Supplementary Materials for "Free as in Free Word Order: An Energy Based Model for Word Segmentation and Morphological Tagging in Sanskrit"

Amrith Krishna[1], Bishal Santra[1], Sasi Prasanth Bandaru[2], Gaurav Sahu[3]
Vishnu Dutt Sharma[4], Pavankumar Satuluri[5] and Pawan Goyal[1]
[1]Dept. of Computer Science and Engineering, IIT Kharagpur
[2]Flipkart, India [3]School of Computer Science, University of Waterloo
[4]American Express India Pvt Ltd [5]Chinmaya Vishwavidyapeeth
amrith@iitkgp.ac.in, bsantraigi@gmail.com,
pawang@cse.iitkgp.ernet.in

The code and the edge vectors are uploaded in https://zenodo.org/record/1035413#.W35s8hjhUUs.

1 Sentence Graph Construction

Sandhi: The phonemes at the word boundaries are often merged using phonetic transformations called as "*sandhi*". "*sandhi*" is primarily an outcome of the euphonic assimilation in speech, that gets reflected in writing as well (Goyal and Huet, 2016). The proximity between phonemes is the sole criteria for applying *sandhi*

The analysis of a construction with sandhi in it can often lead to a word splits with possible gaps and overlaps between them. In Figure 1b and 1c we can find the word splits have overlaps between them. This happens as the multiple phonemes join together to form a single phoneme at the time of sandhi. Rarely, instance like Figure 2d occurs. Here the sandhi led to generation of an additional phoneme 'n' it it. Figure 1a is a case where the word splits do not have overlap with the sandhied version. At the same time, the phonetic change in the sandhied version can be observed.

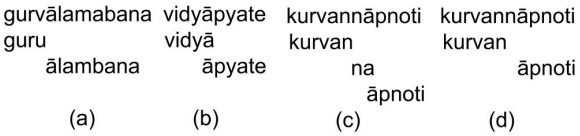

Figure 1: Instances of sandhi formation in Sanskrit. a) Phonetic transformation of '*u*' and '*ā*' to '*v*' in the joint form. b) '*ā* ' and '*ī*' at the word boundaries join together to form a single '*e*' in the final form, resulting in both the split words having an overlap at the juncture (Goyal and Huet, 2016). c) and d) Two possible analysis for the sandhied chunk '*kurvannāpnoti*

Figure 2 shows the possible segments and the desired segmentation solution for a sentence, '*satyaṃbrūyātpriyaṃbrūyānnabrūyātsatyamapriyaṃpriyaṃcanānṛtambrūyādeṣadharmaḥsanātanaḥ*'. The Sanskrit Heritage Reader essentially shows all the unique segments that are part of at least one segmentation solution. For example, in Figure 2, the word '*satī*', numbered as 9, is part of 264 out of possible 1056 segmentation solutions. We call such a representation of the segments as a shared forest representation.

Word order in a sentence Kulkarni et al. (2015) perform an analysis over the extent to which free word ordering is allowed in Sanskrit. The authors identifies that constructions in Sanskrit follow weak non-projectivity. But, their analysis reveals that constructions in poetry violates this especially for adjectival and genitive relations. In pedagogical systems, the poetry is converted to corresponding prose follows the regularities mentioned in Kulkarni et al. (2015). For example the prose order for the sentence given above is 'satyam brūyāt priyam brūyāt , satyam apriyam na brūyāt. priyam ca na anṛtam brūyāt , eṣa sanātanaḥ dharmaḥ'. The order of adjectives and modifiers are violated in poetry as it can be seen from the last 3 words of the prose. We use the Digital Corpus of Sanskrit for our experiments where the constructions primarily are written in poetry format.

Lexical juncture System The Sanskrit Heritage Reader uses finite state methods in the form of a lexical juncture system to obtain the possible segments for a given sentence. We follow the definitions as used in Goyal and Huet (2016) and it is recommended to refer the work for a detailed review of the system.

A *lexical juncture system* on a finite alphabet Σ is composed of a finite set of words $L \subseteq \Sigma^*$ and a finite set R of rewrite rules of the form $u|v \rightarrow f/x\_\_$ (Kaplan and Kay, 1994), with $x, v, f \in \Sigma^*$ and $u \in \Sigma^+$. In this formalisation, Σ is the set of

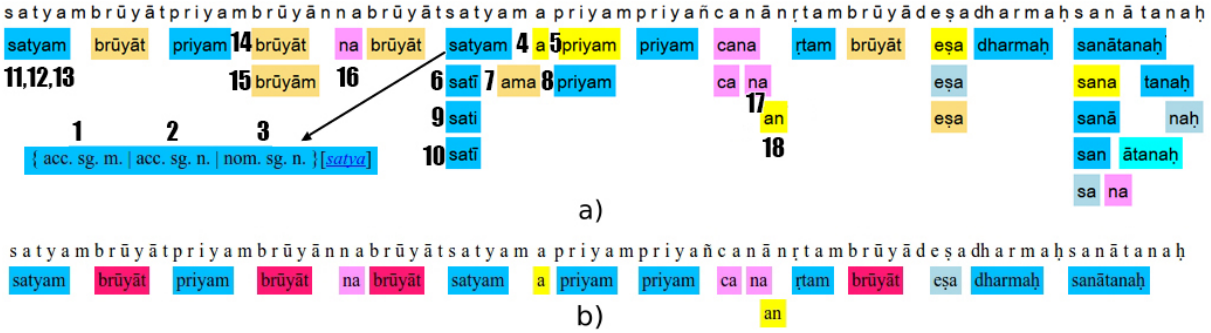

Figure 2: a) All the phonetically valid segmentations for '*satyaṃbrūyātpriyaṃbrūyānna-brūyātsatyamapriyaṃpriyaṃcanānṛtambrūyādeṣadharmaḥsanātanaḥ*' from (*subhāṣitam*) as output by Sanskrit Heritage Reader (SHR) and b) correct segmentation selected from the candidate space.

phonemes, R is the set of sandhi rules, and L is the vocabulary as a set of lexical items. Though every entry $z \in L$ is an inflected word form, it additionally contains the morphological analysis of the word as well. For clarity, we will denote every entry z additionally as a 3-tuple (l, m, w), where l denotes the lemma of the word, m denotes the morphological class of the word, w denotes the inflected word form generated from l and m. Given a sentence s, a valid segmentation solution/sandhi analysis, S_i, can be seen as a sequence $\langle z_1, \sigma_1, k_1\rangle; ...\langle z_p, \sigma_p, k_p\rangle$. Here, $\langle z_j, \sigma_j, k_j\rangle$ is a segment with $z_j \in L$, $k_j \in \mathbb{N}$ denotes the position at which the word w_j begins in the sentence s and $\sigma_j = [x_j]u_j|v_j \rightarrow f_j \in R$ for $(1 \leq j \leq p)$.

For s, there can be multiple possible *sandhi* analyses. Let $\mathcal{S}$ be the set of all such possible analyses for s. We find a shared forest representation of all such *sandhi* analyses

$$D(\mathcal{S}) = \bigcup_{S_i \in \mathcal{S}} S_i$$

A segment $\langle z_j, \sigma_j, k_j\rangle \in D(\mathcal{S})$, iff $\langle z_j, \sigma_j, k_j\rangle$ exists in at least one S_i. Two segments $\langle z_i, \sigma_i, k_i\rangle$ and $\langle z_j, \sigma_j, k_j\rangle$ are said to be 'conflicting' if $k_i \leq k_j < k_i + |w_i| - 1$ or $k_j \leq k_i < k_j + |w_j| - 1$. No two conflicting segments exist in a valid segmentation solution.

Converting the candidate space into graph

We convert the shared forest representation of the candidate segments into a sentence graph $G(V, E)$. For the graph G, a node $v \in V$ is a segment $\langle z_j, \sigma_j, k_j\rangle$, where z_j is a 3-tuple (l, m, w). This representation scheme for the segment is indispensable for the task. For example, in Figure 2a, the nodes 1, 2 and 3 differ from each other only based on the morphological tag they carry. It is represented using the m attribute of the 3-tuple (l, m, w). Similarly nodes 1 and 11 differ from each other only based on their position information represented by k_j in the segment.

Figure 3: A subsequence of sentence from Figure 2

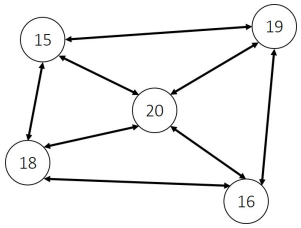

Figure 4: Sentence Graph constructed for the subsequence in Fig. 3. Edges are bidirectional. For better illustration of construction details, node 17 is omitted.

The proximity of words to one another is not suggestive of the syntactic or semantic compatibility between the nodes. Hence, we form an edge between every pair of non-conflicting nodes. There exists an edge $e \in E$ between every pair of vertices $v_i, v_j \in V$, provided v_i, v_j are not '*conflicting*' with each other, and thus, can potentially be part of the same segmentation solution. Using the character sequence shown in Figure 3, which is a sub-sequence taken from the sentence in Figure 2a, we construct its corresponding sentence graph as shown in Figure 4. In Figure 4, node 15 has edges to all other nodes except the nodes 16 and

17 (node 17 not shown in the figure). The three nodes share the same lemma and inflected word-form but differ by their morphological class. Similarly, nodes 18 and 19 are conflicting as they overlap in their input text positions and are suggested as alternatives.

2 Inference Procedure

Visualisation of Inference in Clique-EBM Figure 5 shows our approach for obtaining one maximal clique. Here we show the clique selection approach for the input sequence in Figure 3. The construction approach of sentence graph ensures that we always receive a clique. By this method, we find a maximal clique starting from each of the node $v_i \in V$ We obtain the score for a clique by factoring it as the sum of the scores of its edges.

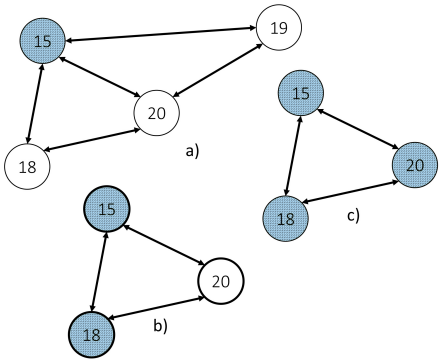

Figure 5: Maximal clique selection for the input sequence in Figure 3. The nodes in V_{T_i} at each step are shown as coloured nodes.

Inference for Tree-EBM For Tree-EBM, like Takahashi (1980), we use the Prim's algorithm as our search procedure for finding the relevant candidate nodes. The inference procedure identifies a tree that spans over a subset of nodes, such that the subset of ndoes forms a phonetically valid solution. For every node $v_i \in V$, we perform the following:

1. Initialize a tree $T_i(V_{T_i}, E_{T_i})$ with v_i as the only vertex and remove all the vertexes in V which are conflicting with v_i.
2. Add the vertex $v_j \in V - V_{T_i}$ to V_{T_i}, where v_j forms the minimum weighted directed edge with any of the vertexes in the V_{T_i}.
3. Remove v_j and all the vertexes which are conflicting with v_j from V.
4. Repeat Steps 2 - 3 till $V = \varnothing$

3 Edge vector creation

Once the sentence graph is constructed, what remains is to obtain feature vectors for the edges. In this section we discuss how we use a corpus to automatically obtain feature vectors for edges in the sentence graph G. We use the Path Ranking Algorithm (PRA) (Lao and Cohen, 2010) to automate the edge vector generation process. Here, we will use the terms Path Ranking Algorithm and Path Constrained Random Walks interchangeably (PCRW). Although, PCRW is essentially random walks performed over metapaths filtered using PRA.

3.1 Constructing the Corpus Graph

Consider a large corpus of Sanskrit denoted as $\mathcal{C}$. A sentence in our corpus essentially contains the segmented words and their corresponding morphological analysis. We construct a graph $G_{glob}(V_{glob}, E_{glob})$ from the corpus which accommodates not just the words but also their morphological analysis. Essentially, the nodes in the graph are of heterogeneous types. As Sanskrit is morphologically rich and is a low resource language, the use of lemma as a type can also be seen as a means of combating sparsity of data. It is often not possible to obtain enough distributional evidence for an inflected word-form independently. Further, when a rare or an out of vocabulary word appears, the nodes indicative of morphological information can serve as a back-off mechanism.

Preliminaries: Heterogeneous Information Networks The typed network is better known as Heterogeneous Information Networks in the Information Networks community. Here, we follow the terminology prevalent in the community (Meng et al., 2015). G_{glob} is represented as a large Heterogeneous Information Network (HIN) $G_{glob}(V_{glob}, E_{glob})$. A HIN is a network where the nodes or edges can be of different types. Shi et al. (2017) defines an HIN as a directed graph $G^{'} = (V^{'}, E^{'})$, with an node type mapping function $\phi : V^{'} \rightarrow \mathcal{A}$ and a link type mapping function $\psi : E^{'} \rightarrow \mathcal{R}$, where $|\mathcal{A}| > 1$ or $|\mathcal{R}| > 1$. Each node $v' \in V'$ belongs to one of the particular types from the node type set $\mathcal{A} : \phi(v^{'}) \in \mathcal{A}$, and similarly for each edge $e^{'} \in E^{'}$ in relation type set $\mathcal{R} : \psi(e^{'}) \in \mathcal{R}$. In our setting, we consider only nodes to be of heterogeneous type, while all the edges are of homogeneous type.

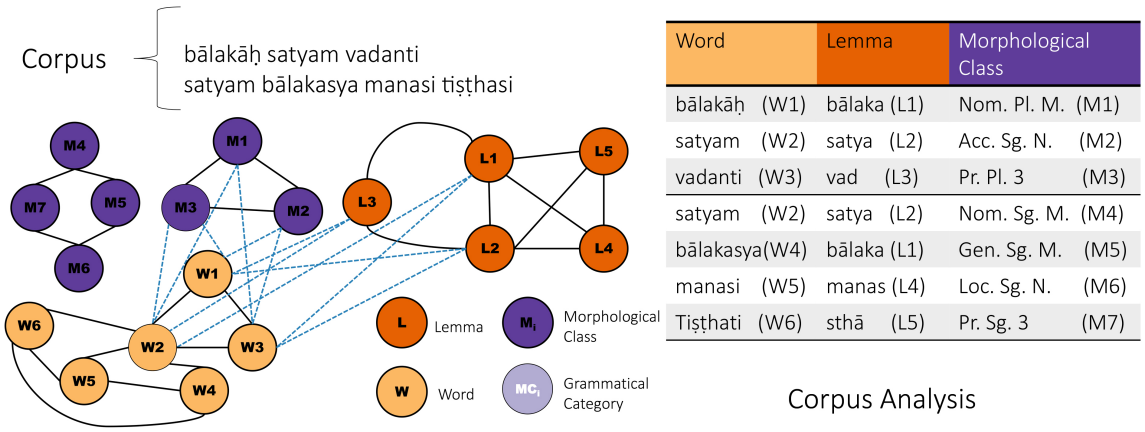

Figure 6: Construction of Corpus Graph G_{glob} for a given Corpus. The edges are bidirectional. For better clarity, we only show partially constructed corpus. We omit the nodes of the Morphological Group type. The edges between nodes of Lemma and Morphological Class types and the edges between all the inter-type nodes for Sentence 2 are not shown in the figure. The inter-type edges are shown in dotted format. But all the edges are homogeneous and represent co-occurrence between the nodes in a sentence.

Graph Creation Co-occurrence of two items in a sentence is the sole criteria for edge formation in G_{glob}. The type of the node does not matter here. The edges in G_{glob} are directed. Hence for every pair of nodes that have a co-occurrence in the corpus, there exist two directed edges. Given two nodes $v_g, v_h \in V_{glob}$, the weight for the edge from v_g to v_h, $P_{co}(v_h|v_g)$ is calculated as

$$P_{co}(v_h|v_g) = \frac{count(v_g, v_h)}{count(v_g)} \quad (1)$$

Here $count(v_g, v_h)$ represents the count of sentences in which both v_g and v_h co-occur. Similarly, $count(v_g)$ represents the count of sentences in which the node v_g occurs.

Figure 6 shows the construction of our corpus graph from a corpus of 2 sentences. The corresponding morphological analysis for the sentences is also shown in the figure. The HIN G_{glob} has four object types, namely, the lemma, word, morphological class and grammatical category. The word type corresponds to all the unique inflected wordforms that exist in the corpus, while the lemma type corresponds to all the unique lemmas in the corpus. The morphological class corresponds to the exact morphological tag a word form takes in a sentence.

In Sanskrit, an inflection is indicative of multiple grammatical categories i.e. a morphological tag generally represents more than one grammatical categories. We abstract this and form a node type to capture the category specific data. Grammatical category type is an abstraction over morphological class, where we form sets of morphological classes that share one or more grammatical categories.

Obtaining Grammatical Category From Morphological Tags To be specific, the morphological class of a word in Sanskrit is indicative of the gender, number and declension/tense it carries, depending on whether the word is a noun or verb (Scharf, 2009). For example, if we consider a noun, '*rāmasya*' (of Rāma), it denotes the genitive case, masculine gender and singular inflection of the noun '*rāma*'. However, only a specific grammatical category (e.g., 'genitive' case implying possessiveness) might be relevant as a feature for the task while the other categories (masculine, singular) may result from the morphological agreement requirements from the words. To address this, we form a node of type 'grammatical category', in this case a node representative of the 'genitive' case. This node is essentially the union of all the genitive case morphological classes and represents the occurrence of any genitive class entity in the corpus. Similarly 'genitive singular' will be another node of the same type where we take union of all the three genitive singular nodes which vary only based on the gender.

For our HIN G_{glob}, we thus have object type set $|\mathcal{A}| = 4$ and the relation type set $|\mathcal{R}| = 1$. Table 1 shows the number of nodes for each of the type.

Type	No. of nodes
Grammatical Category	310
Lemma	66,914
Word	217,535
Morphological Class	218

Table 1: Node types and the number of corresponding nodes in G_{glob}

3.2 Metapaths

The corpus graph constructed represents the co-occurrence information between the nodes as obtained from the corpus. However, considering co-occurrence as the sole criteria can be noisy. For example, the adjective of the subject in a sentence may not be of any relevance to the object in the sentence. We need to identify refined contexts that can be potentially beneficial for the task. We identify typed paths that provide connectivity between nodes which will provide us with discriminative contexts. So, while existence of all the paths is equally valid in the corpus graph, some of the paths are more valid for the task. Once a set of typed paths is defined, we look for connectivity between node pairs only via the typed paths and other sources of adjacency are ignored.

The typed paths that we use here are formally called as metapaths in the Information Networks literature. A metapath $\mathcal{MP}$ is a path defined on the schema $(\mathcal{A}, \mathcal{R})$ and is denoted in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} ...A_l \xrightarrow{R_l} A_{l+1}$, which defines a composite relation $R = R_1 \circ R_2 \circ ... \circ R_l$ between A_1 and A_{l+1}, where $A_l \in \mathcal{A}$ and $R_l \in \mathcal{R}$ (Shi et al., 2017). In our case, metapaths are sequences of object types as our edge relations are of the same type.

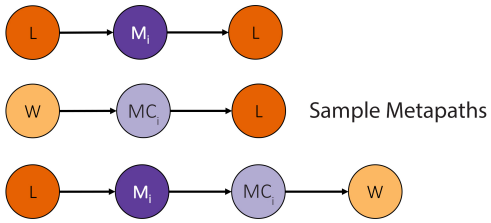

Figure 7: Sample metapaths from the HIN G_{glob}. The paths can be of lengths 1,2 or 3. The internal nodes use the nodes rather than the type in our setting.

Constraints on the Metapath Formation We consider metapaths of length 1, 2 and 3 (i.e., the number of edges). The external nodes, i.e., nodes at both ends of the metapaths can only be from 3 of the 4 node types, namely, lemma, word and morphological class. The internal nodes in the metapaths can belong to only 2 of the 4 node types, namely, grammatical category and morphological class. Though, traditionally, each unique sequence of node types is considered to be a separate metapath, we make a deviation from the standard definition of a metapath. For the internal nodes, we consider every unique node rather than the node type to construct the metapath. Figure 7 show some sample metapaths. The internal nodes are sub-scripted to show the use of nodes (different instances of the type) and not the types. The linguistic motivation for the deviation is quite intuitive as we need to capture the distinctive syntactical patterns prevalent in the corpus via different metapaths. Please note that when the type "morphological class" is used as an external node, only the type information is used.

3.3 Automatic Metapath Generation and Filtering

We use the Path Ranking Algorithm (PRA) (Lao and Cohen, 2010) for metapath enumeration and subsequent filtering. From any given HIN, the first step of PRA is to identify potential metapaths beneficial for a task using a supervised feature selection approach. This is a one time process. The second step in PRA is performed at run-time (Gardner and Mitchell, 2015). The first step requires only the HIN.

As with the standard procedure for PCRW, we enumerate all the possible paths $\mathcal{MP}$, upto a length of 3. From the corpus, we obtain a set of node pairs which acts as the training dataset for metapath filtering. Now for a given node pair, each of the metapath acts as a template for obtaining a path that connects the nodes. The path so formed should satisfy the metapath constraints. Now, the path so obtained will have a non-negative real value score, which is a function of the edge weights of the edges that forms the path. Thus each metapath acts as a feature for the node pair.

Then we use a supervised feature selection approach for identifying a fixed set of metapaths $\mathcal{MP}_{final}$ from $\mathcal{MP}$. The path enumeration and filtering is a one time process. We experiment with 8 different metapath filtering approaches. We first limit the maximum possible path length of the metapaths to be at values 2 and then 3. This setting enumerated about 4800 and 800,000 metapaths respectively, having sufficient evidence in the corpus. We kept a threshold that a metapath should exist between at least 10 different pair of nodes. For filtering the paths we use two different feature selection approaches. One method is recursive feature elimination (RFE) with linear regression as the estimator and the other is Mutual Information Regression (MIR) (Kraskov et al., 2004).

The estimators use regression approaches and the training data require labels for feature selec-

tion. We experimented with point-wise mutual information (PMI) in one setting and co-occurrence probability, henceforth to be referred to as bigram. We can see this as a bigram due to the directionality of the word pairs involved. But, 'bigram' is also part of one of the metapaths we generate and hence we remove it from our feature set, when used as a label. While performing feature selection for the set with all the metapaths up-to length 3, we initially used a correlation based filtering to reduce the feature space to 10,000 metapaths as it was costly to perform the aforementioned supervised regression tasks on 800,000 features. For each feature, we find its correlation with the label and experimented with different sizes at which the filtering process was stopped. For the filtering of metapaths by the estimators, we use a training set of 10,000 word pairs from the corpus. The training set was obtained by stratified sampling based on frequency of word co-occurrence. In Lao and Cohen (2010), the authors used a logistic regression classifier setting for the selection of metapaths, while we use a regression setting.

Generating PCRW Vectors for the edges in the sentence graph: For each edge in the sentence graph G, we obtain a vector where each component of it is a non-negative real valued score. This forms the second step of PRA where the path scores for the node pairs via each of the filtered metapath from $\mathcal{MP}_{final}$ are calculated.

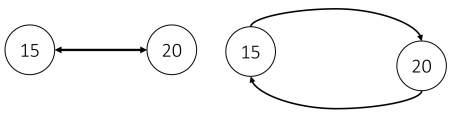

Figure 8: A pair of nodes from the sentence graph G for input sequence in Figure 3. Every node pair has 2 directed edges.

Figure 9: Metapath based feature vector for any directed edge in the sentence graph G

For a given node pair in G, the product of edge weights for each path is calculated as the path weight. Then, the negative logarithm of the values so obtained becomes the feature value. Given the nodes $v_h, v_g, v_k \in V_{glob}$, where v_h, v_g are the external nodes and v_k is an internal node, the values for paths of length 1 and 2 (this can be generalised to length 3 as well) respectively are

System	P	R	F
EdgeGraphCRF	77.6	79.68	78.63
supervisedPCRW	73.28	82.73	77.72
Tree-EBM	88.7	91.25	89.96
Clique-EBM	94.41	96.31	95.35

Table 2: Performance evaluation of the competing systems for the word-prediction task. P is precision, R is recall, and F is the F-Score.

$$S(Path_{v_h,v_g}) = -log_{10}P_{co}(v_g|v_h) \quad (2)$$

$$S(Path_{v_g,v_k,v_h}) = -log_{10}(P_{co}(v_h|v_k)\cdot P_{co}(v_k|v_g)) \quad (3)$$

Figure 10: Sample scores obtained for each metapath for a given directed edge in the sentence graph G.

Figure 10 shows the sample metapath score obtained for a specific edge in the sentence graph G. All the values obtained will be non-negative real valued numbers.

4 Results

Word prediction task results of all models on DCS10K: Table 2 provides the best results for word prediction in terms of macro averaged precision, recall and F-score for each of the system when tested on 'DCS10K'.

Hyper-Parameters: We vary our PCRW feature vector size from 400 to 2000 in steps of 100. A vector size of 1500 gives the best results for our system. The neural network in our system by default has 1 hidden layer for learning the energy function. Adding additional hidden layers to the neural network did not improve over the best results obtained. Marginal improvement was observed by increasing the hidden layer size from 500 to 800 and finally to 1200 which was the final setting for our model. We used dropout in the neural network with a dropout probability of 0.4 for both the input layer and hidden layer, after experimenting with different settings. We experiment with different loss functions as discussed in LeCun et al. (2006). The standard Hinge loss, with the fixed margin replaced with squared Hamming distance margin function for trees, was finalized after comparing the performance of system

PCRW Vector Size	1500
Hidden Layer size	1200
Learning Rate	$\alpha = 1 \times 10^{-5}$
Margin Loss Discount	$\nu = 1$
Dropout Rate	$p = 0.4$

Table 3: Hyper-parameter settings for Clique-EBM

with square-exponential loss and Log loss function. The reason for better performance of the modified Hinge loss may be due to taking into account the structural similarity of the system output and gold standard clique.

Wall time analysis of our model: Our architecture primarily requires 2 important tasks. One is the generation of PCRW vectors and the other is the search for the clique using our energy based model. For the former, we use the 2-Step PRA algorithm. The first step is an offline one time process, and takes about 34.73s to generate all the 800,000 path scores for a word pair. We had 10,000 such word pairs during the filtering. But we used 40 threads to parallelise the path score generation. The whole process took less than 3 hours including the overhead involved in the threads.The correlation based approach to identify 10,000 metapaths from the original set took 2 hours to complete with 12 threads. For feature selection, RFE took 395 minutes to filter 1500 metapaths while MIR took 220 minutes to complete.

Now at runtime, it takes approximately 66 ms to generate a feature vector with 1500 metapaths for an edge. On an average, the input graph contains about 1178 edges. So, in addition to time for generating the vectors, a graph with 1,178 edges takes about 1.94 seconds for training. For testing, a similar graph would take 0.69s.

Our overall training was completed in 10.5 hours and the testing on DCS10k was completed in 2 hours. All our computations are performed on an Intel Xeon machine with 48 threads (2 CPU x 12 cores x 2 threads) and 256 GB RAM.

Morphological class specific assessment: Table 6 shows top 5 such instances where the mispredictions are skewed towards a particular morphological class for Tree-EBM. We find that the corresponding morphological classes for Nominative case and Accusative case often do get confused by the system. We suspect, this is due to the presence of sentences in active and passive voices where a word which takes in the role of object in the sentence appears in accusative and nominative cases respectively. The rules 2.3.1 and 2.3.2 of *Aṣṭādhyāyī*, the grammar treatise in Sanskrit, describes the phenomenon (Kiparsky, 2009; Sharma, 1987). So it is possible that the same set of lemmas might have been frequently appearing in both the morphological classes. Similarly the masculine and neuter gender nouns for the instrumental, ablative and genitive cases also gets confused by the system. Interestingly it can be observed that these morphological classes often have similar inflections across different lemmas. The final column in the Table 6 presents the share of lemmas having the same final form for the correspodning classes from a sample of 200,000 sentences from DCS. Table 7 shows the mispredictions for the same pair of morphological classes in Clique-EBM. In Clique-EBM we couldn't find any such regularities in the mispredictions. We hypothesise that for incorporating knowledge of dependency structure will be beneficial to reduce the mispredictions further.

We find similar trends for the Lattice-EBM models and the EdgeGraphCRF model as well. We find that majority of the mispredictions happen across those same surface form with same but different morphological tags. The share of arbitrary mispredictions in EdgeGraphCRF were found to be more than the other systems. Empirically we observe that not considering pairwise potentials (between multiple words and at least of nearby words) between other words in the prediction will affect the performance of such systems adversely

Effect of PCRW Vectors: We study the effect of different sets of PCRW vectors generated when

Max Path Length	Feature Selection	Label	P	R	F
2	MIR	PMI	92.81	95.19	93.98
3	MIR	PMI	82.12	86.08	84.05
2	MIR	Bigram	**94.41**	**96.31**	**95.35**
3	MIR	Bigram	93.57	95.47	94.51
2	RFE	PMI	80.85	87.61	84.09
3	RFE	PMI	84.74	86.86	85.79
2	RFE	Bigram	83.59	88.95	86.19
3	RFE	Bigram	85.94	89.31	87.59

Table 4: Effect of PCRW Vector sets on the proposed model (Word Prediction)

Type	Word Recall	Word++ Recall
Noun	96.869	88.998
Verb	95.911	94.424
Compound	93.518	91.067
Indeclinable	97.095	96.472

Table 5: System performance on the coarse level POS

Original Morph	Mispredictions (in %)	Morph Confused to (proportion in mispredictions)	Count in DCS10K	lemmas with same inflection for both the classes in corpus sample (in %)
Nominative Plural Neuter	13.18	Accusative Plural neuter (90.24)	311	99.78
Accusative Singular Neuter	22.77	Nominative Singular Neuter (85.56)	2494	99.90
Accusative Plural Neuter	13.2	Nominative Plural neuter (82.5)	303	99.78
Instrumental Plural Neuter	7.58	Instrumental Plural Masculine (80.0)	266	99.31
Nominative Dual Masculine	9.02	Accusative Dual Masculine (70.83)	266	99.96

Table 6: Top 5 entries from Tree-EBM based on the skewedness towards misprediction to a single morphological class (column 3 parenthesis provides the % of the mispredictions to that morphological class).

Original Morph	Mispredictions (in %)	Morph Confused to (proportion in mispredictions)
Nominative Plural Neuter	3.85	Accusative Plural neuter (25)
Accusative Singular Neuter	12.44	Nominative Singular Neuter (50.62)
Accusative Plural Neuter	6.67	Nominative Plural neuter (52.63)
Instrumental Plural Neuter	3.42	Instrumental Plural Masculine (0.0)
Nominative Dual Masculine	3.76	Accusative Dual Masculine (50.0)

Table 7: Results from Clique-EBM for the pairs of morphological classes shown in Table 6

input to the model. The vector sets were generated based on the settings discussed in Section 3.3 Table 4 reports the Precision, Recall and F-score for all the eight different sets of PCRW vectors generated. We find that the best performing set is 2-MIR-Bigram with an F-Score 95.35. It can be observed that all the 2 path sets perform better than the corresponding 3 path sets. 3-MIR-PMI scores higher in the average precision than its corresponding 2 path vector set, but the 2 path set still scores higher when it comes to the F-Score.

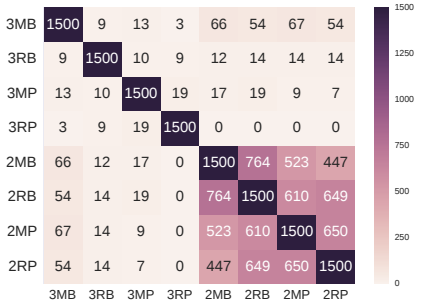

Figure 11: PCRW Feature sets - Pairwise comparison of common features in the set

We find that the features returned by each of the set is quite diverse as the union of all the feature sets has a cardinality of 9055. Figure 11 shows the heatmap for pairwise overlap between different feature vector sets. The best performing set, 2-MIR-Bigram has 8 of the 9 possible metapaths of length 1, i.e., features signifying co-occurrence between 2 nodes without any internal nodes. The remaining one, i.e., of the type word - word co-occurrence is used as the label for feature selection. Interestingly, no other set has all the metapaths of length 1 in them. In 2-MIR-Bigram, metapaths which end with 'lemma' object type are more discriminating than others as 953 of top 1000 paths end with 'lemma' as its object type. In fact all of the top 100 paths end with a lemma node, while 76 start with lemma node as well.

Clique Enumeration Heuristic Unlike other works such as McDonald et al. (2005), where Bron and Kerbosch (1973) was used for clique enumeration, we cannot use it as we cannot prune the edges in the graph. Our heuristic results in a speedup of 2.74 as compared to the Bron Kerbosch (Bron and Kerbosch, 1973) for the enumeration of cliques. Also, our approach can easily be parallelised resulting in further speedups. Our approach guarantees that at least one clique will be obtained for a graph. Also, Both the conditions are important for our requirement of finding subset of nodes. Our sampling heuristic samples only about 0.57 % of the total possible cliques on an average and yet empirically the results are promising. Our choice of energy based model do not require the noramlisation of the solution space, a choice that enables the use of the heuristic. All our reported values are based on 3300 sentences sampled by stratification from the training set.

4.1 Fine Grained Performance Analysis of Clique-EBM

We analyse the performance of Clique-EBM based on its prediction on DCS10k. Figures 12 and 13 show the point plots for those experiment settings. Only entries with an evidence of at least 10 observations are considered.

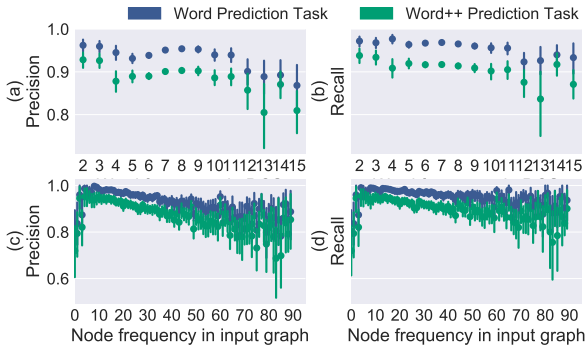

Figure 12: Performance of the system in terms of Precision and Recall for entries in DCS10K grouped over a-b) Number of words in ground truth c-d) Number of nodes in input graph G.

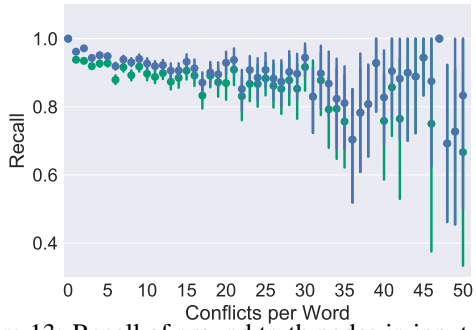

Figure 13: Recall of ground truth nodes in input graph grouped on conflicts per node

Effect of number of words in the ground truth solution: Figures 12a-b show the WPT and WP3T macro level precision and recall for 'DCS10K' where the sentences are grouped based on the number of words in the ground truth solution.

Effect of number of nodes in the sentence graph G: Figures 12c-d show the aforementioned results where the sentences are grouped based on the number of nodes in the corresponding input graph G.

In both these settings, the mean recall and precision for the WPT, marked with filled circles, never go below 90 % and 85 %, respectively. For WP3T, the mean precision and recall never go below 80 %. This shows that our system is robust to the variation in the size of the input sentences and the sentence graph as well.

Effect of number of conflicting nodes for the predicted nodes: Figure 13 shows the micro-averaged recall for the system based on the number of conflicts each token has in the input graph G. In both the tasks, almost all the bins have a mean recall of 70 % or above. The lowest in WPT is 69.23 % and that for WP3T is 66.67 %. On an average, there are 3.94 conflicts per token.

References

Coen Bron and Joep Kerbosch. 1973. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577.

Matt Gardner and Tom Mitchell. 2015. Efficient and expressive knowledge base completion using subgraph feature extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1488–1498, Lisbon, Portugal. Association for Computational Linguistics.

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

Ronald M. Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20,3:331–378.

Paul Kiparsky. 2009. *On the Architecture of Pāini's Grammar*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. 2004. Estimating mutual information. *Physical review E*, 69(6):066138.

Amba Kulkarni, Preethi Shukla, Pavankumar Satuluri, and Devanand Shukl. 2015. *How Free is free Word Order in Sanskrit*. The Sanskrit Library, USA.

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. 2006. A tutorial on energy-based learning. *Predicting structured data*, 1:0.

Ryan McDonald, Fernando Pereira, Seth Kulick, Scott Winters, Yang Jin, and Pete White. 2005. Simple algorithms for complex relation extraction with applications to biomedical ie. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 491–498. Association for Computational Linguistics.

Changping Meng, Reynold Cheng, Silviu Maniu, Pierre Senellart, and Wangda Zhang. 2015. Discovering meta-paths in large heterogeneous information networks. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 754–764, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Peter M. Scharf. 2009. *Modeling Pāinian Grammar*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Ram Nath Sharma. 1987. The astadhyayi of panini, volume i–vi.

Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. 2017. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):17–37.

Hiromitsu Takahashi. 1980. An approximate solution for steiner problem in graphs. *Math. Japonica*, 24(6):573–577.