A Graph Based Semi-Supervised Approach for Analysis of Derivational Nouns in Sanskrit

Amrith Krishna, Pavankumar Satuluri[*], Harshavardhan Ponnada, Muneeb Ahmed[#], Gulab Arora, Kaustubh Hiware and Pawan Goyal
[*]School of Linguistics & Literary Studies, Chinmaya Vishwavidyapeeth CEG Campus ;
[#]Dept. of Electrical Engineering, Indian Institute of Technology BHU;
Dept. of Computer Science & Engineering, Indian Institute of Technology Kharagpur
amrith@iitkgp.ac.in

Abstract

Derivational nouns are widely used in Sanskrit corpora and is a prevalent means of productivity in the language. Currently there exists no analyser that identifies the derivational nouns. We propose a semi supervised approach for identification of derivational nouns in Sanskrit. We not only identify the derivational words, but also link them to their corresponding source words. The novelty of our work is primarily in its design of the network structure for the task. The edge weights are featurised based on the phonetic, morphological, syntactic and the semantic similarity shared between the words to be identified. We find that our model is effective for the task, even when we employ a labelled dataset which is only 5 % to that of the entire dataset.

1 Introduction

Derivational affixes are a prevalent means of vocabulary expansion used in natural languages. Derivational affixes are non meaning preserving affixes, that when applied to a word induce a new word. The affixes signify one or possibly more than one semantic senses that is passed onto the new derived word (Marchand, 1969). For example, the noun 'driver' is derived from the verb 'drive' and the adverb 'boldly' is derived from 'bold', where the derivational affixes '-er' and '-ly' are used. However, affixes that modify only the morphological or syntactic role of a word in its usage are not considered derivational, but as inflectional (Faruqui et al., 2016).

Whenever a new word comes into existence in a language, all of its derived forms are potent to be part of the language's vocabulary as well. But, whenever a derived word is used in conversation, a human does not require an explicit knowledge about the derived word to infer its meaning. The knowledge about the source word and the affix is sufficient for her to infer the derived word's meaning. For example, if a new country is formed with name *nauratia*, an English speaker can infer the meaning for the word *nauratian* as "a person residing in nauratia", in spite of never hearing the derived word previously. Similarly, It is desirable to identify a derived word and link it to its corresponding source word computationally. It is of great practical value if we can obtain a semantic word representation for a derived word from the semantic word representation of its source word. It is often the case that corpus evidence for the source word might be abundant, but the corpus evidence for all the possible derived words need not be available readily (Cotterell and Schütze, 2017). Lazaridou et al. (2013) proposed multiple approaches, all being modifications of Compositional Distributional Semantic Model (CDSM) (Mitchell and Lapata, 2010), for obtaining the semantic word representations for a derived word by combining the representations for source word representation and the representation of the affix .

Identifying derived words from a corpus is challenging. Usage of pattern matching approaches in strings are often inept for the task. Tasks that rely on string matching approaches alone, often result in a large number of false positives. For example, while the word 'postal' is generated from 'post', the word 'canal' is not generated from 'can'. String matching approaches often result in low recall as well, due to the variations in patterns in the derived and source word pairs, even for the same affix. Both 'postal' and 'minimal' are derived using the affix 'al', but the source word for postal is 'post', while the source word for minimal is 'minimum'. Soricut and Och (2015), re-

cently proposed an approach for analysis and induction of morphology in words using word embeddings. But, the authors find that their approach, though effective for inflectional affixes, has limitations with derivational affixes.

In this work, we propose an approach for analysis of derivational nouns in Sanskrit. The rules for generation of derivational nouns are well documented in the ancient grammar treatise on Sanskrit, *Aṣṭādhyāyī*. In fact, it can be observed that the grammar treatise has devoted about a 1115 of 4000 rules for dealing with derivational nouns, which is indicative of the prevalence of derivational noun usage in Sanskrit. Currently, there exists no analyser for Sanskrit that deals with the derived words. This leads to issues with large scale processing of texts in Sanskrit. The recent surge in digitising attempts of ancient manuscripts in Sanskrit, like the Digital Corpus of Sanskrit, The Sanskrit Library, GRETIL, etc. provides us with abundance of unlabeled data. But, lack of labeled data and other resources led us to development of a semi supervised approach for identification and analysis of derived words in Sanskrit. We use the Modified Adsorption algorithm (Talukdar and Crammer, 2009), a variant of the label propagation algorithm for the task. In this task, we effectively combine the diverse features ranging from rules in *Aṣṭādhyāyī*, variable length character n-grams learnt from the data using Adaptor grammar (Johnson et al., 2007) and word embeddings for the candidate words using word2vec (Mikolov et al., 2013).

The novel contributions of our task are:

1. We propose a semi-supervised framework using Modified adsorption for identification of derived words and their corresponding source words for Sanskrit.
2. We are able to scale our approach onto unlabelled data by using a small set of labelled data. We find that our model is effective even under experimental settings where we use a labelled dataset of 5 % size as that of the entire dataset. In other words, we we can label upto 20 times more data than the labeled data we have, and we perform a human evaluation to validate our claim on the unlabeled datasets.
3. By leveraging on the rules from *Aṣṭādhyāyī*, we not only find different pattern differences between the source and derived word pairs, but we also group patterns that are likely to emerge from the same affixes. Currently, given a pattern we can narrow down the possible affixes for a pair to a maximum of 4 candidates from a set of 137 possible affixes.

2 Challenges in Sanskrit Derivational Nouns

In this section, we discuss the challenges in identifying the derivational nouns computationally. The section uses some terms, which bear technical definitions as used in the lingusitic discipline of Sanskrit. Table 1, gives the definitions for all such technical terms that we use in this paper. Here, we attempt to build a semi supervised model that can identify usage of derived words in a corpus and map them to their corresponding source words. Here, we are specifically interested in the usage of secondary derivative affixes in Sanskrit, known as *Taddhita*. '*Taddhita*' refers to the process of derivation of a '*prātipadika*' from a '*prātipadika*'. In Sanskrit, a '*prātipadika*' may refer to a noun or an adjective. Hence, Taddhita covers non-category changing derivations, and can be recursive as well (Bhate, 1989).

The derivation procedure proceeds by use of affixation on a word where the affix modifies the source word to form a derived word. While some affixes substantially modify the derived word from its source word, some other affixes tend to form minimal variation. In fact, the variations need not occur only at the word boundary but also at internal portions of a word. Table 2 illustrates some cases which are discussed here. In case of '*upagu*', the derived word gets an internal change and forms '*aupagava*'. But in case of '*daṇḍa*', '*daṇḍin*' is derived, where no internal modifications occur.

In Sanskrit, there are 137 affixes used in Taddhita. The edit distance between the source and derived words due to the patterns tends to vary from 1 to 6. For example, consider the word '*rāvaṇi*' derived from '*rāvaṇa*', where the edit distance between the words is just 1. But, '*Āśvalāyana*' derived from '*aśvala*' has an edit distance of 6. Since, the possible variations that can be expected are quite high, this might lead to a large candidate space when the said patterns are used for matching the words. Additionally, a number of affixes used in *taddhita* are used for other purposes as well. For example, *kṛdanta*, nouns derived from verbs, share some of the affixes with *taddhita*. In Table 2, '*stutya*', a *kṛt*, derived from '*stu*' follows similar pattern as with the derivation

Term	Definition
prātipadika	A prātipadika is a technical term which is used to collectively address nouns and adjectives in Sanskrit. In Sanskrit, both nouns and adjectives belong to the same category
taddhita	Set of secondary derivative affixes i.e. affixes used for deriving a prātipadika from an existing prātipadika. Incidentally the prātipadikas so derived are also called as taddhita (or taddhitānta)
kṛt	Set of primary derivative affixes used for deriving a prātipadika from a verbal roots.
kṛdanta	Prātipadikas derived from verbal roots by affixing primary derivative suffixes (kṛt) are called kṛdanta.
vṛddhi	The sounds '*ā*', '*ai*' and '*au*' are designated as *vṛddhi*. In *taddhita*, it is observed that the first occurrence of a vowel in words often gets transformed to one of the *vṛddhi* vowels. This operation is also termed as *vṛddhi*.
guṇa	The sounds'*a*','*e*' and '*o*' are called as *guṇa*. Whenever the *guṇa* operation is invoked in, the mentioned vowels will be replaced in place of other vowels.

Table 1: Technical terms in Sanskrit and their definitions

of '*dākṣhiṇātya*', a *taddhita* word, derived from '*dakṣhiṇā*'. Now, '*kālaśa*' is derived from '*kalaśa*', where only an internal change is visible. But the similar pattern between '*karaṇa*' (Instrument) and '*kāraṇa*' (Reason) is a mere coincidence.

We can find that for deriving the word '*vainateya*' (Son of *Vinatā*) from *vinatā* (Wife of sage *Kaśyapa*, a mythological character), the *'ā'* at the end gets replaced with '*eya*', and an internal modification happens from 'i' to 'ai'. So ([*i*→*ai*], [*ā*→ *eya*]) is a valid pattern transformation. Similarly, *gāṅgeya* (Son of the river Ganges) is formed from the word *gaṅgā* (River Ganges). The pattern ([*a*→*ā*], [*ā*→*eya*]) is followed. We could find more than 400 different such patterns induced by the 137 affixes.

With our knowledge from *Aṣṭādhyāyī*, we can abstract out some of the regularities in the modifications made, especially those happening at the internal portions of a word. We see those modifications as result of specific operations performed on the word. In this work, we consider two such operations, important for *taddhita* which we define now.

Vṛddhi - The sounds '*ā*', '*ai*' and '*au*' are designated as *vṛddhi*. In *taddhita*, it is observed that the first occurrence of a vowel in words often gets transformed to one of the *vṛddhi* vowels. This operation is also termed as *vṛddhi*. In Table 2, *upagu*, *pramukha*, *aśvala* and *kalaśa* are some *taddhita* words that show *vṛddhi* of its words. The operation is not exclusive to taddhita and occurs in other instances as well. *sṛ*, *kṛ* are some examples.

Guṇa - The sounds '*a*','*e*' and '*o*' are called as *guṇa*. Whenever the *guṇa* operation is invoked in *Aṣṭādhyāyī*, the mentioned vowels will be replaced in place of other vowels. In case of '*aupagava*', at a certain point of derivation, it takes the form '*aupagu a*', and the '*u*' gets converted to '*o*' by virtue of *guṇa*, finally resulting in *aupagava*. This operation is called *guṇa*. It is important to note that, the pattern 'ava' in the derived form instead of 'u' in the source word is result of the transformation sequence u → o → av → ava, which would not have been possible without applying the guṇa operation. For the complete derivation procedure of the derivational noun '*aupagava*' from *upagu* as prescribed in *Aṣṭādhyāyī*, please refer to *Table 1* in Krishna and Goyal (2015).

We define the character sequence which gets modified or eliminated from the source word during the derivation as 'source pattern' or *'sp'*, and the character pattern that appears in the derived word is termed as 'end-pattern' or *'ep'*. The patterns contain all the other changes apart from *guṇa* and *vṛddhi*. With this knowledge, now if we look into the patterns ([*i*→*ai*], [*ā*→ *eya*]) and ([*a*→*ā*], [*ā*→ *eya*]), we can abstract the first component in both the pattern transformations as vṛddhi operation. For, *vinatā* and *gaṅgā* the source pattern (sp) is the phoneme 'ā'. The end pattern for both the words is the phoneme 'eya'. With this abstraction, we narrow down the pattern variations to about 70 end-patterns (ep). We originally had 400 patterns altogether but now we group the possible (derived word, source word) pairs based on their end-pattern only. Thus such a pair can only belong to one of the 70 possible end-patterns. Table 2 shows the end-patterns for the *taddhita* words provided in it.

3 Method

We define our task over a dataset of finite set of vocabulary $\mathcal{C}$. We enumerate all the possible 70 end-patterns as mentioned in Section 2, that can be applied on a source word. With the ex-

Word	Derived Word	Type	ep
upagu (Name of a person)	aupagava (male offspring of Upagu)	Taddhita	a
śiva (Name of a Hindu god)	śaiva (male offspring of śiva)	Taddhita	a
rāvaṇa (A mythological character)	rāvaṇi (male offspring of Rāvaṇa)	Taddhita	i
tila (Sesame)	Tilya (Which is beneficial to sesame)	Taddhita	ya
pramukha (Prominent)	prāmukhya (Prominence)	Taddhita	ya
daṇḍa (Stick)	daṇḍin (One who carries stick)	Taddhita	in
sṛ (To go)	sārin (One who moves)	Kṛt	–
kṛ (To do)	kāraka (One who does)	Kṛt	–
aśva (Horse)	aśvaka (bad horse)	Taddhita	ka
aśvala (Holy priest of King Janaka)	āśvalāyana (male offspring of aśvala)	Taddhita	āyana
stu (To praise)	stutya (Worthy of praise)	Kṛt	–
dakṣhiṇā (South direction)	dākṣhiṇātya (Southern)	Taddhita	–
kalaśa (Pitcher)	kālaśa (related to Pitcher)	Taddhita	a
karaṇa (Instrument)	kāraṇa (Reason)	Random	–

Table 2: Derivational nouns and their corresponding source words in Sanskrit. Additionally, possible cases of false positives that follow similar patterns to derivational nouns are provided as well

tracted patterns, we identify word pairs $wp_i = (w_j, w_k) \in \mathcal{C}^2$ and represent each such pair as a tuple $t_{wp_i} = \langle w_j, w_k, sp, ep, vṛddhi = o, guṇa = p, a_{wp_{i1}}, a_{wp_{i2}}, a_{wp_{i3}} \rangle$, where $o, p \in \{0, 1\}$ and sp, ep are the source pattern in w_j and the end-pattern added to the derived word w_k respectively. The variables o, p assume the value 1 if the pattern is considered to be obtained only after the application of the corresponding operation. For each wp_i, we encode a vector $a_{wp_{i1}} \in \{0, 1\}^{|\mathcal{A}_1|}$, where $\mathcal{A}_1$ is the set of all rules in *Aṣṭādhyāyī* relevant for derivational nouns and $a_{wp_i,l} = 1$ indicates that the rule l is applicable to the word pair. Similarly, the vector $a_{wp_{i2}} \in [0, 1]^{|\mathcal{A}_2|}$ represents probability value for each of the variable length character n-grams in $\mathcal{A}_2$ learnt from Adaptor grammar (Johnson et al., 2007). $a_{wp_{i3}}$ represents a word embedding for w_j in $\mathcal{A}_3$ obtained using word2vec (Mikolov et al., 2013). For example, the word '*daṇḍin*', derived from '*daṇḍa*' can be represented as a tuple $\langle daṇḍa, daṇḍin, a, in, vṛddhi = 0, guṇa = 0, a_{wp_{i1}}, a_{wp_{i2}}, a_{wp_{i3}} \rangle$.

With the extracted pairs $W_{candidates} \subseteq C^2$, we propose a binary relevance model that trains a separate classifier for every unique end-pattern.

We use Modified Adsorption (MAD) algorithm, a graph based semi-supervised approach for our task (Talukdar and Crammer, 2009). MAD fits to our requirements specifically on two aspects. Primarily the semi supervised setting helps us to use minimal set of labelled nodes as seed nodes and incorporate other unlabeled nodes into the system. The objective function penalises the results when similar nodes are assigned with different labels. Unlike other semi-supervised algorithms (Zhu and Ghahramani, 2002; Zhou et al., 2003), MAD allows us to design the network structure explicitly as required. In MAD, every node has a label associated with it and is seen as a distribution of the labels rather than a binary assignment. The unlabeled nodes initially have no label assignments, but as the algorithm is executed, every node is updated with a distribution of the labels in the label space. The seed nodes are also allowed to be provided with a label distribution rather than hard-assigned labels. In $MAD(G, \mathcal{V}_{seed})$, the algorithm inputs a graph structure $G(V, E, W)$ and additionally a seed distribution, $\mathcal{V}_{seed}$, for the seed nodes in the vertex set, $V_{seed} \subseteq V$. The algorithm outputs a label distribution $\mathcal{V}$, for every $v \in V$.

For our setting, we find that $W_{candidates} = \mathcal{U} \cup \mathcal{S} \cup \mathcal{G}$, where $\mathcal{U}$ is the set of unlabelled nodes, $\mathcal{S}$ is the set of seed nodes used as labelled nodes for training and $\mathcal{G}$ is the set of gold nodes which is used as the test data for evaluation of the model[1]. For the system a node obtained from $\mathcal{U}$ and $\mathcal{G}$ are indistinguishable. Also, all the three sets are mutually disjoint. For every end-pattern, ep_i, we construct a classifier $MAD_i = \{MAD_{i1}(G_{i1}, \mathcal{S}_i) | MAD_{i2}(G_{i2}, \mathcal{V}_{i1}) | MAD_{i3}(G_{i3}, \mathcal{V}_{i2})\}$, where G_{ik} is a graph $G_{ik}(V_i, E_{ik}, W_{ik})$, and '|' is the pipe symbol signifying that, the output at the left of the operator is used as input to the right of the operator. Note that

[1] We follow the same naming conventions as Faruqui et al. (2016) wherever possible

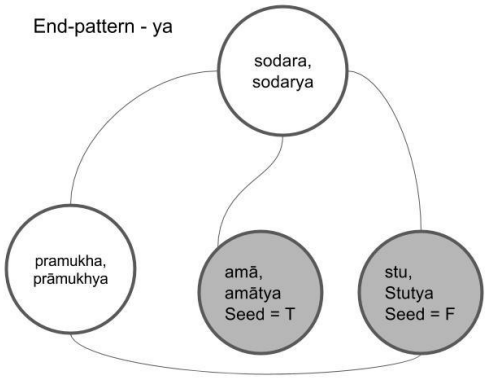

Figure 1: Graph structure for the end-pattern 'ya'. The nodes are possible candidate pairs in $W_{candidates}$. Nodes in grey denote seed nodes, where they are marked with their class label. The Nodes in white are unlabelled nodes.

the vertex set V_i remains the same for all the three graphs G_{i1}, G_{i2}, G_{i3}. Also, $\mathcal{V}_{ik}$ is the label distribution output for $MAD_{i(k)}$ and the seed label distribution for $MAD_{i(k+1)}$. Figure 1 shows the graph structure G_{ik}, for the $ep_i = ya$. Our classifier is a sequential pipeline of 3 graphs, where each graph structure uses label distribution from previous MAD run as its seed. We provide our manually labelled seed set only for the MAD run on MAD_{i1}. In MAD_i, the vertex set V_i remains same in all the runs and is essentially a set of all word pairs that follow a certain end-pattern ep_i.

In our approach, the network structure is influenced by the edge sets $\{E_{i1}, E_{i2}, E_{i3}\}$ and the corresponding weight sets $\{W_{i1}, W_{i2}, W_{i3}\}$, and both are decided by 3 different set of attributes $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ that provide the adjacency and the weights for the relation between the nodes. We explain how the edge set and weight set are defined in each of the phases.

3.1 Phase 1: *Aṣṭādhyāyī* rules

Aṣṭādhyāyī is a grammar treatise on Sanskrit with about 4000 rules, estimated to be written somewhere between fourth century BC and sixth century BC by Pāṇini. About 1115 rules of the 4000 in *Aṣṭādhyāyī*, i.e., more than 25 % of the rules, are devoted to affixation of derivational nouns. The rules related to *Taddhita* either are string rewriting rules, conditional rules, or attribute assignment rules (Krishna and Goyal, 2015). Table 3 illustrate some of the rules related to *Taddhita*, the sense they carry and effect on source word due to the affixation. We consider only the conditional rules used by Pāṇini for the task, which can further be sub-categorised as given below.

1. Phonological and phonemic - Pāṇini uses presence of certain phonological or phonemic entity in the source word as a condition for affixation. For example, the rule '*A.4.1.95 - ata iñ*', states that a lemma ending in '*a*' will be given the affix '*iñ*' when the affix is used to denote the sense of patronymy.
2. Morphological and lexical properties - Pāṇini incorporates a predefined set of lexical lists like gaṇapāṭha where words that are suitable for similar affixal treatment are grouped together. For example, the rule 'A 4.1.112' in Table 2, states to apply the affix '*aṇ*' to all the words in the lexical list headed by '*Śiva*'.
3. Semantic and pragmatic - *Aṣṭādhyāyī* which was intended for human usage, relies on semantic and pragmatic conditions as well. We use additional lexical lists instead of the semantic and pragmatic aspects for the purpose. For example, the rule 'A.4.2.16' applies to those words that signify 'food that is processed or prepared'. Here Pāṇini does not enumerate list of such foods, but just mentions the quality.

In Phase 1 we consider all the rules that deal with any of the phonological, phonemic, morphological and some of the semantic properties. We do not consider the pragmatic conditional rules. Each rule is considered a separate attribute at Phase 1 and the collection is represented as $\mathcal{A}_1$. We define the vertex score $\wp_k$, for $v_k \in V_i$ with the tuple t_k, the weight set W_{i1} and edge set E_{i1} as follows.

$$\wp_k = \sum_{l=1}^{|\mathcal{A}_1|} a_{k_1,l} \tag{1}$$

$$W_{i1}^{v_k,v_j} = \frac{\sum_{l=1}^{|\mathcal{A}_1|} a_{k_1,l} \cdot a_{j_1,l}}{max(\wp_k, \wp_j)} \tag{2}$$

$$E_{i1}^{v_k,v_j} = \begin{cases} 1 & W_{i1}^{v_k,v_j} > 0 \\ 0 & W_{i1}^{v_k,v_j} = 0 \end{cases} \tag{3}$$

In Equation 1, $a_{k_1,l}$ is a component of the vector $a_{k_1} \in \mathcal{A}_\infty$, which indicates whether the 'lth' rule in our filtered set of *Aṣṭādhyāyī* rules is applicable for the word pair represented as the node $v_k \in V_i$ and a_{k_1} is part of the tuple t_k. A source word might satisfy multiple rules and only one of the rules will emerge as the final rule that gets

Rule No	Rule	Semantic Relation	Source Word	Derived Word
4.1.95	ata iñ	Patronym	daśaratha	dāśarathi
4.1.112	śivādibhyo'ṇ	Patronym	śiva	śaiva
4.1.128	catakāya airak	Patronym	catakā	cātakaira
4.2.16	saṃskṛtaṃ bhakṣāḥ	Processed	kalaśa	kālaśāḥ

Table 3: conditional rules related to selection of suitable affix for derivational nouns from *Aṣṭādhyāyī*.

applied (Scharf, 2009). Rules that carry different affixes might find the eligibility for a given pair. For example, consider the rules 'A.4.1.95' and 'A.4.1.112'. For the word *'Śiva'* both the rules apply, and both the affixes *iñ* and *aṇ* find eligibility to be applied. But, according to *Aṣṭādhyāyī*, *Śiva* will get *aṇ* (Krishna and Goyal, 2015). But, in this setting we keep all the attributes that the word qualifies to. The complete derivation history of a word needs to be examined in order to identify the exact rule that can be applied, which is a challenging task by itself.

We consider all the rules that are relevant to an end-pattern and we form an edge between two nodes, if the source words in both the nodes share at least one of the listed property.

3.2 Phase 2: Character ngrams similarity by Adaptor grammar

Pāṇini had an obligation to maintain brevity, as his grammar treatise was supposed to be memorised and recited orally by humans (Kiparsky, 1994). In *Aṣṭādhyāyī*, Pāṇini uses character sub-strings of varying lengths as conditional rules for checking the suitability of application of an affix. We examine if there are more such regularities in the form of variable length character n-grams that can be observed from the data, as brevity is not a concern for us. Also, we assume this would compensate for the loss of some of the information which Pāṇini originally encoded using pragmatic rules. In order to identify the regularities in pattern in the words, we follow a grammar framework called as Adaptor grammar (Johnson et al., 2007). Adaptor grammar is a non-parametric Bayesian approach for learning productions for a Probabilistic Context Free Grammar (PCFG). In the grammar, we provide a skeletal grammar structure, along with the non-terminals to be used in the grammar. The grammar learns the productions and the probabilities associated with each of the productions from the observed data. The productions are variable length character n-grams.

The grammar learns a distribution over trees rooted at each of the adapted non-terminal (Zhai et al., 2014; Krishna et al., 2016). In Listing 1, 'Word' and 'Stem' are non-terminals, which are adapted. The non-terminal 'Suffix' consists of the set of various end-patterns. In this formalism, the grammar can only capture sequential aspects in the words and hence attributes like *vṛddhi* that happen at the internal of the word, non-sequental to rest of the modified pattern, need not be effectively captured in the system.

$Word \rightarrow Stem\ Suffix$
$Word \rightarrow Stem$
$Stem \rightarrow Chars$
$Suffix \rightarrow a|ya|.....|Ayana$

Listing 1: Skeletal CFG for the Adaptor grammar

The set $\mathcal{A}_2$ captures all the variable length character n-grams learnt as the productions by the grammar along with the probability score associated with the production. We form an edge between two nodes in G_{i2}, if there exists an entry in $\mathcal{A}_2$, which are present in both the nodes. We sum the probability value associated with all such character n-grams common to the pair of nodes $v_j, v_k \in V_i$, and calculate the edge score $\tau_{j,k}$. If the edge score is greater than zero, we find the sigmoid of the value so obtained to assign the weight to the edge. Equation 4 uses the Iverson bracket (Knuth, 1992) to show the conditional sum operation. The equation essentially makes sure that the probabilities associated with only those character n-grams gets summed, which is present in both the nodes. We define the edge score $\tau_{j,k}$, weight set W_{i2} and Edge set E_{i2} as follows.

$$\tau_{j,k} = \sum_{l=1}^{|\mathcal{A}_2|} a_{k_2,l}[a_{k_2,l} = a_{j_2,l}] \tag{4}$$

$$E_{i2}^{v_k,v_j} = \begin{cases} 1 & \tau_{j,k} > 0 \\ 0 & \tau_{j,k} = 0 \end{cases} \tag{5}$$

$$W_{i2}^{v_k,v_j} = \begin{cases} \sigma(\tau_{j,k}) & \tau_{j,k} > 0 \\ 0 & \tau_{j,k} = 0 \end{cases} \tag{6}$$

As mentioned, we use the label distribution per

node obtained from phase 1 as the seed labels in this setting.

3.3 Phase 3: Semantic Word vectors

In phase 3, we try to leverage the similarity between word embeddings (Mikolov et al., 2013) to propagate the labels. Due to limited resources at our disposal, we find it difficult to train word embeddings for Sanskrit. We resort to finding synonyms of words using the digitised version of Monier-Williams Sanskrit-English dictionary and then use the corresponding pre-trained English word vectors for the task. We find the word vectors only for the source words as the dictionary entries for derived words are even scarcer to obtain. Since we perform only a dictionary lookup for finding the synonyms of a word, we do not get embeddings for named entities from the dictionary. A given word might have multiple senses in English and hence multiple English synonyms. In such cases, we find all possible similarity scores and take the maximum score among them.

We use cosine similarity between the word vectors as the edge weight in this phase. For each node, for which we were able to obtain a word vector, we find its cosine similarity with that of every other node in the graph for which there exists a word vector. We find that our graph structure G_{i3} for many end-patterns results in multiple disconnected components, as not all words in $W_{candidates}$ has an entry in the dictionary. We assign teleportation probability to every node in the graph in order to handle this issue.

4 Experiments

We explain the experimental settings and evaluation parameters for our model in this section.

4.1 Dataset

We use multiple lexicons and corpora to obtain our vocabulary $\mathcal{C}$. We use IndoWordNet (Kulkarni et al., 2010), the Digital Corpus of Sanskrit[2], a digitised version of the Monier Williams[3] Sanskrit-English dictionary, a digitised version of the Apte Sanskrit-Sanskrit Dictionary (Goyal et al., 2012) and we also utiilise the lexicon employed in the Sanskrit Heritage Engine (Goyal and Huet, 2016). We obtained close to 170,000 unique word lemmas from the combined resources.

[2]http://kjc-sv013.kjc.uni-heidelberg.de/dcs/

[3]http://www.sanskrit-lexicon.uni-koeln.de/monier/

Obtaining Ground Truth Data - For our classifier MAD, we obtain the seed labels $\mathcal{S}$ and the gold labels $\mathcal{G}$ from a digitised version of Apte Sanskrit-Sanskrit dictionary. The dictionary has preserved the etymological information of the entries in the dictionary. For each end-pattern we filtered out the pair of words which are related by *Taddhita* affixes. Seed nodes for the negative class were obtained using candidate pairs which were either marked as *kṛdanta* words in the Apte Dictionary or were found in the dictionary, but are not related to each other. Additionally, we manually tagged some word pairs so as to obtain a balanced set of labels. We narrowed to 11 separate end-patterns for which we have at least 100 candidate pairs and have at least 5 % of word pairs as seed nodes in comparison to the the size of the candidate pairs for the end-pattern. Table 4 shows the statistics related to each of the 11 end-patterns on which we have performed our experiments.

4.2 Baselines

We propose the following systems as the competing systems. We use label propagation (Zhu and Ghahramani, 2002) as a strong baseline and we also compare the output at each of the phase as separate baseline systems. Altogether we compare four systems as follows:

1. Label Propagation (LP_i) - We propose a label propagation based semi supervised classifier (Pedregosa et al., 2011) for each of the end-pattern. For each node, we find the top K similar nodes and assign edges to only those nodes, where K is a user given parameter. The similarity is obtained from a feature vector that defines a node, with features from the first 2 phases incorporated into a single feature vector. We do not use the word embeddings from Phase 3 directly, but find the cosine similarity between the embeddings of the words and perform a weighted sum with the similarity score obtained from the similarity obtained from the combined feature vector.

2. $MADB1_i$ - We report the performance of the system $MADB1_i = \{MAD_{i1}(G_{i1}, \mathcal{S}_i)\}$, where we define the network structure only based on the Phase 1 in Section 3

3. $MADB2_i$ - We report the performance of the system $MADB2_i = \{MAD_{i1}(G_{i1}, \mathcal{S}_i)| MAD_{i2}(G_{i2}, \mathcal{V}_{i1})\}$, where we define the settings for MAD_{i1}, MAD_{i2} based on the de-

End-pattern	$W_{candidates}$	Seed $\mathcal{S}$	Gold Labels $\mathcal{G}$	Recall	Precision	Accuracy
a	2500	350	88	0.77	0.72	73.86
aka	1200	120	30	0.67	0.77	73.33
in	1656	270	68	0.82	0.74	76.47
ya	1566	258	64	0.72	0.7	70.31
i	1455	166	42	0.52	0.55	54.76
ika	803	122	30	0.6	0.69	66.67
tā	644	34	12	0.5	0.6	58.33
la	360	48	12	0.67	0.8	75
tva	303	22	12	0.67	0.8	75
īya	244	40	12	0.67	0.67	66.67
eya	181	34	12	0.83	0.71	75

Table 4: Recall (R), Precision (P) and Accuracy (A) for the candidate nodes evaluated on the gold labels.

scription in Phase 1 and Phase 2 respectively, as defined in Section 3

4. MAD_i - This is the proposed system, as defined in Section 3

4.3 Results

Table 4 shows the final results of our proposed system MAD_i, for each of the 11 end patterns. We report the Precision, Recall and Accuracy for each of the classifier w.r.t the true class. Our results are calculated based on the predictions over the test data in $\mathcal{G}$. Seven of Eleven patterns have an accuracy above 70 %. End-pattern '*i*' is reported to perform the least among the 11 patterns provided. We find that the average degree for G_{i1} for the pattern '*i*' is about 77.62, much higher than the macro average degree for G_{i1} for all the patterns, which is 43.86. This is primarily due to the restrictive nature of node selection that is employed for the pattern 'i' as per *Aṣṭādhyāyī*. We have selected only those nodes which have the *vṛddhi* attribute set to 1 and only those source words which end in '*a*'. This has led to higher average degree among the nodes that got filtered as per *Aṣṭādhyāyī* rules. In order to keep uniform settings for all the systems, we do not deviate from the design. But, for pattern '*i*',when we randomly down-sample the number of neighbours to 44 (to match with the macro average), the accuracy increases to 61.9 %.

Table 5 shows the results for the competing systems. We compare the performance of 5 end-patterns, selected based on the vertex set size V_{i1}. Our proposed system, MAD_i performs the best for all the 5 patterns. Interestingly, $MADB2_i$ is the second best-performing system in all the cases beating LP_i. For the pattern 'aka', the share of word vectors available was $< 10\%$ overall. So, in effect, only one of the false positive nodes got the true negative label, after the third step is performed. Thus the recall remains the same after both the steps.

Pattern	System	P	R	A
a	MAD	0.72	0.77	73.86
	MADB2	0.68	0.68	68.18
	MADB1	0.49	0.52	48.86
	LP	0.55	0.59	55.68
aka	MAD	0.77	0.67	73.33
	MADB2	0.71	0.67	70
	MADB1	0.43	0.4	43.33
	LP	0.75	0.6	70
in	MAD	0.74	0.82	76.47
	MADB2	0.67	0.70	67.65
	MADB1	0.51	0.56	51.47
	LP	0.63	0.65	63.23
ya	MAD	0.7	0.72	70.31
	MADB2	0.61	0.62	60.94
	MADB1	0.53	0.59	53.12
	LP	0.56	0.63	56.25
i	MAD	0.55	0.52	54.76
	MADB2	0.44	0.38	45.24
	MADB1	0.3	0.29	30.95
	LP	0.37	0.33	38.09

Table 5: Comparative performance of the four competing models.

In Label Propagation, we experimented with the parameter K with different values, $K \in \{10, 20, 30, 40, 50, 60\}$, and found that $K = 40$, provides the best results for 3 of the 5 end-patterns. We find that for those 3 patterns ('a','in','i'), the entire vertex set has *vṛddhi* attribute set to the same value. For the other two ('ya','aka'), $K = 50$ gave the best results. Here, the vertex set has nodes where the *vṛddhi* attribute is set to either of the values. We report the best result for each of the system in Table 5.

4.4 Evaluation for Unlabeled Nodes

In order to evaluate the effectiveness of our system, we pick nodes from unlabelled set $\mathcal{U}$ and evaluate the word-pairs based on human evaluation. We take top 5 unlabeled nodes predicted as *taddhita* and top 3 unlabelled nodes predicted as not *taddhita* from each of the the 11 end-patterns. We collate the predictions and divide them into 3 lists of 22 entries each, as the remaining 22 of

the original 88 were filtered out. Seven experts, with background in Sanskrit linguistics labelled the dataset, of which one of the expert evaluator is an author. We divide the set of 66 nodes into 3 mutually disjoint sets, and each set is evaluated by 3 experts. We altogether receive 9 impressions of which the author evaluator and one of the other expert evaluator performed 2 impressions each. In case of a conflict, we go with the majority votes for each of the set. Since the entries are selected from the top scoring nodes, we expected the results to be better than the macro-average performance of the system. We find that the evaluation of our system provides a precision of 0.84, recall of 0.91 and an accuracy 81.82 micro averaged over the 66 predictions.

5 Related Work

Computational analysis of derivational word forms is gaining some traction in the NLP community. Lazaridou et al. (2013) used CDSM (Mitchell and Lapata, 2010) for derivational nouns, originally designed to learn representation for phrases. Cotterell and Schütze (2017), extended the concept of CDSM for derivational word forms with neural models. The authors put forward the idea of jointly handling the segmentation of words into morphemes and semantic synthesis of the word forms to improve the performance of a system for both the tasks. Bhatia et al. (2016), does not make a distinction of inflected word-forms or derivational affixes, but their work can be employed to learn embeddings for a word-form from its morphemes.

Soricut and Och (2015) introduced an unsupervised method of inducing affixal transformations between words using word embeddings. Faruqui et al. (2016) further propose a semi supervised graph based approach for morpho-syntactic lexicon induction. The authors show the effectiveness of their model for inflectional morphology over multiple languages. In Sanskrit, Krishna and Goyal (2015) automated the derivation of Taddhita, where the authors follow an object oriented framework. Deo (2007) have preformed an in depth linguistic analysis of inheritance network used by Pāṇini in handling affixation in Taddhita.

6 Discussion

In Sanskrit, multiple affixes may give rise to similar patterns. In fact, an affix in Sanskrit contains two parts, where one part pertains to the pattern to be induced, and other is a marker which gets elided before the affixation. The presence of the marker, termed as '*it*' marker, also plays a role in determining the type of rules that get triggered during the derivation. For example, consider the word '*prāmukhya*' derived from '*pramukha*' and the word '*sodarya*' from '*sodara*'. Both the words have the same end-pattern '*ya*'. However, only in the case of the former, *vṛddhi* operation takes place but not in the latter. Now, affixes that carry the same pattern might differ by the '*it*' markers. Now, by encoding every candidate word pairs with the suitability of rules of *Aṣṭādhyāyī* in $\mathcal{A}_1$, we can narrow down the possible candidates for the affix to at most 4 candidates of the 137 possible affixes. In order to disambiguate further, we require semantic and pragmatic level information, which is currently unavailable. In this work, we only consider the derivations in *taddhita*, as we find that jointly modelling a system for both *kṛdanta* and *taddhita* is challenging. The rule arrangement for *kṛdanta* is different from that of *taddhita* in *Aṣṭādhyāyī*, thus we require a different model design for organising the rules in $\mathcal{A}_1$, i.e., the phase 1 in Section 3. Hence, in this work we restrict ourselves to resolving *taddhita* nouns, which is the larger section in *Aṣṭādhyāyī* among the two.

7 Conclusion

In this work, we developed a graph based semi supervised approach for analysis of derivative nouns in Sanskrit. We successfully integrate the rules from *Aṣṭādhyāyī*, variable length character n-grams learnt from Adaptor grammar and word embeddings to build a 3 step sequential pipeline for the task. We find that our work outperforms label propagation, which primarily shows the effect of explicit design of network structure. We find that using the label distribution outputs at each phase, for the input at the successive phases improve the results of the model. Our work will be beneficial to the Sanskrit Computational Linguistic community for analysis of derivational words in the digitised ancient manuscripts, as no other analyser in Sanskrit currently handles derivational nouns. Our work doubles as a tool for pedagogy, as we are able to abstract out regularities between the patterns and narrow down the possible affix candidates for a word pair to four.

References

Saroja Bhate. 1989. *Panini's taddhita rules*. University of Poona.

Parminder Bhatia, Robert Guthrie, and Jacob Eisenstein. 2016. Morphological priors for probabilistic neural word embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 490–500. https://aclweb.org/anthology/D16-1047.

Ryan Cotterell and Hinrich Schütze. 2017. Joint semantic synthesis and morphological analysis of the derived word. *Transactions of the Association for Computational Linguistics* .

Ashwini Deo. 2007. Derivational morphology in inheritance-based lexica: Insights from pāini. *Lingua* 117(1):175–201.

Manaal Faruqui, Ryan McDonald, and Radu Soricut. 2016. Morpho-syntactic lexicon generation using graph-based semi-supervised learning. *Transactions of the Association for Computational Linguistics* 4:1–16.

Pawan Goyal and Gérard Huet. 2016. Design and analysis of a lean interface for sanskrit corpus annotation. *Journal of Language Modelling* 4(2):145–182.

Pawan Goyal, Gérard P Huet, Amba P Kulkarni, Peter M Scharf, and Ralph Bunker. 2012. A distributed platform for sanskrit processing. In *COLING*. pages 1011–1028.

Mark Johnson, Thomas L Griffiths, Sharon Goldwater, et al. 2007. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. *Advances in neural information processing systems* 19:641.

Paul Kiparsky. 1994. Paninian linguistics. *The Encyclopedia of Language and Linguistics* 6:2918–2923.

Donald E Knuth. 1992. Two notes on notation. *The American Mathematical Monthly* 99(5):403–422.

Amrith Krishna and Pawan Goyal. 2015. Towards automating the generation of derivative nouns in sanskrit by simulating panini. *arXiv preprint arXiv:1512.05670* .

Amrith Krishna, Pavankumar Satuluri, Shubham Sharma, Apurv Kumar, and Pawan Goyal. 2016. Compound type identification in sanskrit: What roles do the corpus and grammar play? In *Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 1–10. http://aclweb.org/anthology/W16-3701.

Malhar Kulkarni, Chaitali Dangarikar, Irawati Kulkarni, Abhishek Nanda, and Pushpak Bhattacharyya. 2010. Introducing sanskrit wordnet. In *Proceedings on the 5th Global Wordnet Conference (GWC 2010), Narosa, Mumbai*. pages 287–294.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositional-ly derived representations of morphologically complex words in distributional semantics. In *ACL (1)*. Citeseer, pages 1517–1526.

Hans Marchand. 1969. *The categories and types of present-day English word-formation: A synchronic-diachronic approach*. Beck.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive science* 34(8):1388–1429.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

P Scharf. 2009. Rule selection in the a ādhyā yi or is pāinis grammar mechanistic. In *Proceedings of the 14th World Sanskrit Conference, Kyoto University, Kyoto*.

Radu Soricut and Franz Josef Och. 2015. Unsupervised morphology induction using word embeddings. In *HLT-NAACL*. pages 1627–1637.

Partha Talukdar and Koby Crammer. 2009. New regularized algorithms for transductive learning. *Machine Learning and Knowledge Discovery in Databases* pages 442–457.

Ke Zhai, Jordan Boyd-Graber, and Shay B Cohen. 2014. Online adaptor grammars with hybrid inference. *Transactions of the Association for Computational Linguistics* 2:465–476.

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2003. Learning with local and global consistency. In *NIPS*. volume 16, pages 321–328.

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation .