Prediction Of Cognate Reflexes

Varshaa Shree Bhuvanendar
G01269710
George Mason University
vshree@gmu.edu

Ishana Vikram Shinde
G01268126
George Mason University
ishinde@gmu.edu

Kavya Sudha Kollu
G01272848
George Mason University
kkollu@gmu.edu

Deeksha Gangadharan Srinivas
G01291097
George Mason University
dgangadh@gmu.edu

1 Introduction

History of linguistics defines cognates, also called lexical cognates, as sets of words in different languages that have been derived directly from an etymological ancestor in a common parent language. The SIGTYP 2022 Shared Task investigates the research question to predict cognate reflexes from multilingual wordlists. Cognates share a common origin regardless of their meaning and don't contain borrowed words. Individual members in the cognate set, also known as a cognate reflex, show similar sound patterns to other members of the cognate set. This allows mapping across the individual phoneme systems of the individual languages where most of the time the mappings depend on contextual conditions that differ based on their positions in the word.

1.1 Task and Motivation

The main task is to predict cognate reflexes from multilingual wordlists. The goal of cognate prediction is to identify in a given language the words that are most likely to be cognates of terms in related languages, where cognates are words that have developed from a common ancestor word. Cognates allow language learners of all ages to use their new words right away. A learner can build their vocabulary and gain confidence in their new language by beginning with cognate words. It is a task for which there is little data available and which can help linguists find previously unidentified relations. Solving this task would help the linguists address many issues that have been handled solely manually up until now.

1.2 Proposed Approach

We proposed to implement GCNN (Graph Convolutional Neural Networks) to predict cognate reflexes. Our proposed model is a graph-based deep learning model which retains context along the dataset and overcomes limitations of traditional machine learning models. Since the representation of the data will be in the form of graphs, the nodes would be the cognate reflexes with their immediate neighbors being the other reflexes in the cogent set. Along with this, we aim to add relations across cogent sets which will help to get context over cognate sets globally across the dataset. Because this representation would better capture the relations across cognate sets, we believe to get better scores for the cognate prediction task.

1.3 Summary

Through our GCNN implementation, we were able to get the results until training over epochs. Certain integrations with GCNN were unresolvable so we further tried to implement a CNN autoencoder model. One of the key takeaways would be that the CNN model provided better results in comparison to both Graph CNN and CNN Autoencoder. One of the reasons for this could be adding more steps or more relations leading to the model getting overfitted considering the datasets are very small. Another takeaway is that when we were trying to code for Graph CNN one of the steps involved the creation of an adjacency matrix. This matrix is the one that determines the relation in general CNN the relation is determined over cognates in a single cognate set. We tried to establish relationships over cognate sets. This is very complex and does not help the model much.

2 Approach

2.1 Background

2.1.1 Cognates

Cognates or lexical cognates are sets of words in different languages that have been inherited in direct descent from an etymological ancestor in a common parent language.

2.1.2 CNN Inpainting Model

The inpaint model takes inspiration from NVIDIA paper image restoration. The convolution model used in that paper.All available cognates are stacked into an image.In the input pixels are masked out representing the missing cognates. Then the input is passed to a simple convolution and deconvolution layer inorder to recover the character logits. The logits were then decoded back to produce cognate sounds.

2.1.3 CNN AutoEncoder Model

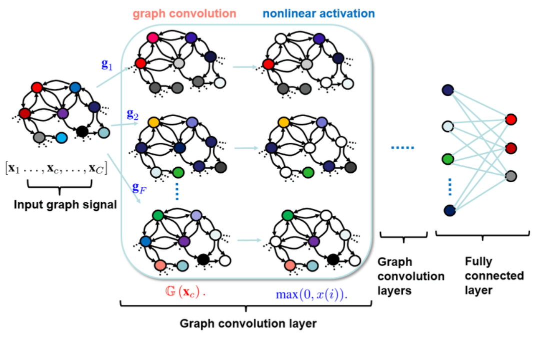
Autoencoders are a unique type of generative network. They consist of an encoder, a latent dimension space, and a decoder. It comprises a neural network architecture that tries to perform the conversion of the higher dimensional data into the lower dimensional vector space. The latent space contains the essential extracted features of the particular image, the data, in a compressed, representative form.

CNN also can be used as an autoencoder for image noise reduction or coloring. When CNN is used for image noise reduction or coloring, it is applied in an Autoencoder framework, i.e, the CNN is used in the encoding and decoding parts of an autoencoder.

A convolutional autoencoder is a neural network (a special case of an unsupervised learning model) that is trained to reproduce its input image in the output layer. An image is passed through an encoder, which is a ConvNet that produces a low-dimensional representation of the image.

2.1.4 Graph Convolutional Network

A Graph Convolutional Network, or GCN, is an approach for semi-supervised learning on graph-structured data. It is based on an efficient variant of convolutional neural networks which operate directly on graphs. The choice of convolutional architecture is motivated via a localized first-order approximation of spectral graph convolutions. The model scales linearly in the number of graph edges and learns hidden layer representations that encode both local graph structure and features of nodes.



2.2 Approach and Motivation

2.2.1 Approach 1: Graph Convolution Neural Network

- Graph CNN uses the concept of nodes representing cognates for each language within a cognate set, that is each cognate as a node and the edges are represented as the similarity between cognates over cognate sets.
- We then create embeddings for each cognate.
- We create an adjacency matrix using the above concept that is by finding the similarity in our case we are using cosine similarity.
- We then initialize the Graph CNN using the adjacency matrix and train the network using train cognate values.
- Next, the model is trained on the initialized model for 500 epochs
- We then generate the prediction file
- Evaluation is done using the predicted values using metrics BLEU Score, Edit Distance, and B-Cubed F1 Score

2.2.2 Approach 2: Convolution Auto Encoder

- We constructed a convolutional encoder and decoder model with the premise that the data is in image format.
- The model is first passed through a 2D convolution layer, and after that, we downsample the data by passing it through a MaxPooling layer with a pool size of 2. The convolution layer is then given the pooled data. The encoder network would be made up of this.
- The decoder network takes the input from the encoder layer, pass it through a convolution layer, upsample it to its original shape, and then pass it on to the final convolutional layer, which would return the logits.

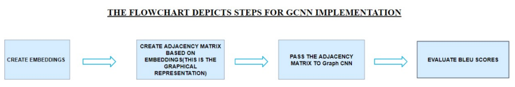
- We then decode the model to get the cognates back.

2.2.3 Motivation

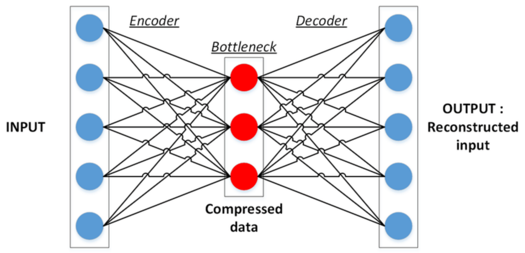
The reason why we tried using GCNN is to give more context generally in CNN the model learns similarity within the cognate set. By using GCNN we are trying to create edges of the graph or determine similarities over the cognate set. So we are creating an adjacency matrix over embedding values for each cognate over the cognate sets.

2.3 Diagrams related to Approach and Dataset

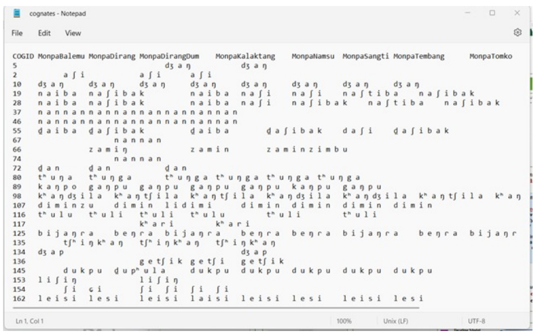
The first diagram illustrates the steps for GCNN implementation.


THE FLOWCHART DEPICTS STEPS FOR GCNN IMPLEMENTATION
CREATE EMBEDDINGS
CREATE ADJACENCY MATRIX BASED ON EMBEDDINGS(THIS IS THE GRAPHICAL REPRESENTATION)
PASS THE ADJACENCY MATRIX TO Graph CNN
EVALUATE BLEU SCORES

The second diagram illustrates the steps for the CNN encoder-decoder model.



The third diagram shows us the data present in the training files.



```
COGID MonpaBalemu MonpaDirang MonpaDirangDum    MonpaKalaktang    MonpaNamsu  MonpaSangti MonpaTembang      MonpaTomko
5                                 dʒ a ŋ          dʒ a ŋ
2           a ʃ i           a ʃ i      a ʃ i
10    dʒ a ŋ     dʒ a ŋ     dʒ a ŋ     dʒ a ŋ     dʒ a ŋ     dʒ a ŋ     dʒ a ŋ     dʒ a ŋ
19    n a i b a   n a ʃ i b a k          n a i b a   n a ʃ i    n a ʃ i    n a ʃ t i b a    n a ʃ i b a k
28    n a i b a   n a ʃ i b a k          n a i b a   n a ʃ i    n a ʃ i b a k    n a ʃ t i b a    n a ʃ i b a k
37    n a n n a n n a n n a n n a n n a n n a n n a n
46    n a n n a n n a n n a n n a n n a n n a n n a n
55    ɖ a i b a   ɖ a ʃ i b a k          ɖ a i b a        ɖ a ʃ i b a k    d a ʃ i    ɖ a ʃ i b a k
67                n a n n a n
66          z a m i ŋ              z a m i n        z a m i n z i m b u
74                n a n n a n
72    ɖ a n      ɖ a n            ɖ a n
80    tʰ u ŋ a   tʰ u ŋ g a       tʰ u ŋ g a tʰ u ŋ g a tʰ u ŋ g a tʰ u ŋ g a
89    k a ŋ p o   g a ŋ p u   g a ŋ p u   g a ŋ p u   g a ŋ p u   k a ŋ p u   g a ŋ p u
98    kʰ a ŋ dʒ i l a   kʰ a ŋ tʃ i l a   kʰ a ŋ tʃ i l a   kʰ a ŋ dʒ i l a   kʰ a ŋ dʒ i l a   kʰ a ŋ tʃ i l a   kʰ a ŋ
107   d i m i n z u    d i m i n   l i d i m i     d i m i n   d i m i n   d i m i n   d i m i n
116   tʰ u l u   tʰ u l i   tʰ u l i   tʰ u l u        tʰ u l i         tʰ u l i
117                     kʰ a r i         kʰ a r i
125   b i j a ŋ r a    b e ŋ r a   b i j a ŋ r a    b e ŋ r a   b e ŋ r a   b i j a ŋ r a    b e ŋ r a   b i j a ŋ r
135        tʃʰ i ŋ kʰ a ŋ    tʃʰ i ŋ kʰ a ŋ    tʃʰ i ŋ kʰ a ŋ
134   dʒ a p                                dʒ a p
136                     g e tʃ i k  g e tʃ i    g e tʃ i k
145        d u k p u   ɖ u pʰ u l a     d u k p u   d u k p u   d u k p u   d u k p u   d u k p u
153   l i ʃ i ŋ            l i ʃ i ŋ
154        ʃ i   ɕ i       ʃ i   ʃ i   ʃ i   ʃ i
162   l e i s i   l e s i    l e i s i   l a i s i   l e i s i   l e s i    l e i s i   l e s i
```

3 Experiments

3.1 Datasets

Training data:
We received our training data from the Lexibank repository. Ten distinct language families make up the training data and they are as follows Abrahammonpa, Allenbai, Backstrom Northern Pakistan, Castrosua, Davletshinaztecan, Felekesemitic, Hantganbangime, Hattorijaponic, List Sample Size, and Mann Burmish.

Test Data:
We obtained ten different language family datasets, which together made up the test data. Bantubvd, Hillburmish, Bodtkhobwa, Beidazihui, Birchallchapacuran, Deepadungpalaung, Kessler Significance, Luangthongkumkaren, Wangbai and bremerberta.

Explanation:
Each dataset was divided into five training and test divisions, with the amount of data used for testing varying from 10% to 20% to 30% to 40% to 50%. The primary input for all systems' training phases was the training data, which was not changed further. However, the test partition was purposely used to create the test data. Prior to creating individual test sets from each cognate set, we iterated through all cognate sets. Then, we deleted each word in a row while iterating over all of the cognate sets. This would result in n test cases for a cognate collection of n words, where each word in each language would need to be predicted once.

The following link provides the access to the data we intend to use:
Train Data:
https://github.com/sigtyp/ST2022/tree/main/data
Test Data:
https://github.com/sigtyp/ST2022/tree/main/data-surprise

3.2 Baseline Comparison

Baseline : Inpainting Model - The Baseline method is the Inpaint model by Google which uses the concept of Convolution Neural Network to predict cognate sets. The unknown cognate values are marked as 0. The steps for the infiller model are as follows:

- The first step in the inpaint model is to mask certain values in the training dataset.
- Next we create embedding for each cognate in the cognate set
- The CNN has one convolution layer and a single deconvolution layer
- In the training phase we import the train data

for a given language family and train the model over 500 epochs.

- Once we get the logits from the convolution network we map it back to respective cognate values.
- We then apply evaluation metrics to compare our outputs with the expected results.
- Once we train the model then we see its performance over dev surprise data.

3.3 Evaluation Metrics

Metrics used include Edit Distance, B-CUbed FS, and BLEU

3.3.1 Edit Distance

In computational linguistics and computer science, edit distance is a string metric, i.e. a way of quantifying how dissimilar two strings (e.g., words) are to one another, that is measured by counting the minimum number of operations required to transform one string into the other.

3.3.2 B-CUbed FS

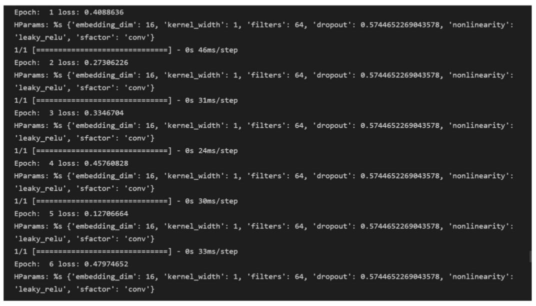
Frequently used in historical linguistics to evaluate automatic cognate detection techniques. This metric is mainly used to avoid overly penalizing systematic errors a system might make.

3.3.3 BLEU

BLEU (Bilingual Evaluation Understudy) is a metric for automatically evaluating machine-translated text. The BLEU score is a number between zero and one that measures the similarity of the machine-translated text to a set of high-quality reference translations.

3.4 Results

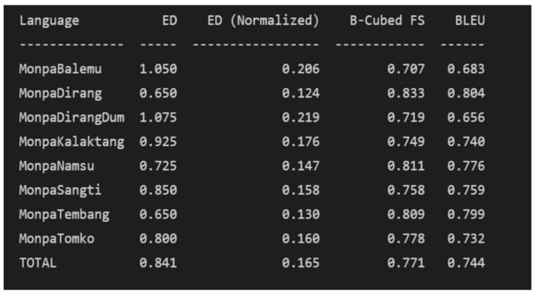
- **Results from GCNN:**



```
Epoch:  1 loss: 0.4088636
HParams: %s {'embedding_dim': 16, 'kernel_width': 1, 'filters': 64, 'dropout': 0.5744652269043578, 'nonlinearity':
'leaky_relu', 'sfactor': 'conv'}
1/1 [==============================] - 0s 46ms/step
Epoch:  2 loss: 0.27306226
HParams: %s {'embedding_dim': 16, 'kernel_width': 1, 'filters': 64, 'dropout': 0.5744652269043578, 'nonlinearity':
'leaky_relu', 'sfactor': 'conv'}
1/1 [==============================] - 0s 31ms/step
Epoch:  3 loss: 0.3346704
HParams: %s {'embedding_dim': 16, 'kernel_width': 1, 'filters': 64, 'dropout': 0.5744652269043578, 'nonlinearity':
'leaky_relu', 'sfactor': 'conv'}
1/1 [==============================] - 0s 24ms/step
Epoch:  4 loss: 0.45760828
HParams: %s {'embedding_dim': 16, 'kernel_width': 1, 'filters': 64, 'dropout': 0.5744652269043578, 'nonlinearity':
'leaky_relu', 'sfactor': 'conv'}
1/1 [==============================] - 0s 30ms/step
Epoch:  5 loss: 0.12706664
HParams: %s {'embedding_dim': 16, 'kernel_width': 1, 'filters': 64, 'dropout': 0.5744652269043578, 'nonlinearity':
'leaky_relu', 'sfactor': 'conv'}
1/1 [==============================] - 0s 33ms/step
Epoch:  6 loss: 0.47974652
HParams: %s {'embedding_dim': 16, 'kernel_width': 1, 'filters': 64, 'dropout': 0.5744652269043578, 'nonlinearity':
'leaky_relu', 'sfactor': 'conv'}
```

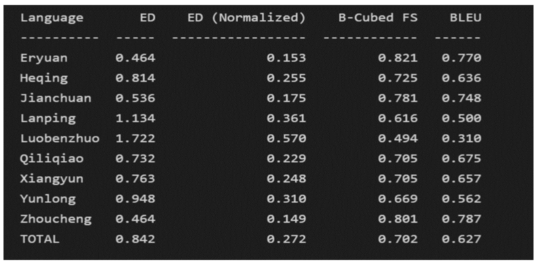
- **Results from CNN Autoencoder** The below tables are showing the results for various languages within various language family sets respectively in our dataset.

  – Dataset: abrahammonpa
  This dataset has 8 different languages which belong to the Tshanglic family.



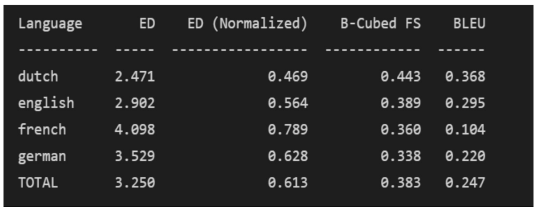
```
Language            ED   ED (Normalized)   B-Cubed FS   BLEU
-------------  -----  ----------------  -----------  ------
MonpaBalemu    1.050             0.206        0.707   0.683
MonpaDirang    0.650             0.124        0.833   0.804
MonpaDirangDum 1.075             0.219        0.719   0.656
MonpaKalaktang 0.925             0.176        0.749   0.740
MonpaNamsu     0.725             0.147        0.811   0.776
MonpaSangti    0.850             0.158        0.758   0.759
MonpaTembang   0.650             0.130        0.809   0.799
MonpaTomko     0.800             0.160        0.778   0.732
TOTAL          0.841             0.165        0.771   0.744
```

  – Dataset: Allenbai
  This dataset has 9 different languages which belong to the Bai family.



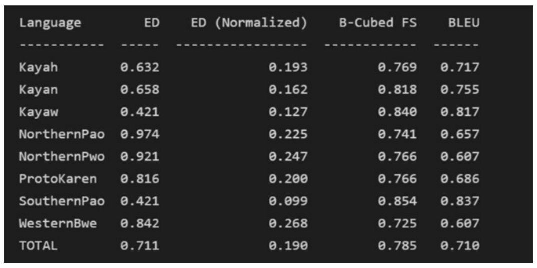
```
Language       ED   ED (Normalized)   B-Cubed FS   BLEU
----------  -----  ----------------  -----------  ------
Eryuan      0.464             0.153        0.821   0.770
Heqing      0.814             0.255        0.725   0.636
Jianchuan   0.536             0.175        0.781   0.748
Lanping     1.134             0.361        0.616   0.500
Luobenzhuo  1.722             0.570        0.494   0.310
Qiliqiao    0.732             0.229        0.705   0.675
Xiangyun    0.763             0.248        0.705   0.657
Yunlong     0.948             0.310        0.669   0.562
Zhoucheng   0.464             0.149        0.801   0.787
TOTAL       0.842             0.272        0.702   0.627
```

  – Dataset: listsamplesize
  This dataset has 4 different languages which belong to the Indo-European family.



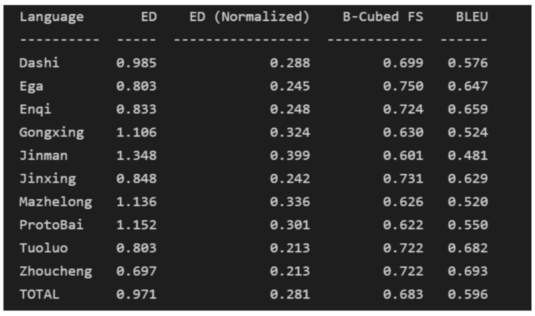
```
Language       ED   ED (Normalized)   B-Cubed FS   BLEU
----------  -----  ----------------  -----------  ------
dutch       2.471             0.469        0.443   0.368
english     2.902             0.564        0.389   0.295
french      4.098             0.789        0.360   0.104
german      3.529             0.628        0.338   0.220
TOTAL       3.250             0.613        0.383   0.247
```

  – Dataset: luangthongkumkaren
  This dataset has 8 different languages which belong to the Sino-Tibetan family.



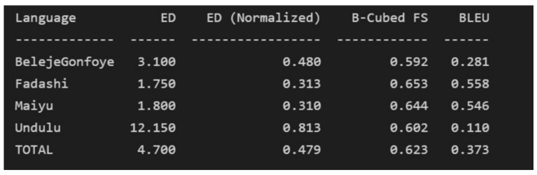
```
Language        ED   ED (Normalized)   B-Cubed FS   BLEU
-----------  -----  ----------------  -----------  ------
Kayah        0.632             0.193        0.769   0.717
Kayan        0.658             0.162        0.818   0.755
Kayaw        0.421             0.127        0.840   0.817
NorthernPao  0.974             0.225        0.741   0.657
NorthernPwo  0.921             0.247        0.766   0.607
ProtoKaren   0.816             0.200        0.766   0.686
SouthernPao  0.421             0.099        0.854   0.837
WesternBwe   0.842             0.268        0.725   0.607
TOTAL        0.711             0.190        0.785   0.710
```

  – Dataset: wangbai
  The dataset has 10 different languages which belong to the Sino-Tibetan family.



```
Language        ED    ED (Normalized)    B-Cubed FS    BLEU
----------   -----   ----------------   ------------  ------
Dashi        0.985              0.288          0.699   0.576
Ega          0.803              0.245          0.750   0.647
Enqi         0.833              0.248          0.724   0.659
Gongxing     1.106              0.324          0.630   0.524
Jinman       1.348              0.399          0.601   0.481
Jinxing      0.848              0.242          0.731   0.629
Mazhelong    1.136              0.336          0.626   0.520
ProtoBai     1.152              0.301          0.622   0.550
Tuoluo       0.803              0.213          0.722   0.682
Zhoucheng    0.697              0.213          0.722   0.693
TOTAL        0.971              0.281          0.683   0.596
```

- Dataset: Brembreta
  This dataset has 4 different languages which belong to the Berta family.



```
Language           ED    ED (Normalized)    B-Cubed FS    BLEU
------------   ------   ----------------   ------------  ------
BelejeGonfoye   3.100              0.480          0.592   0.281
Fadashi         1.750              0.313          0.653   0.558
Maiyu           1.800              0.310          0.644   0.546
Undulu         12.150              0.813          0.602   0.110
TOTAL           4.700              0.479          0.623   0.373
```

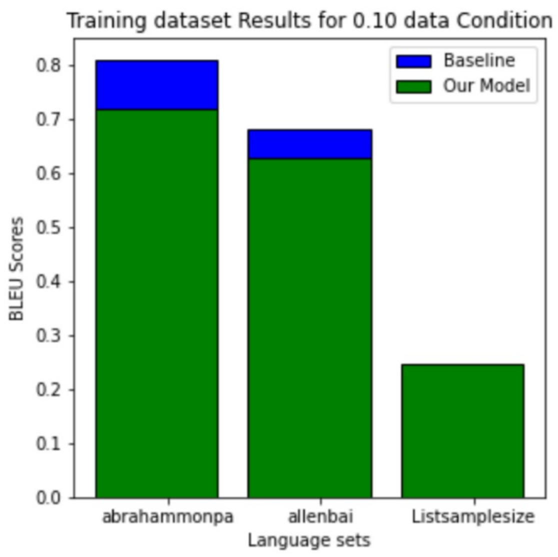
3.5 Result Analysis

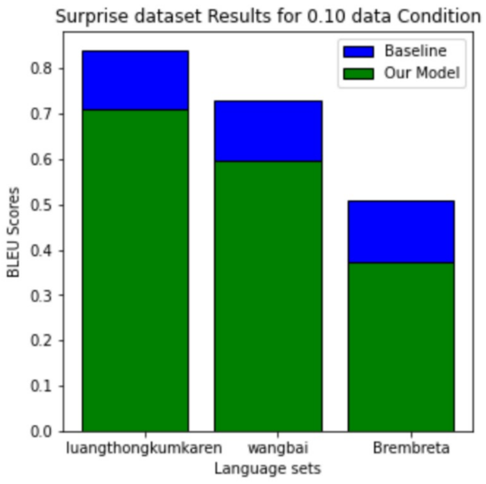
The results show BLEU Score, ED, ED(normalized) and BCubed FS Score for various language families. At the end of each language family, we take an average over scores of different languages.

As we can see loss is reducing over epochs as the model is learning. Also, the surprise language seems to give a higher difference in metric score compared to train languages as seen in the graphs below. The reason is for surprise language the code takes the last checkpoint as the best checkpoint which necessarily need not be the case.

3.6 Comparison of Results with Baseline

Ten datasets, each containing data for training and testing in various related languages, are available. We have considered six such datasets from various language families. The results we obtained are presented below:





4 Related Work

The task provided was a part of a competition called SIGTYP 2022. The baseline provided in the competition were as follows:

- **STAGE 1:**
  - Alignment of Cognate Sets
  - Trimming of aligned cognates
  - Enrichment of alignments by doing conditioning context
  - Alignment is converted to a matrix.
- **STAGE 2:**
  - Apply Classification Models: Used both Support Vector Machine and CORPAR(Graph-Based Model) for Classification.

Various teams that participated in the competition took the following approaches-

- **Team CrossLingference:** This model was implemented in Julia programming language that makes use of Bayesian phylogenetic inference, which was an extension of the phonological reconstruction
- **Team MockingBird:** Neighbor Transformer Model: This model treats the cognate sets as a neighborhood and predicts the pronunciation of target features given its neighbors.
- **Team LeipZig:** This team uses a transformer-based architecture with character and position embeddings. The languages are one hot encoded and the model is trained on individual reflex pairs. In order to get words in each language first we the probabilities and then average them to produce predictions.
- **Team CEoT:** Used the models similar to the baseline but skipped the trimming procedure and used a different technique by taking the preceding and the following context into account. They used Random Forest Classifier rather than Support Vector Machine

5 Conclusion and Future Work

We explored the area of phonemes and understood the importance of cognates while working on our project. We tried to work with different models after understanding the baseline implementation and we found how CNN and GCNN models can be used for the task of cognate prediction.

Using Deep CNN autoencoder we were able to get a difference of a maximum of 0.2 numerical value in comparison to baseline results as shown in the above graphs for various evaluation metrics.

For GCNN we were able to code until training we were able to run epochs successfully with loss reducing over epochs as shown in the result section, but once we got the result we were facing difficulties in translating them back to cognates.

As part of the GCNN Model, we were able to train the model but were unable to map back the logits to their cognate values. This could be considered in the future. Another exploration could be to try some preprocessing steps like trimming the cognates etc.

References

Gerhard Jäger. 2022. Bayesian phylogenetic cognate prediction. In *Proceedings of the 4th Workshop on Research in Computational Linguistic Typology and Multilingual NLP*, pages 63–69.

Christo Kirov, Richard Sproat, and Alexander Gutkin. 2022. Mockingbird at the sigtyp 2022 shared task: Two types of models for the prediction of cognate reflexes. page 70.

Johann-Mattis List. 2019a. Automatic inference of sound correspondence patterns across multiple languages. pages 137–161. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info .

Johann-Mattis List. 2019b. Beyond edit distances: Comparing linguistic reconstruction systems. volume 45, pages 247–258. De Gruyter Mouton.

Johann-Mattis List, Robert Forkel, Simon J Greenhill, Christoph Rzymski, Johannes Englisch, and Russell D Gray. 2022a. Lexibank, a public repository of standardized wordlists with computed phonological and lexical features. volume 9, pages 1–16. Nature Publishing Group.

Johann-Mattis List, Robert Forkel, and Nathan W Hill. 2022b. A new framework for fast automated phonological reconstruction using trimmed alignments and sound correspondence patterns.

Johann-Mattis List, Ekaterina Vylomova, Robert Forkel, Nathan W Hill, and Ryan D Cotterell. 2022c. The SIGTYP 2022 shared task on the prediction of cognate reflexes. In *Proceedings of the 4th Workshop on Computational Typology and Multilingual NLP (SIGTYP 2022)*, pages 52–62. Association for Computational Linguistics.

Rasmus Kristian Rask. 1818. Undersøgelse om det gamle nordiske eller islandske sprogs oprindelse. Popp.

Tiago Tresoldi. 2022. Approaching reflex predictions as a classification problem using extended phonological alignments.