences given tagged and chunked English translations using label propagation.

5. **Hierarchical Clustering with Python and Scikit-Learn:** This online stackabuse article explains in detail the concept of Hierarchical Clustering and how to implement it using Python and library packages like Sickit-Learn.

Chunking

Chunking is a process of extracting phrases from unstructured text, which means analyzing a sentence to identify the constituents(Noun Groups, Verbs, verb groups, etc.) However, it does not specify their internal structure, nor their role in the main sentence.

The process of chunking works on top of POS tagging. It uses POS-tags as input and provides chunks as output.

A chunk tag comprises of chunk label and chunk boundary. Chunk boundary is marked using BI notation where 'B-' prefix indicates beginning of a chunk and 'I-' prefix indicates that the word is inside a chunk.

A sentence typically follows a hierarchical structure consisting of the following components.

> sentence $\rightarrow$ clauses $\rightarrow$ phrases $\rightarrow$ words

Group of words make up phrases and there are five major categories.

1. Noun Phrase
2. Verb Phrase
3. Adjective Phrase
4. Prepositional Phrase

For the scope of this project, we consider only two labels, that is, the Noun Phrase(NP) and Verb Phrase(VP), and Other(which encompasses other categories) along with marking of punctuation tags.

Since we had to approach this problem with an unsupervised algorithm, the very first thing that occurs to mind for chunking sentences is that we have a clustering problem. Thus to solve this clustering problems, we tried two methods, namely, K-Means Clustering and Label Propagation. Due to shortage of resources regarding label propagation, we decided to opt for Hierarchical Clustering as our Base++ System.

Methods

Proposed Chunking System - Baseline: K Means Clustering

K-Means is an unsupervised learning algorithm that allows one to identify similar groups or clusters of data points within any given set of data points. Formally, any K-Means Algorithm comprises of the following steps:

1. Random initialization of K centroids.
2. Assignment of each datapoint to its nearest centroid using some formula for distance.
3. Recomputation of centroid location based on formation of clusters
4. Repetition of above steps until convergence

Now, to apply this algorithm to the problem of chunking, we are faced with certain implementation difficulties. We define certain parameters to implement the algorithm as follows:

- We need to have the value of K before implementing the algorithm. Usually this is obtained via different methods, one of them being the Elbow Method. However, in our case, since we tag only Noun Phrases and Verb Phrases, we initialise K, or the number of centroids, or to say, clusters as the total number of Nouns and Verbs in the sentence.
- We also need a formula to know the distance between centroids. We take each word in the sentence as a data point, and thus centroid are nothing but words of the sentence. To calculate distances, we use the number of words between two words as the standard metric.
- In case two words are equidistant to a centroid, we apply some linguistic knowledge. We use the given POS tags to us as reference and in case the word is a PSP (Postposition), we assign the preceding cluster, else the following one.

- Final centroid were allocated based on POS Tags, chunks containing Nouns were marked as NP while those containing verbs were marked as VP. In case there were no nouns or verbs, the chunk was marked as OTHER, while in case of both nouns and verbs occurring, higher priority was given to whatever occurred before.

Proposed Chunking System - Baseline++: Hierarchical Clustering

As the name suggests this is an unsupervised clustering algorithm which helps to identify similar group of data points within a scattered dataset. There are two major types of Hierarchical Clustering: Agglomerative and Divisive. The former carries out clustering from bottom to top by clustering individual data points accordingly and then combining clusters to form bigger clusters and ultimately ending up with one big cluster at the end of algorithm. Divisive algorithm on the other hand follows the top to bottom approach. It treats all the data-points as one large cluster and then begins to divide the large cluster into smaller clusters until it is left with only individual data points.
Ideally, we should be able to follow any of the two forms to try and build a chunking algorithm but we will stick to Agglomerative form as it resembles the hands on paper approach taught to chunk sentences manually. The steps of any Agglomerative Hierarchical Clustering algorithm include:

1. Consider every data point as an individual cluster. Therefore, the number of clusters in the starting will be the number of words.

2. Form a cluster by joining two closest data points.

3. Form a new cluster by joining two smaller clusters which are close to each other.

4. Repeat steps 2 to 3 until you are left with one big cluster. When a single cluster is left, decide the optimal number of clusters you want to divide the data-points into. This can be automated or done manually using dendrograms.

Since chunking is not something that can be implemented inherently using hierarchical clustering, there were implementation difficulties. The following parameters needed to be considered specifically due to the nature of our problem statement:

- A distance matrix is required to identify clusters or data-points that are more similar to each other than other clusters or data-points. For the same, we convert each word in the sentence to a (x,y) coordinate where 'x' represents the distance of the word from the closest noun in the sentence and 'y' represents the distance of the word from the closest verb in the sentence. We only considered verbs and nouns as noun phrases and verb phrases are the fundamental chunks of any sentence. To calculate the distance between any two data-points we opt for the Euclidean approach.

- The next most important parameter is identifying the optimal number of clusters that we want. As stated earlier this process can be automated or done manually. Since there is no gold method for finding the number of clusters automatically, we decided to find the optimal number of clusters manually using dendrograms. In order to find the optimal number of clusters, we look at longest vertical line of the dendrogram without any horizontal line crossing it. This line signifies the largest 'dis-similarity' between the two clusters that it joins. Then draw a long horizontal line at the base of the longest vertical line and count the number of vertical lines it crosses. This number will the optimal number for number of clusters.

Observations and Insights

K-Means Clustering

- Two nouns occurring together can at times cause problems with a generic K

Means algorithm. Therefore, we handled this special case where we treated two consecutive occurrences of nouns as one cluster and adjusted the value of K accordingly. For example,

- Since allocation of centroids is random in first step, final results also depends upon initial choice, and thus keeps changing on multiple iterations.
- The chunks obtained were not expected to have good quality, but the model outperformed our expectations in some cases. Applying almost only two linguistic rules increased our accuracies significantly.

Hierarchical Clustering

- One striking observation regarding this algorithm was that although we had given importance to only nouns and verbs when building the coordinates, the algorithm still managed to find pre-positional and adjectival phrases at times.
- Since the final chunks depend directly on the number of clusters chosen manually, there were cases when the number of chunks found were more or less than the gold number of chunks.
- Since we did not make use of any explicit Linguistic Rule in our algorithm it was expected that the chunks formed may not be correct. For example, in some cases Hindi auxiliary *tha* along with the punctuation was deemed to be a Verb phrase chunk in itself even though there existed a head verb just in one word vicinity of it.

Evaluation Methodology

To evaluate our obtained results, we defined a gold standard to compare with. Initially we had decided to test our data on sentences which had been chunked using the NLTK Chunk Library. However, we need to define out own rules to chunk using the same, which might lead to ineffectiveness in the Gold Standard.

Therefore, we decided to obtain manually chunked data for our input set, and we used the same as our gold standard. We compared the chunk tags for each word in our output with the gold standard in multiple iterations, and divided the same with the total number of chunks to obtain our accuracy.

Results and Analysis

For the K-Means Clustering Algorithm, we achieved an average overall accuracy of **71.04 %**.
The average overall accuracy for Noun Phrase allocation was found to be **78.25 %**

Since the Hierarchical clustering algorithm involved manual selection of the optimal number of clusters, it was not possible to evaluate all the chunked sentences against the gold standard chunks. However, 20 sentences were evaluated manually. 4 of them were correctly chunked considering the number of chunks found and the words that are part of those chunks. The rest of the 16 sentences had either incorrect number of chunks or/and the words in the chunks were incorrect.

Drawbacks and Challenges

Since centroid assignment assumes number of clusters, and hence number of chunks, our K-Means model is already biased based on the total number of nouns and verbs. This restriction can be uplifted by a better approximation of number of clusters. Similarly in our Hierarchical model, the number of clusters is decided manually which introduces bias early on.

Furthermore, as mentioned earlier, the consecutive occurrence of two nouns causes problems: both in clustering and in decision of number of clusters to allocate. We tried to solve this problem by specially handling this case, but the Hindi language has too many exceptions in this to handle everything perfectly.

Since our centroids are assumed only on the basis of nouns and verbs, the model can only look for two chunk categories, that is, Noun Phrase(NP) and Verb Phrase(VP).

We can try to add more linguistic rules in our hierarchical model to further refine the chunks formed. Also, different techniques should be thought of forming the distance matrix rather than only relying on closest Noun and Verb distance.

Conclusion and Future Work

1. **Hyperparameter Tuning:** A better algorithm to approximate K in the K Means Clustering Algorithm might make our accuracies better. Even the allocation of centroid heads with more linguistic knowledge of POS tags occurring together can lead to a better choice of hyperparameters, further leading to higher accuracy.
2. **Addition of Linguistic Knowledge:** The hierarchical model will surely perform better when trained using linguistic knowledge. As of now, the only linguistic knowledge being used in the model is that noun phrases and verb phrases are the fundamental chunks in a sentence. Introduction of more linguistic rules regarding adjectival and prepositional phrases will benefit the hierarchical model as well.
3. **Automated Methods:** We can try to use automated methods to find optimal number of clusters in the hierarchical algorithm. Different manual selection method can be tried as well if it improves the performance. For example, instead of drawing a horizontal line only at the base of the longest vertical line, another one at the middle could be drawn that will reduce the overall number of chunks.
4. **Other unsupervised approaches:** If we had access to bilingual data or a parallel corpus, we can apply graph propagation in a manner similar to what Derek F. Wong does for Mandarin and English.

*****