RNI: MAHENG/2016/68030 **VOLUME 9 (2024), ISSUE 3 : 33-47** **ISSN: 2456-5342 (print)**

MGMUNIVERSITY
JAWAHARLAL NEHRU
ENGINEERING COLLEGE

VISTA INTERNATIONAL JOURNAL ON ENERGY, ENVIRONMENT & ENGINEERING

VISTA

Usage of Computer Intelligence In Understanding the Natural Language Processing in Sanskrit Texts : Using Python-A Review

Preenon Bagchi and Shivani V.
Karnataka Samskrit University, Bengaluru, India

***Corresponding Author's E-mail:** prithish.bagchi@gmail.com **Mob.:** +91-7975292390

A B S T R A C T

Natural language processing (NLP) denotes the computer's ability to comprehend human language, encompassing spoken and written forms in languages such as English, Hindi, and Sanskrit. This falls under the domain of artificial intelligence (AI). Processing natural language with inputs and outputs in Sanskrit poses a considerable challenge for human-machine interaction. NLP, rooted in linguistics and with a history spanning over 50 years, finds applications in diverse real-world sectors such as medical research, search engines, and business intelligence.

But the sanskrit natural language ambiguity poses a major hurdle in the processing. This work works with Sanskrit sentences and Sanskrit words input to python modules. The relatively unambiguous nature of this machine language and its well laid-out grammatical structure is studied here and also this language is promoted as the language for processing.

Keywords : *NLP, NLTK, parsing, quantum, artificial intelligence-machine learning, sandhi*

1. Introduction

Artificial intelligence penetrates into a myriad number of fields, one of them being natural language processing. With man's continuous efforts to stay away from the jargon of machines,arose the need to feed natural languages as inputs to machines [1, 2]. All natural languages express a large amount of ambiguity. Though, the languages are correctly interpreted by humans out of usage, for a computer that lacks the capability to distinguish between the various interpretations on a contextual basis, this ambiguity proves to be a vice. Hence, the present day is in need of a language that could to a great extent eliminate this ambiguity and at the sametime be suitable for knowledge representation in artificially intelligent systems [1, 2].

Sanskrit which has been in prevalence since thousands of years and has not worn out and moulded in the course of human usage can be used for such knowledge representation [1, 2].

1.1 Natural Language Processing

NLP combines the field of linguistics and computer science to decipher language structure and guidelines and to make models which can comprehend, break down and separate significant details from text and speech [3, 4].

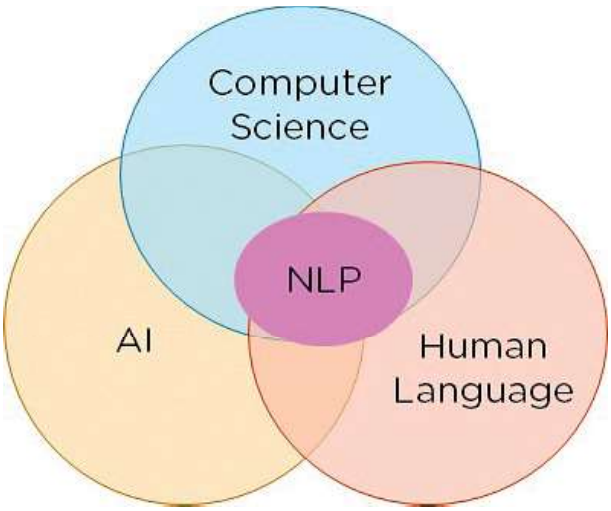

Fig 1: Constituents of NLP

The steps to perform pr-eprocessing of data in NLP include:

1.2 Segmentation : It is needed to first break the entire document down into constituent sentences. This can be done by segmenting the article along with its punctuations like full stops and commas[3, 4].

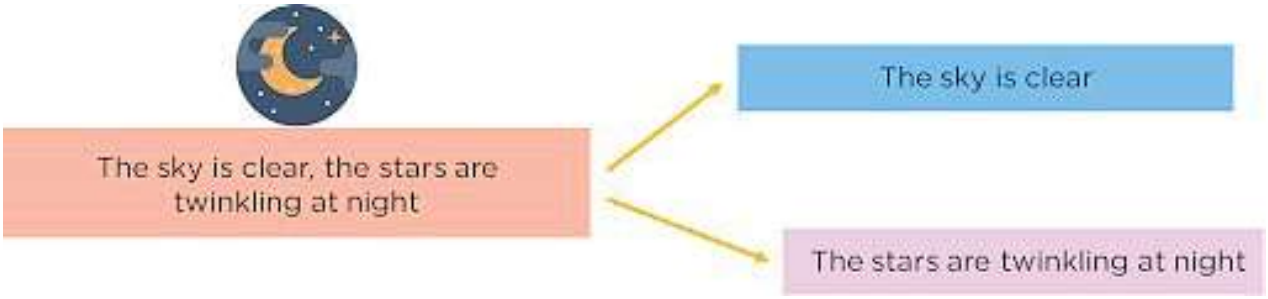

Fig. 2: Segmentation

1.3 Tokenizing : For the algorithm to understand these sentences, it is important to get the words in a sentence and explain them individually to algorithm used. Hence it is essential to break down the sentence into its constituent words and store them. This is called tokenizing, and each world is called a token [3, 4].

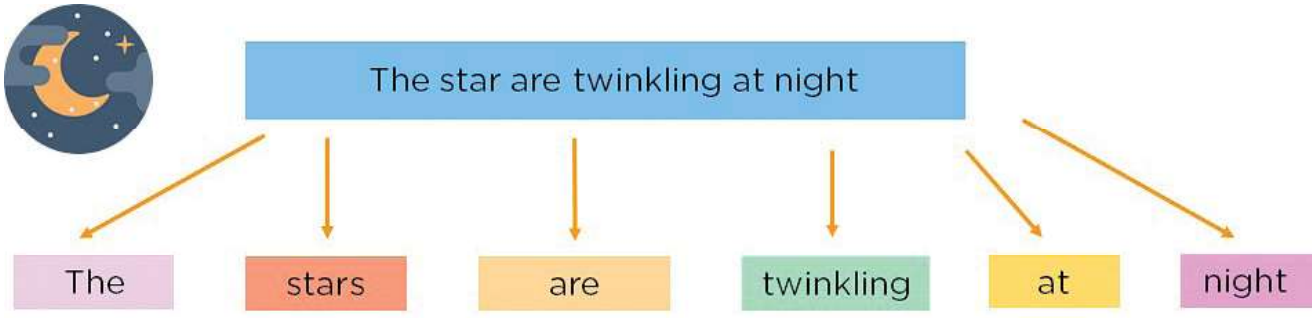

Fig. 3: Tokenization

1.4 Removing Stop Words : It is essential tomake the learning process faster by getting rid of non-essential words, which add little meaning to our statement and are just there to make our statement sound more cohesive. Words such as was, in, is, and, the, are called stop words and should be removed [3, 4].

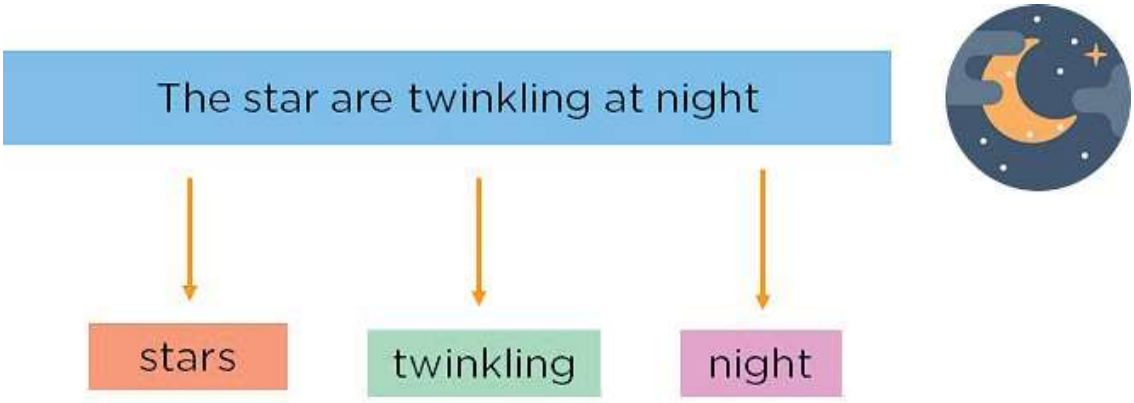

Fig. 4: Stop Words

1.5 Stemming : It is the process of obtaining the Word Stem of a word which gives new words upon adding affixes to them [3, 4].

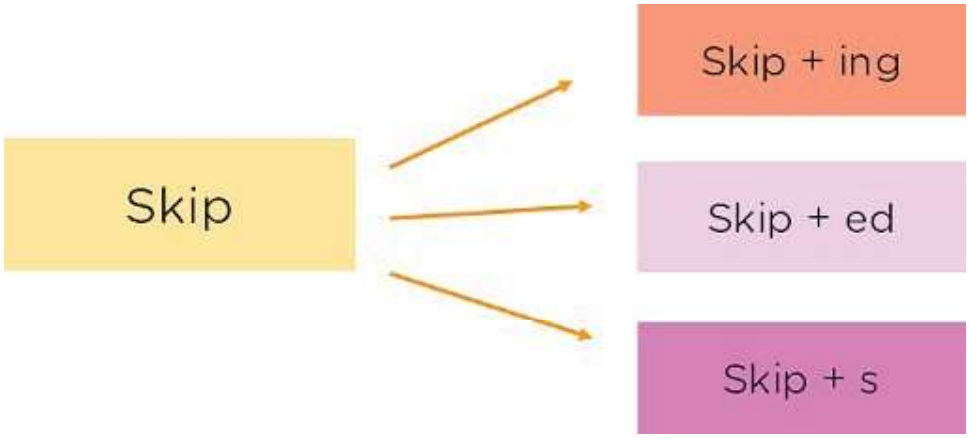

Fig. 5: Stemming

1.6 Lemmatization : The process of obtaining the Root Stem of a word is called lemmatization. Root Stem gives the new base form of a word that is present in the dictionary and from which the word is derived which helps us to identify the base words for different words based on the tense, mood, gender, etc [3, 4].

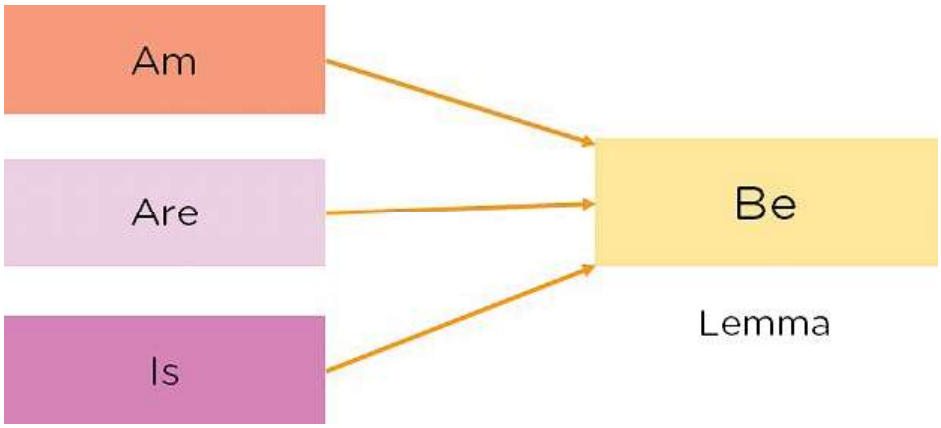

Fig. 6: Lemmatization

1.7 Part of Speech Tagging : Then we need to explain the concept of nouns (संज्ञा), verbs (क्रिया), articles, and other parts of speech to the machine by adding these tags to our words. This is called 'part of' [3, 4].

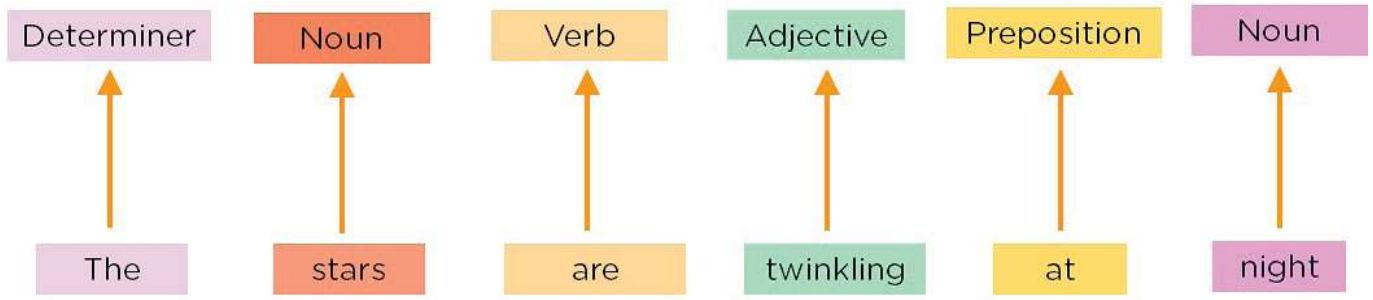

Fig. 7: Part of Speech Tagging

1.8 Named Entity Tagging : Finally we introduce our machine to pop culture references and everyday names by flagging names of movies, important personalities or locations, etc that may occur in the document. We can do this by classifying the words into subcategories. This helps you find any keywords in a sentence. The subcategories are person, manuscripts, value, quantity, organization, theatre [3, 4].

Next, after performing the preprocessing steps, we then give our resultant data to a machine learning algorithm like Naive Bayes, etc., to create your NLP application [3, 4].

NLP embedded computer programs understand natural language as humans do either the language is spoken or written, natural language processing uses artificial intelligence to take real-world language input, process it, and make sense of it in a way a computer can understand. Just as humans have different sensors ---- computers have NLP programs to read and microphones to collect audio. And just as humans have a brain to process, computers have a NLP program to process their respective inputs. At some point in processing, the input is converted to code that the computer can read, process understand [3, 4].

2. Two main phases to natural language processing :

Once the data has been preprocessed, algorithm are used to process them. Two main NLP algorithms are:

2.1 Rules-based system : This is based on linguistic approach. This approach is used early on in the development of natural language processing, and is still widely used.

2.2 Machine learning (ML)-based system : ML algorithms use statistical methods which perform tasks based on training data they are fed, and adjust their methods as more data is processed. Using a combination of ML, deep learning (DL) and neural networks (NN), NLP algorithms make their own rules through repeated processing and learning [3, 4].

3. NLP uses diagram

These are some of the key areas in which a business can use natural language processing (NLP).

These are the types of vague elements that frequently appear in human language and that machine learning algorithms have historically been bad at interpreting. Now, with improvements in deep learning and machine learning methods, algorithms can effectively interpret them. These improvements expand the breadth and depth of data that can be analysed [3, 4].

Semantics involves the use of and meaning behind words. Natural language processing applies algorithms to understand the meaning and structure of sentences. Semantics techniques include:

Word sense disambiguation. This derives the meaning of a word based on context. Example: Consider the sentence, “The pig is in the pen.” The word pen has different meanings. An algorithm using this method can understand that the use of the word pen here refers to a fenced-in area, not a writing implement [3, 4, 5].

Named entity recognition. This determines words that can be categorized into groups. Example: An algorithm using this method could analyze a news article and identify all mentions of a certain company or product. Using the semantics of the text, it would be able to differentiate between entities that are visually the same. For instance, in the sentence, "Lord Rama is the main figure of the epic Rama-yan," the algorithm could recognize the two instances of "Rama's" as two separate entities one a person and one a book [3, 4, 5].

3.1 Natural language generation : This uses a database to determine semantics behind words and generate new text. Example: An algorithm could automatically write a summary of findings from a business intelligence platform, mapping certain words and phrases to features of the data in the BI platform. Another example would be automatically generating news articles or tweets based on a certain body of text used for training [5].

Current approaches to natural language processing are based on deep learning, a type of AI that examines and uses patterns in data to improve a program's understanding. Deep learning models require massive amounts of labeled data for the natural language processing algorithm to train on and identify relevant correlations, and assembling this kind of big data set is one of the main hurdles to natural language processing [4, 5].

Earlier approaches to natural language processing involved a more rules-based approach, where simpler machine learning algorithms were told what words and phrases to look for in text and given specific responses when those phrases appeared. But deep learning is a more flexible, intuitive approach in which algorithms learn to identify speakers' intent from many examples -- almost like how a child would learn human language [4, 5].

There are general five steps in NLP as follows :

3.1.1 Lexical Analysis – It involves identifying and analyzing the structure of words. Lexicon of a language means the collection of words and phrases in a language. Lexical analysis is dividing the whole chunk of txt into paragraphs, sentences, and words [3, 4, 5].

3.1.2 Syntactic Analysis (Parsing) – It involves analysis of words in the sentence for grammar and arranging words in a manner that shows the relationship among the words. The sentence such as "The school goes to boy" is rejected by English syntactic analyzer [3, 4, 5].

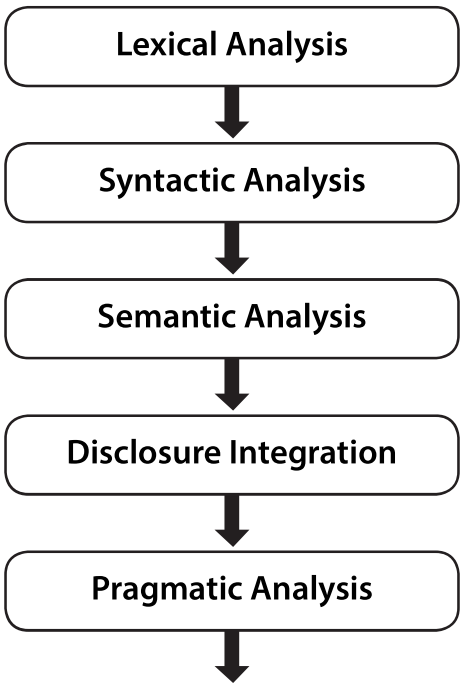

Fig. 8: NLP steps

3.1.3 Semantic Analysis − It draws the exact meaning or the dictionary meaning from the text. The text is checked for meaningfulness. It is done by mapping syntactic structures and objects in the task domain. The semantic analyzer disregards sentence such as "hot ice-cream" [3, 4, 5].

3.1.4 Discourse Integration − The meaning of any sentence depends upon the meaning of the sentence just before it. In addition, it also brings about the meaning of immediately succeeding sentence [3, 4, 5].

3.1.5 Pragmatic Analysis − During this, what was said is re-interpreted on what it actually meant. It involves deriving those aspects of language which require real world knowledge [3, 4, 5].

3.2 Implementation Aspects of Syntactic Analysis

It is the grammar that consists rules with a single symbol on the left-hand side of the rewrite rules. Let us create grammar to parse a sentence −

"The bird pecks the grains"

Articles (DET) − a | an | the

Nouns − bird | birds | grain | grains

Noun Phrase (NP) − Article + Noun | Article + Adjective + Noun

= DET N | DET ADJ N

Verbs − pecks | pecking | pecked

Verb Phrase (VP) − NP V | V NP

Adjectives (ADJ) − beautiful | small | chirping

The parse tree breaks down the sentence into structured parts so that the computer can easily understand and process it. In order for the parsing algorithm to construct this parse tree, a set of rewrite rules, which describe what tree structures are legal, need to be constructed [3, 4, 5].

These rules say that a certain symbol may be expanded in the tree by a sequence of other symbols. According to first order logic rule, if there are two strings Noun Phrase (NP) and Verb Phrase (VP), then the string combined by NP followed by VP is a sentence. The rewrite rules for the sentence are as follows −

S → NP VP

NP → DET N | DET ADJ N

VP → V NP

Lexocon −

DET → a | the

ADJ → beautiful | perching

N → bird | birds | grain | grains

V → peck | pecks | pecking

The parse tree can be created as shown −

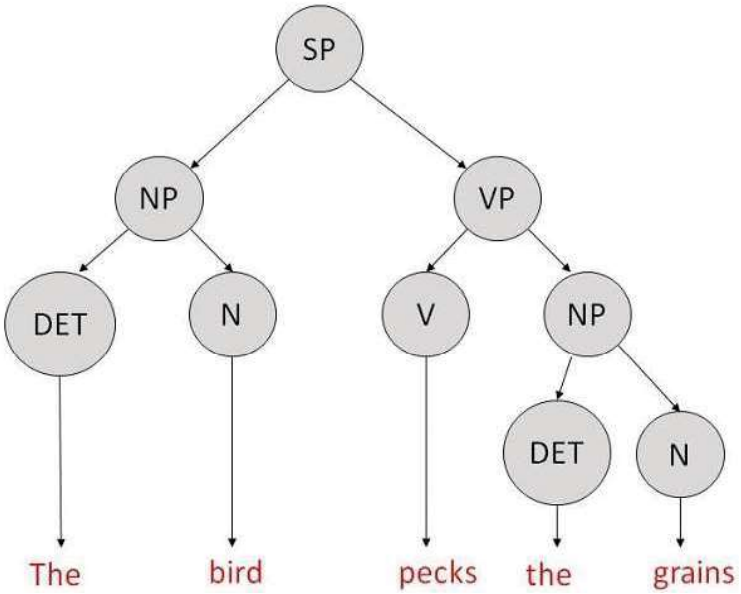

Fig. 9: The parse tree

Now consider the above rewrite rules. Since V can be replaced by both, “peck” or “pecks”, sentences such as “The bird peck the grains” can be wrongly permitted. i. e. the subject-verb agreement error is approved as correct [3, 4, 5].

3.3 Python for NLP

Three tools used commonly for natural language processing include Natural Language Toolkit (NLTK), Gensim and Intel natural language processing Architect. NLTK is an open source Python module with data sets and tutorials. Gensim is a Python library for topic modeling and document indexing. Intel NLP Architect is another Python library for deep learning topologies and techniques [4].

3.4 Mapping Sanskrit as Quantum Language [5]

Quantum Computing needs a cross platform language. Oldest language of the world Sanskrit has attributes that fit into that requirement, since it’s scripts resemble the Circuits and gates. Sutras are strands of Circuits,where the logic gates (Sutra Script) connects [5].

Triangle of Quantum computing in Indian science:

1. Yantra: Quantum Hardware.
2. Mantra: Quantum Operating system.
3. Tantra:Quantum Algorithm.

Layered architecture of Quantum hardware, is similar to the 7 layer chakra system of Yantra.It has hidden geometry and numbers.

3.4.1 Findings: Sanskrit language is seen to have a strong influence on astronomy, in large object and planetary movement prediction, which is actually meant for atomic movement and spins.

Atomic Systems -ANUSHASANAM: Anu means atom and shasanum means system in Sanskrit. The Yoga Sutras of Patanjali describes it in systematic way.

Quantum Machine (Yantra) as seen in Borobudur:

Quantum sensors

9th-century Borobudur architecture resonate the quantum computing with 8 layers of gates.This bottom up design can work at room temperature and it can be used for Autonomous Vehicle and Quantum mobile phones. The Top down design of helium based quantum computer works in sub zero condition .

Spiritual Engineering: Borobudur top circles/diamond shape walls creates unique reflection during dawn sunlight, this can be replicated using crystal Prisms gates for light dispersion and studied

Zen sand Mandala design are used colored crystals to design ,similar to the photonic crystals used in quantum research.

3.4.2 Maya : Every quantum particles appears and dissolves in cyclic way,in every moment of time, creating illusion of materiel forms as seen Anithya in Zen.

The wave of raising and falling of energy creates this forms by pull craving of electrons (Thanha) and push-repulsion(Dosha) creates this illusion (moha).

3.4.3 AUM : symbal has 4 curves, representing,wave like mirage world.The Turya is the dot representing particle state and reality.

3.4.4 Parabhrama : Still, unwavering energy which is unified all energy states negative positive -without any character

3.4.5 Akshara : Quantum software describing quantum state.

Quantum Travel: Higher dimensions are in quantum single dimension.Dharma is the scientific principles which hold Quantum higher dimension together with heavier lower dimension.

Quantum Entanglement — Quantum Yoga:

Yoga from the Sanskrit root "yuj," meaning "to yoke". This is similar to two bulls are connected through a Yolk beam, to give an output and similarly 2 entangled atoms can give output in quantum. That can be called (Quantum level) yoga in Sanskrit.

Bio Quantum Computer:Beyond Neural network, much finer network is bio quantum. Zen masters are the Quantum scientists.

3.4.6 Samadhi : Reaching the initial state of universe is the quantum state

By reducing noise of 5 sensory data of body, by stilling the body and mind, our body reaches the inner core quantum computer that is buzzing silently.

It is similar to the sub-zero, noise reducing in Quantum machines environment now.

Quantum Optimization: cost function: lotus has the Math function like cost function graphically represented. Each symbol has some math formula embedded behind the scene. Lotus represents purity which means optimized.

Bio Quantum Instinct: Mindfulness gives output as insight, a gut feeling, thus is a quantum result.

Quantum Music: The Institute for Quantum Computing and the Kitchener-Waterloo Symphony teamed up to create "Quantum: Music.Each listener of music have different experience, based on their different awareness. Similarly, the Sanskrit hymns has rhythmic lyrics which has layer of meanings.Each gets different meaning based on their understanding and awareness. Hence the verses came from quantum level.

4. Bio Quantum Maintenance : Ayurveda

This balances 5 basic elements to correct body blockage and errors. Ayur in sanskrit means longevity. This tells how to preserve and extend the quantum state [5].

Aṣṭādhyāyī [6-11]

The study of Aṣṭādhyāyī can be classified into three broad areas of academic research:

- Analysis of the grammatical corpus in order to understand its organization and functioning,
- formalization of the grammatical system, and
- its computer implementation or automation.

We are living in an era of rapidly changing technology, virtualisation, SDN/NFV, advent of 4G LTE technology and in the near future, 5G is redefining businesses across the globe. The digitalisation wave has brought artificial intelligence and machine learning, and with it a wave of organizations trying to understand natural language and intent of customers or visual clues using deep neural network techniques. We are still far away from understanding "understanding" via machines but the steps towards those have started [6-11].

While these steps have moved the needle towards realization, a large population in India seems to be unaffected by this change. The "Siris", "Cortanas" and the "Alexas" of the world have captured a global market by their limited natural language capabilities and building huge datasets of languages in their cloud, however about 85% of Indian population seems unaffected by this change. The reason is the language used viz. "English" [6-11].

While we move towards the mid of 21st century, it is imperative that about 1/8th of the world population can converse with machines the way the rest of the world does, in its own dialect and in its own way, and this is the reason for me and my team in Makers Lab (R&D incubator of Tech Mahindra) researching upon one of the first communication languages spoken by man, "Sanskrit" [6-11].

The objective of this work is to showcase how communications with machines can improve if alternative techniques and idioms are utilized [6-11].

The objective is also to look through the lens of AI (artificial intelligence) and algorithms and see which one of them apply in the world of Sanskrit for NLP(natural language processing) [6-11].

By definition Natural language processing (NLP) is used for communication between computers and human (natural) languages in the field of artificial intelligence, and linguistics. Being concerned with human-computer interaction, NLP works to enable computers to make sense of human language to make interactions with machinery and humans as user friendly as possible [6-11].

Sanskrit is a language of ancient India with a history going back about 3,500 years. In the early 1st millennium CE, along with Buddhism and Hinduism, Sanskrit migrated to Southeast Asia, parts of East Asia and Central Asia, emerging as a language of high culture and of local ruling elites in these regions.

Sanskrit is an Old Indo-Aryan language. As one of the oldest documented members of the Indo-European family of languages, Sanskrit holds a prominent position in Indo-European studies. It is related to Greek and Latin, as well as Hittite, Luwian, Old Avestan, and many other extinct languages with historical significance to Europe, West Asia, and Central Asia. It traces its linguistic ancestry to the Proto-Indo-Aryan language, Proto-Indo-Iranian, and the Proto-Indo-European languages [6-11].

What makes Sanskrit unique is the rule set that it formulates and the grammar that was formulated much before the language became widely accepted and spoken in the Indian sub-continent. While most languages we speak are “natural”, Sanskrit by definition is a synthetic language [6-11].

Since this is a comparison between two languages it seems logical to start with how languages were earlier used to speak and communicate rather than write, and so let us start with some phonetics discussion. [6-11]

Phonetics

The phonetic sounds of alphabets in English and their counterparts in Sanskrit Varna-mala is different. Varna-mala is the Sanskrit corpus of alphabets. In English the sounds of the alphabets clash with their counterparts in quite a few occasions. Let us take the alphabets of English and the Varna-mala in Sanskrit for example [6-11].

In Sanskrit, the alphabets are called Varna-mala .Every word in Sanskrit is formed because of the combination of two elements Swar (स्वर) and a Vyanjan (व्यंजन). In Sanskrit, there are 13 Swaras, 33 Vyanjanas and about 2 Swarakrashits {Special words} . All in all Sanskrit has a total of 49 Varnas of the Varna-mala. By its definition itself, Sanskrit has more alphabets, characters and building blocks than any other language [6-11].

Out of these Swaras 5 are pure:

Remaining 9 are: आ, , ई, ऊ,, ऋ,, ल, ए, ऐ, ओ, , औ

Well for any observer, the difference is quite apparent. The way we these are arranged is primarily because how air can be modelled within the mouth itself. When you open a mouth and take a sound from the glottis, the word sound has (अ) there, whereas when the mouth opens up wider it is an (आ) sound. English or any other language in comparison has only an (A) equivalent to ए, which completely misses the way the glottis performs when open the mouth wide [6-11].

From just observing the tables above, one this which is visible is that number of alphabets do not compare to the number of Varnas in Sanskrit. In fact they are much lesser in number, but on close examination, something interesting appears. The range of English vocabulary also becomes lesser because the phonetics of a lot of alphabets do not map to individual phonetics of the Sanskrit Varnas A map is shown below purely how phonetics is used [6-11].

The present work deals with the latter two areas, namely, formalization andcomputer implementation of Aṣṭādhyāyī. It seeks to study the content andprocesses of the Pāṇinian system of Sanskrit grammar and represent them interms of logical relations and operations. A formal representation is attemptedin order to facilitate an examination of the underlying grammatical structures [6-11].

It also enables an implementation of the grammatical processes on computer.

Methodology with Results

Sentences

Sentences consist of one or more language-components. They represent thewhole unit of a typical linguistic expression with the possibility of a numberof inflected words. This class is necessary since the rules of grammar considerthe whole sentence and not just one word to be the unit of derivation [6-11].

From the point of view of a formal representation of grammatical processes,sentences can simply be defined as a sequence of language-components. Accordingly,the class of Sentences is implemented as a list of LangComps [6-11].

```
1 class Sentences:
2 def __init__(self ,list_of_LangComps =[ LangComps ()]):
3 self.sentence = list_of_LangComps
```

The present formal framework uses three levels to represent any linguisticexpression. Sentences correspond to the whole unit of a particular linguisticexpression, while sound-sets correspond to the individual sounds. Languagecomponentsare an intermediate level between the two and are tentatively relatedto an inflected word. Depending upon the level from which conditionalinformation can be gathered in a sufficient manner, the grammatical operationscan be distinguished as those that apply to a sound-set or to a languagecomponentor at the level of the whole sentence [6-11].

The process of derivation is carried out when an operational statement is appliedto a sentence or to its constituents, i.e. the language-components or thesound-sets. Together with an operational statement, a sentence forms the nextdata-structure of the system, namely the class DStates [6-11].

```
1 class DStates: # Derivational state
2 def __init__(self, (sentence, statement_string)=(None, None)):
3 self.dState = sentence
4 self.applied_statement_str = statement_string
```

The application of a particular statement brings about some change in thecurrent state of a sentence. This change may be at the level of a sound-set if, for example, it gets a new attribute, or at the level of a language-component, for example, addition of a new sound-set, or even at the level of a sentenceitself, in the case of addition of new language-components. The dState variableof the class saves the changed state of the sentence after the applicationof some statement. The operational statement which is applied is stored in thevariable applied_statement_str [6-11].

The current state of a sentence saved in a derivational state is the result ofapplication of an operational statement on the previous state of that sentence.The sequence of such derivational states is stored in a slice and is implementedby the class Slices. From the point of view of data-structures, a slice is simplya sequence or list of derivational states or DStates [6-11].

```
1 class Slices:
2 def __init__(self ,list_of_DStates =[]):
3 self.slice = list_of_DStates
```

There are two kinds of changes that the operational rules of grammar bringabout: either the derivational state is saturated or it progresses towards completion. The process of saturation is associated with attachment of attributes, while addition of new components is related with incremental steps of completionof the derivational

process. Slices contain only those changes wherethe derivational process is saturated, i.e. only when attributes are added tothe components. The other case, when a new component is introduced, leads to the formation of a new slice [6-11].

5. Processes of grammar :

During the process of their derivation, linguistic expressions are represented through a sentence which consists of one or more language-components corresponding to the individual inflected words. Each language-component contains a sequence of sound-sets. Each sound-set corresponds to a singlephoneme [6-11].

The derivational process is effected through a number of operational statementswhich are applied to a sentence. A sentence, together with an operational statement, constitutes a derivational state. There are two fundamental types of operations:

(i) to saturate a sentence, in that all attributes that canbe attached are added to it, and

(ii) to add a new component and graduatetowards completion of the derivational process.

Accordingly, a slice contains a sequence of derivational states that arise during the process of saturation.A new slice is added, once a new component is introduced. A process-strip records a sequence of slices and thus registers the process of completion [6-11].

Given the above framework and corresponding data-structures, the general algorithm of the derivational process can be specified as follows.

```
1 initialize a process -strip
2 repeat the following steps:
3 saturate the process -strip
4 look for completing statements
5 if there is no statement to be applied
6 return the process -strip
7 else:
8 select a completing statement
9 apply it to the process -strip
```

After initialisation, the process-strip is populated with new components and saturated repeatedly, till there is no admissible component available. This brings the process of derivation to an end [6-11].

Program 1 [program courtesy 12-14]:

```
# हलन्त्यम्
धातुः= औधातुः= "डुकृञ्"
हल्_क= ("ह", "य", "व", "र", "ल", "ञ", "म", "ङ", "ण", "न", "झ", "भ", "घ", "ध","ज","ब","ग",
"ड","द","ख","फ","छ","ठ","थ","च","ट","त","क","प","श","ष","स")
हल्_ख= tuple([व+ "क्"[-1] for वin हल्_क])
if औधातुः[-1] == "क्"[-1]:
    धातुः= औधातुः[:-1]
    # this condition is not required
    if धातुः endswith(हल्_क):
        धातुः= धातुः[:-1]
print(धातुः)
```

The program takes हल्_क as the variable with datas as all the halanta letters. हल्_ख variable is assigned a tuple having the हल्_क letters with halanta. The program proceeds with the condition ifऔधातुः(with halanta) equals any halanta letters, it will print that particular halanta letter.

Program 2 [program courtesy 12-14]:

```
पदम्= "कोरोना"
अट्= ["अ", "आ", "का"[-1], "इ", "कि"[-1], "की"[-1], "ई", "उ", "कु"[-1], "कू"[-1], "कृ"[-1], "कॄ"[-1], "कॢ"[-1], "के"[-1], "कै"[-1], "को"[-1], "कौ"[-1], "ऊ" "ऋ", "ऌ", "ए", "ऐ", "औ", "ह", "य", "व"]
कुपु= ["क", "ख", "ग", "घ", "ङ", "प", "फ", "ब", "भ", "म"]
अनुमतम्= अट्+ कुपु

def णत्वपरीक्षा (पदम्):
    रेफस्थानम्= पदम्.index("र")
    नस्थानम्= पदम्.index("न")

    print("रेफस्थानम् =", रेफस्थानम्)
    print("नस्थानम् =", नस्थानम्)

    व्यवहिताः= पदम् [रेफस्थानम्+1:नस्थानम्]
    णत्वम्= False

    for वर्णःin व्यवहिताः
        if वर्णःin अनुमतम्:
            णत्वम्= True
        else:
            णत्वम्= False
            break

    if णत्वम्== True:
        पदम्= पदम् [:नस्थानम्] + "ण" + पदम् [नस्थानम्+1:]
    return पदम्

try:
    पदम्= णत्वपरीक्षा (पदम्)
except:
    pass

print("पदम् =", पदम्)
```

This program takes input any पदम् and tries to identify that it is रेफस्थानम् or नस्थानम् and it ends with न or ण.

Program 3 [program courtesy 12-14]:

pip install inltk

pip install torch==1.3.1+cpu -f https://download.pytorch.org/whl/torch_stable.html

API

Setup the language

from inltk.inltk import setup

setup('<code-of-language>') // for hindi, then setup('hi')

Note: We need to run setup('<code-of-language>') when you use a language for the FIRST TIME ONLY. This will download all the necessary models required to do inference for that language.

Tokenize

from inltk.inltk import tokenize

tokenize(text ,'<code-of-language>') // where text is string in <code-of-language>

Get Embedding Vectors

This returns an array of "Embedding vectors", containing 400 Dimensional representation for every token in the text.

from inltk.inltk import get_embedding_vectors

vectors = get_embedding_vectors(text, '<code-of-language>') // where text is string in <code-of-language>

Example:

>> vectors = get_embedding_vectors('भारत', 'hi')

>> vectors [0].shape

(400,)

Predict Next 'n' words

from inltk.inltk import predict_next_words

predict_next_words(text , n, '<code-of-language>')

// text --> string in <code-of-language>

// n --> number of words you want to predict (integer)

Program 4 [program courtesy 12-14]

Installation

This work is done using Python 2.7. (Python 3 support is in progress)

pip install sanskrit_parser

Usage

Lexical Analyzer

Use the SanskritLexicalAnalyzer to split a sentence (wrapped in a SanskritObject) and retrieve the top 10 splits:

>>> from sanskrit_parser.lexical_analyzer.SanskritLexicalAnalyzer import SanskritLexicalAnalyzer

>>> from sanskrit_parser.base.SanskritBase import SanskritObject, SLP1

>>> sentence = SanskritObject("astyuttarasyAMdishidevatAtmA")

>>> analyzer = SanskritLexicalAnalyzer()

>>> splits = analyzer.getSandhiSplits(sentence).findAllPaths (10)

>>> for split in splits:

... print split

6. Morphological Analyzer :

The Sanskrit Morphological Analyzer class has a similar interface to SanskritLexicalAnalyzer, and has a constrainPath() method which can find whether a particular split has a valid morphology, and output all such valid morphologies.

>>> from sanskrit_parser.base.SanskritBase import SanskritObject, SLP1

>>> from sanskrit_parser.morphological_analyzer.SanskritMorphologicalAnalyzer import SanskritMorphologicalAnalyzer

As mentioned previously, both over-generation and under-generation are possible with the Sandhi class.

MaheshvaraSutras

Get varnas in a pratyahara:

>>> from sanskrit_parser.base.MaheshvaraSutras import MaheshvaraSutras

7. Sanskrit Object :

SanskritObject is a base class used in all modules. It supports automatic detection of input encoding and transcoding to any encoding supported by the indic_transliteration package.

>>> from sanskrit_parser.base.SanskritBase import SanskritObject, SLP1

>>> sentence = SanskritObject("astyuttarasyAMdishidevatAtmA")

>>> print sentence.transcoded(SLP1)

astyuttarasyAMdiSidevatAtmA

Command Line Usage

All the classes described above can also be used from the command line. The corresponding examples are below. Please run the tools with --help/-h to get help on the options

SanskritMorphologicalAnalyzer

$ python -m sanskrit_parser.morphological_analyzer.SanskritMorphologicalAnalyzer astyuttarasyAm --input-encoding SLP1 --need-lakara

Input String: astyuttarasyAm

Input String in SLP1: astyuttarasyAm

Start Split: 2017-10-01 11:16:10.489660

End DAG generation: 2017-10-01 11:16:10.496199

End pathfinding: 2017-10-01 11:16:10.497342

Splits:

8. Conclusions :

"Processing of Sanskrit texts poses several challenges owing to the high lexical productivity of the words, free word order in poetry, euphonic assimilation of sounds at the word boundaries and phonemic orthography followed in writing. Keeping these in mind, we proposed a generic graph-based framework that takes advantage of the free word order nature of the language. Further, linguistic insights from the traditional Sanskrit grammar for learning the feature function and applying the relevant constraints." are used this proposed framework substantially reduces the training data requirements to as low as 10%, as compared to that of the neural state-of-the-art models.

In all the Sanskrit-related tasks discussed in the work, we either achieve state-of-the-art results or ours is the only data-driven solution for those tasks. The specific and unambiguous nature and at the same time, the

vast literature and vocabulary of the Sanskrit language provides a gateway for implementing this language in a way that a computer can understand. NASA scientists believe that Sanskrit language can provide a huge impetus to the development to the field of artificial intelligence. One of the oldest well-formed language is now being relegated to scriptures. It is intended to ensure this language becomes the core of understanding machines and also relaying information not just for a wide variety of native population but for the world.

References

[1] Chandana, Bathulapalli., Drumil, Desai., Manasi, Kanhere. (2016). Use of Sanskrit for natural language processing, International Journal of Sanskrit Research; 2(6): 78-81

[2] Tyagi V, 2020, Robotics and AI (Artificial Intelligence) Possible By "SANSKRIT" an Ancient Language, International Journal of Scientific & Engineering Research, 11(7):421-424

[3] Bharati, Akshar & Chaitanya, Vineet & Sangal, Rajeev & Gillon, Brendan. (2002). Natural Language Processing: A Paninian Perspective.

[4] Bird S, Klein E, Loper E, (2009). Natural Language Processing with Python, O'Reilly, 504, ISBN 978-0-596-51649-9

[5] Hacker Dojo CA:https://www.eventbrite.com/e/qsangate-quantum-language-evolution-mapping-sanskrit-as-quantum-language-tickets-83755108785

[6] Hopcroft, JE, Motwani, R, Ullman, JD. 2002. Introduction to Automata Theory, Languages and Computation. 2nd Ed, Pearson Education Pvt. Ltd., 2002.

[7] Briggs, Rick. 1985. Knowledge Representation in Sanskrit and artificial Intelligence, pp 33-39. The AI Magazine.

[8] Analysis of Sanskrit text: parsing and semantic relations by Goyal P, Arora V, Behera L, 2007, First International Sanskrit Computational Linguistics Symposium, INRIA ParisRocquencourt, Oct 2007, Rocquencourt, France. ffinria-00203459

[9] A case for Sanskrit as a computer programming language by P. Ramanujum

[10] Computer Simulation of Astadhyayi by Goyal P, Singh H, Kulkerni A and Behera L.

[11] Mishra A, Modeling the Pāninian System of Sanskrit Grammar, HEIDELBERG UNIVERSITY PUBLISHING

[12] https://www.kaggle.com/datasets/sayantankirtaniya/indic-nlp-library (accessed on May 17, 2024)

[13] https://inltk.readthedocs.io/en/latest/api_docs.html (accessed on May 17, 2024)

[14] https://pypi.org/project/sanskrit-parser/0.0.1.dev4/ (accessed on May 17, 2024)