Deep Learning Based Unsupervised POS Tagging for Sanskrit

Prakhar Srivastava
ABV-Indian Institute of Information Technology and Management
Gwalior, Madhya Pradesh, India
prakhar.sriv1996@gmail.com

Kushal Chauhan
ABV-Indian Institute of Information Technology and Management
Gwalior, Madhya Pradesh, India
ipg_2015045@iiitm.ac.in

Deepanshu Aggarwal
ABV-Indian Institute of Information Technology and Management
Gwalior, Madhya Pradesh, India
ipg_2015113@iiitm.ac.in

Anupam Shukla
Indian Institute of Information Technology
Pune, Maharashtra, India
director.iiitp@gmail.com

Joydip Dhar
ABV-Indian Institute of Information Technology and Management
Gwalior, Madhya Pradesh India
jdhar@iiitm.ac.in

Vrashabh Prasad Jain
Mahatma Gandhi Antarrashtriya Hindi Vishwavidyalaya
Wardha, Maharashtra, India
vrashabh.jain@gmail.com

ABSTRACT

In this paper, we present a deep learning based approach to assign POS tags to words in a piece of text given to it as input. We propose an unsupervised approach owing to the lack of a large Sanskrit annotated corpora and use the untagged Sanskrit Corpus prepared by JNU for our purpose. The only tagged corpora for Sanskrit is created by JNU which has 115,000 words which are not sufficient to apply supervised deep learning approaches. For the tag assignment purpose and determining model accuracy, we utilize this tagged corpus. We explore various methods through which each Sanskrit word can be represented as a point multi-dimensional vector space whose position accurately captures its meaning and semantic information associated with it. We also explore other data sources to improve performance and robustness of the vector representations. We use these rich vector representations and explore autoencoder based approaches for dimensionality reduction to compress these into encodings which are suitable for clustering in the vector space. We experiment with different dimensions of these compressed representations and present one which was found to offer the best clustering performance. For modelling the sequence in order to preserve the semantic information we feed these embeddings to a bidirectional LSTM autoencoder. We assign a POS tag to each of the obtained clusters and produce our result by testing the model on the tagged corpus.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**; **Unsupervised learning**; *Artificial intelligence*; *Cluster analysis*;

ACAI '18, December 21–23, 2018, Sanya, China

ACM ISBN 978-1-4503-6625-0/18/12…$15.00
https://doi.org/10.1145/3302425.3302487

KEYWORDS

NLP, POS tagger, tagset, Sanskrit JNU corpus, word2vec, n-grams, autoencoder, LSTM, BiLSTM

ACM Reference Format:
Prakhar Srivastava, Kushal Chauhan, Deepanshu Aggarwal, Anupam Shukla, Joydip Dhar, and Vrashabh Prasad Jain. 2018. Deep Learning Based Unsupervised POS Tagging for Sanskrit. In *2018 International Conference on Algorithms, Computing and Artificial Intelligence (ACAI '18), December 21–23, 2018, Sanya, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3302425.3302487

1 Introduction

Processing of natural languages in computer science generally requires parts-of-speech information of the language corpus under study. This parts-of-speech data provides information about the context and usage of the respective words and has been proven to be incredibly useful in Natural Language Processing applications. For languages like English, French and Spanish, this POS information is inferred from a given untagged text to facilitate further NLP tasks. The models used for POS tagging are trained in a supervised manner from large human tagged corpora, and thus achieve excellent performance. The unavailability of tagged corpora for languages like Sanskrit, act as a bottleneck for these traditional supervised learning algorithms as they typically require rich datasets. To overcome these problems, rule based techniques have been employed for POS tagging in Sanskrit, but these achieve moderate performance at best and are very difficult to create as they require extensive knowledge about the grammar and semantics of the language.

Sanskrit is an ancient Indian language whose sufficiently large annotated POS corpora are not readily available to go with supervised learning methods. When supervised deep learning approaches where adopted and the model was trained on relatively small JNU tagged corpus split into 70% training data and 30% test data, it didn't perform well on the unseen text that is not present in the vocabulary on which is is trained. Though much work has been done for other languages like English, Chinese and French, not much attention has been given to the most scientific and logical language, i.e. Sanskrit. The work that has been done in the domain of Sanskrit POS tagging do not employ novel deep learning methods which have shown much promise recently. There are many applications

of POS tagging which further contributes in the motivation. Some of the applications are given below:

Language Modelling: Language modelling is the process of restrictive appropriation on the recognition of the ith word in an arrangement, given the personalities of every single past word.

Machine Translation: Machine translation is the interpretation of content by a PC, with no human contribution.

Word Morphology: Morphology, in phonetics, is the investigation of the types of words, and the courses in which words are identified with different expressions of a similar dialect.

2 Features of Sanskrit Language

(1) **Punctuation Marks:** There are only two types of punctuation marks in Sanskrit for demonstrating the break of a sentence, unlike English, | is used to denote the end of a sentence while || is used to denote the end of a verse.

(2) **Phoneticism:** Our speech is generally the same as our composition. With this, we can conclude that there is a balance of correspondence in the spelling of a word and elocution. This is valid for most of the Indian languages and dialects as they are phonetic in nature but for English, it's distinctive in nature.

(3) **Genders:** In Sanskrit, there are three sexual orientations i.e. masculine, feminine and neuter. Any word in the language can be specified in one of these orientations. By usage of these words, any of the sexes can be chosen.

(4) **Numbers:** There exist three types of numbers in Sanskrit which are solitary, double and plural. Depending upon usage in a sentence, things experience changes to show the best suitable number.

(5) **Cases:** In Sanskrit, for every word, there exists a root word. According to the usage in a sentence, the best suitable form of the root word is being utilized. This gives an impetus for usage of Sanskrit for scientific purposes. This is not the same for English and Hindi.

(6) **Adjectives:** It is a descriptive word that qualifies noun, pronoun or any other adjective. In Sanskrit ordering of the words in a sentence does not make any difference. The word described by the adjective can be known by its gender, number and case.

(7) **Persons:** The type of verb depends upon the individual and number of the subject to which it is linked.

(8) **Active and Passive Voice:** All verbs are terminated by *aatmnepad* in passive voice and do not depend on the verb. *Dhatu* can be *aatmnepad*, *parsmaipad* or *vbhyapad*.

(9) **Moods and Tenses:** There are four moods and six tenses in Sanskrit. Three tenses are for past, one is for the present, and two are for the future.

(10) **Sandhi:** The inclination of the beginning syllable of a word to blend with the last syllable of its former word is frequent in nature. Syllables tend to blend in Sanskrit when words are spoken steadily in a progression. This procedure of associating of syllables is called *Sandhi*.

(11) **Compound words:** In Sanskrit language, it is not possible and practically infeasible to escape compound words. The way towards making it is *Samas*.

Sanskrit alphabets contain vowels, consonants, numerals, and conjunct-consonants, which are respectively shown in 1, 2, 3 and 4.

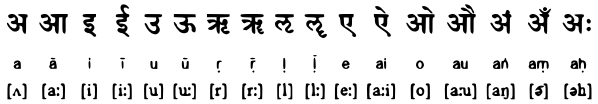

Figure 1: Vowels and vowel diacritics

Figure 2: Consonants

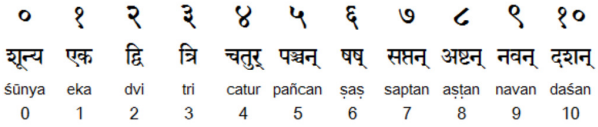

Figure 3: Numerals

Figure 4: Conjunct-Consonants

3 Related Work

Tree tagger has been developed in the past in [11]. POS tagging has been executed in two phases in this paper - namely preparing stage and testing stage. The most accurate POS tags are associated with the unlabeled words in the corpus.

Stochastic tagger has been actualized in [6] for raw Sanskrit text. In the proposed approach, Hidden Markov Model (HMM) is used for POS tagging and with modifications, the Viterbi algorithm is also utilized.

The Singular Valued Decomposition approach along with clustering has been used in [12] for unsupervised POS tagging. In this approach was modified by employing 'two step SVD' or SVD2 approach, for extracting the latent features from the corpora, followed by K-Means clustering. This implementation achieved more tagging accuracy than the previous method with lesser computational cost.

In the paper [2], unsupervised POS tagging was carried out for Bangla language by using a Hidden Markov Model. The states in the HMM were the POS tags and the observed symbols were the words in the corpus. A sequence model was thus constructed for Bangla text, by finding the HMM parameters using the Baum-Welch algorithm.

In a paper [3], a Conditional Random Field Autoencoder was used to predict the latent representation of the input. The latent representation thus generated is converted back into the input by a generative model. A simple categorical distribution over the input vocabulary was used as the input to the CRF Autoencoder to generate its latent representation. This approach enables construction of feature rich representations without resorting to independence assumptions, such as those used in Hidden Markov Models. Competitive results were obtained by employing this approach on problems of POS inductions and bitext word alignment.

In a proposed approach, instead of using a simple multinomial distribution over the vocabulary for generating word vectors, a multivariate Gaussian distribution is used to produce word embeddings for each word token. Specifically, two types of word embeddings are used to capture syntactic similarities:

(1) **Skip Gram embeddings:** These embeddings predict surrounding context words given a current word.
(2) **Structured Skip Gram embeddings:** These embeddings are an extension of Skip Gram embeddings which also take into account the relative position of words for a given context while generating word vectors.

This approach was tested for unsupervised POS induction using Hidden Markov Models and Conditional Random Field Autoencoders and achieved competitive results [9].

A rule-based POS tagger has been created for Sanskrit in [13]. All the inferred rules are stored in a database. Sanskrit sentences are processed with a specific goal to be allocated a POS tag. For this, the longest addition is sought from postfix table and labels are allocated. 100 words with 15 labels are utilized for testing purposes. 90% accuracy was achieved utilizing this Rule-based approach.

A POS tagger for Bengali has been actualized using deep learning in the paper [7]. A Deep Belief Network (DBN) was used to construct many layers of latent representations. A DBN is a deep learning architecture where multiple Restricted Boltzmann Machines (RBMs) are stacked one over the other where the hidden layer of an RBM acts as the visible layer for the next. Some essential features of words in the context of POS tagging were identified such as:

(1) Word Length.
(2) Suffix/Prefix of the word.
(3) Information about POS tag of the previous words.

4 Research Gaps

Work done in the domain of POS tagging for Sanskrit is very less. Moreover, the tagging that has been done is supervised in nature which requires large well tagged Sanskrit corpora [1, 11, 13]. As the amount of human tagged corpus for Sanskrit is limited, such approaches are unable to achieve high accuracies.

Approaches used for POS tagging in Sanskrit have not employed modern deep learning architectures which show much promise [1]. Significant improvements in POS tagging have been observed in other foreign languages and Indic languages such as Bangla [7]. However, the potential of employing novel methods of deep learning in the domain of Sanskrit POS tagging is yet to be explored.

The use of Hidden Markov Models in the task of POS tagging has shown some promise [2]. However, the use of HMM requires resorting to independence assumptions which generally are not true in language modelling tasks. Words occurring in natural language are generally dependent on words that occur previously, and influence words that may occur in the future.

A deep learning architecture such as Deep Belief Network, which has been used for the task of POS tagging in [7], has to be given inputs which explicitly capture relevant features of previous occurring words, to accurately predict the POS tag of the current word. Feature engineering of this kind is not reliable, but it has to be used in models which lack memory of past inputs.

5 Model Architecture

The block diagram of our model architecture is shown in 5. Details of the individual components of our model has been described below.

A **Initial representations:**

Word2Vec word embeddings: For any task which involves machine processing of natural language, it is required to represent words in a form which the machine can understand. These word representations typically take the form of fixed length real valued vectors for each word. In our approach, we use Word2Vec word embeddings which represent words in a continuous vector space where semantically similar words are mapped to nearby points. In the first step of our model pipeline, we use an encoder which transforms each word in the text corpus into Word2Vec word embeddings. The Word2Vec encoder is trained on a text with fixed size word vocabulary $V^{word} = \{w_1, w_2, w_3, ..., w_n\}$. For each word w_i in our training corpus, we obtain a corresponding word vector v_i if $w_i \in V^{word}$, where $v_i \in R^{emb\_size}$. For training the Word2Vec encoder, we employ the skipgram approach. The skipgram approach as in [10] is based on a log bilinear model which is trained to predict an unordered set of context words given an center word of a training context window. Given a sequence of training words $w_1, w_2, \cdots w_T$ and context window C_t, the skipgram approach maximizes the

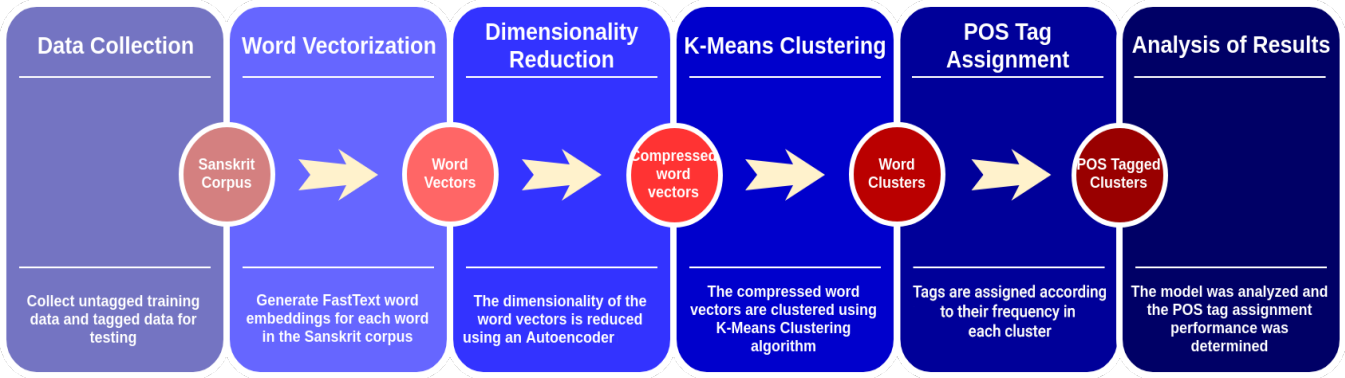

Figure 5: Model Architecture

following:

$$\sum_{t=1}^{n} \sum_{c \in C_t} log\ p(w_c|w_t)$$

The probability of observing a context word $p(w_c|w_t)$ is parametrized by the word vectors via the following softmax scoring relation:

$$p(w_c|w_t) = \frac{e^s(w_t, w_c)}{\sum_{j=1}^{n} e^{s(w_t, j)}}$$

Limitations of skipgram Word2Vec approach:

(a) This approach can only form word embeddings for words that it has already seen in the training data i.e. words that are already in the vocabulary. This is a limitation, for languages with large vocabularies and small training data.
(b) The approach encoded entire words into word vectors and thus ignores the internal structure of words. This subword information is highly descriptive, especially in our case wherein Sanskrit the part of speech is largely determined by the prefix/suffix attached to the root word.

Character n-grams: To overcome these limitations, we employ the proposed skipgram based approach in [4] where each word is represented as a bag of character n-grams. In this approach, a given word viewed as a set of character n-grams $G_w \subset G$ where G is the vocabulary of character n-grams of the model. The model encodes each character n-gram g to a real valued vector representation z_g. The embedding for each word is obtained as the sum of the vector representations of its constituent character n-grams. Hence, the scoring function is modified as:

$$\sum_{g \in G_w} z_g^T v_c$$

In this way, reliable word embeddings can be obtained for unseen words, for which the vector representations of its constituent character n-grams are known.

B **Dimensionality Reduction:**
The high dimensional vector representations obtained in the preceding step are not suitable for clustering approaches due to the curse of dimensionality. However, reducing the dimensions of these vector representations incurs a significant reduction in performance. Hence, we use autoencoders for compression of these high dimensional embeddings.

Autoencoders: Autoencoders are neural networks which are used to learn low dimensional, but richer representations of a given input in an unsupervised manner. An autoencoder consists of two parts - an encoder network (E), which encodes a high dimensional input into a compressed form, and a decoder network (D), which decodes the encoded representation to reconstruct the input. For a given input vector $v_i \in R^{emb\_size}$, the encoder maps v_i to its latent representation as $e_i = E(v_i) \in R^{enc\_dim}$ where $enc\_dim < emb\_size$. The decoder attempts to reconstruct the original vector v_i from the latent representation e_i. $\overline{V_i} = D(e_i)$ where $\overline{V_i} \in R^{emb\_size}$. The autoencoder is trained to minimize the reconstruction loss $L_r = \sum V_i - \overline{V_i}^2$

C **Modelling Sequence information:** The embeddings obtained from the Word2Vec model only considers the words that are present in the context window and hence, learns from the local dependencies. In order to model long term sequence information, we use an autoencoder with recurrent architecture.

LSTM Networks: Standard RNNs have difficulty in learning long term temporal dependencies due to the vanishing gradient problem in which the gradients become smaller and smaller in backpropagation through time and lead to inconsequential gradient updates for timesteps further back in time. In LSTM networks, Long Short-Term Memory cells as in figure 3.2 are used instead of vanilla RNN cells in the layers of a network with recurrent architecture. LSTM cells have a gated structure which can regulate gradient flow and their inclusion helps to solve the problem of vanishing gradients of vanilla RNNs.

Bi-LSTM Autoencoders: An autoencoder with Bi-LSTM architecture as in figure 3.3 can successfully model sequential information of word occurrences in both directions, in the training data. This architecture allows us to condition

a word's encoded representation on the encoded representations of both - previously occurring words and words occurring in the future and thus helps in imparting long term sequential dependencies in the encoded representations.

D **Clustering:**
K-Means Clustering: The method of K-Means Clustering is used in partitioning input data points into a fixed number of clusters in a continuous high dimensional vector space such that data points closer to each other (shorter Euclidean distance) are clustered together.

Common Noun (*nāmapada*)	N
Proper noun	NA
Proper Noun Country (*nāma abhidhāna deśa*)	NAD
Patronymic Noun	NAP
Metronymic Noun	NAS
Proper Noun Nationality (*nāma abhidhāna tadrāja*)	NAT
Noun Desiderative	NS
Coordinative Enumerative Compound (*dvandva itaretara*)	NCDI
Coordinative Collective Compound (*dvandva saṃhara*)	NCDS
Determinative Compound Accusative	NCT2
Determinative Compound Instrumental	NCT3
Determinative Compound Dative	NCT4
Determinative Compound Ablative	NCT5
Determinative Compound Genitive	NCT6
Determinative Compound Locative	NCT7
aluk Compound	NCAl
Negative Compound	NCNT
karmadhāraya Compound	NCK
dvigu Compound	NCD
bahuvrthi Compound	NCB

Adverbial Compound	NCA
Pronoun	SN
Pronoun First Person	SNU
Pronoun Second Person	SNM
Pronoun Reflexive	SNA
Pronoun Demonstrative	SNN
Pronoun Interrogative	SNP
Pronoun Relative	SNS
Parasmai Pada Verb	AV
Atmane Pada Verb	PV
Cardinal Number	SAM
Ordinal Number	SAMY
Past Active Participle	KV1
Past Middle/Active Participle	KV2
Past Passive Participle	KB1
Past Active Participle	KB2
Future Active Participle	KAa
Future Passive Participle	KAb
kṛdanta vidhyarthaka 1	KVI1
kṛdanta vidhyarthaka 2	KVI2
kṛdanta vidhyarthaka 3	KVI3
Particles	AV
Negative Particle	AVN
Conjunctive Particle	AVC
Disjunctive Particle	AVD
Interrogative Particle	AVP
Infinitive Particle	AVT
avyaya ktvānta	AVK
avyaya lyabanta	AVL
Adverb Particle	AVKV
Interjection Particle	UD

Figure 6: JNU Sanskrit Tagset

6 Experimental Setup and Implementation Details

A **Dataset Used**
POS Tagset: The POS tagset used was the JNU Sanskrit Tagset (JPOS) as shown in 6. The JPOS tagset has been created by [5]. This tagset is entirely based on Paninian linguistic standards. It contains mainly three types of labels - word class main labels, sub-labels and punctuation labels. It includes a total of 134 labels, which comprises of 65 word class labels, 43 highlight sub-labels and 25 punctuation labels and one label AJ to label obscure words.
Sanskrit Corpus: The Dataset used for training our model was created at Jawaharlal Nehru University under the supervision of Dr. Girish Nath Jha. The dataset consists of unlabeled training data consisting of 42000 words (10000 unique) and test data of 115,000 words (37000 unique) with associated POS tags in compliance with the JPOS Tagset.

B **Establishing A Baseline**
[12] used unsupervised methods for POS tagging for the first time. His approach included dimensionality reduction of word vectors through Singular Value Decomposition (SVD). Schutze's algorithm categorized words in context instead of on word types. [8] modified this approach using SVD2 and performed KMeans Clustering on the obtained vectors. POS tags were associated with these clusters by mapping techniques. These approaches were used primarily for POS tagging for English. Using these approaches for Sanskrit, we achieved test accuracies of 47.5% and 55.6% respectively.

C **Preliminary Approach**
We used the Facebook's FastText library which provides pretrained word embeddings for 294 languages, including Sanskrit. The word embeddings obtained are 300-dimensional real valued vectors. These word vectors were encoded into low-dimensional representations using an autoencoder with three fully connected layers with 150, 64 and 150 units respectively. Adam optimizer was used for performing weight updates with a learning rate of 0.003. The model was trained for 50 epochs. The loss function for the autoencoder was RMSE which accounted for the reconstruction loss. K-Means Clustering was performed on the 64-dimensional encoded representations. Word vectors were clustered in vector space depending upon the Euclidean distance from the cluster centers. POS labels were assigned to these clusters depending upon the count of POS tags. The classification accuracy achieved using this approach was 55.2%.
Many of the words from our training corpus were classified as UNK because they were different from the vocabulary on which FastText's word vector model was trained on. The vocabulary size on which it was trained had 42000 unique words and word vectors corresponding to these many words were only known. This lead us to collect more Sanskrit data for training our word vector model. We collected data from Rig Veda, Ramayana, Bhagavad Gita, Puranas, Ayurveda and scraped Wikipedia pages for Sanskrit text. We managed to collect a corpus of 2368500 words with 510000 unique words and trained our word vector model with skip gram approach using this corpus. This adaption enhanced the classification accuracy to 58.7%.

D **Proposed Approach**
In our proposed approach, we use character n-grams instead of traditional Word2Vec implementations. By incorporating n-grams in our skip gram word vector model we intend to address the problem of finding vector representations of words for our training corpus that are missing from our word vector model's vocabulary. These representations take into account the internal structure of the words as word vectors is formed by a combination of vectors of the character n-grams which constitute the word under consideration. Keeping rest of the architecture same as in our preliminary approach, we get the classification accuracy of 84.9%. This increase in accuracy can be accounted by:

Word Vectorization Method	Dimensionality Reduction	Accuracy
Frequency of occurrence	SVD	47.5
Frequency of occurrence	SVD-2	55.6
Skipgram Word2Vec Facebook FastText library	Autoencoder	55.2
Word2vec (using skip gram) trained on collected text	Autoencoder	58.7
Character embeddings using n-grams	Autoencoder	84.9
Character embeddings using n-grams	LSTM Autoencoder	89.7
Character embeddings using n-grams	Bidirectional LSTM Autoencoder	93.2

Table 1: Comparison of results

(a) Previously the words for which vector representations were unavailable were classified as unknown thus decreasing our classification accuracy because their reference in the corpus was unknown.
(b) In Sanskrit, words having the same prefix/suffix as in different *Dhatu Roop's* of a root word will generally be of the same POS.

E **Modifications in our Proposed Approach**
We replaced the autoencoder in our proposed model architecture with an LSTM autoencoder. This allows us to incorporate long term sequential dependencies in our encoded representations. The LSTM autoencoder consisted of 64 units. The model was trained with a batch size of 256 and Adam optimizer. This approach yielded us an accuracy of 89.7%. An unidirectional LSTM network can only model dependencies for word that have previously occurred in the text. To incorporate both - words previously occurring and words occurring in the future, we use a Bidirectional-LSTM Autoencoder. With the use use of BiLSTM autoencoder, the encoded representations of size 128 were clustered and a POS tagging accuracy of 93.2% was achieved.

7 Results

Results comparing test accuracies obtained using various methods that we explored for POS tagging are presented in table 1. The best test accuracy we achieved was 93.2%. As is evident from the table, significant improvements were observed when character n-grams were used instead of traditional Word2Vec for obtaining vector representations.

8 Discussion

(1) Due to an imbalance in the occurrence frequency of the POS tags, there is bias in POS classification task. More frequent tags tend to show and chances are tags which have a rare occurrence won't be classified, thus causing true positives and false positives for that specific tag to be zero. This problem can only be solved by creating balance in the frequency of occurrence of the POS tags while creating the corpus.
(2) The reconstruction loss incurred for an input word vector having dimensionality as high as 1000 or 2000 was much higher than a vector of length 300. Consequently, classification accuracy was better when 300 dimensional encoded as compared to a vector with higher dimensionality.
(3) The dimensions of encoded vectors also played a crucial role in determining the POS tags. K-Means having a curse of higher dimensionality yielded better results when encoded representations were of dimensionality 64 or 32 rather than having dimensionality of 128 or higher.

9 Future Work

A corpus having a balanced frequency of various POS tags should be created for training the model thus facilitating the formation of proper clusters for rarely occurring POS tags and reducing the number of false positives. This approach can be used for other Indian languages and dialects that currently do not possess a sufficiently rich tagged corpus like Tamil, Bengali, Marathi, Telugu, Gujarati, Gurmukhi, Oriya, etc.

REFERENCES

[1] S. Adinarayanan and N. S. Ranjaniee. 2015. Part-of speech tagger for sanskrit. *A state of art survey, International Journal of Applied Engineering Research* 10 (2015), 24173–24178.
[2] Hammad Ali. 2010. *Unsupervised Parts-of-Speech Tagger for the Bangla language.* Department of Computer Science. University of British Columbia.
[3] Waleed Ammar, Chris Dyer, and Noah A. Smith. 2014. Conditional Random Field Autoencoders for Unsupervised Structured Prediction. In *NIPS 14 Proceedings of the 27th International Conference on Neural Information Processing, Systems.* 3311–3319.
[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.
[5] R. Chandrashekar. 2007. *Parts-of-Speech Tagging for Sanskrit.* Ph.D. Dissertation. Jawaharlal Nehru University.
[6] Oliver Hellwig. 2007. SanskritTagger : a stochastic lexical and pos tagger for Sanskrit. In *First International Sanskrit Computational Linguistics Symposium.*
[7] Md. Fasihul Kabir, Khandaker Abdullah-Al-Mamun, and Mohammad Nurul Huda. 2016. Deep Learning Based Parts of Speech Tagger for Bengali. In *International Conference on Informatics, Electronics and Vision.*
[8] Michael Lamar, Yariv Maron, Mark Johnson, and Elie Bienenstock. 2010. SVD and Clustering for Unsupervised POS Tagging. In *Proceedings of the ACL 2010 Conference Short Papers.* 215–219.
[9] Chu-Cheng Lin, Waleed Ammar, Chris Dyer, and Lori Levin. 2015. Unsupervised POS Induction with Word Embeddings. In *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL.* 1311–1316.
[10] TomÃąÅą Mikolov, Ilya Sutskeve, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality.. In *Adv. NIPS.*
[11] M. S. Kumar R. M. Prashanthi and R. R. Sree. 2013. Pos tagger for sanskrit. *International Journal of Engineering Sciences Research (IJESR)* 4 (2013), 32–41.
[12] Hinrich Shutze. 1995. Distributional part-of-speech tagging. In *Seventh Conference on European Chapter of the Association for Computational Linguistics.* Morgan Kaufmann Publishers Inc., 141–148.
[13] Namrata Tapaswi and Suresh Jain. 2012. Treebank based deep grammar acquisition and part-of-speech tagging for Sanskrit sentences. In *In Software Engineering (CONSEG), 2012 CSI Sixth International Conference on.* IEEE, 1–4.