Glottochronology of Indic Languages

244854

Under the supervision of Prof. Enrico Scalas

Degree Programme: MSC. Data Science

This report consists of approximately 15000 words

Abstract

The development of data science techniques in the field of linguistics has taken a huge rise since last few decades. The work of dating the languages to its evolution has been studied since a long time. The old methods were known as comparative methods in which all the work was done manually by the historical linguistics. The objective of this research is to perform glottochronology of Indic languages for reconstruction of Phylogenetic tree.

Table of Contents

Statement of Originality

I declare that this research has been composed and performed solely by me and that it has not been submitted, in a whole, in any previous application for a degree. Except where states otherwise, referenced pr acknowledged, the work presented is entirely my own.

I confirm that this research presented for the degree [degree sought], has

i. Been composed by myself
ii. Been solely the result of my own work
iii. Not been submitted for any other degree or professional qualification

Acknowledgement

Here, I would like to take a moment and thank Professor Enrico Scalas for all the support he has provided me through this journey of my Research work. Professor Enrico has been very considerate and understanding and helped me with every conceptual difficulty faced during this time.

Introduction

The ability to communicate through language is a key characteristic that distinguishes the human race from other animals. Nonetheless, despite a spectacular increase in our scientific understanding about the genesis of life, the universe, and (nearly) everything else that we have seen fit to consider, [1]we know surprisingly little about how our unique aptitude for language began and evolved into the sophisticated linguistic systems we use today. Although it is accepted that Homo sapiens initially used language between [2]30,000 and [2]100,000 years ago, [2] the exact process of language evolution is still a mystery, giving two primary schools of thought. One widely held belief is that language emerged as a result of evolutionary adaptation, which occurs when a population changes its process through time in order to survive. That is where natural selection comes into play, which is the theory that a population's distinctive physical qualities make that population more likely to survive in its environment. The notion here is that language was established to help humans in their survival for two reasons, one, for humans to successfully hunt, farm, and protect themselves against the hostile environment around them, they needed to be able to speak with one another. The ability to use language for communication gave the human species a certain survival edge. And two, those who believe in the adaptation theory claim that language was necessary for social interaction. Researchers Steven Pinker and Paul Bloom hypothesize in their study "Natural Language and Natural Selection" that a series of calls or gestures evolved over time into combinations, giving humans complex communication, or language [2]. As the world around them became more intricate, humans required a more complex mechanism to communicate with one another, which eventually evolved into a language. And that, in a nutshell, is the adaptation theory.

The alternative competing view, proposed by linguist Noam Chomsky and evolutionary biologist Stephen Jay Gould, is that language emerged as a byproduct of other evolutionary processes, rather than as a specific adaption. [2] Gould used the term "spandrel" to describe the belief that language defied the laws of natural selection. In fact, Gould and Chomsky proposed the hypothesis that many human behaviors are spandrels. These diverse spandrels evolved as a result of what Darwin termed "pre-adaptation," which is now known as exaptation. This is the notion that a species makes use of an adaptation for a function other than what it was originally intended for. According to Chomsky and Gould, language may have evolved simply because the brain's physical structure changed or because the same cognitive structures that were employed for learning rules or manufacturing tools also worked well for sophisticated communication [2].

With little knowledge of languages, language change, or language history, scholars around the end of the 18th century began to become increasingly intrigued by the startling similarities between some languages, such as Sanskrit- The classical language of India, Latin, and Old Greek [3]. Instead of making a strict difference between the precise patterns of similarity between languages that come from linguistic change and the process of change itself, they directly associated the patterns with the process. As a result, the linguistics assumed that the Indo-European languages were derived from Sanskrit. It took researchers more than a half-century to recognize that there was no direct line of ancestry between Sanskrit and the European languages [3] and that they were more likely the offspring of a common, unknown ancestor language. Today, after 200 years of historical linguistics research, we now have a considerably clearer picture of language, language history, and language change. All languages go through change during the process of evolution as the old words are replaced by new words, pronunciation of words changes over time, etc. When speakers of a language leave, their speech continues to develop separately in the two communities, until the independent changes become so significant that the language is divided into two different languages [3]. These changes are

phonological, semantic, and syntactic in nature. How can the relationship between two languages that are geographically separated, such as Sanskrit and Latin established? Proving that the two languages are derived from the same Proto-Language is one of the major tasks of historical linguistics. Language change came to be viewed as a process of descent with the alteration from a common ancestor language.

In the 1950's Morris Swadesh developed a comparative method known as Lexicostatistics to determine the relationship between languages. In this method, [4]Swadesh analyzed the languages by comparing the percentage of shared cognates between them. In historical linguistics, cognates, also known as lexical cognates, are sets of words in distinct languages that have been inherited in direct lineage from an etymological predecessor in a shared parent language. However, lexicostatistics does not recreate proto-language. Therefore, to overcome this, Morris Swadesh developed a comparative method known as "Glottochronology," [4] which employs lexicostatistical methods to evaluate the rate of change occurring in the vocabularies of the languages and calculates the period of time during which the two related languages have developed separately.

Data science has had a significant impact on linguistics. Having a thorough understanding of the true extent of variety among languages, within dialects, and among dialects is essential to all areas of the subject. With the invention of computers, the subfield of corpus linguistics—which is probably as old as the field itself—gave birth to many fundamental techniques used in data science. When the first huge search engines appeared in the late 1990s, linguists immediately began utilizing them to identify examples, investigate trends, and reveal subtle patterns in usage that emerge only at a very large scale. Even the largest corpora seemed little in relation to the open Web; this is arguably the first big data application in the linguistic sector. Data Science methods like Natural Language Processing and Machine learning have emerged as important counterparts to traditional and comparative methods in linguistics. The goal of this research is to study the Glottochronology of Indo-Aryan (Indic) languages using modern Data Science techniques.

Linguistic Background

The Indo-Aryan language family is the easternmost branch of the Indo-European language family. South Asia, is home to speakers of Indo-Aryan (Indic) languages. India, Pakistan, Bangladesh, Nepal, Bhutan, and the islands of Sri Lanka and the Maldives are among the countries represented by this region. [5]These seven nations, collectively referred to as the SAARC (South Asian Association of Regional Cooperation) countries represent the majority of the Indo-Aryan (Indic) speaking region. There are more than 200 Indo-Aryan languages, out of which a few major languages are Sanskrit, Hindi, Marathi, Pali, Gujarati, Punjabi, Rajasthani, Sindhi, Assamese, Oriya, Bengali, Bihari, Kashmiri, Urdu. The Indo-Aryan Languages are divided into three subgroups based on their periods of origin. These subgroups are as follows.

- Old Indo-Aryan
- Middle Indo-Aryan
- Modern Indo-Aryan

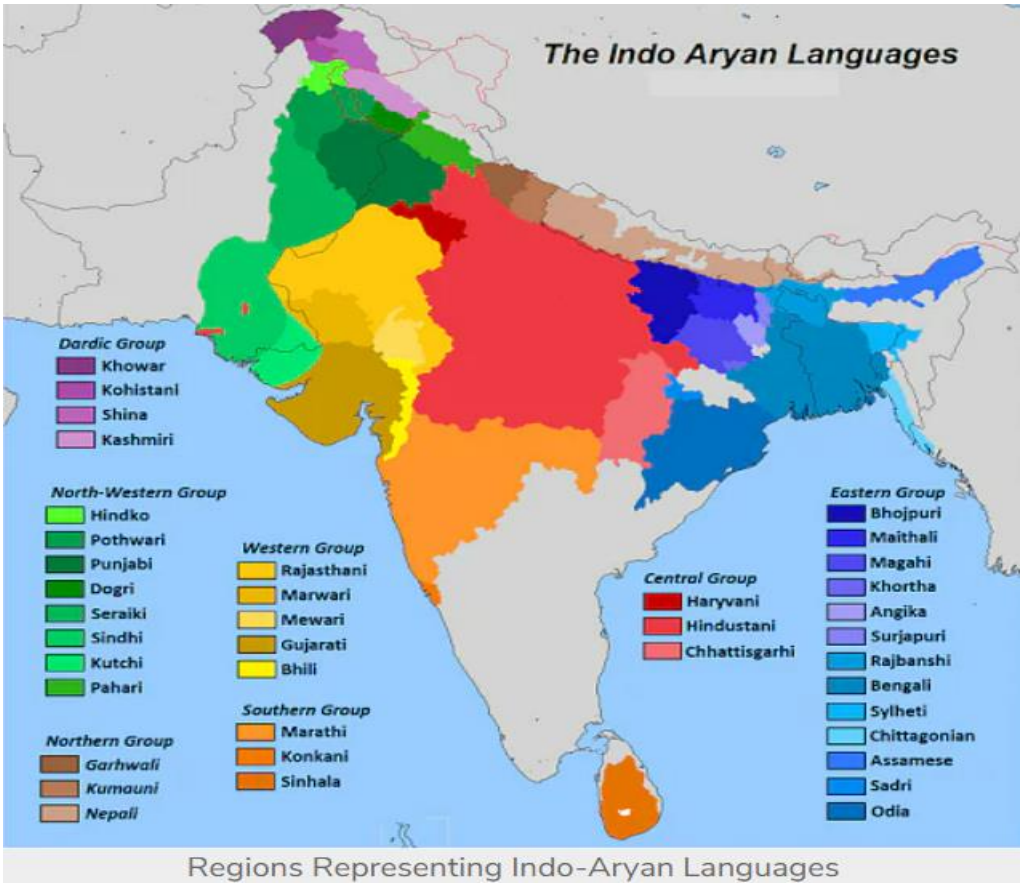


Figure 1: : Representation of Indo-Aryan Languages with respect to geographical locations

Old Indo-Aryan

The old Indo-Aryan group of languages was formed around 1500 BCE. Sanskrit which is the mother of all Indian languages originated from Old Indo-Aryan languages written by Indo-Aryan tribes migrating from what is now Afghanistan through northern Pakistan and into northern India between 1500 and 1200 BCE. It is one of the oldest Indo-European languages for which extensive documentation exists, and is thought to have been the general language of the greater Indian Subcontinent in ancient times. The only language that has crossed regional and geographical barriers is Sanskrit. Every region of India, from the north to the south and from the east to the west, has contributed to or been influenced by the Sanskrit language. Sanskrit literature began with the spoken or sung literature of the Vedas approximately 1500 BCE [6] and continued with the oral legacy of the Sanskrit Epics of Iron Age India, around 1200 BCE, the time following the Bronze Age. Vedic Sanskrit began the transition from a first language to the second language of religion and learning around 1000 BCE. Panini, an early scholar, standardized 3,959 rules of syntax, semantics, and morphology in Vedic Sanskrit grammar around 500 BCE (the study of words and how they are formed and relate to each other). The most significant Vyakarana text still in existence is Panini's Astadhyayi, [6] which consists of eight chapters outlining his rules and their origins. Panini contributed to the development of what is now known as Classical Sanskrit through this standardization.

Middle Indo-Aryan

The Middle Indo-Aryan group of languages was formed around 600 BC to 1000 AD [7]. It is distinguished by the growth of the Prakrit language. Prakrit was widely spoken by the populace in an informal setting. On the other hand, Sanskrit was orthodox, had set laws, and was only used by elites or scholarly people, mainly Brahmins. Languages such as Pali, Ardha-Magadhi, and Apabhramsa are a part of the Prakrit language. The earliest traces of the Middle Indo-Aryan language can be found in Ashoka's inscriptions (about 260 BCE), [7] as well as the earliest forms of Pali, the language of the Theravada Buddhist canon. In comparison to Sanskrit, Prakrit literature evolved more slowly. Sanskrit and Prakrit are linked, although there are various ways in which they diverge and contrast.

Modern Indo-Aryan

The Modern Indo-Aryan languages developed after 1000 AD. These languages descended from the Old Indo-Aryan languages and the Middle Indo-Aryan languages. This group of languages is spoken in India's western and eastern regions. The largest such languages in terms of first-speakers are Marathi, Bengali, Hindi, Urdu, Gujarati, Rajasthani, Sindhi, Punjabi, Bhojpuri, Oriya, Nepali, Romani, Assamese, Sindhi

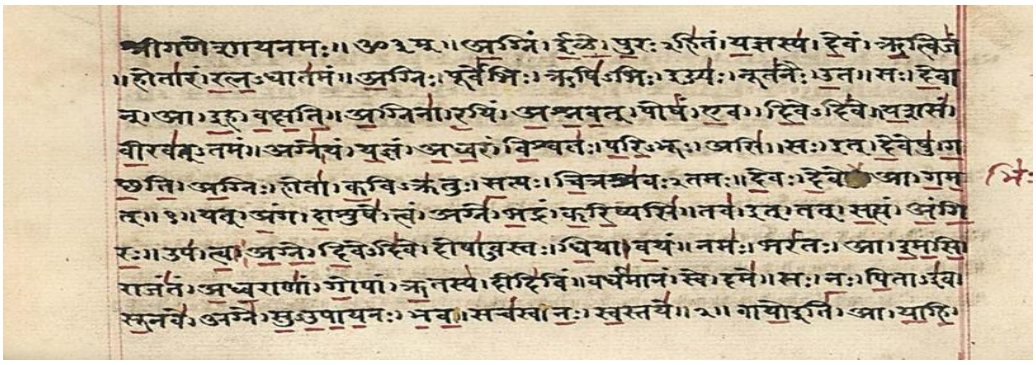


Figure 2:: Sanskrit manuscript from 'Rigveda'

Figure 3:Burmese Kammavaca manuscript written in Pali in the 'Burmese' script.

Cognates

In historical linguistics, cognates, also known as lexical cognates, are groups of words in distinct languages that have been inherited in direct lineage from an etymological predecessor in a shared parent language. In order to determine if two lexemes are cognate or not, one must frequently conduct a thorough analysis of historical sources and employ the comparative approach because linguistic change can have extremely radical consequences on both the sound and meaning of a word. The meanings of cognates do not necessarily have to be the same; they could differ as a result of divergent linguistic development. For example, the English word 'starve', German word 'sterben' (to die) and Dutch word 'sterven' (die) are derived from the same Proto-Germanic root 'sterbana. Cognates are distinguished from other kinds of relationship. Cognates differ from other types of relationships between words like loanwords or doublets. Words that have been borrowed from one language to another are called loanwords. For example, the English word 'beef' is borrowed from Old French word ' boef '(meaning "ox"). Despite belonging to the same etymological stemma, linguists do not typically refer to these words as cognates. Doublets are paired terms from the same language that share a common etymology but may have different but related usage and meanings. Frequently, one is a loanword while the other is the native term, or they have co-occurred in a current standard language after first developing in separate dialects. The old French word 'boef', for example, is cognate with the English word 'cow', hence English cow and beef are doublets.

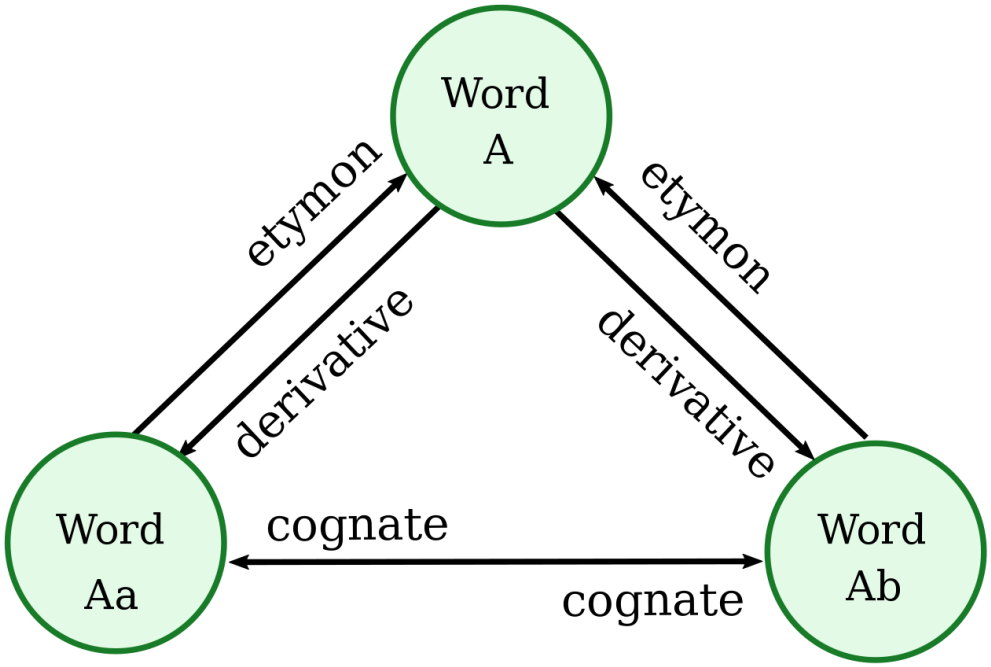


Figure 4:Example of cognates

Phylogenetic Tree

A phylogenetic tree, also called a phylogeny, [8] is a diagram that shows the evolutionary branches from which various species, creatures, languages, or genes have descended from one another. Phylogenies are helpful for classifying organisms, and languages, organizing knowledge of biological diversity, and shedding light on evolutionary processes.

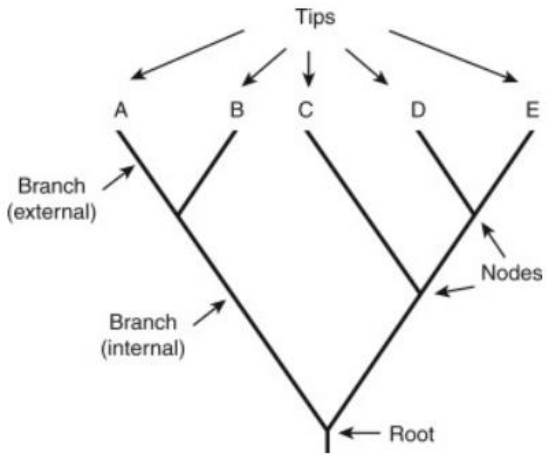


Figure 5:Phylogenetic tree example

A tree's leaves, often known as its tips, might represent species, populations, people, languages, or even DNA. Taxa are used when the tips represent a formally identified group (singular: taxon). A group of organisms at any level of the hierarchy, [8] such as a family, genus, or species, is referred to as a "taxon." The endpoints of extinct lineages, proto-language or fossils may alternatively be represented as the tips of a phylogenetic tree. A branch may contain one or more leaves and stands for the continuity of a lineage over time. Branches connect to other branches at nodes, which reflect the last shared ancestors of taxa at the tips of descendent lineages. An internal branch is one that connects two nodes, whereas an exterior branch connects a tip to a node.

A node and all of the lineages that are descended from it are grouped together as "clades" [8] on a tree. When a clade is said to be "monophyletic," it means that it contains every member of an ancestor lineage's line of offspring. In Figure 6, we may state that the tree supports taxa C, D, and E forming a clade or, to put it another way, that C, D, and E are monophyletic.

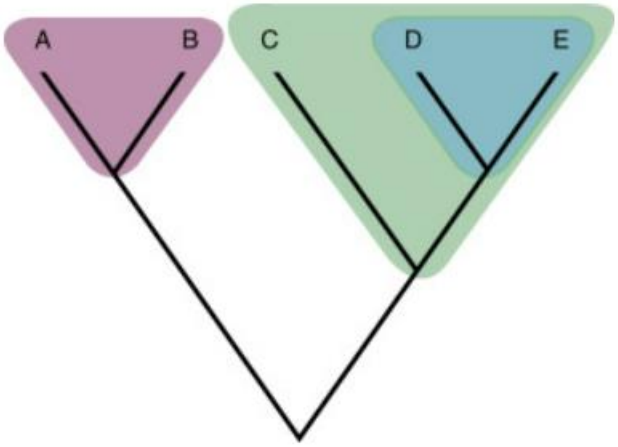


Figure 6:Example of Clade on a tree

Literature Review

State of the Art

The earliest work on glottochronology was done in the 1950s by Morris Swadesh, who created a statistical approach in the article for Salish Internal Relations to investigate comparative linguistics [9], quantify the degree of relatedness across languages, and reconstruct the language evolution. The process used by Swadesh to study the glottochronology of languages is described as follows

1. If a language's morpheme inventory, or a defined subset of it, is observed over a period of time [10]
2. If individual members of the inventory at a given period are designated as cognates of members from a prior time [10]
3. and some statable regularity can be found in time-depth at which the words in the inventory disappear and are replaced by new words, then the number of words in a certain subset, 2 which are present at any one point of time can be used as a measure to calculate the time elapsed since some previous point -time for which a similar count is available [10]

In his research, Swadesh employed a limited vocabulary in 30 Salishan languages and dialects during the investigation. The vocabulary was obtained from Boas' Comparative Salish vocabulary, which is a compilation of information gathered by numerous field researchers [9] and is housed in the American Philosophical Society's Boas Collection. Swadesh employed grammar rules based on the dialects' phonology to identify the cognates. Due to the incomplete development of Salish phonology at the time, certain errors in the prediction of cognates were made. Therefore, it was determined that the outcomes of this comparing procedure were roughly accurate.

Swadesh's test vocabulary was made up of 165 carefully selected words, all of which could be found in Comparative vocabulary and were appropriate for usage in regular conversation. If two languages were discovered to contain the same forms for a given notion, [9] which were derivable from the same prototype through regular phonological processes in each language, a correspondence was established, and these words were classified as cognates of each other. For example, the words upnkst, pancs, and upn ten from the three Salish languages of Clallam, Chehalis, and Thompson were grouped together as cognates of one another [9].

The number of correspondences (cognates) between each pair of languages was transformed into a percentage of total instances. [9] The following formula was used to translate them into units of time depth.

$$i = \frac{logC}{2logr}$$

Where,

i = indicated periods of depth

C = percentage of basic vocabulary in common

r = percentage of basic vocabulary retained after one standard period of time

The standard rate value of r used was 85%, [9] which is the percentage discovered to remain in Modern English by reference to Old English during a 1000-year period.

Derivation of the above formula given by Swadesh

In the study, assuming a constant average rate of vocabulary drift, [9] the effect of passing time on an initial vocabulary will be to reduce the vocabulary by a specified percentage each period by reference to the percentage at the beginning of that period. For instance, if 85% of the vocabulary is still present after one period of time, there will be 85% of 85% [9] after a second equal period of time. This is determined by the formula,

$$R = r^p$$

Where,

R = Percentage retained after a given number of standardized periods

r = percentage retained after one such period.

p = number of periods

If two languages a and b begin with the identical initial stock of basic vocabulary and suffer reductions to R_a percent in language a and R_b percent in language b independently of each other, [9] the probable shared vocabulary will be the product of maintained percentage in each, that is R_a times R_b. If the percentage maintained is the same for both languages, as it would be if they are contemporary languages evolved from the same original language after a similar time of drift at the same rate, then it is given by the formula,

$$C = R^2 = r^{2p}$$

On the other hand, if the two languages have been in contact for the entirety of the time since their common era, they will have exerted influence on each other. If one of the two languages replaces a word from the original stock, the second language may replicate the displacement or the first language may finally return to its original form. Because these effects can either promote or impede change, the trends may cancel one other out. Although the overall amount of change may be the same between the two languages as it would be between a single language and its related

languages, the two languages will most likely remain together. As a result, [9] they will deviate from one another less frequently than indicated by the calculation above. Therefore, Swadesh modified the formula to give the minimum probable common vocabulary under conditions of full divergent drift. Below is the formula given by Swadesh

$$r^{2p} = Min.C$$

To determine the indicated units of temporal depth in a comparative research, start with C, which is the proportion of common vocabulary. [9] Following is the formula's derivation:

$$C = r^{2i}$$

$$log\ C = \log r^{2i}$$

$$\log C = \log r * 2i$$

$$2i = \frac{\log C}{\log r}$$

$$i = \frac{\log C}{2 \log r}$$

In the research by Robert B. Lees, [10] the author explained the concepts of Glottochronology with respect to Swadesh's findings. The author compared the inventory of the words with the atoms in the radioactive element. As the age of the earth's crust can be calculated by analysing decay agents in mineral products, similarly, morpheme decay analyses can yield an accurate lexical history chronology [10]. In the research work done by the author, the total number of words in the inventory morphemes contained a subset of words which was the active common vocabulary of an average speaker. Some of these common words were affixes or other patterned units of grammatical structure [10]. The remaining words were classified as root morphemes, each with its own set of meanings. Although the meanings of these morphemes differ by culture, certain meanings, such as body parts, numerals, and geographical names, are likely to be found in all civilizations. The author referred to these morphemes as BASIC-ROOT-MORPHEME INVENTORY denoted by I. The size of I was expected to be essentially constant in time and from language to language because the meanings of words in BASIC-ROOT-MORPHEME INVENTORY are universal. Many morphemes can be found as cognates when morphemes from one subset of language are compared to morphemes from another subset of language with similar meanings at a later time. The size of the original subset is reduced by the remaining morphemes that were discovered to be non-cognates as few old terms are replaced with new ones. This decrease in the size of original subset is known as morpheme decay. There are many reasons for

morpheme decay like change in vocabulary, confusion in etymology, semantic shifts, new words for slang, borrowing words from another language, etc. To calculate the morpheme decay, the author described the following law

$$-\frac{dN}{dt} = \frac{N}{V} R = \lambda N$$

V = number of items in I

I = BASIC_ROOT_MORPHEME INVENTORY

R = time-rate at which these items are exchanged

λ = rate constant

For the term $\frac{N}{V} R$ consider a subset S of V items in I for a given language, which contains N_0 Items at the time t_0, out of which only N items are left after certain time t.

$\frac{N}{V} R$ = Items lost from S

Solving the equation $-\frac{dN}{dt} = \frac{N}{V} R = \lambda N$ for N as a function of time to calculate the number of morphemes left from sample S, from the original sample of N_0 morphemes after time t has elapsed. Here, e is the mathematical constant (e = 2.718), we get

$$N = N_0 e^{-\lambda t}$$

The time required by a language with a morpheme decay-rate λ to decrease the size of the sample S from N_0 to N is given by the time-depth equation,

$$t = -\frac{1}{\lambda} ln \frac{N}{N_0}$$

To measure the time during which two languages started diverging, it was assumed that "The selection of items for replacement in one language is statistically independent of the selection in other languages" [10]. The mathematical representation of the statement is given as,

$$F_0 = \frac{N_a}{N_0} . N_b . \frac{1}{N_0} = \frac{N_a N_b}{{N_0}^2}$$

Na = Items preserved from language a after time t

N_b = Items preserved from language b after time t

Using the above equations, Swadesh prepared a word list for French and English to calculate the time-depth of Germanic languages. The list consisted of 202 morphemes out of which 56 (27.7%) morphemes were identified as cognates and the divergence was recorded at 1000 B.C. The author also calculated the shared cognates for Gujarati and Modern Rajasthani, from 191 morphemes. The shared cognates were 54.5% which indicates that the split of these two Indic languages occurred at 550 A.D.

Automatic Methods for Glottochronology

Glottochronology makes the implicit assumption that vocabularies evolve at a constant rate in order to estimate the time at which languages diverged. The concept, originally put forth by Swadesh, is to limit the comparison to a list of concepts that are universally understood and that relate to the fundamental activities of humans. The decision is prompted by the fact that certain phrases are taught in childhood and change slowly over time. Swadesh lists have been used in glottochronology for almost a half-century. Glottochronologists calculate the distances between languages based on the proportion of shared cognates. On average, it is thought that divergence times are logarithmically proportional to these lexical distances.

Cognates are words that are assumed to have a shared linguistic past; yet, determining their identity frequently requires judgement and insider information. Because cognates do not always have a comparable appearance, it is far from simple to tally the number of cognate words in the list. As a result, subjectivity is important. Additionally, results are frequently skewed because it is simpler for American or European experts to identify those cognates that belong to western languages. For instance, the words leche in Spanish and gala in Greek are cognates. Leche actually derives from the Latin lac with the genitive form lactis, whereas gala has the genitive form galactos. The words "wheel" and "cakra" are cognates in English and Hindi. Our historical records have made it possible to identify these two languages, which would not have been the case for languages, for instance, from Central Africa or Australia.

The method proposed by Swadesh was used to calculate the cognates and generate the phylogenetic tree manually. Detecting the cognates manually requires deep knowledge of linguistics and hours of manual work. In the last two decades, there has been a significant interest in automatic approaches for detecting cognates and generating phylogenetic trees. These automated methods are known as Computational historical linguistics. The goal of the relatively new field of computational historical linguistics is to offer automated solutions for historical linguistics issues that have previously only been solved manually. There has been a significant increase in work on developing techniques for automatic cognate detection over the past twenty years. Even completely automated procedures in which phylogenies are created from automatically inferred cognates do not significantly differ from phylogenies derived from expert's cognate assessments, current approaches get good accuracy ratings when compared to human experts. Some of the work in this area is described in the below sections

In the research by authors in [11], an algorithm to detect the cognates automatically using Consonant Class Matching Algorithm was proposed. This approach considers that the sounds belonging to the [11] same sound class are most likely descended from the same ancestral language. Therefore, for analyzing the cognates, the authors made use of linguistic data taken from a collection of databases prepared by participants of the Evolution of Human Languages Project and the tower of Babel Project. Each root node of the 100-word list is coded by replacing its first two consonants with the generic consonant classes. The authors used 9 generic consonants [11]classes which are given in the below table. The bold capital letters refer to the consonant class code. In the research, the authors treated all the vowels as a single class. Therefore, each root was represented as a sequence of consonantal classes. The process was performed for 53 Eurasian and North African languages. The words which had the same first two consonant classes were stated as cognates and the words which differed in their first two classes were stated as noncognates [11].

The authors estimated the statistical significance of the observed fraction of matches between wordlists of two languages using the bootstrapping approach to analyze the findings. For this

technique, the authors chose a root word at random from List 1 and [11] matched it with a different root word at random from List 2. To calculate the statistical significance and roughly estimate the probability distribution, this method was done 10,000 times. The lower this estimated likelihood, [11] the more certain the belief that the proportion of observed matches could not have occurred by chance.

Table 1:Consonant Classes used in the research

	Front: labial, labiodental	Central: dental, alveoral, postal-veolar, palatal, retrofle	Back: velar, uvular
Nasal	**M**: m, m, ɱ	**N**: n, n, ɲ, ɳ	**N** : ŋ, N
Plosive, implosive, ejective	**P**: p, ph, b, p', ɓ, ɸ, β, f, v	**T**: t, th, d, ɗ, t'	**K**: k, g, k', q, ɣ x
Fricatives, approximants		**S**: s, z, ʂ, ʐ ʑ, ʃ, θ	
Affricates		**C**: c, ʒ,	
Laterals		**L**: l, ɭ	

In Another approach by [12], the authors identified the cognates by performing Levenshtein Distance also known as the Edit distance approach for all the word pairs. Levenshtein Distance is used broadly I speech recognition, genetics and bird song ethology. The basic idea behind Levenshtein Distance is that it calculates distance between two strings by calculating the minimum number of single-character edits like insertions, deletions, and substitutions between two words. For the experiment, the authors used 101 [12] words from 40 different Dutch dialects. All of the pronunciations of the words were taken from Dialect Atlas. For better results, the authors made use of feature vectors generated by replacing each phonetic symbol with a vector of features. Each feature was then regarded as a phonetic property [12] and was used to classify the sounds. To cluster the words into cognate sets, flat version of UPGMA (Unweighted Pair Group Method with Arithmetic Mean) was used. The UPGMA is a sequential clustering method generally used for phylogenetic tree construction. The algorithm takes an input of a distance matrix containing pairwise statistical estimation of aligned sequences generated from the normalized edit distance method. UPGMA algorithm is the simplest and fastest method to generate phylogenetic trees. This proposed method also takes into account a user-defined threshold of maximum distance between words.

Another similar approach was developed by [13] which used Normalized Edit Distance above along with Sound Class Alignment based model. The main objective of the SCA method is to distinguish between internal and external representation of phonetic sequences [13]. In this method the pairwise distance is calculated with the help of The Sound class-based Phonetic Alignment algorithm. The algorithm consists of four steps, described below.

1. In the first step tokenization is performed on the input sequence and the sequence is transformed into phonetic sequence.
2. Next, each sentence is further represented by its accompanying sound class sequence and prosodic profile

3. The third step involves doing a pairwise or multiple alignment analysis.
4. In the last step, the sequences are converted back to their original format.

The SCA method uses pre-processing which uses the consistency-based scoring system and post-processing methods which uses iterative refinement. The formula used to convert the similarity scores to distance is

$$D = 1 - \frac{2.S_{AB}}{S_A + S_B}$$

S_A and S_B = Similarity scores of the sequences aligned with themselves S_{AB} = Similarity score of the alignment of both the sequences

The LexStat method proposed by the same author in [13] builds on the SCA method described above, but has separate scoring functions for each language pair. It uses permutation to compute log-odds scores from the expected and attested distribution of sound classes. LexStat integrates the most significant components of the comparative method with modern approaches to sequence comparison in historical linguistics and evolutionary biology. The approach makes use of automatically extracted language-specific scoring algorithms and calculates distance scores from pairwise alignments of the input data. These language-specific scoring systems resemble the idea of regular sound correspondences in historical linguistics. The LexStat method is implemented in 5 stages

1. The input sequences are transformed into tuples made up of prosodic strings and sound classes.
2. In second step, preliminary cognate sets are derived using a straightforward language-independent ACD technique.
3. The next stage is to build language-specific log-odds scoring methods for all language pairs using a Monte-Carlo permutation test.
4. Based on language-specific scoring algorithms, all word pairings' pairwise distances are calculated.
5. The sequences are clustered into cognate sets whose average distance exceeds a predetermined threshold in the last phase.

The sound-pair PMI matrix is estimated using the Online PMI method in [14]. The method begins with a blank PMI matrix and a list of synonymous word pairings from all language pairs. Using the current

PMI matrix, [14]the technique proceeds by constructing the PMI matrix from alignments calculated for each minibatch of word pairs. The current PMI matrix is then mixed with the most recent minibatch's generated PMI matrix. This process is repeated a predetermined number of times. To determine the pairwise word similarity matrix for each meaning, we use the final PMI matrix. In a subsequent phase, the distance score was created by applying the sigmoid transformation to the similarity score: 1.0−(1+exp(−x)) −1. The Label Propagation technique is then used to infer cognate clusters using the word distance matrix as an input.

To reconstruct the phylogenetic tree from identified cognates with the help of the Sound Class Alignments method the authors of [15] used Bayesian Phylogenetic Inference to reconstruct the phylogenetic tree. To calculate the posterior distribution, the Metropolis-Hastings algorithm of Markov Chain Monet Carlo was used. For each dataset, the authors ran the chains for 15 million generations and the first 50% [15] of chains were thrown away as a part of burnin. For this experiment, MrBayes was used to perform the analysis. The findings from this research imply that, while expert-annotated phylogenies are still generally superior, automated phylogenies derived from cognate sets are nearly equivalent to them. In the research, the authors also concluded that automatic cognate detection can greatly benefit future work on phylogenetic reconstruction. The evaluation of the reconstructed phylogenetic trees in the research [15] , was performed using Generalized Quadrant Distance. The quadrant distance measures the distance between two trees in terms of quartets. A quartet is a set of four leaves selected from a set of leaves without replacement.

In another research by [16] the authors also used Markov Chain Monte Carlo permutations to reconstruct the phylogenetic tree. The authors used 207 words 150 languages from the IELEX dataset. The phylogenetic analysis was carried out using BEAST which is used to sample the posterior distribution. BEAST model outputs a trace [16]of first-order parameters, which includes tree, topology, and chronology. The first half of the trace is discarded here as well (similar to the above method), and the second half is used as the posterior sample.

One of the straightforward approaches to reconstructing a phylogenetic tree from a distance matrix is by using the UPGMA method. In UPGMA, phylogenetic lineages and time-dependent substitution rates are implicitly assumed to be constant. Because this assumption is frequently violated, this method is no longer commonly utilized.

Research Objectives

The main goal of this research is to Reconstruct phylogenetic tree for Indic languages. The research is divided into two parts.

1. In the first part, cognates of all the 18 languages are determined by using Automatic Cognate Detection Methods, Natural Language Processing and clustering algorithms.
2. The cognates identified from the first objective, will be then used to generate phylogenetic tree using Machine Learning techniques.
3. Comparison of the Phylogenetic tree obtained from Machine learning technique with the phylogenetic tree obtained by historical linguistics. Here we will consider the phylogenetic tree obtained by the linguistics as the gold standard. In Data Science "Gold Standard" is considered the ground truth and accepted as the most accurate result. Figure 7, represents the Gold Standard tree obtained by historical linguistics for Indo-Aryan (Indic) Languages.

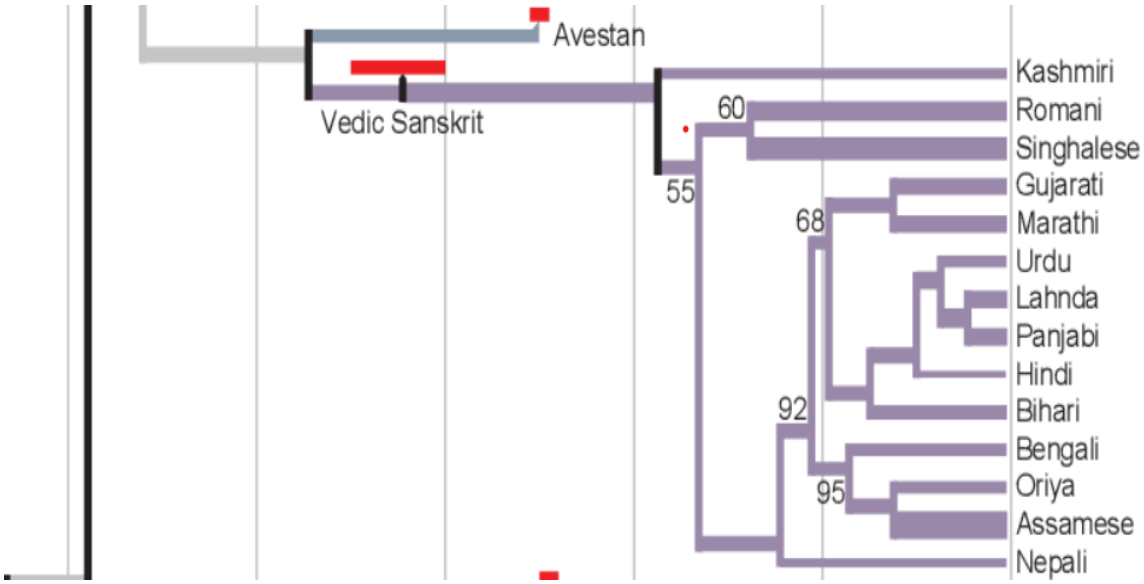
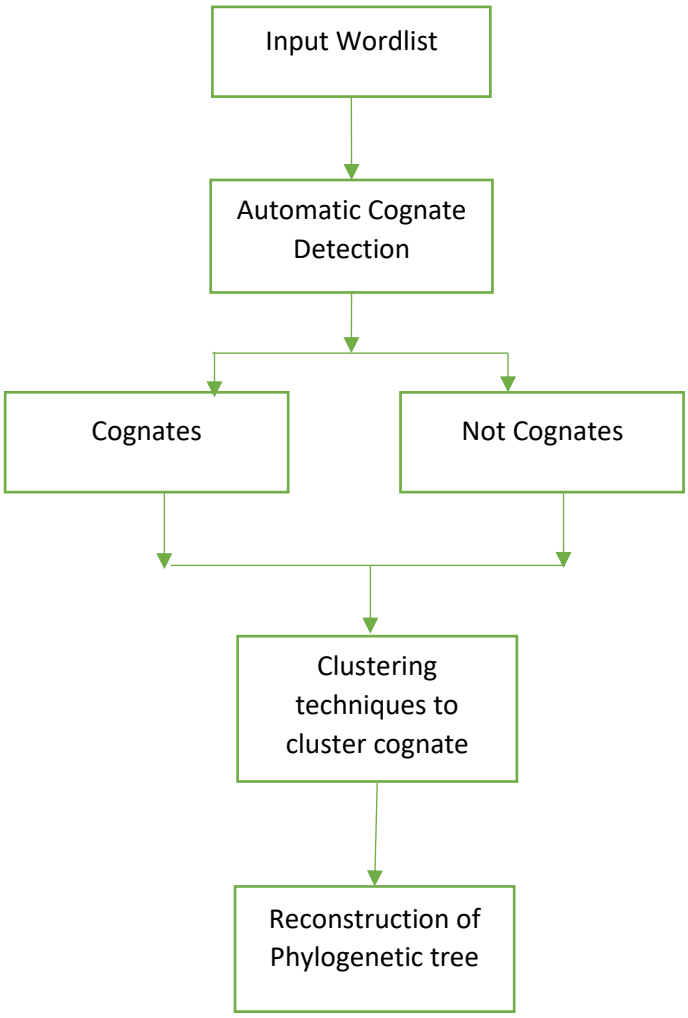


Figure 7:: Gold Standard tree for Indo-Aryan Languages

The process required for generating phylogenetic tree is given below. In the research the process is followed as given in the below diagram



Dataset

In the research, words from Indic (Indo-Aryan) languages are used for classification and reconstruction of phylogenetic trees. To keep the data at manageable level, Swadesh list of Indic languages from Wiktionary is used. The Swadesh list is compiled by Morris Swadesh and consists of carefully chosen words for the purpose of linguistic study majorly in lexicostatistics and glottochronology. The list consists of 207 words which are basic root morphemes obtained by translating English words into most common words which are used in everyday language. These words in the Swadesh list are the words which are universal in every language and likely to be found in every culture like body parts, numerals, geographical terms, simple activities, etc. The Swadesh list words are already converted to IPA transliterations. The International Photonic Alphabet (IPA) gives visual representations of speech sounds by the means of symbols. The IPA was first devised by International Phonetic Association in the 19th century [17] as the standardized representation of speech sounds in written format. The IPA was created to represent phonetics, phonemes, intonation, and the division of words and syllables as they appear in lexical sounds in oral language. The 18 languages in the below table are taken into account to study glottochronology. These languages represent the primary languages spoken by Indo-Aryans.

Table 2:List of Indic languages used in research

Sanskrit	Gujarati
Pali	Marathi
Hindi	Konkani
Urdu	Assamese
Nepali	Bengali
Bhojpuri	Oriya
Punjabi	Kashmiri
Sindhi	Sinhalese
Dhivehi	Romani

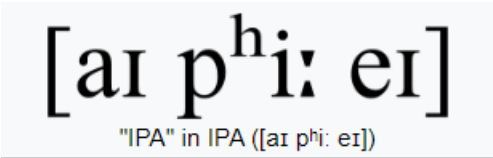


Figure 8:Example of IPA

Methodology

Web Scraping

Web scraping is the procedure for gathering and cleaning raw material from the Internet. To allow for future manipulation and analysis, this data is typically saved in a local file. In the research we will use Beautiful Soup library in python to scrape the data. Python's Beautiful Soup package is used to extract data from HTML and XML files for web scraping purposes. From the source code of the website, it generates a parse tree that can be used to extract data in a hierarchical and more comprehensible way.

Automatic Cognate Detection

In the research, cognate detection is carried out using 4 different methods namely Edit Distance, LexStat, SCA (Sound Class Alignment), and Pointwise Mutual Information. After detecting the cognates, the cognates are then clustered into groups using Infomap Algorithm. The detailed methodology of each method is described in the below sections.

Levenshtein Distance

Levenshtein distance also known as Edit Distance is a technique in which two strings are compared with each other by calculating number of distances between them. The smallest number of point mutations needed to replace one string with another is known as the Levenshtein distance between two strings. The point mutation comprises of the following

1. Substitution
   Substitution is an operation required to exchange a character in a string with another character. For example, to replace the word ‘fire’ with the word ‘fine’, the character ‘r’ is replaced by ‘n’.

2. Insertion
   Insertion is an operation that adds a character to a string. For example, to replace the word ‘for’ with the word ‘fore’, we need to add the string ‘e’ to the word ‘for’. Characters can be added to the words in the insertion operation not only at the end of the word but also in the middle or beginning of the word.

3. Deletion
   The deletion operation deletes the character from the string in order to replace a string with another string. For example, to replace the string ‘where’ with the string ‘were’, the character ‘h’ is deleted.

The below algorithm is used to calculate Levenshtein Distance between two words

Let word1 be a = a1, a2, ...an and word2 be b= b1, b2, …bn, then Levenshtein distance between word1 and word2 is calculated by

$$d_{i0} = \Sigma_{k=1}^{i} w_{del}(b_k), \quad \text{for } 1 \le i \le m \tag{1}$$

$$d_{0j} = \Sigma_{k=1}^{j} w_{ins}(a_k), \quad \text{for } 1 \le j \le m \tag{2}$$

$$d_{ij} = \begin{cases} d_{i-1,j-1} & \text{for } 1\, a_j = b_j \\ \min \begin{cases} d_{i-1,j} + w_{del}(b_k) \\ d_{i-1,j} + w_{ins}(a_j) \\ d_{i-1,j-1} + w_{sub}(a_j, b_j) \end{cases} & \text{for } 1 \le i \le m, 1 \le j \le n \\ & \text{for } a_j \ne b_j \end{cases} \tag{3}$$

Where,

a = Words represented as rows

b = Words represented as columns

i = Letter index of the word a

j = Letter index of the word b

n = length of the word a

m = length of the word b

w = value of deletions, insertions and substitution

ins, del, sub = insertions, deletions and substitutions

Sound Class Based Phonetic Alignment (SCA)

The Sound Class Based Phonetic Alignment technique is built into the Python Lingpy library. The SCA technique is based on the distinction of an external and interior representation of phonetic sequences. A specialized module is responsible for managing the conversion of external to internal format of phonetic sequence and vice versa. It manages the transition [18]between internal and external phonetic representation of sequences. The implementation of SCA algorithm is completed in 8 major steps as below.

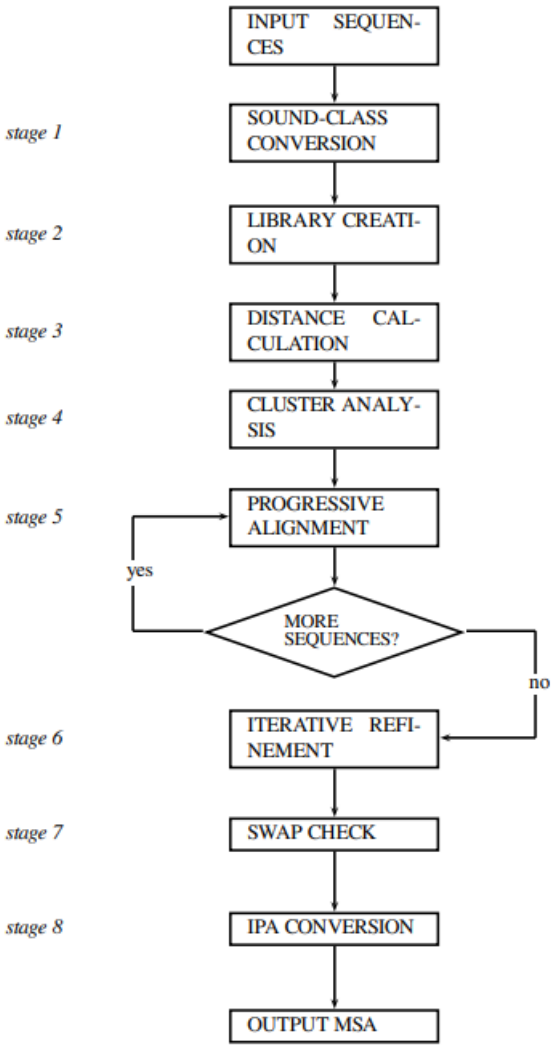


Figure 9:Working Procedure of SCA method

In Step 1, the input sequences of words are converted into sound class [18].

In Step 2, the library of pairwise alignment is created. This is done by suing Needleman-Wunsch algorithm. To illustrate the working of Needleman-Wunsch algorithm, [18] consider two sequences x and y, with lengths m and n. Let x_i denote the i-th element of x and y_i denote the j-th element of y. A scoring system that penalizes the fundamental matching types is required to score the alignment. The scoring function score (A, B) is used in this scoring scheme to represent the matching of segments. It returns -1 if segments A and B are different and 1 if they are similar. With the gap penalty g set to -1, [18]empty matches are expressed. After calculating the scores, an alignment matrix is created by the Lingpy Module.

In Step 3, pairwise similarity scores of all the sound class sequences are used to calculate the distance matrix by the formula,

$$D = 1 - \frac{2.S_{AB}}{S_A + S_B}$$

Where [18]

S_A and S_B = Similarity scores of the sequences aligned with themselves

S_{AB} = Similarity scores of alignments of both sequences

In Step 4, the Neighbor-joining algorithm [18] is used to cluster the sound class strings. The Neighbor-joining algorithm is built-in in the Lingpy library. The clustering procedure yields the guide tree that is used for the progressive alignment in the next step.

In Step 5, all the sequences are aligned stepwise with each other, after the branching of the guided tree.

In Step 6, Iterative refinement is carried out to avoid possible errors due to misaligned sequences. The standard Needleman-Wunsch approach will result in computing durations that increase exponentially when the number of sequences being evaluated increases to three and beyond. Therefore, it is usual practice to use specific heuristics that can only ensure the identification of a close to an ideal solution for multiple sequence alignments. The so-called progressive alignment approaches are the algorithms used in multiple sequence analyses that are most frequently used. [18]There are two main phases to these strategies: A guide tree is built in the first step to depict the relationships among all sequences. This guide tree is utilized in the second step to sequentially align all sequences with one another, working from its branches all the way down to its root.

In Step 7, swap detection [18] is carried out with the help of Edit Distance as well as building an additional penalty of 1 for crossed matches. The method's core idea is to take use of the fact that the alignment of swaps in phonetic sequences frequently follows a similar pattern. Spreading the swapped

region over three columns rather than two allows us to compel the algorithm to avoid mismatches by [18]using a new scoring system where gaps are rated as 1, but mismatches are marked as 2.

In the last step, all the sequences are converted from internal representation to their original IPA format [18].

Lingpy LexStat

The LexStat Method for detecting cognates automatically also works with Lingpy library in python and was proposed by [18] in the year 2014. LexStat integrates the most significant components of the comparative method with modern approaches to sequence comparison in historical linguistics and evolutionary biology. The method leverages automatically extracted language-specific scoring algorithms and computes distance scores from pairwise alignments of the input data. These language-specific scoring methods resemble the idea of regular sound correspondences [18] in historical linguistic theory. The basic working of the LexStat method is given in Figure 8.

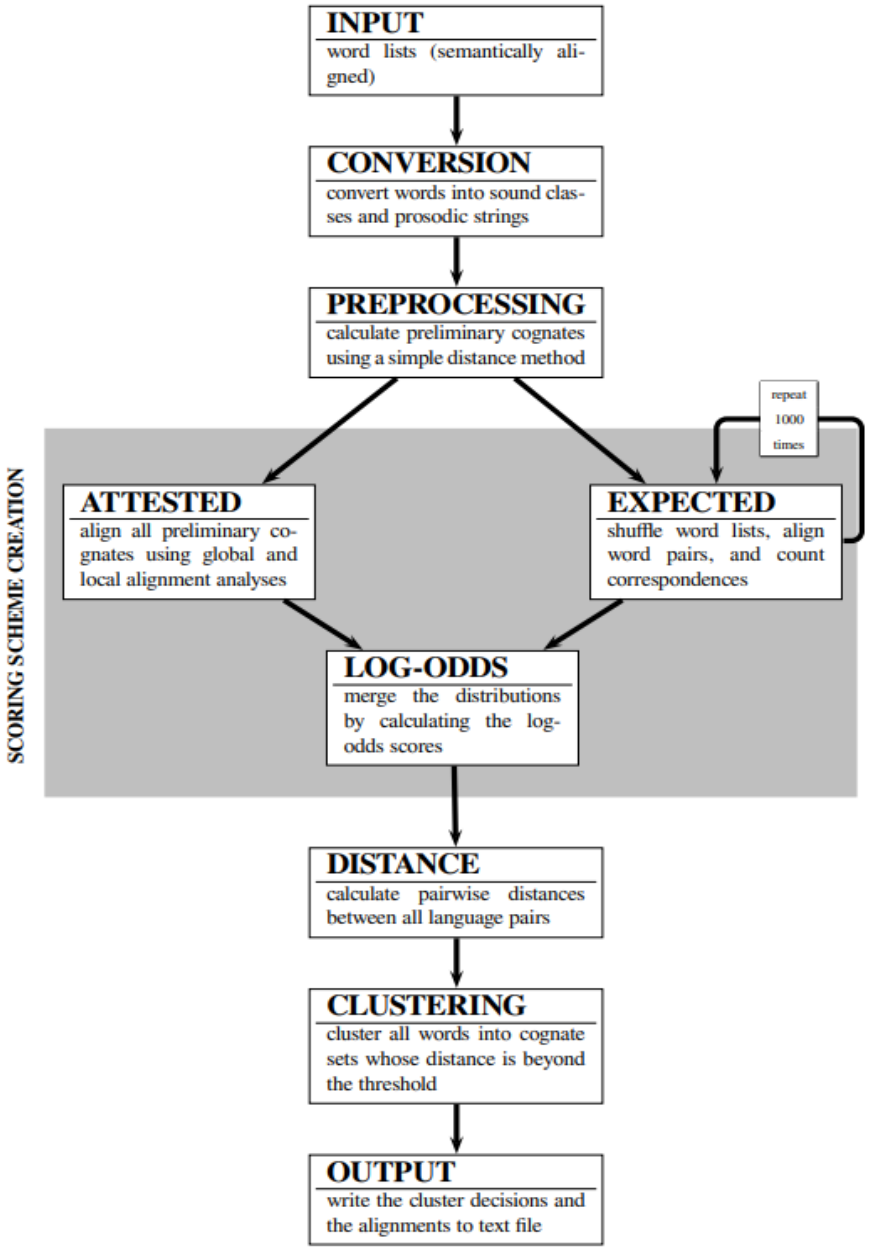


Figure 10: LexStat Process

In the first step, an input file containing wordlists is provided to the algorithm. For this input file, there is no restriction on the order of columns and the format of the file is flexible. If user wants to include additional data in file, it can be included in separate columns. For each word in the file, 4 columns containing word, IPA, dialect, and concept is necessary for comparing the cognates for every word [18].

In the sequence conversion step, all the input sequences are converted into tuples [18], which consist of entries of sound classes and the prosodic strings. Prosodic strings are the features of sound classes that appear when all the sound classes of a word are put together in connected speech.

For the next steps, a scoring system is created. The general technique used for finding similarities between two languages is to create an attested distribution [18] by calculating all the possible links between all the word pairs in the language pair word lists and comparing the distribution with the expected distribution. Here, LexStat uses consistency-based alignments in which local and global alignments are computed and the resulting distribution is averaged. The main advantage of consistency-based alignments is that more information can be taken into account, especially if the words exhibit "local" similarities, for example, prefixes and suffixes.

The issue that all techniques must address is how to deal with noise in the data. Regular sound correspondences can only be identified for cognate words, which are not all of the words in the data. Linguists manually determine regular sound correspondences by counting the number of correspondences in words that they presume are cognate [18]. Therefore, in the LexStat method, the author has used an effective strategy to pre-process the words. The main idea here is to use a list of multilingual words instead of bilingual words. The generation of the predicted distribution is more difficult than the process used to create the attested distribution of matching residue pairs. Under the presumption that all aligned residue pairs are statistically independent of one another [18], it is common practice in biology to quantitatively infer the anticipated frequencies of matching residue pairs. Sadly, the LexStat technique does not allow for this because the confirmed frequencies are obtained from an alignment process that inherently favors and disfavors specific matches. To generate the expected distribution of matching residue pairs, the words in the word lists for a given language pair are repeatedly resampled by shuffling one of the entry columns. The new word pairs are pairwise aligned in each resampling step using the same procedures that were used to create the attested distribution. Then, to reflect the anticipated frequencies, the average of the residue pair frequencies over all samples is calculated. In the default settings, the number of repetitions is set to 1000 [18].

The next step is to compute the log odds, the output from attested and expected distribution is used to create a language-specific scoring matrix [18] for an alignment algorithm. The similarity score for $S_{x,y}$ for each residue pair x and y is calculated by the below formula,

$$S_{x,y} = \frac{1}{r_1 + r_2}\left(r_1 log_2\left(\frac{a_{x,y}^2}{e_{x,y}^2}\right) + r_2 d_{x,y}\right)$$

Where, [18]

$a_{x,y} =$ Attested frequency of segment pair

$e_{x,y} =$ Expected Frequency of segment pair

$r_1 and\ r_2 =$ Scaling factor

$d_{x,y} =$ Similarity score of the original scoring function used to retrieve the attested and expected distributions

In the Distance Calculation step, after calculating the language specific scoring system above, distance between all the word pairs is calculated by using semi-global alignment analysis algorithm [18]. LexStat method uses the same technique as SCA method described in above section to calculate the similarity score and uses Monte Carlo permutation method to calculate similarity of a sequence compared with itself.

In the last and final step, clustering techniques are applied to cluster the sequences into group of cognates. In this technique, LexStat uses Flat cluster UPGMA algorithm with a user defined threshold. This method clusters all the words in the same similarity slot into set of cognates.

Pointwise Mutual Information

The concept of Pointwise Mutual Information was first introduced in 1961 by Robert Fano by the name mutual information, but today it is the most important concept in Natural Language Processing and is used to find the related measure of dependence between random variables. Pointwise Mutual Information compares the likelihood of two events occurring simultaneously to the likelihood of these events occurring independently. As shown below, the technique computes the (log) chance of co-occurrence multiplied by the product of the single likelihood of occurrence:

$$\mathrm{PI}(\mathrm{x}, \mathrm{y}) = log \frac{p(x, y)}{p(x)p(y)}$$

Where,
P (x, y) = Joint distribution of X and Y
P(x) and p(y) = Marginal distributions

Knowing that while 'x' and 'y' are independent, their joint probability is equal to the product of their marginal probabilities, when the ratio equals 1 (thus the log = 0), it indicates that the two words do not jointly constitute a unique notion; rather, their co-occurrence is accidental.

On the other hand, if both words—or even just one of them—has a high joint probability when combined with the other term, it indicates that the two words are likely to communicate a similar notion.

Infomap Algorithm

Infomap is a map-based network clustering technique that partitions data based on the flow caused by the links in a particular network. Infomap algorithm's goal is to reduce the cost function [19]. Assuming a sender pretends to convey a random path inside a network to a receiver, the following assumption is made: the size of this message is supposed to be kept as little as possible. A quick solution would be to give each node a unique name (code) and broadcast the corresponding sequence to the receiver. Based on an appropriate codebook [19], the latter may decode the message. Given that the path is given in binary language by N codes of the same length, the minimum length L of each code word is,

$$L = log_2 \mathrm{N}$$

Using Huffman coding is an additional, [19]concise approach to represent the same path. This method involves assigning codes of various lengths to each node based on the random path's ergodic node visit frequency. By itself, Huffman coding is an effective approach to send discrete nodes that make up the random path we pretend to encode intermittently or discontinuously. According to Shannon's source coding theory, the entropy of the random variable itself determines the minimum description length (MDL) when we use n codewords to represent n states of a random variable X and is given by,

$$\mathrm{H(X)} = -\sum_{1}^{n} p_i \log\,(p_i)$$

When transmitting the complete path or important sequences of the associated nodes, [19]the description can be made more effective. To do this, the network is divided into modules, and a distinct code book is developed for each module. This makes it possible to use the same identifications across various nodes. For these layered descriptions, the MDL is calculated using a weighted average of the frequency that each module and index codebook are used by the below formula,

$$\mathrm{L} = (\mathrm{M}) = qH(Q) + \sum_{m=1}^{n_m} p_{\odot}^{m}\ H(P_m)$$

Where, [19]

q = ∑(m)(j=1) qj: the sum of the exit probability for each community

H(Q) = Average length of code required for movement between two communities

$p_{\odot}^{m}$= qi + ∑(m)(β∈i) pβ: Probability for a random walk-in community c

$H(P_m)$ = Average code length of module codebook m

Substituting the values of H(Q), $p_{\odot}^{m}$= qi + ∑(m)(β∈i) pβ, q = ∑(m)(j=1) qj and H(P_m) in the equation we get,

$$\mathrm{H(Q)} = -\sum_{i=1}^{m} \frac{q_i}{\sum_j^m qj} \log\left(\frac{q_i}{\sum_{j=1}^{a} qj}\right) \qquad (1)$$

$$H(P_m) = -\frac{q_i}{q_i + \sum_{\beta\in i}^{m} \mathrm{p}\beta} \sum_{i=1}^{m} log\left(\frac{q_i}{q_i + \sum_{\beta\in i}^{m} \mathrm{p}\beta}\right) - \sum_{\alpha\in i} \frac{\mathrm{p}_\alpha}{q_i + \sum_{\beta\in i}^{m} \mathrm{p}\beta} log\left(\frac{\mathrm{p}_\alpha}{q_i + \sum_{\beta\in i}^{m} \mathrm{p}\beta}\right) \qquad (2)$$

Substituting equation (1) and equation (2) in L(M), we get

$$L(M) = \left(\sum_{m\in M} q_m\right) log\left(\sum_{m\in M} q_m\right) - 2\left(\sum_{m\in M} q_m \log(q_m) - \left(\sum_{\alpha\in V} \mathrm{p}_\alpha \log\left(p_\alpha\right)\right)\right) + \sum_{m\in M} (q_m + \sum_{\alpha\in M} \mathrm{p}_\alpha) \log\left(q_m + \sum_{\alpha\in M} \mathrm{p}_\alpha\right)$$

In a network that is both unweighted and undirected, the relative weight w is calculated by dividing the total weight of the edges connecting to a module by twice the total weight of all the links in the graph. The infomap method demonstrates that compression issues can also be solved using community finding algorithms.

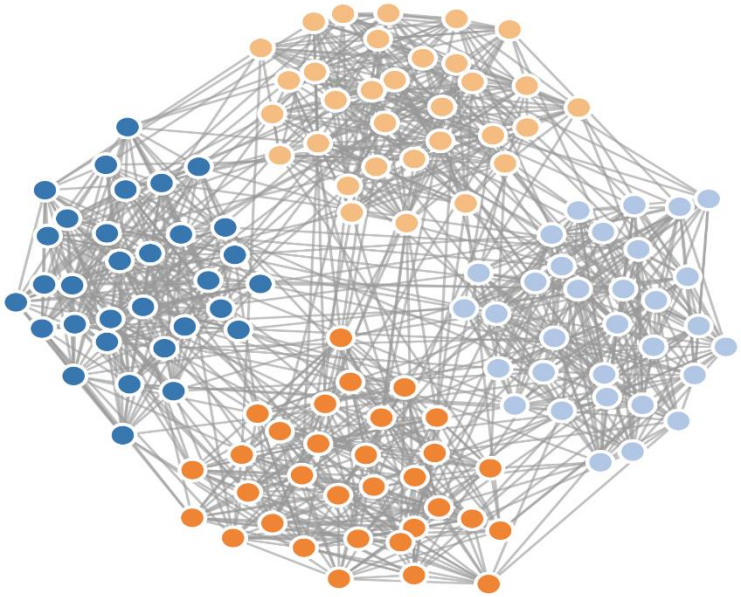


Figure 11:Infomap Clustering

Reconstruction of Phylogenetic Tree

The next step after detecting the cognates is to reconstruct the phylogenetic tree. For reconstruction of the phylogenetic trees, Phylogenetic Inference combined with Markov Chain Monte Carlo model is utilized.

Bayesian Phylogenetic Inference with Markov Chain Monte Carlo

Statistical inference is the process through which we learn about what we don't observe based on what we do notice. In other terms, it is the process of deriving inferences, such as punctual estimations, confidence intervals, or distribution estimations, about some latent variables (typically causes) in a population, based on some observed variables (often effects) in this population or in a sample of this population. Bayesian inference, in particular, is the process of making statistical inference from a Bayesian perspective. The Bayesian paradigm, to put it simply, is a probabilistic/statistical paradigm in which prior knowledge, described by a probability distribution, is updated each time a new observation, whose uncertainty is modelled by a different probability distribution, is recorded.

Bayesian phylogeny inference uses information from the prior and the data likelihood to get the posterior probability of trees, which is the probability that the tree is right given the data, the prior, and the likelihood model. The idea of the Bayesian phylogeny is based on the law,

$$f(\tau, \upsilon, \theta \mid X) = \frac{f(X \mid \tau, \upsilon, \theta) f(\tau, \upsilon, \theta)}{f(X)}$$

Where,

X = Data Matrix, which is a binary matrix with dimensions M × C where M is the number of languages and C is the number of cognate clusters.

τ = Topology of the tree

υ = Branch lengths

θ = Substitution model parameters

Instead of dealing with tedious computations, MCMC can be utilized in Bayesian inference to create samples straight from the "not normalized part" of the posterior. To calculate the posterior distribution $f(\tau, \upsilon, \theta | X)$, we have utilized Markov Chain Monte Carlo Method. Markov Chain Monte–Carlo (MCMC) is a prominent method for acquiring distribution information, particularly for calculating posterior distributions in Bayesian inference. It allows one to evaluate a distribution without understanding all of the mathematical features of the distribution by randomly sampling values from it. The term MCMC is a combination of two methods: Monte–Carlo and Markov chain. Monte–Carlo is a method of evaluating the parameters of distribution by studying random samples from it. The concept behind the Markov chain attribute of MCMC is that random samples are generated through a chain of process with sequence. The "Markov" property of the chain states that while each new sample is dependent on the one before it, new samples are not dependent on any samples before the prior one. Because of the emphasis on posterior distributions, which are often

difficult to work with via analytic study, MCMC is particularly effective in Bayesian inference. In the dissertation, we will be using the Metropolis-Hastings (MH) algorithm of MCMC to sample the phylogenies from the posterior distribution. The MH method creates a Markov chain of parameter states by suggesting a change to a single parameter or a block of parameters in ψ. Here $\psi = f\ (\tau, \upsilon, \theta)$. In the Markov chain, the present state has a parameter θ, and a new $\theta\ *$ value is offered from a distribution q ($\theta\ * \ |\theta$), which is then accepted with a probability. The MH algorithm is given by the formula,

$$r = \frac{f(X|\tau,\upsilon,\theta^*)f(\theta^*)q(\theta|\theta^*)}{f(X|\tau,\upsilon,\theta)f(\theta)q(\theta^*|\theta)}$$

Evaluation of Results

To evaluate the identified cognates and compare it with the previous work, Bcubed F score method for evaluation is used. To calculate the result below metrics are required.

$$\text{Precision} = \frac{\text{True Cognates Identifed}}{\text{All the positive Classification (TP + FP)}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{Actual Postives}}$$

The B-Cubed metrics assigns precision and recall to each item (here word) in a set of clusters

$$\text{F score} = \frac{2 * \text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})}$$

Implementation

Data Collection

To collect the data from the Wiktionary website, web scraping with beautiful soup library is used to iterate over the Swadesh list and saved the dataset in a tsv (Comma-separated format) file format. Figure 7 shows the data extracted by web scraping for the word 'I' in all the Indic languages.

The procedure to collect data with web scraping is completed in 4 steps.

1. In the Wiktionary website all the data is given in table format. Therefore, first, all the words from the table were extracted in a list. There were a few missing entries for the words in the 'Kashmiri' language, these missing entries were replaced by 'None'. All of these words in the table have a hyperlink that redirects the page to another page that contains the IPA of that particular word. To collect the IPA, all such hyperlinks were found and appended to another list. A tuple of list is created for every concept which consists of 3 entries for every concept-Language, word, and the link to IPA.

2. In the second step we create a file to store IPA for each word we found in step 1. For this, every word and the link in the tuple of list is iterated and IPA entry for each word is extracted with web scraping's beautiful soup method from the page and stored in text file.

3. In step 3, we read this IPA file and join the IPA entries with the Cognancy, Concept, Word, and dialect extracted from the above steps together in a list. This list is created for every word. Here to select the words, which can be treated as concepts, words from the Sanskrit language were chosen.

4. In step 4 we create a tsv file that contains all the entries from step 3. Throughout the research, we will use this file for analysis.

5. IPA entries for some of the words were not present in the Wiktionary website, therefore in the last step, the IPA entries with None value were replaced to their proginal IPA format manually. To calculate the IPA for such words, Wikipedia was utilized and a deep study of grammar was done

Out[31]:

	COGNACY	CONCEPT	COUNTERPART	IPA	DOCULECT
0	0	ahám	ahám	ɐɦɐm	Sanskrit
1	0	ahám	ahaṃ	ˌɑːˈhɑṃ	Pali
2	0	ahám	maĩ	mɛː	Hindi
3	0	ahám	maĩ	mɛː	Urdu
4	0	ahám	ma	mə	Nepali
5	0	ahám	ham	ˈhəm	Bhojpuri
6	0	ahám	maĩ	mɛː	Punjabi
7	0	ahám	ãũ,	ɑːũː	Sindhi
8	0	ahám	hũ	hũː	Gujarati
9	0	ahám	mī,	mi	Marathi
10	0	ahám	hãv	ɦɑːʊ	Konkani
11	0	ahám	moi	mɔɪ	Assamese
12	0	ahám	ami	ami	Bengali
13	0	ahám	muṁ	mũ	Oriya
14	0	ahám	bụ	bɨ	Kashmiri
15	0	ahám	mama,	ˈmɑmə	Sinhalese
16	0	ahám	ahareṅ,	əhərɛn	Dhivehi
17	0	ahám	me	miː	Romani

Figure 7: Example of the data extracted from Wiktionary for the word 'I'

The details of all the entries in the file are given below

Cognacy

The column 'COGNACY' is used to store ID number for each concept. As there are 18 languages for evaluation, each concept is given an ID which will be common for all the words belonging to the same concept. Therefore, as seen in Figure 7, all the words belonging to the concept 'ahám' have cognacy 0.

Concept

A concept is a linguistic and philosophical unit that categorizes and specifies each potential meaning of a word and its evolution. Here the words in Sanskrit language are chosen as concepts as it is the Proto-language of all the Indic languages.

Counterpart

The column counterpart represents the words in Indo-Aryan languages for each concept. In this column, the words are also given with their IPA transliterations.

IPA

In this column, IPA of each word is extracted from the Wiktionary website. There were few missing entries for IPA, which were then filled manually after a thorough research and learning grammar rules of all the Indic languages.

Doculect

This column represents the Dialect (language) for every entry. There are total 18 languages taken from the Swadesh list. All these languages are primary languages of Indo-Aryan language family

Automatic Cognate Detection

This section describes the methods implemented for detecting the cognates automatically. The cognates are found using two different techniques given below.

1. Automatic Detection of Cognates using Lingpy library
    - Sound Class Based Alignments
    - LexStat Method with three different clustering techniques- UPGMA, edit-distance and Infomap
2. Automatic Detection of Cognates using Pointwise Mutual Information

The results obtained from all these methods are then evaluated to determine the most accurate method. The following sections describes the implementation and results obtained from the above method.

Automatic Detection of cognates using Lingpy library

Sound Class Based Alignment

The sound class-based alignment algorithm begins with an input file containing concept, words, IPA, Language and the concept ID. This input file is created during web scraping which is in TSV file format. The Lingpy file was then given to the SCA algorithm, which first converted the words present in the file into their respective tokens of sound class. After the sound classes are created, pairwise alignment was carried out for each pair of words in the file using Needleman-Wunsch algorithm. During this process the SCA module creates a scoring system. The gap penalty for scoring system was set to -1. Therefore, for giving the scores to the word pairs Word1 and Word2, a score of -1 was given if w1 (ith element of Word1) and w2 (jth element of Word2) are different and a score of 1 was given if w1 and w2 are similar and an alignment matrix was created. The output of the scoring system calculated was then imported into a CSV file. The figure below shows example of the scoring system created, for the concept 'agni' which means fire in English.

	ID	COGNACY	CONCEPT	COUNTERPART	IPA	DOCULECT	TOKENS	SONARS	PROSTRINGS	CLASSES	LANGID	NUMBERS	WEIGHTS	DUPLICATES
37	2989	166.0	agní	agní	əgni:	Sanskrit	ə g n i:	7 1 4 7	XBCZ	EKNI	15.0	15.E.V 15.K.C 15.N.C 15.I.V	1.5 1.75 1.5 0.8	0.0
38	2990	166.0	agní	aggi	əggi:	Pali	ə g g i:	7 1 1 7	XMBZ	EKKI	12.0	12.E.V 12.K.c 12.K.C 12.I.V	1.5 1.1 1.75 0.8	0.0
39	2991	166.0	agní	āg	ɑ:g	Hindi	ɑ: g	7 1	XN	AK	6.0	6.A.V 6.K.c	1.5 0.8	0.0
40	2992	166.0	agní	āg	ɑ:g	Urdu	ɑ: g	7 1	XN	AK	18.0	18.A.V 18.K.c	1.5 0.8	0.0
41	2993	166.0	agní	āgo	ägo	Nepali	ä g o	7 1 7	XBZ	AKU	10.0	10.A.V 10.K.C 10.U.V	1.5 1.75 0.8	0.0
42	2994	166.0	agní	āg	ɑ:g	Bhojpuri	ɑ: g	7 1	XN	AK	3.0	3.A.V 3.K.c	1.5 0.8	0.0
43	2995	166.0	agní	agga	əggɑ:	Punjabi	ə g g ɑ:	7 1 1 7	XMBZ	EKKA	13.0	13.E.V 13.K.c 13.K.C 13.A.V	1.5 1.1 1.75 0.8	0.0
44	2996	166.0	agní	bāhi	bɑ:ɦi	Sindhi	b ɑ:ɦi	1 7	AX	PA	16.0	16.P.C 16.A.V	2.0 1.5	0.0

Figure:

Figure 12:Scoring system created with the SCA model, Example word 'agni'

After creating a scoring system, a pairwise similarity for all the words in the list was calculated to form a distance matrix. The distance matrix determines the distance between two phonetic sequences. In the next step, Neighbor-joining algorithm is used for clustering the sound class strings calculated in the previous steps. To cluster the strings, Lingpy has built-in library in the module for Neighbor-joining algorithm. The threshold for clustering the cognates was kept at 0.55 which gave the best results. For the thresholds > 0.55 the results did not give better results than 0.55.

After clustering, the Lingpy module carries out the alignments for all the sequences, using progressive alignment approach and then the Lingpy algorithm carries out swap analysis where the mismatched alignments are penalized by 2 and gap penalty is set to -1. The output of the analysis was then imported into a TSV/CSV file. Figure 10, shows the output of cognate detection using lingpy's SCA approach.

COGNACY	CONCEPT	COUNTERPART	IPA	DOCULECT	TOKENS	SONARS	PROSTRINGS	CLASSES	LANGID	NUMBERS	WEIGHTS	DUPLICATES	SCA_ID
166.0	agní	āga	ɑːga	Gujarati	ɑː g a	7 1 7	XBZ	AKA	5.0	5.A.V 5.K.C 5.A.V	1.5 1.75 0.8	0.0	26.0
166.0	agní	āg	ɑːg	Marathi	ɑː g	7 1	XN	AK	9.0	9.A.V 9.K.c	1.5 0.8	0.0	26.0
166.0	agní	ujo	uːzɔː	Konkani	uː z ɔː	7 3 7	XBZ	YSU	8.0	8.Y.V 8.S.C 8.U.V	1.5 1.75 0.8	0.0	15.0
166.0	agní	zui	zui	Assamese	z ui	3 7	AX	SY	1.0	1.S.C 1.Y.V	2.0 1.5	0.0	15.0
166.0	agní	agun	agun	Bengali	a g u n	7 1 7 4	XBYN	AKYN	2.0	2.A.V 2.K.C 2.Y.V 2.N.c	1.5 1.75 1.3 0.8	0.0	26.0
166.0	agní	niaṁ	None	Oriya	N o n e	0 7 4 7	AXBZ	0UNE	11.0	11.0.C 11.U.V 11.N.C 11.E.V	2.0 1.5 1.75 0.8	0.0	21.0
166.0	agní	None	None	Kashmiri	N o n e	0 7 4 7	AXBZ	0UNE	7.0	7.0.C 7.U.V 7.N.C 7.E.V	2.0 1.5 1.75 0.8	1.0	21.0
166.0	agní	gini	gini	Sinhalese	g i n i	1 7 4 7	AXBZ	KINI	17.0	17.K.C 17.I.V 17.N.C 17.I.V	2.0 1.5 1.75 0.8	0.0	21.0
166.0	agní	alifāṅ	əlifəːn	Dhivehi	ə l i f əː n	7 5 7 3 7 4	XBYBYN	ELIBEN	4.0	4.E.V 4.L.C 4.I.V 4.B.C 4.E.V 4.N.c	1.5 1.75 1.3 1.75 1.3 0.8	0.0	18.0

Figure 13:Result of cognate detection using SCA for example word 'agni'

Understanding the meaning of each filed of the output files for scoring function and clustering method

ID: The column ID represents the ID given to each concept by the Lingpy library. Each concept is therefore identified with its ID

Cognacy: The column cognacy was created during the creation of Lingpy file, which 'is used to store ID number for each concept. As there are 18 languages for evaluation, each concept is given an ID which will be common for all the words belonging to the same concept.

Concept: A concept is a linguistic and philosophical unit that categorizes and specifies each potential meaning of a word and its evolution. Here the words in Sanskrit language are chosen as concepts as it is the Proto-language of all the Indic languages.

Counterpart: The column counterpart represents the words in Indo-Aryan languages for each concept. In this column, the words are also given with their IPA transliterations.

Doculect: This column represents the Dialect (language) for every entry. There are total 18 languages taken from the Swadesh list. All these languages are primary languages of Indo-Aryan language family.

IPA: The IPA column consists of the phonetic symbols for each word.

Tokens: The column token was created by Lingpy method, in which the IPA entry for each word is tokenized into characters.

Sonras: Lingpy uses the tokens generated from IPA entry to determine the sound class of the word. This column contains the sound classes for each IPA character.

Prostrings: This column contains, the prosodic of the phonetic characters. Prosodic represents the rhythm, stress of characters, intonation of speech of all the phonetic characters combined together and represents the sound of the IPA characters when combined during speech.

Classes: This column represents the sound class for each word. The sound classes are defined in Lingpy library.

LangID: The Lingpy library has ID attached to each language. This column therefore represents the language ID associated with the Lingpy library for each language and word.

Numbers: The number column consists of language specific triples containing language ID, Sound class string and prosodic string.

Weights: The weights column represents the gap weights penalized for each IPA character during the scoring system creation.

Duplicates: This column represents the number of times a word has appeared to be a duplicate of after word.

SCA_ID: This is the most important column for our analysis. The SCA_ID column contains the cognate ID's given to each word after clustering. If two words have the same SCA_ID, that means those words are cognates of each other. For example, from the above figure, we have taken the word 'agni'. If we see the SCA_ID for the word in all the languages, the Lingpy clustering algorithm has given a cognate ID '26' to the word in Gujarati, Marathi and Bengali. This indicates that the words āga(Gujarati), āg(Marathi) and agun(Bengali) are cognates of each other.

The loan words in the Lingpy module are represented by negative cognate ID. The SCA algorithm could not deal with borrowing detection/loan word detection efficiently. Out of 3726 words 60 words represented a borrowing relation, which is not very accurate.

The SCA employs pairwise alignment to determine how far apart two words are. In alignment analyses, two sequences are placed in a matrix so that all corresponding segments are in the same column and non-corresponding segments are filled in the vacant matrix cells with gap symbols. The figure below shows the pairwise alignment calculated by SCA for the word 'agni' which means fire. The Lingpy model allows to export the output of the alignments in the form of HTML file, which gives a clear and better visual

Concept: *agni* (ID: 1)			
CogID	**Language**	**Entry**	**Aligned Entry**
16	Assamese	zui	– – – z ui – –
16	Bengali	agun	a g – – u n –
16	Bhojpuri	ɑ:g	ɑ: g – – – – –
16	Dhivehi	əlifə:n	ə l i f ə: n –
16	Gujarati	ɑ:ga	ɑ: g – – – – a
16	Hindi	ɑ:g	ɑ: g – – – – –
16	Kashmiri	None	– N – – o n e
16	Konkani	u:zɔ:	– – u: z ɔ: – –
16	Marathi	ɑ:g	ɑ: g – – – – –
16	Nepali	ägo	ä g – – – – o
16	Oriya	niaṁ	– – – n ia ṁ –
16	Pali	əggi:	ə g – – – g i:
16	Punjabi	əggɑ:	ə g – – – g ɑ:
16	Romani	dʒæg	– d – ʒ æ g –
16	Sanskrit	əgni:	ə g – – – n i:
16	Sindhi	bɑ:ɓi	– – – b ɑ:ɓi – –
16	Sinhalese	gini	– g – – i n i
16	Urdu	ɑ:g	ɑ: g – – – – –

Figure 14:Pairwise Alignment for the word 'agni'

Automatic Cognate Detection with LexStat

The LexStat Method for detecting cognates is used with 2 different types of clustering Methods.

LexStat takes the same kind of input file as that of SCA, and therefore the same Lingpy file with 207 words with their IPA, Language, Concept and ID is given to LexStat. Similar to SCA, LexStat also first the input sequence is tokenized and transformed into tuples of prosodic strings and sound classes and created a scoring system. The output of the scoring system by LexStat is given below.

ID	COGNACY	CONCEPT	COUNTERPART	IPA	DOCULECT	TOKENS	SONARS	PROSTRINGS	CLASSES	LANGID	NUMBERS	WEIGHTS	DUPLICATES
2989	166.0	agní	agní	əgniː	Sanskrit	ə g n iː	7 1 4 7	XBCZ	EKNI	15.0	15.E.V 15.K.C 15.N.C 15.I.V	1.5 1.75 1.5 0.8	0.0
2990	166.0	agní	aggi	əggiː	Pali	ə g g iː	7 1 1 7	XMBZ	EKKI	12.0	12.E.V 12.K.c 12.K.C 12.I.V	1.5 1.1 1.75 0.8	0.0
2991	166.0	agní	āg	ɑːg	Hindi	ɑː g	7 1	XN	AK	6.0	6.A.V 6.K.c	1.5 0.8	0.0
2992	166.0	agní	āg	ɑːg	Urdu	ɑː g	7 1	XN	AK	18.0	18.A.V 18.K.c	1.5 0.8	0.0
2993	166.0	agní	āgo	ägo	Nepali	ä g o	7 1 7	XBZ	AKU	10.0	10.A.V 10.K.C 10.U.V	1.5 1.75 0.8	0.0
2994	166.0	agní	āg	ɑːg	Bhojpuri	ɑː g	7 1	XN	AK	3.0	3.A.V 3.K.c	1.5 0.8	0.0
2995	166.0	agní	agga	əggɑː	Punjabi	ə g g ɑː	7 1 1 7	XMBZ	EKKA	13.0	13.E.V 13.K.c 13.K.C 13.A.V	1.5 1.1 1.75 0.8	0.0
2996	166.0	agní	bāhi	bɑːʰi	Sindhi	b ɑːʰi	1 7	AX	PA	16.0	16.P.C 16.A.V	2.0 1.5	0.0

Figure 15:Scoring System created by LexStat

The scoring system created by LexStat is exactly similar to the scoring system created by the SCA method. All the information regarding the columns is given in the section of SCA implementation. The output of the scoring system was then used by the LexStat Algorithm to calculate a preliminary list of cognates. With the result of the preliminary cognates, LexStat calculates the pairwise alignments as well as performs shuffling the wordlist by iterating 1000 times. After this step, the LexStat method calculates the log-odds of the preliminary cognates and the expected cognates and computes the pairwise distance to compute cognates. After this step, to cluster the cognates into groups, two methods are used. First the cognates are clustered with edit-distance clustering algorithm with a threshold of 0.75, which gave the best cognate results.

For clustering the cognates with Infomap Clustering, all the above steps are followed first, followed by Infomap algorithm. The threshold values tried here were 0.55, 0.6, 0.75 out of which threshold 0.55 gave the best and efficient results. Both the clustering libraries are included in the Lingpy library, and hence in the cognate detection, this library is used to cluster. The results of cognate detection by both the methods are then imported to a TSV/CSV file.

As compared to the SCA method, the LexStat Algorithm could determine the cognates more precisely, but the borrowing/loan detection results remained the same. The figures below show the cognate judgements for the word ‘agni’ for both LexStat-edit-distance and LexStat-Infomap.

COGNACY	CONCEPT	COUNTERPART	IPA	DOCULECT	TOKENS	SONARS	PROSTRINGS	CLASSES	LANGID	NUMBERS	WEIGHTS	DUPLICATES	EDITID
166.0	agní	āga	ɑːga	Gujarati	ɑː g a	7 1 7	XBZ	AKA	5.0	5.A.V 5.K.C 5.A.V	1.5 1.75 0.8	0.0	37.0
166.0	agní	āg	ɑːg	Marathi	ɑː g	7 1	XN	AK	9.0	9.A.V 9.K.c	1.5 0.8	0.0	37.0
166.0	agní	ujo	uːzɔː	Konkani	uː z ɔː	7 3 7	XBZ	YSU	8.0	8.Y.V 8.S.C 8.U.V	1.5 1.75 0.8	0.0	36.0
166.0	agní	zui	zui	Assamese	z ui	3 7	AX	SY	1.0	1.S.C 1.Y.V	2.0 1.5	0.0	36.0
166.0	agní	agun	agun	Bengali	a g u n	7 1 7 4	XBYN	AKYN	2.0	2.A.V 2.K.C 2.Y.V 2.N.c	1.5 1.75 1.3 0.8	0.0	37.0
166.0	agní	niaṁ	None	Oriya	N o n e	0 7 4 7	AXBZ	0UNE	11.0	11.0.C 11.U.V 11.N.C 11.E.V	2.0 1.5 1.75 0.8	0.0	42.0
166.0	agní	None	None	Kashmiri	N o n e	0 7 4 7	AXBZ	0UNE	7.0	7.0.C 7.U.V 7.N.C 7.E.V	2.0 1.5 1.75 0.8	1.0	42.0
166.0	agní	gini	gini	Sinhalese	g i n i	1 7 4 7	AXBZ	KINI	17.0	17.K.C 17.I.V 17.N.C 17.I.V	2.0 1.5 1.75 0.8	0.0	42.0
166.0	agní	alifāṅ	əlifəːn	Dhivehi	ə l i f əː n	7 5 7 3 7 4	XBYBYN	ELIBEN	4.0	4.E.V 4.L.C 4.I.V 4.B.C 4.E.V 4.N.c	1.5 1.75 1.3 1.75 1.3 0.8	0.0	39.0
166.0	agní	jag	ʤæg	Romani	ʤ æ g	1 3 7 1	ACXN	TSEK	14.0	14.T.C 14.S.C 14.E.V 14.K.c	2.0 1.5 1.5 0.8	0.0	49.0

Figure 16:Cognate detection for word 'agni' by LexStat-Edit distance

CONCEPT	COUNTERPART	IPA	DOCULECT	COGID	TOKENS	SONARS	PROSTRINGS	CLASSES	LANGID	NUMBERS	WEIGHTS	DUPLICATES	INFOMAPID
agní	āga	ɑːga	Gujarati	27.0	ɑː g a	7 1 7	XBZ	AKA	5.0	5.A.V 5.K.C 5.A.V	1.5 1.75 0.8	0.0	2.0
agní	āg	ɑːg	Marathi	27.0	ɑː g	7 1	XN	AK	9.0	9.A.V 9.K.c	1.5 0.8	0.0	2.0
agní	ujo	uːzɔː	Konkani	334.0	uː z ɔː	7 3 7	XBZ	YSU	8.0	8.Y.V 8.S.C 8.U.V	1.5 1.75 0.8	0.0	1.0
agní	zui	zui	Assamese	334.0	z ui	3 7	AX	SY	1.0	1.S.C 1.Y.V	2.0 1.5	0.0	1.0
agní	agun	agun	Bengali	-4.0	a g u n	7 1 7 4	XBYN	AKYN	2.0	2.A.V 2.K.C 2.Y.V 2.N.c	1.5 1.75 1.3 0.8	0.0	2.0
agní	niaṁ	niaṁ	Oriya	673.0	n ia ṁ	4 7 4	AXN	NIM	11.0	11.N.C 11.I.V 11.M.c	2.0 1.5 0.8	0.0	5.0
agní	None	None	Kashmiri	627.0	N o n e	0 7 4 7	AXBZ	0UNE	7.0	7.0.C 7.U.V 7.N.C 7.E.V	2.0 1.5 1.75 0.8	1.0	4.0
agní	gini	gini	Sinhalese	-2.0	g i n i	1 7 4 7	AXBZ	KINI	17.0	17.K.C 17.I.V 17.N.C 17.I.V	2.0 1.5 1.75 0.8	0.0	9.0
agní	alifāṅ	əlifəːn	Dhivehi	531.0	ə l i f əː n	7 5 7 3 7 4	XBYBYN	ELIBEN	4.0	4.E.V 4.L.C 4.I.V 4.B.C 4.E.V 4.N.c	1.5 1.75 1.3 1.75 1.3 0.8	0.0	3.0
agní	jag	ʤæg	Romani	27.0	ʤ æ g	1 3 7 1	ACXN	TSEK	14.0	14.T.C 14.S.C 14.E.V 14.K.c	2.0 1.5 1.5 0.8	0.0	8.0

Figure 17:Cognate detection for word 'agni' by LexStat-Edit Infomap

As we can see from the results of both the methods, the cognate IDs assigned by LexStat-Edit-Distance in the column edited and the cognate IDs assigned by LexStat-Infomap in the column infomapid are different, but both of these methods identify the words āga(Gujarati), āg(Marathi) and agun(Bengali) are cognates of each other. This shows that both the techniques work sufficiently and almost gave almost similar results.

Pointwise Mutual Information

The PMI approach starts with an empty PMI matrix and a list of cognate word pairs from the Swadesh list. Using the current PMI matrix, the technique proceeds by constructing the PMI matrix from alignments calculated for each minibatch of word pairs. The current PMI matrix is then combined with the most recent minibatch's generated PMI matrix. This process is repeated a certain number of times. To determine the pairwise word similarity matrix for each meaning, we use the final PMI matrix. In a final stage, the distance score was created by applying the sigmoid transformation to the similarity score given by,

$$1.0 - (1 + \exp(-x))^{-1}$$

Where x is the similarity score obtained. The Label Propagation technique is then used to infer cognate clusters using the word distance matrix as an input.

The PMI scores between two sounds i, j is calculated by the formula,

$$\mathrm{PMI(i,j)} = \log \frac{\mathrm{p(i\,,j)}}{\mathrm{q(i).\,q(j)}}$$

p (i, j) = probability of i, j being cognates

q(j) = the probability of the chosen segment in the chosen word equals j

q(i) = the probability of the chosen segment in the chosen word equals i

If the PMI value is positive, it indicates that the chances of I aligning with j in a pair of cognates is high, but a negative PMI value indicates that the alignment of I and j is more likely the result of chance rather than shared heritage. Levenshtein distance and the vanilla Needleman-Wunsch algorithm (VNW) are used in the PMI weighted alignment. In terms of similarity, the Levenshtein distance is analogous to the standard Needleman-Wunsch approach. In VNW, the similarity score is enhanced by 1 when a character or sound segment match occurs, and by -1 when a character mismatch does. The Levenshtein distance technique treats all insertions, deletions, and replacements the same way, but

the VNW algorithm includes a deletion operation called the gap opening penalty parameter that must be provided individually. Since deletion occurs in chunks, a different parameter termed gap extension, which has a lesser or equivalent penalty than gap opening, should is used instead. The weighted Needleman-Wunsch algorithm establishes the alignment between two input strings by maximizing the pairwise similarities of matched segment pairs, which requires a similarity score for each pair of segments. The PMI weighted alignment is used in conjunction with Online Expectation Maximization (EM). Following an initial configuration of model parameters, the EM method uses a model parameter to realign the words in a sentence pair. The model parameters are re-estimated using the word alignments from the previous iteration. The EM technique re-estimates the model parameters after each complete scan of the training data. The Online EM technique uses the following equation to combine the parameters estimated from the current update step k with the previous parameters θ_{k-1},

$$\theta_k = (1 - \eta k)\theta_{k-1} + \eta k^s$$

$\eta k = (k + 2)^{-\alpha}$

θ = PMI scores of all the segment pairs

α = 0.5 ≤ α ≤ 1

k belongs to the mini batch parameter, m = $\frac{D}{k}$, here D is the size of training data which computes the total iterations.

The above procedure was applied for the cognate detection phase as following,

1. In the first step, the input file we generated with web scarping was first converted into ASJP format, and the data was cleaned as per ASJP standards.
2. Apply normalized Levenshtein distance algorithm to obtain the list of possible word pairs from Swadesh list which as most likely cognates. In the research we considered all the word pairs with a length normalized Levenshtein distance (LDN) of less than 0.5 to be plausible cognates.
3. Used the Needleman-Wunsch technique to align the list of possible cognates.
4. We take advantage of this trait to create a partial graph and feed the subsequent network into the InfoMap method.
5. Using the Needleman-Wunsch algorithm and result obtained from step 2, created a new set of alignments.
6. Steps 2 and 3 were repeated until the average similarity between repetitions remained constant.
7. After completing the above steps, a PMI based score was generated for each word pair. To get the PMI score in the range of [0, 1], the score was transformed into distance score using the formula of sigmoid transformation described above.
8. The word distance matrix is then supplied to the label propagation algorithm with a threshold of 0.5

9. The output of the cognate judgements was then converted into a nexus file, which will be used later for reconstructing phylogenetic tree.

The below figures show few output results of the PMI algorithm.

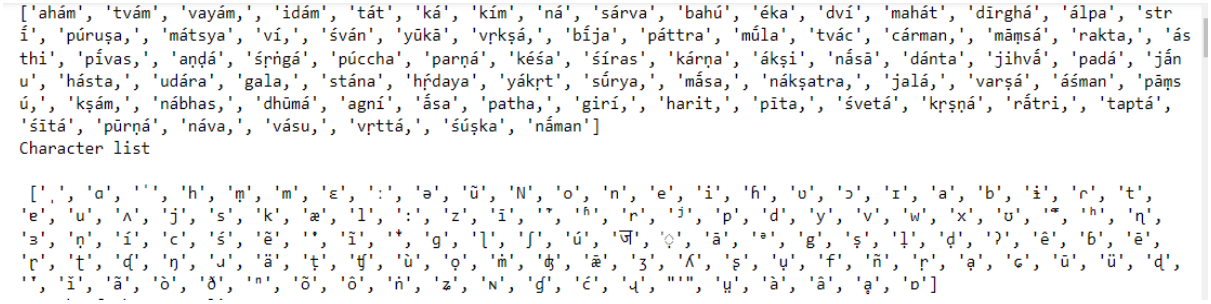


Figure 18:List of words and the tokenized IPA characters selected for first minibatch

```
Size of initial list  1888
Iteration  0
Non zero examples  1888 1611.0  number of updates  8
Iteration  1
Non zero examples  1611 1609.0  number of updates  15
Iteration  2
Non zero examples  1609 1608.0  number of updates  22
Iteration  3
Non zero examples  1608 1604.0  number of updates  29
Iteration  4
Non zero examples  1604 1603.0  number of updates  36
Iteration  5
Non zero examples  1603 1603.0  number of updates  43
Iteration  6
Non zero examples  1603 1603.0  number of updates  50
Iteration  7
Non zero examples  1603 1602.0  number of updates  57
Iteration  8
```

Figure 19:Iterations running in PMI algorithm

Evaluation of the Automatic Cognate Detection

For evaluating the methods and performance of cognate detection, bcubes method is used. In this method, the performance is evaluated based on the Precision, Recall and F1 Scores of the cognate detection results. To compare the cognates, a gold standard cognate judgement was necessary. As it is difficult to create a gold stand cognate judgement list, In the research the cognate judgements carried out by the author of [20]. The research contains manually curated cognate IDs for Swadesh List of 100 words. This list of cognate ID's is considered gold standard list. Now to evaluate the performance of our methods, with gold standard metrics, first another Swadesh list of 100 words was created which consisted of only those words which were present in the gold standard list. Below figure shows the gold standard cognate IDs for 100 words of Swadesh list, which were taken from the research of the author [20].

:

Words	Sanskrit	Pali	Hindi	Urdu	Nepali	Bhojpuri	Punjabi	Sindhi	Gujarati	Marathi	Konkani	Assamese	Bengali	Oriya	Kashmiri	Sinhalese	Dhivehi
all	25	22	1	1	1	12.0	1	1	1	270	270	1	1	1	17.0	270	174
ash	62	60	3	1	140	54.0	111	3	3	3	606	152	152	567	57.0	382	382
bark	116	114	4	4	328	107.0	4	4	4	4	4	328	4	4	111.0	383	523
belly	141	141	678	678	678	134.0	678	678	678	-1	-1	678	678	678	134.0	329	329
big	156	154	5	5	358	146.0	5	5	248	248	5	5	5	5	150.0	385	669
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
what	2307	2315	96	96	96	2307.0	96	96	96	96	96	96	96	96	2311.0	96	96
white	2356	2354	97	97	375	2346.0	97	205	267	290	267	134	267	267	2351.0	428	428
who	2358	2358	99	99	99	2358.0	99	99	99	99	99	99	99	99	2363.0	99	99
woman	2449	2450	-7	-7	244	2442.0	135	206	268	268	-8	135	321	268	2446.0	429	244
yellow	2482	2488	100	100	100	2478.0	100	100	100	100	322	322	322	322	2483.0	431	546

99 rows × 18 columns

Figure 20: Gold Strand Cognate ID file

With the help of Pandas, a new Swadesh file of 100 words was created which consisted of the columns, Cognacy, Concept, Word, IPA, Language, ID, gold standard cognate ID. The figure below shows an example of such file. This file was then given as input to all the automatic cognate detection methods described above and the performance for each method was evaluated using bcubes which calculates Precision, Recall and F1-score. Precision is a measure of how many of the positive predictions are correctly identified. Recall is a measure of how many of the positive cases are identifies correctly and F1-score combines both precision and recall and known as harmonic mean of the two.

	COGNACY	CONCEPT	COUNTERPART	IPA	DOCULECT	COGID
0	0	ahám	ahám	ɐɦɐm	Sanskrit	9
1	0	ahám	ahaṃ	ˌɑˈhɑm̩	Pali	8
2	0	ahám	mãĩ	mɛː	Hindi	41
3	0	ahám	mãĩ	mɛː	Urdu	41
4	0	ahám	ma	mə	Nepali	41
5	0	ahám	ham	ˈhəm	Bhojpuri	2
6	0	ahám	maĩ	mɛː	Punjabi	41
7	0	ahám	ãũ̃,	ɑːũː	Sindhi	41
8	0	ahám	hũ	None	Gujarati	41
9	0	ahám	mĩ,	mi	Marathi	41
10	0	ahám	hãv	ɦɑːʊ	Konkani	41
11	0	ahám	moi	mɔɪ	Assamese	41
12	0	ahám	ami	ami	Bengali	41
13	0	ahám	muṁ	mũ	Oriya	41
14	0	ahám	bụ	bɨ	Kashmiri	6
15	0	ahám	mama,	ˈmɑmə	Sinhalese	41
16	0	ahám	ahareṅ,	əhəren	Dhivehi	41
17	0	ahám	me	miː	Romani	41

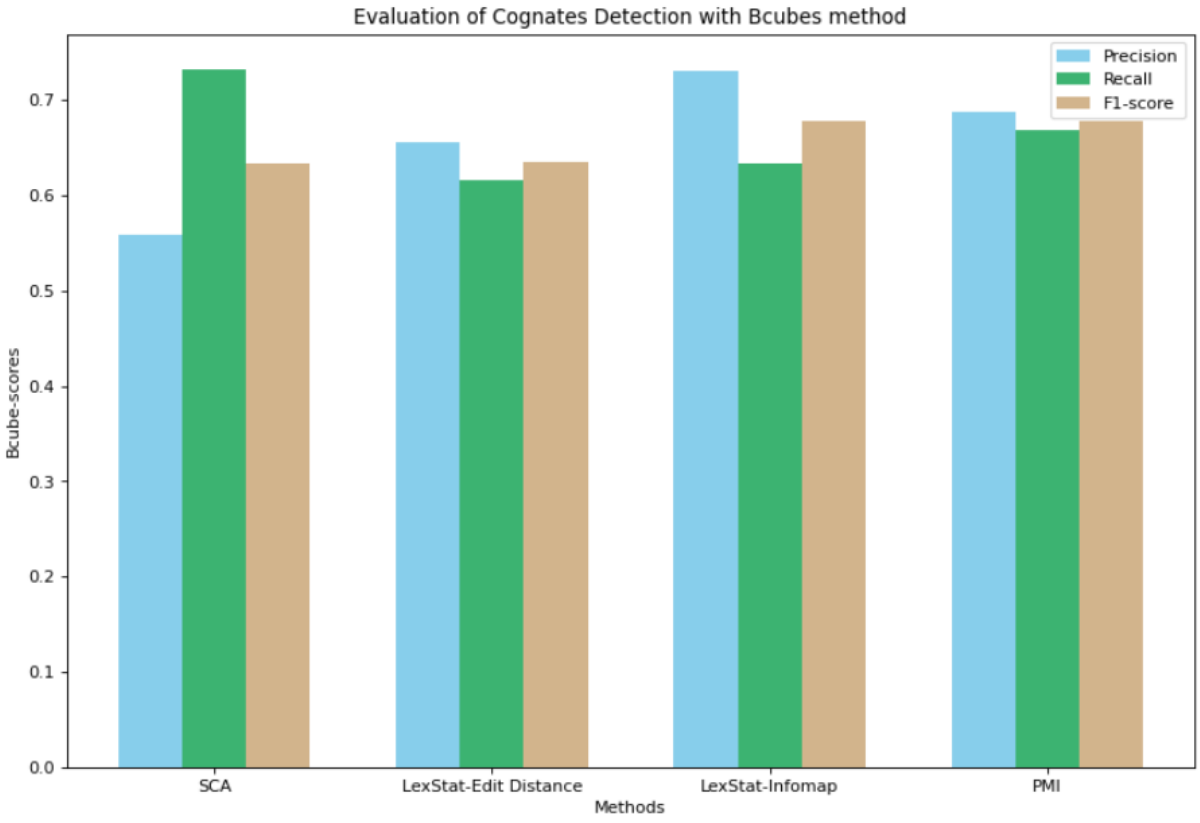
Figure 21:Swadesh list file to evaluate the performance of the methods

Results Obtained

Table 3:: Performance evaluation of all the methods

	Threshold	Precision	Recall	F1-Score
SCA	0.5	0.55910	0.73300	0.63430
LexStat-Edit Distance	0.75	0.65640	0.61570	0.63540
LexStat-Infomap	0.55	0.73050	0.63310	0.67830
PMI	0.5	0.68847	0.66925	0.67873

Table 4:Plot of performance evaluation for automatic cognate detection methods



From the above evaluation scores, it can be seen that The Pointwise Mutual Information approach outperforms all the other methods. The Lingpy based methods SCA, LexStat-Edit-Distance performs good at separating loan words and cognates but fails to detect a large number of cognates correctly. The Pointwise Mutual Information and LexStat-Infomap have a very close score with all the performance metrics and were able to detect most of the loan words correctly as well. One disadvantage of evaluating the performance on small dataset (100 words) is that, if the dataset is very small and the distance between languages is large it is quite difficult to prove the cognacy between words. While the LexStat method also performs very well, it lacks in determining borrowings as it creates a language specific scoring system. With language independent scoring system LexStat can handle borrowings more accurately. To manage the False cognate Identification, the threshold can be minimized for all the Automatic Cognate detection methods we utilized. Hence, the PMI approach with a threshold of 0.5 performed best, but after minimizing the threshold to 0.45, 0.4, the results

remained same. This may mean that the dataset was overfitted and hence could not classify the cognates more accurately.

As the PMI and LexStat-Infomap methods performed best, further these methods were used to calculate the distance between two languages. This distance was calculated as a correlation matrix, which consisted of text file, with one language and its distances on each line, separated by spaces, and the number of languages in the first row. The below figure shows the distance of the languages with each other.

	Languages	Assamese	Bengali	Bhojpuri	Dhivehi	Gujarati	Hindi	Kashmiri	Konkani	Marathi	Nepali	Oriya	Pali	Punjabi	Romani	Sanskrit	Sindhi	Sinhalese	Urdu
0	Assamese	0.0000	0.3247	1.0000	0.4675	0.4868	0.4805	1.0000	0.5195	0.5455	0.4416	0.3766	1.0000	0.4416	0.5325	1.0000	0.4737	0.5325	0.4805
1	Bengali	0.3247	0.0000	1.0000	0.5325	0.4211	0.4156	1.0000	0.4675	0.4675	0.4805	0.2987	1.0000	0.3896	0.5455	1.0000	0.4342	0.5714	0.4156
2	Bhojpuri	1.0000	1.0000	0.0000	1.0000	1.0000	1.0000	0.8182	1.0000	1.0000	1.0000	1.0000	0.8831	1.0000	1.0000	0.7403	1.0000	1.0000	1.0000
3	Dhivehi	0.4675	0.5325	1.0000	0.0000	0.5263	0.5714	1.0000	0.5844	0.5714	0.5584	0.5325	1.0000	0.5325	0.5844	1.0000	0.5395	0.3766	0.5714
4	Gujarati	0.4868	0.4211	1.0000	0.5263	0.0000	0.2895	1.0000	0.4079	0.2895	0.3947	0.3816	1.0000	0.2368	0.5395	1.0000	0.2533	0.5000	0.3026
5	Hindi	0.4805	0.4156	1.0000	0.5714	0.2895	0.0000	1.0000	0.4935	0.4026	0.3117	0.4026	1.0000	0.1299	0.4675	1.0000	0.2632	0.5325	0.0260
6	Kashmiri	1.0000	1.0000	0.8182	1.0000	1.0000	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	0.9481	1.0000	1.0000	0.8182	1.0000	1.0000	1.0000
7	Konkani	0.5195	0.4675	1.0000	0.5844	0.4079	0.4935	1.0000	0.0000	0.3896	0.5065	0.5325	1.0000	0.4416	0.6234	1.0000	0.4342	0.5584	0.4935
8	Marathi	0.5455	0.4675	1.0000	0.5714	0.2895	0.4026	1.0000	0.3896	0.0000	0.4805	0.5065	1.0000	0.3766	0.5714	1.0000	0.3816	0.5455	0.4156
9	Nepali	0.4416	0.4805	1.0000	0.5584	0.3947	0.3117	1.0000	0.5065	0.4805	0.0000	0.4935	1.0000	0.3117	0.4805	1.0000	0.4079	0.5455	0.3117
10	Oriya	0.3766	0.2987	1.0000	0.5325	0.3816	0.4026	1.0000	0.5325	0.5065	0.4935	0.0000	1.0000	0.4026	0.5455	1.0000	0.4474	0.5584	0.4026
11	Pali	1.0000	1.0000	0.8831	1.0000	1.0000	1.0000	0.9481	1.0000	1.0000	1.0000	1.0000	0.0000	1.0000	1.0000	0.8052	1.0000	1.0000	1.0000
12	Punjabi	0.4416	0.3896	1.0000	0.5325	0.2368	0.1299	1.0000	0.4416	0.3766	0.3117	0.4026	1.0000	0.0000	0.4675	1.0000	0.2105	0.5065	0.1169
13	Romani	0.5325	0.5455	1.0000	0.5844	0.5395	0.4675	1.0000	0.6234	0.5714	0.4805	0.5455	1.0000	0.4675	0.0000	1.0000	0.5000	0.5844	0.4675
14	Sanskrit	1.0000	1.0000	0.7403	1.0000	1.0000	1.0000	0.8182	1.0000	1.0000	1.0000	1.0000	0.8052	1.0000	1.0000	0.0000	1.0000	1.0000	1.0000
15	Sindhi	0.4737	0.4342	1.0000	0.5395	0.2533	0.2632	1.0000	0.4342	0.3816	0.4079	0.4474	1.0000	0.2105	0.5000	1.0000	0.0000	0.5132	0.2763
16	Sinhalese	0.5325	0.5714	1.0000	0.3766	0.5000	0.5325	1.0000	0.5584	0.5455	0.5455	0.5584	1.0000	0.5065	0.5844	1.0000	0.5132	0.0000	0.5325
17	Urdu	0.4805	0.4156	1.0000	0.5714	0.3026	0.0260	1.0000	0.4935	0.4156	0.3117	0.4026	1.0000	0.1169	0.4675	1.0000	0.2763	0.5325	0.0000

Table 5:Distance between languages

These distances were calculated to see how closely a language is related to another language. We can see from the figure that Pali is the most distant language from all the other languages. The languages Assamese, Bengali and Oriya are very closely related to each other, as the distance between them is very less. The distance between Hindi and Urdu is 0.260 which is justifiable as both are very similar languages and have almost every word and phonetic characters similar. Urdu is also closely related with Sindhi language. In the distance matrix the language ‘Nepali’ is very close to ‘Punjabi’. But these languages are from different geographical regions and the Nepal region is very close to Gorkhaland territorial region in West Bengal region geographically where the languages spoken are Assamese and Bengali. This relation is very interesting to note.

Further, we will now use these cognate judgements to reconstruct a phylogenetic of Indic Languages.

Reconstruction of Phylogenetic Tree

To reconstruct the phylogenetic tree, Bayesian Inference using Markov Chain Monte Carlo method is used. The methods in Bayesian inference in historical linguistics are based on the below formula,

$$f(\Lambda|X) = \frac{f(X|\Lambda)f(\Lambda)}{f(X)}$$

To utilize this formula for reconstruction of tree, we can use the below formula,

$$\Pr(Tree|Data) = \frac{\Pr(Data|Tree) \times P(Tree)}{\Pr\,(Data)}$$

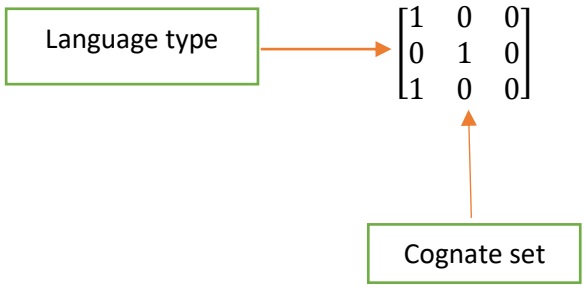
Where,

f = probability density function

Λ= consists of the tree topology τ, branch length vector T and substitution model parameter θ

X = Data Matrix

In the first step for reconstructing Phylogenetic tree, the Data Matrix X is created. The matrix created is of N*K dimension, in which there are N (207) different words from 18 Indic languages and they can be put into K clusters in a language family. As we need a binary data matrix, we construct the data matrix with elements i_{ij}. If the element is 1 it states language i is could be classified into cognate cluster j, and if the element is 0, it indicates that language i cannot be classified into cognate cluster j. This data matrix is crated using python library and converted into a nexus file. The example of such Data matrix is given below.



As the Automatic Cognate detection gave best results for LexStat Infomap and PMI, we will use these data of cognates to reconstruct the phylogenetic tree. The output of both the methods, in transformed into a matrix with the help of Python and stored in nexus file type. This file contains matrix of language and cognates as discussed in above section. For reconstructing phylogenetic tree, Markov Chain Monet Carlo Method is utilized. The procedure of MCMC is described as follows.

1. In the first Step a Random Tree A is selected
2. Then we use Bayes rule to evaluate the Probability of A, P(A)
3. In this step, Tree A is modified in a small amount randomly and second tree B is created
4. Then again Bayes rule is used to calculate the probability of B, P(B)
5. If P(B) > P(A), which suggests that the new tree P(B) is more likely to represent data than tree A, then step 1 is repeated. Here instead of starting with a random tree, the algorithm uses tree B

   If P(A) > P(B), then algorithm chooses any two options between starting with tree B or tree A and select to start with tree A randomly which has the probability of 1-P(B).

The likelihood that the created trees accurately represent the data increases when the process is done numerous times. In order to prevent getting stuck in local maxima and missing a region of the search space with even higher probability, the final step gives these algorithms a way to transition from a higher-probability tree to a lower-probability tree.

To implement reconstruction of Phylogenetic tree, MRBAYES software package is utilized. This package can be downloaded from [21]. The MRBAYES package accepts the nexus we created as an input the run the MCMC analysis. To start the analysis, we need to first create a data encoding file in nexus format as discussed above.

The issue with MYBAYES package us that additional than leaves, no other tree node can be directly specified. As a result, it is difficult to directly declare retentions as the root node of the tree. Therefore, to overcome this, we will set ‘Sanskrit’ language as the outgroup as it is related to every other language in Indo-Aryan language family. The next step of the algorithm is to reduce the distance between the outgroup language and the root node language.

The Input file for MRBAYES has 18 languages with the data encoding. The MRBAYES then takes the file as input, add few users selected configuration options, and begins the process of Bayesian Markov Chain Monte Carlo. In this research, the configuration options used as as follows

1. Generations = one million
2. Chains = 6 (Markov chains for initial random trees)

3. Sampling frequency and burn-in were set to default
4. Outgroup was set as Sanskrit

After putting these settings, the package starts running the analysis. The below code is used to run the analysis.

MrBayes > execute filename.nex

MrBayes > lset nst=6 rates=invgamma

MrBayes > outgroup Sanskrit

MrBayes >mcmc nchains=6

For running the analysis for one million generations it takes 10-12 mins, after which the trees are generated and summarized. The trees are also downloaded automatically in a tree file after the analysis is completed. To summarize the tree and the standard deviations results in the software, the below code is utilized.

MrBayes > sumt burnin 250.

Results for Phylogenetic reconstruction

The results obtained for both the models Pointwise Mutual Information and LexStat-Infomap are given below

Results of MCMC Analysis with input LexStat-Infomap

After running the MCMC analysis with 2 chain swaps, the histogram of the standard deviations of probability distribution is given below. The tracer plots and histograms generated for this run are given in appendix. The tree obtained by MCMC analysis took 12 mins to generate one million generations. The burnin period used in the analysis is 250 and the temperature is varied with values – 1.0, 0.91, 0.83, 0.77, 0.71 and 0.67 for 6 chains. Here,

$$T = \frac{1}{(1 + T * (ID - 1))}$$

T = 0.10

ID = chain number

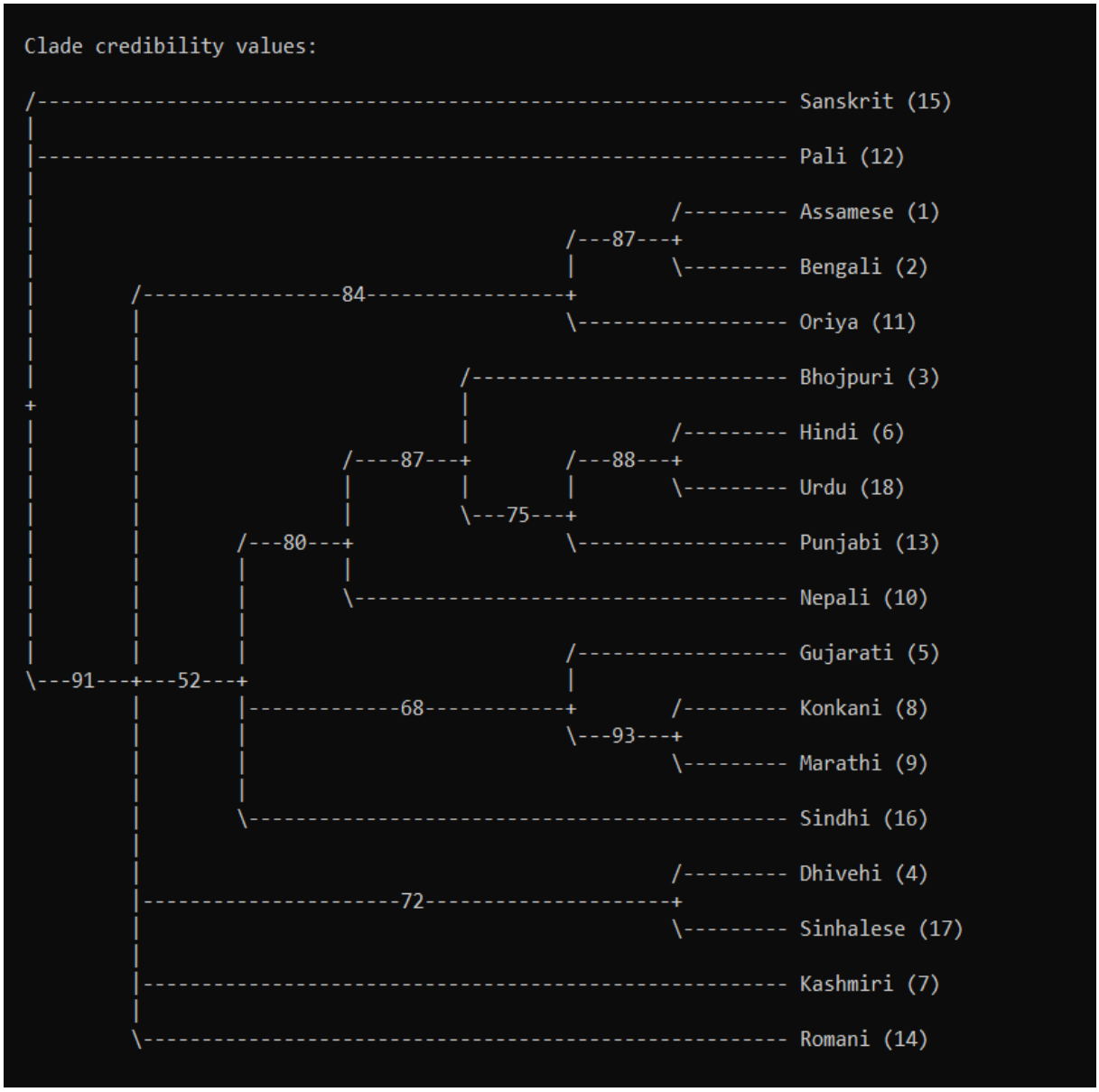


Figure 22:Phylogenetic tree using MCMC and input LexStat-Infomap

The phylogenetic tree generated by MCMC analysis with input of LexStat-Infomap was generated by specifying outgroup as Sanskrit. Without specifying the retentions, the analysis of MCMC did not give the best results. It can be seen from the tree that our analysis in cognate detection was very close to the phylogeny method. The branches of the trees for groups Marathi, Konkani and Gujarati

are correctly analyzed. Similarly other groups like Hindi, Urdu, Punjabi, Bhojpuri are classified almost accurately, The Language Sindhi shows some resemblance to Urdu in our cognate detection, but our phylogeny analysis states it differently. Languages in the group Assamese, Bengali and Oriya are also correctly presented. The dated years shows close resemblance to the phylogenetic tree generated by historical linguistics.

Results of MCMC Analysis with input PMI approach

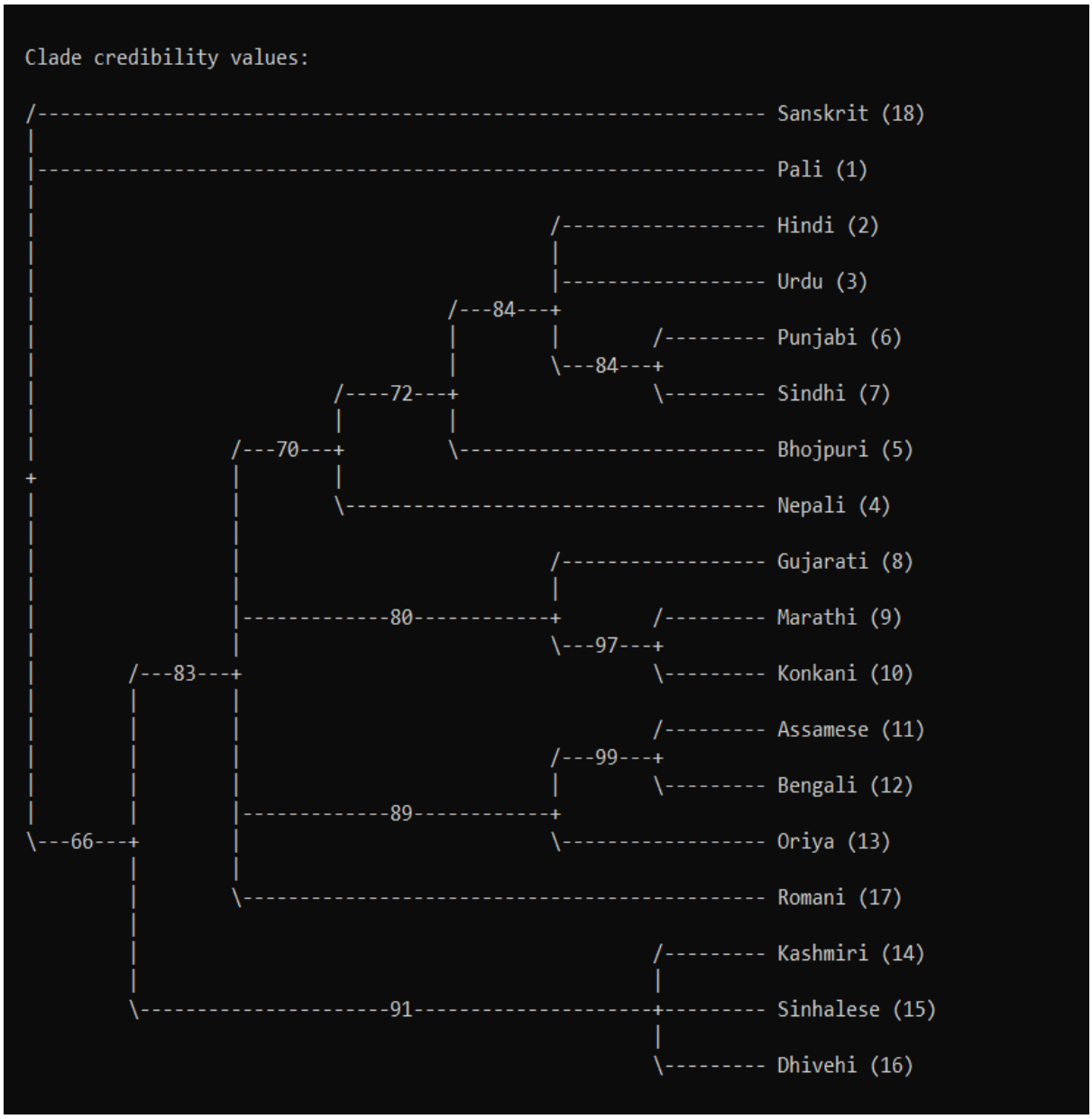


Figure 23:Phylogenetic using MCMC with input PMI

The tree obtained by MCMC analysis took 10 minutes to generate one million generations. All the settings like burnin, outgroup, etc are kept exactly similar to the previous method .The tracer plots and histograms generated for this run are given in appendix.

The phylogenetic tree obtained by the input of PMI method and MCMC analysis looks more detained and accurate than the tree obtained by LexStat-Infomap and MCMC. In this tree firstly, the languages, Urdu, Hindi, Sindhi, Punjabi are in the same branch which is the reality. In this tree the languages Romani, Kashmiri and Sindhi are branched together, and the origin is shown to a mix of Pali and Sanskrit.

Comparison of gold standard Phylogenetic with the trees obtained by Machine Learning

The trees generated by MCMC analysis have a very close resemblance to the trees generated by the historical linguistics. Although, there are few differences which we will discuss below, the overall analysis of phylogenetic tree states that MCMC algorithm performed well and the output is very close to the gold standard cognate tree.

Branches: The gold standard Phylogenetic tree has 15 branches including Sanskrit. These branches contain Languages: Kashmiri, Romani, Singhalese, Gujarati, Marathi, Urdu, Lahnda, Punjabi, Hindi, Bihari, Bengali, Oriya, Assamese, Nepali and Sanskrit. Whereas, our analysis contains 18 branches, the extra languages that we analyzed are Pali, Dhivehi, Bhojpuri and Konkani. The author [20] did not include these languages as few of the languages were not securely datable or these languages contained too many lacunae. As the tree generated by PMI approach is more close to the gold standard tree we will compare these to trees in detail.

Branch Groups and leaves:

Gold Standard Phylogenetic tree

Group 1: In the gold standard tree, the group1 branch has 3 leaves: Kashmiri, Romani, Singhalese.

Group 2: The group 2 has two leaves: Gujarati and Marathi

Group 3: The group 3 has Urdu as its main branch followed by rooted nodes and leaves of: Lahdna, Punjabi, Hindi and Bihari

Group 4: The branches of group 4 are: Bengali, Oriya, Assamese and Nepali

Branches of tree Generated by PMI approach:

Here Sanskrit is kept as the outgroup,

Group 1: Pali

Group 2: Branch contain Hindi, Urdu and further a branch of Punjabi, Sindhi and Bhojpuri.

Group 3: Branch contains Gujarati, Marathi, Konkani.

Group 4: Assamese, Bengali and Oriya and Romani

Group 5: Kashmiri, Sinhalese and Dhivehi

As we calculated above, the gold standard tree has 4 main branches followed by sub-branches, whereas the MCMC PMI tree has 5 main Branches followed by Sub-branches. If we consider the time depth of Gujarati and Marathi in both trees, the gold Standard tree has a time depth of 68 years, while the MCMC PMI tree has time depth of 80 years. Let's take another example of Oriya and Assamese, the time depth in gold standard tree is 95 years and the time depth in MCMC PMI approach is, 99 years, which is very close to the gold standard.

Similar Analysis can be performed between the time-Depth of gold Standard tree and MCMC LexStat-Infomap. The Gujarati Marathi in MCMC LexStat-Infomap is 68 years, which is exactly similar to the gold standard. The time-depth for Oriya and Assamese is 87 years, which is again close to gold standard.

The Glottochronology calculates the time depth by keeping the rate constant which is not accepted as the best approach by many researchers. In MCMC analysis, the value of rates is varied by applying smoothing techniques, and therefore it generates a more efficient as well as close to the gold standard tree.

The tree generated by MCMC in both the approaches have 'Nepali' grouped with languages like Punjabi, Bhojpuri, where as in gold Standard tree the language 'Nepali' is grouped with Assamese, Bengali and Oriya. It made sense to group Nepali with these languages that times, as geographically, Nepal is closer to Bengali, Assamese and Oriya speaking areas in West Bengal, while Speakers of Punjabi live in Northern parts of India. Nepali developed proximity to Pahari languages spoken in Northern regions of India. This language seems to have undergone a change with time, by replacing old words with new. This origin of new and modern Nepali language is believed to be originated from Sinja valley in Jumla, located in Nepal. Overall, the Tress generated with MCMC are close enough with the gold standard trees.

Discussion

The field of linguistics is very vast and requires in-depth knowledge in linguistics. The reconstruction of phylogenetic tree majorly depends on the input of the cognate results. Therefore, a heavy focus was given to detect cognates by various methods and select the cognates which are more reliable. Throughout the experiments, it was observed that for the Lingpy based methods, the size of the data did not matter much, and the focus was only given to the phonetic of the words. Due to which the task of borrowings and loan detection was not as successful as it was expected. This negligence resulted in a low Precision, recall and F1-score. The solution to this problem could be to use a classifier and train some data to detect the cognates and loan words, before directly testing on the raw dataset.

For the task of Automatic cognate detection using PMI method, it was observed that the low minibatch size gives better results, as compared to a large minibatch size. It was also observed that the initial values of α were enough to produce better results. In NLP tasks, the value of m Is required to be large, but in our experiment, the smaller values of m gave better results.

Recent advances in Machine learning and Bayesian techniques, helped to tackle few problems faced by glottochronology. Firstly, the problem of information loss caused by transforming discrete characters into distances can be solved by analyzing the discrete characters themselves in order to determine the best tree. Secondly, Using Bayesian Markov chain Monte Carlo (MCMC) techniques, where the frequency distribution of the sample is close to the posterior probability distribution of the trees, it is possible to estimate uncertainty in the estimation of tree topology, branch lengths, and parameters of the evolutionary model. Therefore, in this research, using MCMC approach has given better results and a future scope of more improved work.

Conclusion

In this research, new and automatic methods for working with linguistics are explored. The methods like Bayesian Inference, Markov Chain Monte Carlo, Pointwise Mutual Information were explored to Detect cognates of Indic languages and reconstruction of phylogenetic tree. The Automatic cognate detection methods identified most of cognates correctly as compared with the gold standard cognates, but these methods lack in detecting the loan words. The performance metrics and evaluation of these methods indicates that more work in this direction is needed as cognates detection is the core of Phylogenetic tree reconstruction. Bayesian Methods used with Markov Chain Monte Carlo can give state-of-the-art results, if used in a correct way. In our research, these methods have given us better results than expected. And the results can further be improved by using smoothing for varying the rates in MCMC analysis.

Future Scope

The cognate identification can be improved by using Convolutional Neural network, in which the dataset is trained against false cognates. This can improve the overall results of cognate detection as well as phylogenetic tree reconstruction.

References

[1] S. K. Morten H.Christiansen, "Language evolution: consensus and controversies," *Science Direct,* vol. 7, no. 7, pp. 300-307, July 2003.

[2] C. W. Bryant, "howstuffworks," April 2021. [Online]. Available: https://science.howstuffworks.com/life/evolution/language-evolve.htm.

[3] J.-M. List, "Sequence Comparison in Historical Linguistics," *ResearchGate,* 2014.

[4] S. C. Gudschinsky, "The ABC'S of Lexicostatistics (Glottochronology)," *WORD,* vol. 12.2, pp. 175-210, 1956.

[5] G. c. Dhanesh Jain, The Indo-Aryan languages, London ; New York : Routledge, 2007.

[6] Boundless, "Early Civilizations in Indian Subcontinent," in *World Civilization*.

[7] N. Grover, "Indo-Aryan Group of Languages - Art and Culture Notes," August 2022. [Online]. Available: https://prepp.in/news/e-492-indo-aryan-group-of-languages-art-and-culture-notes.

[8] D. B. A.D. Scott, "Phylogenetic Tree," *Science Direct,* 2008.

[9] M. Swadesh, "Salish Internal Relationships," *International Journal of American Linguistics,* vol. 16, pp. 157-167, 1950.

[10] R. B. Lees, "The Basis of Glottochronology," *Linguistic Society of America,* vol. 29, pp. 113-127, 1953.

[11] I. P. M. G.-M. Peter Turchin, "Analyzing genetic connections between languages," *Journal of Language Relationship,* vol. 5, pp. 117-126, 2010.

[12] W. H. Jhon Nerbonne, "Measuring Dialect Distnace Phonetically," *Computational Phonology: Third Meeting of the ACL Special Interest Group in Computational Phonology,* 1997.

[13] M. List, "Sequence Comparison in Historical Linguistics," *düsseldorf university press,* vol. 1, 2014.

[14] J. W. P. S. G. J. Taraka Rama, "Fast and unsupervised methods for multilingual cognate clustering," *arXiv,* 2017.

[15] J.-M. L. J. W. G. J. Taraka Rama, "Are Automatic Methods for Cognate Detection Good Enough for," *arXiv,* 2018.

[16] C. C. D. H. A. G. Will Chang, "ANCESTRY-CONSTRAINED PHYLOGENETIC ANALYSIS SUPPORTS THE," *JSTOR,* vol. 91, pp. 194-224, 2015.

[17] wikipedia, "wikipedia, The free encyclopedia," [Online]. Available: https://en.wikipedia.org/wiki/International_Phonetic_Alphabet.

[18] J.-M. List, "Sequence comparison in historical linguistics," *Düsseldorf University Press, Düsseldorf,* 2014b.

[19] L. Rita, "Towards Data Science," 12 April 2022. [Online]. Available: https://towardsdatascience.com/infomap-algorithm-9b68b7e8b86.

[20] A. I. Kogan, "Genealogical classification of New Indo-Aryan languages and lexicostatistics," *Journal of Language Relationship,* vol. 14, pp. 227-258, 2017.

[21] C. Z. J. N. M. S. D. L. A. K. Andreas Kusalananda Kähäri, 10 January 2020. [Online]. Available: https://nbisweden.github.io/MrBayes/download.html.