Orthographic Languages Similarity Measurements

By:
Aayush Mishra (16095001)
Kartikey Singh (16095029)

In this project, we extracted some similar words between languages with their distance by using provided corpora. We have applied the Longest Common Substring (LCS) using suffix trees and n-gram for finding the similar orthographic words using existing libraries from a raw corpus of a related language. Here, we have used three language pairs Hindi and Bhojpuri, Hindi and Magahi, Hindi and Maithili. Other than Hindi all the other languages are part of Indo-Aryan family, spoken mostly in the northern and north-eastern area of India and hence are orthographically similar.

The task can be related to finding cognates between two languages. Cognates are words in different languages that have similar spelling and meaning. They can help a second-language learner on the tasks of vocabulary expansion and reading comprehension.

Preprocessing:

The corpus which we were provided was unfiltered and had a lot of undesired content so to obtain a structured corpus from it we did some preprocessing. First, we removed some of the unknown files and files that were too large. We then created a directory containing all these unfiltered files. After this some of the files had a header that was made up of jumbled characters, those were removed. Then special characters and numerical characters were also deleted. Using Unicode values we made sure that the characters that were left are from the Devanagari script. And finally, after doing all this processing at some places we got long sequences of white spaces, these were collapsed to a single white space. These files were then stored in a new directory. Following this, we read the filtered files and generated a list of unique words of length greater than 2. We saved them as pickled files. This preprocessing procedure was repeated for all the four languages.

After all the preprocessing, the size of the corpus for each language is shown below

Language	Size of Corpus
Hindi	135305
Bhojpuri	36551
Magahi	33497
Maithili	25661

Suffix Tree and Longest Common Substring Matching:

To start this method we started by creating a suffix tree from our Hindi corpus. Then a data frame was created consisting of columns namely,

```
cols = ['Language', 'Word', 'Length', 'Empty Match', 'Partial Matches',
'Correct Match']
```

If no match was found for a particular word then Empty match would store “True” and the other two would be “False”. If a partial match was found then Empty match will be “False”, Partial Match will consist of a list of substring matches. And finally, if an exact match was found Correct Match would be “True”.

To start, we first created a suffix tree from our Hindi corpus.

```
tree = SuffixTree(True, hindi_list)
```

Then picking a word from the lists of orthographic languages. We then used, the function,

```
match_list = tree.findString(word)
```

The function which returned to us a list of matches for that word from the Hindi suffix tree. A function was then used to analyze this match list and find the values of the columns “Empty Match”, “Partial Matches” and “Correct Match”. If the match list was empty then no matches were found of that word. To check whether an exact match was found or not, we took the intersection of the match list with the original word if the answer was true, an exact match was found and the values were stored in the columns accordingly.

After generating these columns for each word of all the three languages, we plotted a few graphs to help us analyze the data which are shown in the appendix.

The following table shows the percentage of finding a match and an exact match for a word of the given language with the Hindi corpus

Language	Percentage of finding a match	Percentage of finding an exact match
Bhojpuri	44.269 %	34.609 %
Magahi	41.890 %	34.558 %
Maithili	39.145 %	30.228 %

n-gram Similarity Measurement:

In this method, we created a function which provides us with n-gram similarity between two strings.

The core idea behind n-gram similarity is to generalize the concept of the longest common subsequence to encompass n-grams, rather than just unigrams. The code for n-gram similarity has been provided below:

```
def ngram_similarity(x, y, n=1):
   k = len(x)
   l = len(y)
   L = [[0]*(l+1) for i in range(k+1)]
   for i in range(k+1):
       for j in range(l+1):
           if i == 0 or j == 0:
               L[i][j] = 0
           else:
               count = 0
               for u in range(n):
                   if i+u <= k and j+u <= l:
                       if x[i-1+u] == y[j-1+u]:
                           count += 1
               pos_ngram = (1/n)*count
               L[i][j] = max(L[i-1][j], L[i][j-1], L[i-1][j-1]+pos_ngram)
   return round(L[k][l]/max(k, l), 3)
```

The following table displays the best approximate match for a Hindi word in different languages along with their distance.

Hindi Word	Bhojpuri	
	Best match	Distance
निर्माणजो	नवनिर्माण	0.722
नके	अके	0.667
गदम	आदम	0.667

Hindi Word	Magahi	
	Best match	Distance
राष्ट्रपिता	राष्ट्रीयता	0.773
गीतिका	रतिका	0.667
सूर्यकुल	मूर्खाकुल	0.667

Hindi Word	Maithili	
	Best match	Distance
प्रांगण	प्रसंगक	0.643
कहानियां	ठेहुनिया	0.625
फसियो	लसिया	0.600

DICE algorithm for similarity measurement:

DICE algorithm works on the concept of character n-gram model. The basic formula that is used in this method is:

$$DICE(x, y) = 2 * |ngram(x) \bigcap ngram(y)|/(|ngram(x)| + |ngram(y)|)$$

For computing n-gram we used the following function:

```
def ngram(q, n=2):
   return [q[i:i+n] for i in range(len(q)-n+1)]
```

The following table displays the best approximate match for a Hindi word in different languages along with their distance.

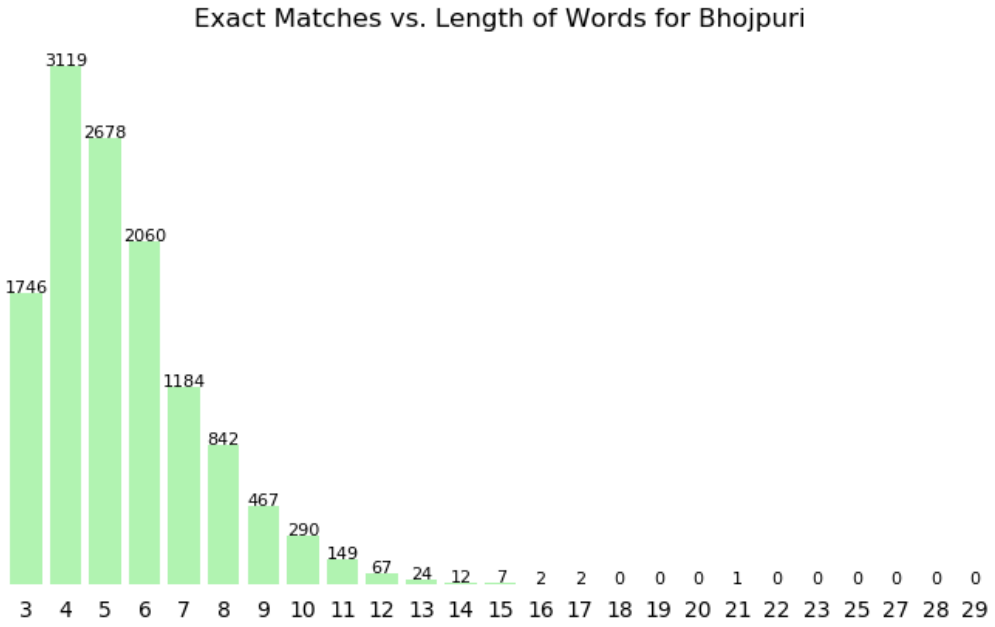
Hindi Word	Bhojpuri	
	Best match	Distance
निर्माणजो	नवनिर्माण	0.750
श्वेतकण	श्वेतकेतु	0.714
नके	सुनके	0.667

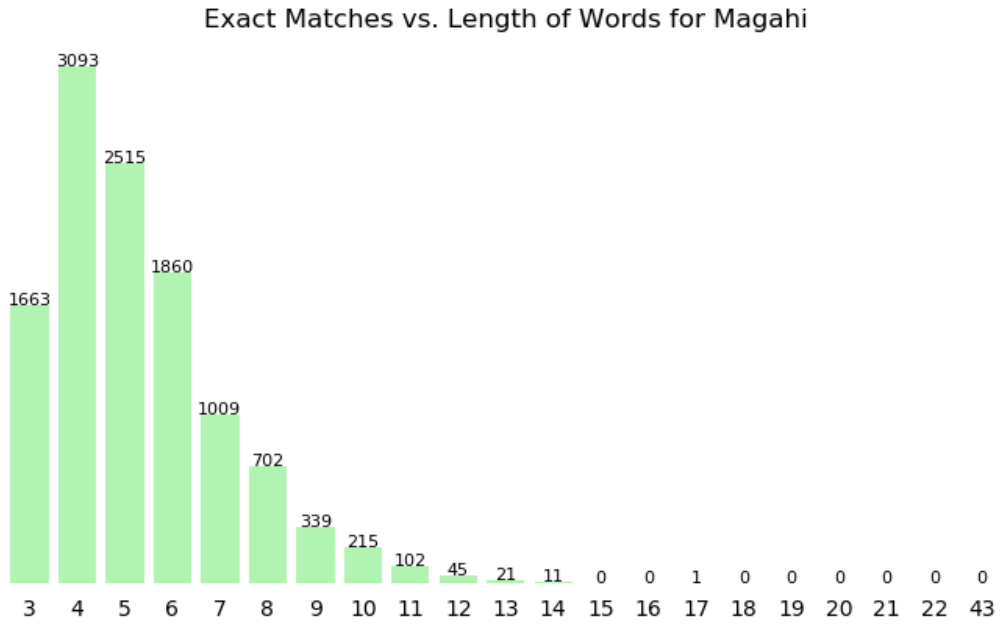
Hindi Word	Magahi	
	Best match	Distance
गारो	दारोगा	0.750
मंझोली	मंझोलका	0.727
राष्ट्रपिता	राष्ट्रीयता	0.700

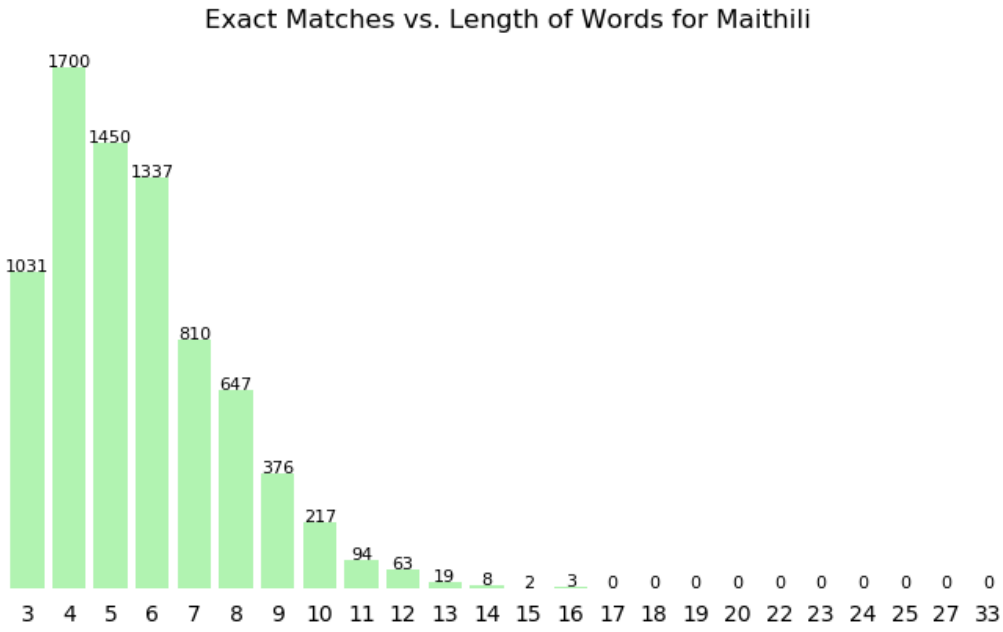
Hindi Word	Mathili	
	Best match	Distance
पहाड़ियां	उड़िया	0.545
मान्यनहीं	मान्यताक	0.533
सिद्धता	सुप्रसिद्ध	0.533

Appendix:

The following graphs represent the relation between the number of exact matches found with the length of the word for a given language using suffix trees







References:

- N-Gram Similarity and Distance[Grzegorz Kondrak]
- Chris Brew and David McKelvie. 1996. Word-pair extraction for lexicography. In Proc. of the 2nd Intl Conf. on New Methods in Language Processing, pages 45–55.
- Suffix trees library:
    - (https://github.com/kvh/Python-Suffix-Tree)