Treatment of optional forms in Mathematical modelling of Pāṇini

Anupriya Aggarwal, Malhar Kulkarni
Indian Institute of Technology Bombay, Mumbai
anupriya@iitb.ac.in, malharku@hss.iitb.ac.in

Abstract

Pāṇini in his *Aṣṭādhyāyī* has written the grammar of Sanskrit in an extremely concise manner in the form of about 4000 *sūtras*. We have attempted to mathematically remodel the data produced by these *sūtras*. The mathematical modelling is a way to show that the *Pāṇinian* approach is a minimal method of capturing the grammatical data for Sanskrit which is a natural language. The *sūtras* written by *Pāṇini* can be written as functions, that is for a single input the function produces a single output of the form y=f(x), where x and y is the input and output respectively. However, we observe that for some input *dhātus*, we get multiple outputs. For such cases, we have written multivalued functions that is the functions which give two or more outputs for a single input. In other words, multivalued function is a way to represent optional output forms which are expressed in *Pāṇinian* grammar with the help of 3 terms i.e. *vā*, *vibhaṣā*, and *anyatarasyām*. Comparison between the techniques employed by *Pāṇini* and our notation of functions helps us understand how *Pāṇinian* techniques ensure brevity and terseness, hence illustrating that *Pāṇinian* grammar is minimal.

1 Introduction

Pāṇini's Aṣṭādhyāyī is 'almost an exhaustive grammar for any human language with meticulous details yet small enough to memorize it' (Kulkarni, 2016). Such an exhaustive grammar is ideal to be used for artificial language processing. Briggs (Briggs, 1985) even demonstrated in his article the salient feature of Sanskrit language that can make it serve as an artificial language. Although, this is not a new concept, various efforts in mathematical modelling of Indian languages have been done before. Joseph Kallrath in his book 'Modeling Languages in Mathematical Optimization' says that 'a modeling language serves the need to pass data and a mathematical model description to a solver in the same way that people especially mathematicians describe those problems to each other' (Kallrath, 2013). Mathematical modelling of languages also impacts our understanding of the language and its grammar. As scholars are delving into the question of formalizing various natural languages, it is also having an impact on how we understand the language itself. Recent work in theoretical and computational linguistics has influenced the interpretation of grammar (Scharf, 2008). We have followed a similar approach, wherein we have modelled the *Pratyayas* in Sanskrit in the form of functions with the help of *Pāṇinian sūtras*.

Similar to mathematical functions which can be expressed as f(x)=y where x is the input and y is the output of function f; 'the *sūtras* too look for their preconditions in an input environment. The effects produced by *sūtras* become part of an ever-evolving environment which may trigger other' (Sohoni & Kulkarni, 2018). For the grammar to fit mathematical functions, we 'need a strong and unambiguous grammar which is provided by *Maharishi Pāṇini* in the form of *Aṣṭādhyāyī*' (Agrawal, 2013).

Statistical analysis of a language is a vital part of natural language processing (Goyal, 2011). According to how components of the target linguistic phenomenon are realized

Proceedings of the 17th International Conference on Natural Language Processing, pages 15–21
Patna, India, December 18 - 21, 2020.

mathematically, available models of language evolution can be classified as rule-based and equation-based models. Equation-based models tend to transform linguistic and relevant behaviors into mathematical equations (Tao Gong, 2013), which is what we have attempted in this paper.

Ambiguity is inherent in the Natural Language sentences (Tapaswi & Jain, 2012), and hence Sanskrit being a natural language also has certain ambiguities. The ambiguity that we are dealing with in this paper is that a single *dhātu* combined with a single pratyaya can result in two or more optional forms. Mathematical modelling of such natural languages can help to remove this ambiguity. Traditionally too, there have been attempts by various scholars like *Kātyāyana*, *Patanjali* and *Bhartṛhari* to provide extensive commentaries which contain explanations for various aspects of the grammar. They do not question *Pāṇini's* basic model, but rather explain it, refine it and complete it (Huet, 2003). Explanations and clarifications in the form of various *vārtikas* also come handy while dealing with ambiguities. However, here we are diverging from the traditional approach and writing functions in order to model the grammatical data.

To account for more than two forms of a word, *Pāṇini* uses optional form rules to state that alternate forms are also possible. For example, *sūtra* (rule) 1.2.3 *vibhaṣorṇoḥ* states that 'After the verb *ūrṇa* 'to cover', the affix beginning with the augment iṭ is regarded optionally like ṅit (Source, 2020)'. We have used multivalued functions to denote such optional forms in our system of representing the *pratyayas* as functions.

2 Methodology

We are here attempting to mathematically model the data produced by the *sūtras* for which we started with compiling the list of *dhātus* and their respective derived *dhātus* with different *pratyayas* like from the Kridantkosh of Pushpa Dikhshita Vol.1 (Dikshita, 2014), sanskritworld.in (Dhaval Patel, n.d.), Siddhananta Kaumudi of Bhattoji Dikshita (S.C.Vasu, 1905), The Madhaviya DhātuVritti (Sayanacarya, 1964) and the roots, verb-forms and primary derivatives of the Sanskrit Language by W.D.Whitney (Whitney, 1885). The list of *dhātus* without the application of any *pratyaya* are considered as x, after the application of the concept of *anubandhas*. *Anubandhas* have a very prominent role to play in the *Pāṇinian* system of Sanskrit grammar. It literally means 'what is attached to'. It has been used by all ancient authorities on Sanskrit grammar who have come after *Pāṇini*, right from *Kātyāyana* to *Nageśa*. However, *Pāṇini* has used the term '*it*' to describe the *anubandhas*. M. Williams dictionary (Williams, 2008 revised) defines *anubandhas* as an indicatory letter or syllable attached to roots etc., marking some peculiarity in their inflection e.g. an '*i*' attached to roots denotes the insertion of a nasal before their final consonant. According to *Nyāyakosa*, *anubandha* is a letter that is attached to the stem (*prakṛti*), termination (*pratyaya*), augment (*āgama*) or a substitute (*ādesha*) to indicate the occurrence of some special modifications such as *vikaraṇa*, *āgama*, *guṇa* or *vṛddhi*, accent etc. But it is dropped from the finished word i.e. *pada*. The use of *anubandha* is one of the crucial steps *Pāṇini* has taken to ensure the brevity and terseness of his work. We can say that *anubandhas* do form part of the *pratyayas* etc. to which they are found appended (Devasthali, 1967). But before we directly start writing our functions, we need to define the input set which comprises of *dhātus* from the *Dhātupatha* as well as the derived *dhātus* without *anubandhas*.

Let A be a set of all the *dhātus* after the *anubandhas* have been removed. These primary *dhātus are* 1943 in total. However, the input *dhātus* are not limited to these *dhātus* in set A. We can also derive a new *dhātu* set B by adding a *san pratyaya* to the *dhātus of* set A. The items in set B can be called *dhātus by* following the grammatical rule laid down by *Pāṇini,* '3.1.32 *sanādyantāḥ dhātavaḥ*' which says that 'all roots ending with

***Sūtra* numbers**	***Pratyaya***
3.1.5	*san*
3.1.8	*kyac*
3.1.9	*kāmyac*
3.1.11	*kyaṅ*
3.1.13	*kyaṣ*
3.1.20	*ṇiṅ*
3.1.21	*ṇic*
3.1.22	*yaṅ*
3.1.27	*yak*
3.1.28	*āy*
3.1.29	*īyaṅ*

Table 1: List of *San pratyayas* in *Aṣhṭādhyāyī* with their respective *sūtra* numbers

the *pratyayas* starting with san are called *dhātu.*

Hence the input x is defined as,

$$x \in (A \cup B)$$

In this paper we will focus on the multivalued functions that give two or more outputs for the same input *dhātu* of the form $f(x)=\begin{cases} y_1 \\ y_2 \end{cases}$ if there are two optional forms; $f(x)=\begin{cases} y_1 \\ y_2 \\ y_3 \end{cases}$ if there are three optional forms and so on.

3 Notation

Let x be the input *dhātu*. For the purpose of writing these functions, we start enumerating the syllables from left to right or from right to left depending upon that particular function. We can denote x as, x= (….,x(2),x(1))= (x'(1),x'(2),….). x can be a consonant (C) or a vowel (V) and they are denoted by

C'(i)= i^{th} consonant from left;
V'(i)= i^{th} vowel from left;
C(i)= i^{th} consonant from right,
V(i)= i^{th} vowel from right.

cura =	*c*	*u*	*r*	*a*
Right to left	x(4)	x(3)	x(2)	x(1)
Left to right	x'(1)	x'(2)	x'(3)	x'(4)

Table 2: The numbers 1, 2, 3,… signify the position of the syllable. The notation x (unprimed) is used when the syllables are counted right to left, and the notation x' is used when the syllables are counted left to right.

For example: If x = *cura*, then

Conversion are denoted by a right arrow with a number on the top. The number denotes the location of the conversion.

For example, $x[a \xrightarrow{2} \bar{a}]$ denotes that in the *dhātu* x, a which is at the 2nd place from the right is getting replaced with $\bar{a}$.

We also define a '+ operator' to explain the change of syllables when two syllables combine. In Sanskrit language when two syllables come closer, for the ease of pronunciation (in most cases) it gets replaced by another syllable or a combination of syllables. For example: *ū+i=vi, e+i=ayi, o+i=avi, d+ta=tta, ch+t= ṣṭa, j+ta=kta, dh+ta=dhda, bh+ta=bdha, h+ta= ṇḍha*. Note that although the '+ operator' may look similar to the concept of *Sandhi* in Sanskrit, it is totally based on our need to fit our dataset and does not encompass the broad concept of *Sandhi*.

4 A function p(x)

This function is not a *pratyaya* function, but it is required to write the *pratyaya* function. Thus, it would be helpful to define it here. The *dhātus* which have two or more vowels are called *udātta*, and when a suffix is added to them an additional '*i*' comes. Such *dhātus* are called *seṭ* (literally meaning 'with *iṭ*'). For *dhātus which* have one vowel, we need to see the instructions given in the *Dhātupaṭha*. They can either be *seṭ* or *aniṭ* depending upon the given instructions given. Example of one such instruction is '*bhu sattayām*| '*udātth parasmaibhāṣḥ*'| It says that '*i*' will come as the *prayogsamavāyī svara* is *udātth*.
The function p(x) is defined by,

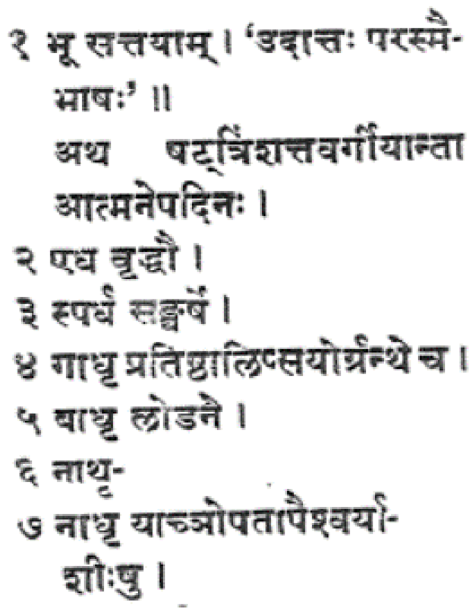
१ भू सत्तायाम् । 'उदात्तः परस्मैभाषः' ॥
अथ षट्त्रिंशत्तवर्गीयान्ता आत्मनेपदिनः ।
२ एध वृद्धौ ।
३ स्पर्ध सङ्घर्षे ।
४ गाधृ प्रतिष्ठालिप्सयोर्ग्रन्थे च ।
५ बाधृ लोडने ।
६ नाथृ-
७ नाधृ याच्ञोपतापैश्वर्याशीःषु ।

Figure 1: Example of an Instruction given in the *Dhātupatha*.

$$p(x) = \begin{cases} \text{i if x is } set, \\ \text{0 if x is } anit. \end{cases}$$

5 Multivalued functions

The words used for optionality by *Pāṇini* are vā, *vibhaṣā*, *anyatarasyām*. *vā* appears 136 times, *vibhaṣā* appears 258 times and, *anyatarasyām* appears 161 times respectively in *Aṣṭādhyāyī*; including the ones that occur in *Anuvritti*[1]. *Pāṇini* and all the commentators have given us no indication that they are supposed to be anything but synonyms. But the modern scholar Paul Kiparsky has wondered how could this be so, because *Pāṇini* has vowed to eliminate every needless extraneous syllable and there must be a deeper reason to suggest the use of three different terms. Hence, he has propounded the hypothesis in his well-argued study *Pāṇini* as a 'variationist' that the three terms *vā, vibhaṣā, anyatarasyām* refer respectively to three different kinds of options: those that are preferable (*vā*), those that are marginal (*vibhaṣā*) and those that are simple options(*anyatarasyām*) (Sharma, 2018).

Word	**Occurrence**	**Usage**
vā	136 times	preferable
vibhaṣā	258 times	marginal
anyatarasyām	161 times	simple options

Table 3: Words used for optionality by *Pāṇini*

One such case which results in such optional forms is represented in the table below where the addition and absence of '*i*' results in two forms and the change of '*h'* syllable to two different syllables further results in two forms. Thus, we end up with three forms of the same word.
Let us look at an example for this case for x = *muh*:

$$tum(muh) = \left\{\begin{array}{l} x\left[i\ u \xrightarrow{2} e\ o\right] + 0 + tum \\ x\left[i\ u \xrightarrow{2} e\ \ o\right] + 0 + \d{d}hum \\ x\left[i\ u \xrightarrow{2} e\ o\right] + i + tum \end{array}\right.$$

$$= \left\{\begin{array}{l} mogdhum \\ mo\d{d}hum \\ mohitum \end{array}\right.$$

[1] The number of times these words appear in *Aṣṭādhyāyī*;
in
ca
A
S
P
P
2(

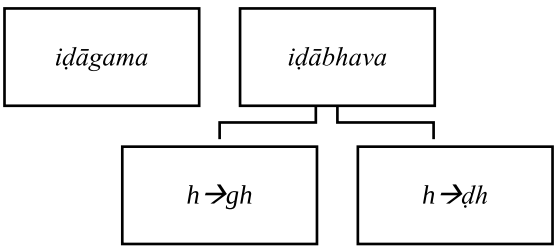

Figure 2:Multivalued functions

6 Cases for multivalued functions

Some cases for multivalued functions are displayed below[2].

Some Multivalued functions for *Tumun Pratyaya*

Case I:
If x ϵ {*svṛ sū dhū*}, then

$$tum(x) = \left\{\begin{array}{l} x\left[i/\bar{\imath}u/\bar{u}\d{r}/\bar{\d{r}} \xrightarrow{1} e\ o\ ar\right] + i + tum \\ x\left[i/\bar{\imath}u/\bar{u}\d{r}/\bar{\d{r}} \xrightarrow{1} e\ o\ ar\right] + 0 + tum \end{array}\right.$$

x	$\left\{\begin{array}{l} x\left[i/\bar{\imath}u/\bar{u}\d{r}/\bar{\d{r}} \xrightarrow{1} e\ o\ ar\right] + i \\ x\left[i/\bar{\imath}u/\bar{u}\d{r}/\bar{\d{r}} \xrightarrow{1} e\ o\ ar\right] + 0 \end{array}\right.$	tum(x)
sū	*svi* *so*	*svitum* *sotum*
svṛ	*svari* *svar*	*svaritum* *svartum*

Case II:
If x has two syllables such that x(1)= *ṝ*, then

$$tum(x) = \left\{\begin{array}{l} x\left[\bar{\d{r}} \xrightarrow{1} ar\right] + i + tum \\ x\left[\bar{\d{r}} \xrightarrow{1} ar\right] + \bar{\imath} + tum \end{array}\right.$$

x	$\left\{\begin{array}{l} x\left[\bar{\d{r}} \xrightarrow{1} ar\right] + i \\ x\left[\bar{\d{r}} \xrightarrow{1} ar\right] + \bar{\imath} \end{array}\right.$	tum(x)
vṝ	*vari* *varī*	*varitum* *varītum*
kṝ	*kari* *karī*	*karitum* *karītum*

Case III:
If x ϵ {gup}, then

$$tum(x) = \left\{\begin{array}{l} x\left[u \xrightarrow{2} o\right] + i + tum \\ x\left[u \xrightarrow{2} o\right] + 0 + tum \\ x\left[u \xrightarrow{2} o\right] + \bar{a}y + i + tum \end{array}\right.$$

[2] An exhaustive list of cases for *Tumun* and *san pratyayas* including the multivalued cases are given in the appendix in Devanagari script.

x	$\begin{cases} x\left[u \xrightarrow{2} o\right] + i \\ x\left[u \xrightarrow{2} o\right] + 0 \\ x\left[u \xrightarrow{2} o\right] + \bar{a}y + i \end{cases}$	*tum*(x)
gup	*gopi* *gop* *gopāyi*	*gopitum* *goptum* *gopāyitum*

Case IV:
If x ϵ {tṛp dṛp}, then

$$\text{tum(x)} = \begin{cases} \text{x}\left[\text{ṛ} \xrightarrow{2} \text{ar}\right] + 0 + \text{tum} \\ \text{x}\left[\text{ṛ} \xrightarrow{2} \text{r}\right] + 0 + \text{tum} \\ \text{x}\left[\text{ṛ} \xrightarrow{2} \text{ar}\right] + \text{i} + \text{tum} \end{cases}$$

x	$\begin{cases} x\left[\text{ṛ} \xrightarrow{2} ar\right] + 0 \\ x\left[\text{ṛ} \xrightarrow{2} r\right] + 0 \\ x\left[\text{ṛ} \xrightarrow{2} ar\right] + i \end{cases}$	tum(x)
dṛp	*darp* *drap* *darpi*	*darpatum* *draptum* *darpitum*

Some Multivalued functions for *San Pratyaya*

Case I:
If x’(1)=c, x’(2)=v= *i u*, x’(3)=c in x(which has exactly 3 letters), then

$$\text{san(x)}=\begin{cases} \text{T(x)} + \text{v}'(1) + \text{x}[i\ u \rightarrow e\ o] + \text{iṣa} \\ \text{T(x)} + v'(1) + x + \text{iṣa} \end{cases}$$

where, T(x)= $c'(1)[k\ g\ bh\ \text{ṣ}\ h\ \rightarrow\ c\ j\ b\ s\ j]$

x	$\text{T(x)} + \text{v}'(1)$	*san*(x)
gud	*ju*	*jugodiṣa* *jugudiṣa*
yut	*yu*	*yuyotiṣa* *yuyutiṣa*
vith	*vi*	*vivethiṣa* *vivithiṣa*
cit	*ci*	*cicetiṣa* *cicitiṣa*

Case II:
If there is only one v in x, such that x(2)=v= *i u* and starts with at least two consonants i.e x’(1)=c, x’(2)=c, then

san(x)=

$$\begin{cases} c'(1) + \text{v}'(1) + x\{v[i\ u \xrightarrow{2} e\ o\ ]\} + p(x) + \text{ṣa} \\ c'(1) + \text{v}'(1) + x + p(x) + \text{ṣa} \end{cases}$$

x	$\text{c}'(1) + \text{v}'(1)$	san(x)
cyut	*cu*	*cucyotiṣa* *cucyutiṣa*
kliś	*ci*	*cicleśiṣa* *cikliśiṣa*

7 Conclusion

According to the mathematical definition of a function, it generates a unique output for every input. However, while mathematically modelling *Pratyayas* in Sanskrit we came across several instances where a single input was generating multiple outputs, which have been represented by multivalued functions.

To ensure brevity, *Pāṇini* has used several tools which have been compared with their equivalent tools in our functional approach.

Functions	***Pāṇinian* tools**
x(2)	*upadhā*
c’1,c’2,…,v’1 if x'1=consonant; c’1,c’2,…,v’2 if x'1=vowel	*ekāc*
Multivalued functions	*vā, vibhaṣā, anyatarasyām*
x(1)	*antya*
-	*anuvṛtti*

Table 1: *Pāṇinian* Techniques vs functions

What we are essentially denoting as x(2) in our functions i.e. the penultimate term is nothing but *upadhā*. *Pāṇini* by convention treats x(1) as the end and calls it *antya*. This is clear from the definition of *upadhā* given by *Pāṇini* in Aṣṭādhyāyī *sūtra* ‘1.1.65 *alontyāt pūrva upadhā’*, which means ‘The letter immediately preceding the last letter of a word is called penultimate (*upadhā*) (Creative Commons, 2020)’. As stated before in the paper, the words *vā, vibhaṣā,* and, *anyatarasyam* are used by *Pāṇini* to denote optional forms that we have demoted by multivalued functions.

Another important feature of *Pāṇinian* grammar is *anuvṛtti*, which is a technique of carrying some parts of the previous *sūtras* to the next *sūtras*. Due to *anuvṛtti*, the order in which various elements appear in the *sūtra* itself are very important. However, we do not need to define any such equivalent tool in our modeling as long as we define some global functions and operators such as p(x) and the '+' operator.

By mathematically modeling *pratyayas*, the reason behind use of these techniques employed by *Pāṇini* to ensure brevity becomes very clear.

Mathematical modelling of *Pāṇinian* grammar in this way helps identify some general patterns, each of which is grouped separately as a case in the functions. These patterns are mainly dependent upon the occurrence of certain specific syllables at certain places. However, we observed that there are some *dhātus* which even after fulfilling the conditions given in the cases, give an output which is different from what is observed in the literature. All such cases needed a separate approach. Hence the for the treatment of such cases input sets for those particular cases have been defined.

The knowledge of *Pāṇinian* rules also helps us reduce the number of individual cases that have been constructed for each function. It helps group certain cases together into a single generalized case. For example: instead of writing three individual functions for *i→e*, *u→o*, and *ṛ→ar*, the knowledge of the rules in *Aṣṭādhyāyī* helps to write a general case of the form *i u ṛ → e o ar*.

Writing such functions for all other *pratyaya* functions may lead us towards a global function for *pratyayas* and for other grammatical tools as well. This technique of mathematical modelling is extremely helpful to understand Sanskrit grammar for people who are non-linguists or do not understand the technicalities of Sanskrit grammar. This mathematical model can also form a base for further processing of the grammatical rules for natural language processing of the language with the help of well-defined input and output sets.

Acknowledgments

Our deepest regards to Prof Swapneel Mahajan from the Department of Mathematics, IIT Bombay whose guidance in terms of inputs and ideas have helped shape the concept of these functions.

References

Agrawal, S. S. (2013). Sanskrit as a Programming Language and Natural Language Processing. *Global Journal of Management and Business Studies. Volume 3*.

Briggs, R. (1985). Knowledge Representation in Sanskrit and artificial Intelligence. *The AI Magazine*, 32-39.

Creative Commons. (2020, 1 4). *Ashtadhyayi*. Retrieved from Paniniya Moolstrot: https://ashtadhyayi.github.io/

Devasthali, G. V. (1967). *Anubandhas of Panini*. Poona: W.H.Golay.

Dhaval Patel, D. (n.d.). *Sanskrit Tool*. Retrieved 12 16, 2019, from Sanskrit World: https://www.sanskritworld.in/sanskrittool/SanskritVerb/tiGanta.html

Dikshita, P. (2014). *kavirasayanmityaparanama kridantkoshah prathamo bhagah*. pratibha prakashan.

Goyal, L. (2011). Comparative analysis of printed Hindi and Punjabi text based on statistical parameters. *Information systems for Indian languages, communications in computer and information science, Volume 139, Part 2*. Berlin, Heidelberg: Springer.

Huet, G. (2003). Lexicon-directed segmentation and tagging in Sanskrit. (pp. 307-325). Helsinki, Finland: In XIIth World Sanskrit Conference.

Kallrath, J. (2013). *Modeling Languages in Mathematical Optimization*. Springer Science & Business Media.

Kulkarni, A. (2016). Brevity in Pāṇini's Aśṭādhyāyī. In B. A. Joseph, *The Interwoven World: Ideas and Encounters in History*. Common Ground Publishing.

S.C.Vasu. (1905). *The Siddhanta Kaumudi of Bhattoji Dikshita*. Allahabad, The Panini office.

Sayanacarya. (1964). *The Madhaviya Dhatuvritti*. Prachya Bharati Prakashan.

Scharf, P. M. (2008). Modeling Paninian Grammar. *International Sanskrit Computational Linguistics Symposium*, (p. 97).

Sharma, D. N. (2018). *Introduction To Panini's Grammar*. CC0 1.0 Universal.

Sohoni, S., & Kulkarni, M. (2018). A Functional Core for the Computational Aṣṭādhyāyī. *Computational Sanskrit and Digital Humanities, Selected papers presented at the 17th World Sanskrit Conference*.

Source, O. (2020). *१.२.३ विभाषोर्णोः*. Retrieved from Ashtadhyayimulstrota: https://ashtadhyayi.github.io//sutra-details/?sutra=1.2.3

Tao Gong, L. S. (2013). Modelling language evolution: Examples and predictions. *Elsivier*, 2.

Tapaswi, N., & Jain, S. (2012). Treebank based deep grammar acquisition and Part-Of-Speech Tagging for Sanskrit sentences. *IEEE*.

Whitney, W. D. (1885). *The roots, verb-forms, and primary derivatives of the Sanskrit language. A supplement to his Sanskrit grammar.* Leipzig, Breitkopf and Härtel.

Williams, M. (2008 revised). *Monier Williams Dictionary.*