Evaluating Neural Word Embeddings for Sanskrit

Jivnesh Sandhan[1], **Om Adideva Paranjay**[3], **Digumarthi Komal**[1], **Laxmidhar Behera,**[1,2] **and Pawan Goyal**[4]
[1]Dept. of Electrical Engineering, IIT Kanpur,[2]Tata Consultancy Services,
[3]Dept. of Electrical and Systems Engineering, University of Pennsylvania,
[4]Dept. of Computer Science and Engineering, IIT Kharagpur
`jivnesh@iitk.ac.in, pawang@cse.iitkgp.ac.in`

Abstract

Recently, the supervised learning paradigm's surprisingly remarkable performance has garnered considerable attention from Sanskrit Computational Linguists. As a result, the Sanskrit community has put laudable efforts to build task-specific labeled data for various downstream Natural Language Processing (NLP) tasks. The primary component of these approaches comes from representations of word embeddings. Word embedding helps to transfer knowledge learned from readily available unlabelled data for improving task-specific performance in low-resource setting. Last decade, there has been much excitement in the field of digitization of Sanskrit. To effectively use such readily available resources, it is essential to perform a systematic study on word embedding approaches for the Sanskrit language. In this work, we investigate the effectiveness of word embeddings. We classify word embeddings in broad categories to facilitate systematic experimentation and evaluate them on four intrinsic tasks.

1 Introduction

The supervised learning paradigm has shown remarkable performance in many downstream Natural Language Processing (NLP) tasks. In this domain, the NLP community has witnessed the emergence of neural-based approaches with the state of the art performance. The dramatic popularity of such approaches has garnered considerable attention from Sanskrit Computational Linguistics (SCL) community. The primary prerequisite for the feasibility of such approaches is the availability of task-specific labeled data. Therefore, as a first step, there was a need for preparing such task-specific resources. As a result, laudable efforts have been put by the community for building task-specific labeled data for various NLP tasks (Krishna et al., 2017; Hellwig and Nehrdich, 2018; Hellwig et al., 2020). The availability of such resources triggered the emergence of the data-driven neural-based approaches into the SCL field. The wide success of neural-based data-driven approaches is greatly justified by the state of the art performance for many downstream tasks for Sanskrit, namely, segmentation (Hellwig and Nehrdich, 2018; Reddy et al., 2018; Krishna et al., 2018), dependency parsing (Krishna et al., 2020c; Krishna et al., 2020a; Sandhan et al., 2021; Krishna et al., 2020b), semantic type identification (Sandhan et al., 2019; Krishna et al., 2016), word order linearisation (Krishna et al., 2019; Krishna et al., 2020c) and morphological parsing (Gupta et al., 2020; Krishna et al., 2018).

The supervised approaches have shown remarkable performance in several downstream NLP tasks and many of these use word embeddings as a primary component. In 2013, Mikolov et al. (2013) proposed a new milestone in the field of word embedding, which completely revolutionized Natural Language Processing (NLP). From there, word embedding has been a buzzword in NLP and has become part and parcel of downstream applications. These word embedding approaches adopt a semi-supervised learning paradigm. They do not depend on any explicitly labeled data or human supervision. Availability of massive unlabelled corpora and ease of integration into downstream NLP tasks made them exciting for the researchers. Sanskrit, the cultural heritage of India, has 30 million extant manuscripts (Goyal et al., 2012). Last decade, there has been much excitement in digitization for Sanskrit, as a result, we have DCS[1], The Sanskrit Library[2] and GRETIL[3]. To reduce the labeled data dependency, it's crucial to take advantage of such cheaply available resources.

This motivates us to systematically investigate word embedding approaches that are originally proposed for resource-rich languages like English. Most of the research in NLP disproportionately focuses on resource-rich languages due to readily available language resources: digitized texts and gold standard labeled data (Joshi et al., 2020). Also, the recent word embedding approaches have been studied for resource-rich languages such as English. However, such analysis is lacking for Sanskrit. In addition to the morphological rich nature of language, the Sanskrit is a relatively free word order language, and most words are compound words (samāsa) that demand special attention for modeling the word embedding approach. To conduct a systematic study on the efficacy of these approaches, we categorize word embeddings into two categories, namely, *static* and *contextualized*. The *static* embeddings generate a single representation for all the senses possible for a word, irrespective of surrounding context words. In contrast to that *contextualized* embeddings generate a different representation for each sense of the meaning depending on the surrounding context words. This work provides fertile soil for investigating the following questions: (1) Which linguistic phenomenons are captured by word embeddings? (2) Out of existing word embedding approaches, which embeddings are most suitable for Sanskrit? (3) What are the shortcomings of existing approaches specific to Sanskrit? Based on our experiments, our major findings are as follows:

1. Our intrinsic evaluation results illustrate that word embedding models trained on Sanskrit corpus are able to capture exceptionally good amount of the syntactic information. However, they perform relatively poorly with the semantic notion of relatedness - similarity mainly due to Out of vocabulary problem.
2. The *contextualized* word embedding model by Peters et al. (2018) outperforms all the word embeddings in all the intrinsic tasks with a large margin except semantic categorization task (Section 5). We employ contextual models without a context to measure efficacy of these approaches compared to static approaches. Applying *contextualized* models without context is similar to just using their underlying static word representation.
3. The qualitative analysis shows that the *contextualized* representations generated by Peters et al. (2018) and Lan et al. (2020) for polysemous words indicating the same sense of meaning forms a cluster for each sense of meaning (Section 6.3). This is indeed a serendipitous outcome for such data-hungry models when trained on negligibly small amount of corpus available for Sanskrit.
4. We release our codebase, datasets and models publicly at: `https://github.com/jivnesh/EvalSan`. This might help the research community to do further analysis and apply these embeddings to downstream NLP applications of Sanskrit.

2 Neural Word Embedding

Neural word embedding methods construct vector space so that semantically related words are projected in closer vicinity compared to unrelated words. Word embedding is a dense representation of words in low-dimensional vector space. It mainly exploits the distributional hypothesis: neighboring words share similar meaning (Firth, 1957; Harris, 1954).

As shown in Table 1, we consider the two broader categories of embedding approaches, namely, *static* and *contextualized*. Further, we divide them based on an atomic unit of input fed to the model, i.e., token level, subword level and character level. First, we consider token level embeddings: `word2vec` (Mikolov et al., 2013) and `GloVe` (Pennington et al., 2014). We consider

[1]`http://www.sanskrit-linguistics.org/dcs/index.php`
[2]`https://sanskritlibrary.org/`
[3]`http://gretil.sub.uni-goettingen.de/gretil.html`

Category	Input level	Models
Static	Token	Word2vec, GloVe
	Subword	FastText
	Character	CharLM
Contextualized	Subword	ALBERT
	Character	ELMo

Table 1: Categorization of embedding models

`word2vec` as a strong baseline because it is the basic backbone of all the neural-based follow-up works in this domain. While `word2vec` focuses on the local context of a target word, `GloVe` is an alternative approach that considers the global context of a target word. Therefore, `GloVe` can be seen as representative of count-based models; hence we choose to include it in our study. However, these token-based approaches suffer from two serious issues: (1) Out of vocabulary (OOV) problem: they fail to obtain representation for words which are not seen during training (2) Polysemy problem: a single representation for multiple meanings of words. In order to handle the first drawback, sub-word (Wieting et al., 2016; Bojanowski et al., 2017; Heinzerling and Strube, 2018) and character level (Kim et al., 2016; Jozefowicz et al., 2016) compositional modeling is proposed. These models learn composition functions to obtain word-level representation of a word using character level or subword level modeling. In the next category, the advent of contextualized embeddings is attributed to the polysemy problem. The year 2018 is an inflection point in the field of NLP with the launching of contextualized embedding by Peters et al. (2018, `ELMo`). The specific meaning of the word can be understood by the context words surrounded by it. Hence, in contextualized embedding approach, the representation of each token word is a function of entire sentence as a input. The emergence of another contextual embedding by Devlin et al. (2019, `BERT`) can be seen as a clever recipe of a bunch of excellent ideas bubbled in the field of NLP in recent years. We consider a derivative of `BERT` known as `ALBERT` (Lan et al., 2020) which is light version of `BERT` with relatively less number of parameters.

3 System Description

Tokenizer: Token-based word embeddings such as `word2vec` and `GloVe` suffer from OOV problem, i.e., they are not capable of producing vector representation for words that are not seen during training. Here, word is defined as a continuous unsandhied string separated by delimiter. This problem is even more prominent for Sanskrit due to the following reasons: (1) morphological-rich nature (2) highly inflectional (3) most of the text is available in *sandhied* format. Hence, token-based approaches are highly ineffective for Sanskrit. We take inspiration from Heinzerling and Strube (2018) and leverage `sentencepiece` tokenizer (Kudo and Richardson, 2018) to overcome OOV problem. The `sentencepiece` is unsupervised tokenizer used for neural-based tasks such as machine translation and sequence generation task. This is a purely data-driven tokenizer and does not require any pretokenization. Also, this is language independent approach where words are considered as a sequence of unicode characters. The `sentencepiece` overcomes the challenge as mentioned earlier by creating a new vocabulary instead of relying on real word boundaries. This method uses a greedy approach to figure out subword units that maximize the likelihood of the language model (Kudo and Richardson, 2018). There exist other unsupervised tokenizers such as `Wordpiece` (Wu et al., 2016) and `BPE` (Sennrich et al., 2016). However, these approaches expect tokenized words as input and most of the available text for Sanskrit is in *sandhied* format. Hence, `sentencepiece` is the most befitting choice as a tokenizer. We demonstrate `sentencepiece` tokenization with an example as shown below.[4] Here, `sentencepiece` considers space character as a normal character and is replaced by "_" character. Note that *manā_bhava* is a word in our newly generated vocabulary, originally part of two words.

Sandhied sentence (original)

man-manā bhava mad-bhakto mad-yājī māṁ namaskuru
mām evaiṣyasi yuktvaivam ātmānaṁ mat-parāyaṇaḥ

Segmented by `sentencepiece` tokenizer

_man - manā_bhava _mad -bhakto _mad - yājī _māṁ _namas kuru
_mām _evai ṣyasi _yukt vaivam _ātmān aṁ _mat - parāyaṇ aḥ

3.1 Static Embeddings

Word2Vec: The work of Mikolov et al. (2013) is a new milestone in the field of word embeddings and popularly known as `word2vec`. This work was the basic backbone of the neural-based word embedding approach, which led to many of word embedding variants tackling various challenges. This model exploits the distributional semantics hypothesis: "semantically similar words are used in similar contexts" (Firth, 1957; Harris, 1954). `Word2vec` often considered as one of the seminal the deep learning models; however, technically, neither the architecture is deep, nor the model uses non-linearities. It adapts language modeling objectives to consider the additional context. Language modeling is the task where the next word is predicted given previous n words. However, `word2vec` also considers next n words as a context. The basic intuition behind this model is to project the words into n dimensional vector space such that semantically similar words are placed in closer vicinity than non-similar words. In other words, the dot product of similar (cosine similarity) words will be higher compare to that with non-similar words. In order to obtain geometric organization of words, authors propose two training methods: (1) Continuous Bag of Words `(CBOW)` (2) `Skip-gram`. While the `CBOW` uses context words to predict the target word, skip-gram predicts context words given the target word as an input. Note that context is considered as a window of n words surrounding target word from forward and backward direction. Unlike prior sparse representations, these learned vector representations are dense vectors of dimension ranging from 50-1000 and these dimensions do not have clear semantic interpretation.

GloVe: Prior to Mikolov et al. (2013), count-based models were prominent in the word embedding field. These count-based models (Bengio et al., 2003; Collobert and Weston, 2008; Collobert et al., 2011) are explicitly designed to model the global statistics of a entire corpus. Predictive models such as `word2vec` solely focus on the target word's local context. To gain the best of two worlds, Pennington et al. (2014) proposed an alternative word embedding technique that integrates global statistics via matrix factorization and local context via the sliding window method. This approach focuses on the co-occurrence of each word with all other words in the entire corpus, therefore called Global Vectors `(GloVe)`. For the unsupervised models, the primary source of information to learn word representation is corpus statistics. However, how to exploit corpus statistics to generate word representations is an interesting problem. Pennington et al. (2014) show that the ratio of co-occurrence probabilities of words is more indicative of their relation than a direct co-occurrence probability . Although, `GloVe` does not use neural network, Levy et al. (2015) consider it as a predictive model due to following reason: Unlike other count-based models, `GloVe` use the Stochastic Gradient Descent (SGD) optimization for non-convex objective function. The `GloVe` outperforms it's peer models (including `word2vec`) in several intrinsic and extrinsic evaluation tasks. The model works well with a small-sized corpus but is highly scalable to a larger corpus. Although the model demands much memory to populate the co-occurrence matrix in large corpora, the authors term it as a "one-time, up-front cost".

[4]Originally, we apply tokenization on SLP1 transliterated data. Just to illustrate, we show an example in the romanized (IAST) transliteration scheme.

FastText: Token-level embeddings like `word2vec` and `GloVe` suffer from *Out of Vocabulary* (OOV) problem. That means these models cannot generate vector representation for words that are not present in vocabulary. For example, even if model learns embeddings for *pācikā* and *bhāryaḥ*, it will not be able to handle the word *pācikābhāryaḥ*, if it is not a part of vocabulary. Hence, such approaches could not serve the need of morphologically rich languages. Also, the model is not capable of handling different possible inflections of a word (for example, *rāmaḥ, rāmābhyām, rāmasya*) if such inflectional variants are not a part of the vocabulary. Hence, to overcome these shortcomings, Bojanowski et al. (2017, `FastText`) proposed a new word embedding technique, specifically, for learning reliable representations for OOV words. To accomplish this, the authors propose to incorporate subword level information into the model. While learning vector representation of target word, simultaneously, representations of all n-grams (n varies from 3 to 6) are also learned. For example, for n=4, the possible 4-grams of a word *ajñānam* are *<ajñ, jñāna, ñānam, ānam>*. The brackets are used for padding purposes to indicate the beginning and end of the word. With this modification, the training procedure similar to word2vec is applied. This subword level representation learning empowers the model to generate a representation for OOV words by simply summing representations of n-grams corresponding to the target word. `FastText` has been shown to improve performance on syntactic word analogy tasks, especially in the case of morphologically rich languages.

CharLM: Kim et al. (2016, `CharLM`) proposed a new variant of language model that learns subword information through a character-level convolutional neural network (CNN). In a standard neural language modeling setting, the model takes word-level vector representation as input and predicts word-level information. This modeling decision makes the model blind to subword information. For example, the inflected forms of a root word should have structurally similar word embeddings because they shared common sub-strings except for affixes. Therefore, to capture subword level information, `CharLM` takes a character-level input and learns composition function to generate word-level representation. This is achieved by extracting the features from CNN followed by a highway network. Finally, these extracted features are fed to a long short-term memory (LSTM) recurrent neural network language model to make word-level predictions. Kim et al. (2016) argues that without using word embeddings as input to the language model, their model gets the additional advantage of fewer parameters with competitive performance. Also, they show that the model is capable of encoding both semantic and orthographic information.

3.2 Contextualized Embedding

Each word has multiple meanings and their uses highly depend on the context they are used. However, irrespective of context, traditional approaches give a single vector representation for a word. Therefore, traditional word embeddings are called *static* embeddings. For example, there are two possible meanings of a word *cakrī*: (1) a potter (2) the holder of the disc and its uses highly depends on the context. In order to solve this polysemy problem, contextualized embeddings like `ELMo` (Peters et al., 2018) and `BERT` (Devlin et al., 2019) are proposed. In literature, there exists two domains for applying language representations to downstream tasks, namely, *feature-based* and *fine-tuned*. The feature-based approach simply augments the learned representation into the task-specific model architecture and well-known example in this category `ELMo`. A fine-tuned approach such as `BERT`, integrates pretrained model architecture into task-specific model and trains all the parameters, including the pretrained component. The emergence of these embeddings is described as a beginning of a new era in a word embedding. Their exceptional popularity in the field launched many of contextualized embedding variants. Hence, we include `ELMo` and `ALBERT` (A Light `BERT`: derivative of `BERT` model with significantly less number of parameters) (Lan et al., 2020) in our study. From an architectural point of view, `ELMo` belongs to LSTM based language model family and `ALBERT` belongs to the transformer-based language model family.

ELMo: Unlike traditional *static* word embeddings, ELMo computes a representation of each token word as a function of all the words in a sentence. Authors employ the language modeling objective; hence, these representations are also called as ELMo (Embeddings from Language Models). There are three salient features of ELMo: (1) *contextual:* the word representation depends on surrounding context words (2) *deep:* the word representation is weighted average of all the layers of model architecture (3) *character based:* character based modeling enables model to encode morphological clues to obtain robust representations for OOV words, especially, for languages with rich morphology. Prior state of the art neural language models (Kim et al., 2016; Jozefowicz et al., 2016) takes context-independent token-level representation as an input to forward language modeling objective where next word is predicted based on previous n words. Similarly, the backward language model predicts a word given n following words, i.e., precisely opposite to the forward model. ELMo coupled both forward and backward language model objective function together. Through ablations and intrinsic evaluation experiments, authors discover that higher-level layers in model architecture capture semantic properties and lower-level layers are syntactic. They show that a linear combination of internal representation helps obtain a rich representation, which is a better decision over just using top layer representation. Simultaneously providing enriched representation via a weighted linear combination of all layers, the model can choose the most relevant features for specific downstream tasks.

ALBERT: BERT is blend of several clever ideas that has been emerging in the field of NLP such as semi-supervised learning (Dai and Le, 2015), ELMO (Peters et al., 2018), ULMFIT (Howard and Ruder, 2018) and Transformer (Vaswani et al., 2017). The foundational concept used in BERT is a transformer architecture popularized by Vaswani et al. (2017). Inspired from Howard and Ruder (2018, ULMFIT), BERT adopts the fine-tuning approach to augment learned contextual representation in downstream applications. Authors argue that the previous language representations are unidirectional language modeling objective and this choice limits the model to learn effective representations. First, ELMo from the LSTM family demonstrated the effectiveness of a bidirectional language modeling for learning enriched representations. Suppose we introduce bidirectional language modeling into a transformer with a vanilla language modeling objective. In that case, the problem will become trivial and it will not be possible to learn robust representations. Therefore, to tackle this challenge, authors adapt a vanilla language modeling objective to a masked language objective. Here, instead of predicting the next word from previous n words, the model masks a few words from a sentence and learns to recover masked words from the remaining words. This is also called a denoising objective. In this way, BERT can take advantage of the surrounding context from both sides simultaneously.

This data hungry model reports the state of the art results in many downstream tasks on the leaderboard at the expense of huge parameters. Despite the great success of BERT, it is highly impractical to train such a monstrous model with limited available resources for Sanskrit. For example, BERT-base has 110 million trainable parameters, making it computationally very intensive and nearly impossible to train on consumer devices. Therefore, we choose a derivative of BERT called as ALBERT, a light BERT, which has significantly few parameters than traditional BERT model with significantly better performance. The authors achieved this by making three key changes to the original BERT's architecture. First, they factorize the large vocabulary embedding matrix used in BERT into two smaller ones such that dependency on the size of hidden layers is obliviated from the size of vocabulary embeddings. Second, they apply cross-layer parameter sharing. This way, the parameters are shared for similar sub-segments, thus ensuring that parameters are learned only once and are then reused for the subsequent blocks. This significantly reduces the number of trainable parameters in the model. Lastly, they propose a new inter-sentence coherence task called Sentence Order Prediction, instead of using original BERT's Next Sentence Prediction (NSP). NSP is a binary classification loss that was specifically created to improve performance on downstream NLP tasks. In summary, the ALBERT model is found to have 18x fewer parameters than BERT-large and trains 1.7x faster while achieving the state of the art results on many downstream tasks. This model is an important breakthrough as it brings the capabilities of `BERT` to a more practical and usable form that can be useful for real-world applications.

4 Intrinsic Evaluation

To evaluate the quality of the embedding model, two primary methods are proposed in literature: (1) Intrinsic evaluation (2) Extrinsic evaluation. Intrinsic evaluation refers to the class of methods to measure the quality of vector space without evaluating them on downstream tasks. For example, semantic similarity is the most traditional method to evaluate meaning representations intrinsically. On the other hand, extrinsic evaluation refers to the class of methods to assess the quality of vector representation when fed as an input to the machine learning-based model in the downstream NLP task. The wide range of NLP tasks that deals with lexical semantics such as chunking, part of speech tagging, semantic role labeling and named entity recognition can be used for the extrinsic evaluation. In this work, we restrict our study to intrinsic tasks. Intrinsic evaluation directly checks for syntactic and semantic properties of words (Mikolov et al., 2013; Baroni et al., 2014). These tasks rely on query inventory datasets consisting of a query word and semantically related target word. These query inventories are readily available for resource-rich languages but not for Sanskrit. We create such inventories for four different intrinsic tasks: relatedness, synonym detection, analogy prediction, and concept categorization. For creating such query inventory dataset for these intrinsic evaluation tasks, we use *Amarakoṣa*[5] (Nair and Kulkarni, 2010), Sanskrit WordNet[6] (Kulkarni, 2017) and Sanskrit Heritage Reader[7] (Goyal and Huet, 2016; Huet and Goyal, 2013). In extrinsic evaluation, feature extracted by word embeddings is fed as an input to downstream NLP task and corresponding performance metric is used as an indicator to compare across different embedding approaches. Although this family of evaluation methods gives signal on the relative strength of different word embedding approaches, it needs not be considered a general proxy for overall quality.

Relatedness: It is essential to distinguish between the notion of relatedness and synonyms (Agirre et al., 2009). Synonym words share many semantic properties and they can be substituted with each other without changing the meaning of the sentences. On the other hand, semantically related words can co-occur in the same context or document and share any one semantic relationship such as meronymy or antonymy. Unlike synonym words, we can not substitute related words in place of each other. The notion of semantic relatedness between word pairs is one of the fascinating concepts in lexical semantics with a variety of NLP applications (Pilehvar and Camacho-Collados, 2020). This task is considered as an in-vitro evaluation framework for judging the quality of word embeddings. The word pairs are annotated based on the degree of relatedness on a numerical scale by human annotators for evaluating semantic relatedness. For example, the distance between the *paricārikā* and *vaidya* in the numerical scale from 0 to 3 could be 2.5 because both words belong often found together. Similarly, the human judgment score for word pair *prādhyāpaka - karkaṭī* could be 0.5 because these two words do not share any semantic property. The word embedding performance is then evaluated by assessing the correlation between the average scores assigned by annotators and the cosine similarities between corresponding vector representations.

For the construction of such human-annotated test data for the new target language, a standard methodology is followed in the literature (Camacho-Collados et al., 2015). It is divided into two steps: (1) English word pairs of a standard English dataset are translated into the target language (2) language experts annotate these word pairs. For the English language, the dataset popularized by Hassan and Mihalcea (2011, WordSim353) has 353 word pairs with a human relatedness score. However, this dataset is criticized due to the human judgment criterion, which

[5]`https://sanskrit.uohyd.ac.in/scl/amarakosha/index.html`
[6]`https://www.cfilt.iitb.ac.in/wordnet/webswn/index.php`
[7]`https://sanskrit.inria.fr/DICO/reader.fr.html`

conflates similarity and relatedness. Another alternative is RG-65 (Rubenstein and Goodenough, 1965) dataset. In addition, we find that many words such as *automobile, cushion, autograph etc.* can not be naturally translated into Sanskrit. Therefore, we use *Amarakoṣa* to get word pairs due to its systematic organization of semantically related words. Here, synonymous words are categorized into synsets and semantically related words into *varga.* Therefore, we decide to select the word pairs from *Amarakoṣa.* This choice obliviates the need for translation and annotation of word pairs into Sanskrit. We exploit systematic categorization based on relatedness presents in the *Amarakoṣa.* We transform the relatedness task into a classification task where the task is to categorize word pairs into three classes based on cosine similarity scores between vector representations generated by corresponding embeddings. For selecting word pairs, we follow the following procedure. First, we sample 13,500 word pairs in such a way that an equal proportion of word pairs are selected from the following classes : (1) word pairs from the same synset (2) word pairs from same *varga* but not from the same synset (3) and the remaining word pairs such that each word from a word pair belongs to different *varga.* Out of 13,500, we use 4,500 word pairs as the development and remaining word pairs in the test set. We use a development set to find out cosine similarity thresholds to classify word pairs into respective classes. Here, we use the macro F-score as an evaluation metric to measure the performance.

Synonym Detection: Synonym detection (Baroni et al., 2014; Bakarov, 2018) is a task where the goal is to find the synonym of a given target word from the four synonym candidates. In this multiple-choice setting, a single answer is correct. For example, for target *sūryaḥ*, synonym candidates are *ādityaḥ* (correct), *cañcalā*, *candraḥ* and *ākāśa.* Here, we choose an option as a correct answer which shows the highest cosine similarity with the target word. We use accuracy as a metric to measure performance. We leverage *Amarakoṣa* (Nair and Kulkarni, 2010) synsets to build MCQs. It has 4,056 synsets categorized into 25 *vargas.* Out of 4,056, only 2,581 synsets have more than one element. We choose a word pair from each such synset, assign one word as the target word and another word as the correct answer. Then, for the remaining options, we use three words from the same *varga* but not from the synset where the target word belongs. The primary motivation of such a choice is that words belonging to the same *varga* are semantically related and picking such related words as synonym candidates make the task more interesting and challenging. Our test data has 3,034 MCQs. We consider accuracy as an evaluation metric for this task.

Analogy Prediction: The analogy prediction task popularized by Mikolov et al. (2013), a question is framed with word pairs exhibiting similar sense. For example, "If *kapi* is similar to *kapibhyām* then in the same sense *yati* is similar to ..." Authors propose that a simple algebraic operations suffice to answer these analogy questions. In above example, in order to find similar word for *yati*, we need to calculate $X = vector(\text{“kapibhyām”}) - vector(\text{“kapi”}) + vector(\text{“yati”})$. Then, we find the nearest neighbour to X as per cosine similarity metric and use it as answer to question.

To probe the information encoded in word embeddings, we build a comprehensive set of syntactic and semantic questions as illustrated in Table 2. We use the Sanskrit Heritage Reader (Goyal and Huet, 2016; Huet and Goyal, 2013) for generating syntactic questions. We consider the primary conjugation table for verbal root words and the declension table for nominal words for syntactic analogies. For conjugations, there are 10 families and popularly also called as *gaṇas.* Each *gaṇa* is further divided into *parasmaipadī* and *ātmanepadī.* Out of these 20 families, we choose 3 representative candidates to generate primary conjugations such that most of the inflectional variations are covered. Similarly, nominal words are broadly divided into *ajanta* and *halanta.* Further, they are divided based on gender and the end of root words (for example, *akārānta, ukārānta, nakārānta, jakāranta etc*). In this way, we consider 25 fine-grained families for nominals. From each family, we select 3 representative words. Words occurring in same family display similar inflections variations. Therefore, to make questions, we strictly choose

Relationship	Word Pair 1		Word Pair 2	
declension	kapi	kapayaḥ	yati	yatayaḥ
conjugation	ci	acīyethām	aś	āśyethām
absolutive	dr	dīrtvā	aś	aśitvā
infinitive	dr	daritum	aś	aśitum
husband - wife	viṣṇuḥ	lakṣmī	rāmaḥ	sītā
son - father	rāma	daśaratha	abhimanyu	arjuna
daughter - father	pārvatī	himavān	sītā	janaka
charioteer - warrior	dāruka	kṛṣṇa	śalya	karṇa
defeated - victorious	kicaka	bhīma	rāvaṇa	rāma
son - mother	kṛṣṇaḥ	yaśodā	rāmaḥ	kauśalyā

Table 2: Examples of syntactic (top) and semantic (bottom) questions in our test set

word pairs from the same family where the inflectional component belongs to the same position in the conjugation/declension table. For semantic questions, we extract semantic relation from *Amarakoṣa* (Nair and Kulkarni, 2010), *Rāmāyanam* and *Mahābhāratam*.[8] In semantic relationships, we consider six categories: husband - wife, son - father, daughter - father, charioteer - warrior, defeated - victorious, and son - mother. There are 6,415 semantic and 10,000 syntactic questions. We only consider single token words in our test set. We use accuracy as a metric to measure the performance. We evaluate accuracy separately for semantic and syntactic questions.

Categorization: As the name suggests, categorization is an evaluation task where the goal is to group nominal sets of concepts into natural categories (Baroni et al., 2014). For example, *narmadā, godāvarī, kāverī* and *gaṅgā* fall under the names of river category. This task is treated as an unsupervised clustering task where word representations are clustered into n clusters (n is the total number of gold standard categories). Here, performance is evaluated in terms of *purity*. This metric represents the degree to which elements from the single gold standard category belong to the same cluster. If unsupervised clustering exactly replicates the gold standard clusters, then the purity score reaches 100%. Otherwise, it decreases based on the inability to replicate gold standard partitions.

Following Baroni and Lenci (2010), we construct a test set of 15 common categories picked from *Amarakoṣa*. We exploit the ontological classification of *Amarakoṣa* based on *Vaiśeṣika* ontology. *Amarakoṣa* is also marked with semantic relations between words. These relations are mainly classified into hierarchical and associative. We use hierarchical relations such as hypernymy - hyponymy and holonymy - meronymy for this task. In each category, we select up to 10 concepts. Our categorization data contains the following categories: *vādyopakaraṇam, vṛttiḥ, grahaḥ, devatā, paśuḥ, pakṣī, nadī, vṛkṣaḥ, upakaraṇam, ābharaṇam, alaukikasthānam, vāhanam, parvataḥ, vṛkṣaphalam* and *puṣpam.* We also build a separate test set consisting of 45 syntactic categories where each category is made up of 25 concepts.[9] Each category represents a particular morphological class.

5 Experiments

Corpus: Our corpus contains texts from the Digital Corpus of Sanskrit (DCS), scraped data from Wikipedia and Vedabase corpus.[10] The number of words in each section is 3.8 M, 1.7 M, and 0.2 M, respectively. DCS and Vedabase are segmented, but the Wikipedia data is

[8]We extract word pairs holding specific relation from wikipedia.
[9]More details can be found at: `https://github.com/jivnesh/EvalSan`
[10]We use SLP1 transliteration scheme for training all the models.

unsegmented. We use this data for training all the embedding models. Most of the data in our corpus is in the form of poetry.

Hyper-parameters: For tackling OOV problem faced by token-based embedding models such as `word2vec` and `GloVe`, we leverage `sentencepiece` tokenizer with 32,000 vocabulary size. We train `word2vec` and `GloVe` on `sentencepiece` segmented corpus. We employ following hyper-parameter settings for `word2vec` and `GloVe` : the embedding dimension size of 300, window size as 11, minimum word count for vocabulary as 1 and the number of epochs equal to 80. We keep all the remaining parameters same as used by original authors of the respective works. Next, we use raw sentences (instead of using `sentencepiece` tokenized data) for training `FastText` and `CharLM`. We use following hyper-parameter setting for `FastText` : the embedding dimension of 300, window size of 11, the number of epochs 80, the minimum count for deciding vocabulary equals to 6, negative sampling loss with 5 negative samples, the learning rate as 0.025, minimum length of n-grams as 3 and maximum length of n-grams as 11. For `CharLM`, we use default parameters of small model architecture as Kim et al. (2016).[11] For `ELMo`, we use hyper-parameter settings of small model architecture.[12] For `ALBERT`, we use the same hyper-parameters as `albert-base-V2` model architecture except vocabulary size of 32,000 and the number of hidden layers equal to 6.

Results: Table 3 reports results on all the intrinsic tasks. We evaluate contextual models without context to assess how effective they are when compared with static models. Applying *contextualized* models without context is similar to just using their underlying static word representation. `ELMo` outperforms all the models in all the intrinsic tasks with a large margin except the semantic categorization task. Surprisingly, with limited training data, the contextualized `ELMo` model achieved impressive overall performance compared to static models across all intrinsic tasks. The subword level segmentation with the `sentencepiece` model empowered the `word2vec` and `GloVe` to such an extent that they outperform `FastText` in analogy prediction tasks and on par in the remaining tasks. For resource-rich languages, `ALBERT` have shown the state of the art performance on many evaluation tasks. However, despite a thorough hyper-parameter search, the performance of `ALBERT` could not compete with static models. This may be due to a lack of massive training data. Similarly, except relatedness and syntactic categorization, `CharLM` model shows low performance with a large margin.

	Relatedness	Categorization		Similarity	Analogy	
Model	f-score	syn	sem	acc	syn	sem
word2vec	30.90	0.22	0.41	37.34	30.65	11.01
GloVe	31.50	0.25	0.39	37.41	30.04	14.01
FastText	31.20	0.14	**0.47**	37.87	16.65	6.01
CharLM	35.21	0.33	0.28	35.94	2.11	0.03
ELMo	**37.17**	**0.86**	0.41	**42.15**	**56.80**	**32.70**
ALBERT	33.00	0.27	0.34	32.70	25.86	0.10

Table 3: Results on intrinsic evaluation tasks. For similarity and analogy prediction tasks, we use accuracy (acc) in terms of percentage as the evaluation metric, purity for categorization and f-score for the relatedness task. The syntactic and semantic tasks are denoted by *syn* and *sem* keywords, respectively.

[11]`https://s3.amazonaws.com/models.huggingface.co/bert/albert-base-v2-config.json`

[12]`https://s3-us-west-2.amazonaws.com/allennlp/models/elmo/2x1024_128_2048cnn_1xhighway/elmo_2x1024_128_2048cnn_1xhighway_options.json`

6 Analysis

6.1 Error Analysis

We conduct a detailed analysis of prominent mistakes done by the word embedding models. For the analogy prediction task, we find that all the models fail to correctly predict analogy questions where the root word gets modified while adding the affix to the root word. For example, in this analogy question (*lip : lepsyāmahe :: gur : gorişyāmahe*) while adding *syāmahe* the root gets modified. On the other hand, all the models perform surprisingly well for analogy questions where affix is added without modifying root word (*śak : śaknutam :: dagh : daghnutam*). In the incorrect answers of the subword enriched `FastText` model, at least the root word is predicted correctly. `CharLM` performs the worst among all models and the predictions made by the model seem almost random with no relation to the root word or the suffix. `CharLM` learns a character level composition function to generate word-level representation. Therefore, it often predicts the correct root with an incorrect suffix or no suffix. While in the case of syntactic analogy prediction task, the `GloVe` and `word2vec` perform comparably. However, the improved performance of `GloVe` at semantic analogy task can be attributed to consideration of the global statistics of word occurrences. We observe that all the embedding models majorly get those test cases correct for similarity and relatedness tasks, which have words already present in the model vocabulary. This indicates that these models are only capable of generating representation for OOV words based on orthographic similarity and fail to capture semantic meanings. Specifically, around 55% words present in the relatedness test set are OOV words.[13]

6.2 Qualitative inspection for rare word

We analyze the word representations captured by different models. We consider *contextualized* models also in this experiment. We use contextual models without a context to assess how effective are they when compared with static models. Applying *contextualized* models without context is very similar to just using their underlying static word representation. For systematic investigation, we choose three representative words from following categories: (1) In Vocabulary words (2) Out of Vocabulary words. Table 4 illustrates the top three nearest neighbors based on cosine similarity between embeddings learned by different models. For frequently occurring words present in vocabulary all the embeddings except `CharLM` and `ALBERT` are able to capture lexical meaning. The quality of nearest neighbour candidates of `CharLM` and `ALBERT` is relatively poor compared to other models. We suspect that the possible reason for this behaviour may be due to less amount of training data. The subword level modeling in `word2vec`, `GloVe` and `FastText` makes these models more inclined to find words which are orthographically similar. For these models, the representations seems to depend on surface string. For example, the nearest neighbours of *nityam* and *sukham* are very close in terms of edit distance. On the other hand, `ELMo` not only captures semantic level features but also orthographic features. This is very well evident from the following examples. The nearest neighbours produced by `ELMo` for *nityam* and *sukham* are lexically similar but orthographically different in terms of edit distance. However, the nearest neighbours of *tābhyām* exhibits suffix similarity. The last three words are OOV where *ahham*, *bahūhuni* are words with incorrect spelling and *sahasra-cakṣo* is the compound word. For OOV words also similar trend holds. We generate representation for OOV for `word2vec` and `GloVe` using `sentencepiece` tokenizer. This tokenizer breaks OOV into subwords available in `sentencepiece` model vocabulary and representation of OOV word is sum of representations of these segmented subwords.

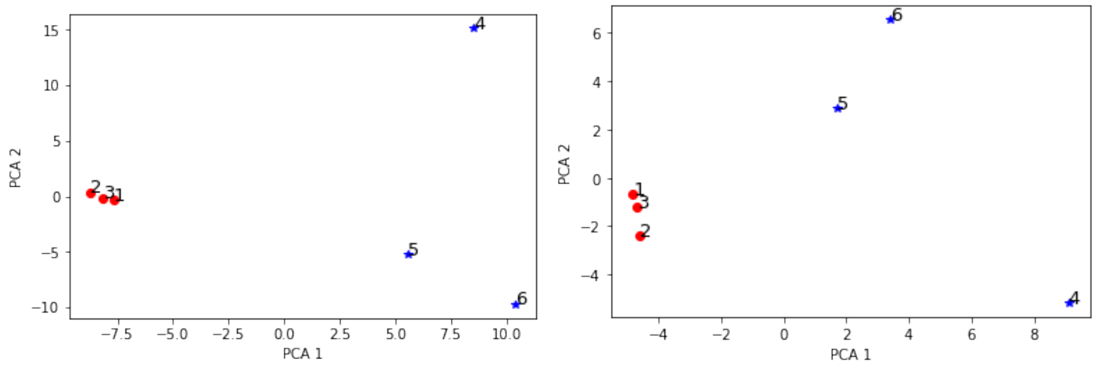
6.3 Qualitative inspection of contextualized embeddings

To investigate the contextualized nature of `ELMo` and `ALBERT`, we analyze the vector representation (using PCA to project the vectors into 2D space) of a two polysemous words in different

[13]Here, OOV referred as words which are not seen during training of embedding model.

	Vocabulary words			**OOV words**		
	nityam	*sukham*	*tābhyām*	*ahham*	*bahūhuni*	*sahasra-cakṣo*
word2vec	nityam-eva	sukhamayaṃ	āvābhyām	allāh	mānini	pracakṣva
	nityamiti	duḥkham	dvābhyām	ahne	sani	suparṇāḥ-ca
	sadā	sukha-duḥkham	tayā	ahrasat	cūrṇitāni	pracakṣmahe
GloVe	nityamiti	sukhamayaṃ	adhikāribhyām	allāh	bahū	tat-ca-eva
	nityam-eva	duḥkham	pārṣṇibhyām	ahne	behulā	tathā-ca
	sadā	duḥkhameva	jaṅghābhyām	ahnuvi	behulāyāḥ	yat-ca-eva
FastText	sadā	su-sukham	tayoḥ	simham	bahuni	sahasraḷaṃṇtra
	satatam	duḥkham	tau	tam	mandmatyoḥ	sahasra-kṛtvas
	nitya-snāyī	sukha-duḥkham	cakṣuḍbhyām	yathā-aham	bahavaḥ-tatra	sahasra-vedhī
CharLM	sṭeḍiyam	somam	sabhāyāḥ	īḍyam	vatsyati	paryaṭanasthaleṣu
	samśayam	saudham	nyāyasabhāyāḥ	bṛhantam	yotsyati	naṣṭa-saṃjñaḥ
	yāpayan	homam	hastābhyām	gayam	setsyati	cākṣuṣasya-antare
ELMO	martyam	sakham	nābhyām	arham	vilāsini	ajāta-śatro
	niṣṭham	kulam	yābhyām	aṅkam	mattakāśini	sahasra-śīrṣā
	yugyam	sukhām	bhūbhyām	andham	jānuni	ahu-rūpaḥ
ALBERT	dānam	satām	ardham	ahrasam	ityadīni	ūrdhva-ge
	loke	padam	mandam	allāh	hrasāni	sahasra-śīrṣā
	bhavitā	kṣayam	dhairyam	ahnuthāḥ	maṃdiram	sarva-mahīkṣitām

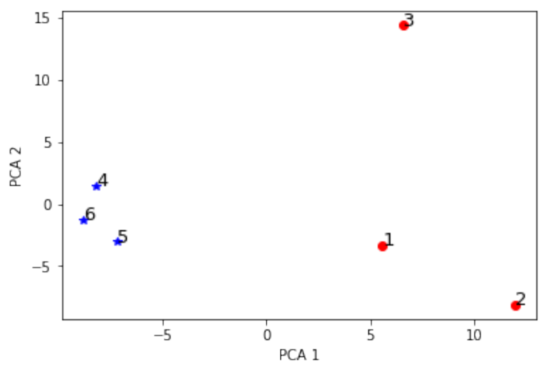
Table 4: Nearest neighbours (based on cosine similarity) for in vocabulary and OOV words for the models trained on Sanskrit corpus. Last three words are OOV words. We use contextual models without a context. Applying *contextualized* models without context is very similar to just using their underlying static word representation.



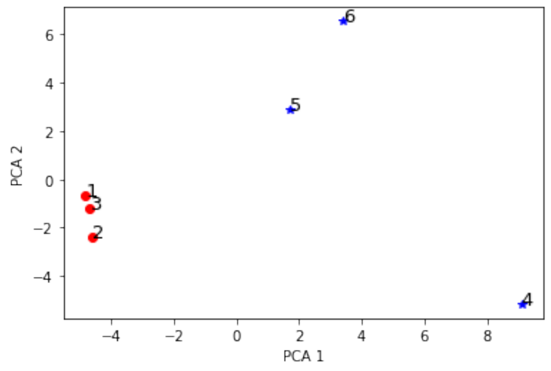
(a) (b)

Figure 1: Projection of contextualized representation (the left side : `ELMo` and the right side `ALBERT`) for different sense of polysemous word *cakrī* denoted by following sentences: *(1) cakrī mṛdā jalena ca kumbhanirmāṇe viśāradaḥ (2) cakrī kevala jala mṛd cakra upajīvakaḥ (3) cakrī cakrasya upari mṛdaḥ kumbhaṃ nirmāti (4) svahaste bibharti sudarśanam sa cakrī (5) sudarśanena cakrī śiśupālam jaghāna (6) rakṣobhyaḥ svabhaktān trātum cakrī sudarśanam bibharti.*

context. In this experiment, each test case consists of 6 sentences in Sanskrit such that all sentences have one common polysemous word. Out of 6 sentences, 3 sentences represent one sense of meaning and the remaining represent another sense of meaning for that common polysemous



(a)



(b)

Figure 2: Projection of contextualized representation (the left side : `ELMo` and the right side `ALBERT`) for different sense of polysemous word *hariḥ* denoted by following sentences: *(1) vāyusutaḥ hariḥ vānarāṇām toṣam vavardha (2) sugrīvaḥ nāma hariḥ vānarāṇām rājā āsīt (3) vānarāṇām vṛddhaḥ hariḥ jāmbavān (4) hariḥ baleḥ trailokyasampadam chalena jahāra (5) hariḥ hiraṇyākṣam nihatya devatānām khedam jahāra (6) hariḥ rukmiṇīm jahāra paśyatām*

word. Ideally, contextualized word vectors representing the same sense of meaning should form a cluster. For the polysemous word *cakrī*, we consider two senses of meaning, namely, potter and the Lord Viṣṇu. This test set consists of 6 sentences as follows: *(1) cakrī mṛdā jalena ca kumbhanirmāṇe viśāradaḥ (2) cakrī kevala-jala-mṛd-cakra-upajīvakaḥ (3) cakrī cakrasya upari mṛdaḥ kumbhaṃ nirmāti (4) svahaste bibharti sudarśanam sa cakrī (5) sudarśanena cakrī śiśupālam jaghāna (6) rakṣobhyaḥ svabhaktān trātum cakrī sudarśanam bibharti.* For the first three sentences, the sense of the meaning of the word *cakrī* is a potter and for the remaining sentences, the sense of meaning is the Lord Viṣṇu. Figure 1 illustrates visualization of polysemous word *cakrī*. Here, we observe that the contextualized representation indicating the same sense of meaning are clustered together. Similarly, we consider another polysemous word *hariḥ* consists of two senses of meaning, namely, monkey and the Lord Viṣṇu. *(1) vāyusutaḥ hariḥ vānarāṇām toṣam vavardha (2) sugrīvaḥ nāma hariḥ vānarāṇām rājā āsīt (3) vānarāṇām vṛddhaḥ hariḥ jāmbavān (4) hariḥ baleḥ trailokyasampadam chalena jahāra (5) hariḥ hiraṇyākṣam nihatya devatānām khedam jahāra (6) hariḥ rukmiṇīm jahāra paśyatām.* In Figure 2, we find the similar trend for this test case. The qualitative analysis suggests that the *contextualized* representations generated by Peters et al. (2018) and Lan et al. (2020) for polysemous words indicating the same sense of meaning forms a cluster for each sense of meaning (Section 6.3).

7 Conclusion and future work

In this work, we focused on evaluating word embedding approaches (initially proposed for resource-rich language English) for the Sanskrit language. Word embedding helps to transfer knowledge learned from readily available unlabelled data for improving the task-specific performance of data-driven approaches. We investigat the effectiveness of word embedding approaches for Sanskrit. To facilitate systematic experimentation, we classify word embeddings in the broad categories and evaluated on 4 intrinsic tasks, namely, relatedness, categorization, similarity identification and analogy prediction tasks. This work can be considered as the fertile soil for investigating the following questions: (1) Which linguistic phenomenons are captured by word embeddings? (2) Out of existing word embedding approaches, which embeddings are more suitable for Sanskrit? (3) What are the shortcomings of existing approaches specific to Sanskrit? Surprisingly, with limited training data, the Peters et al. (2018) achieved impressive overall performance compared to static models across all intrinsic tasks. The qualitative analysis suggests that the *contextualized* representations generated by Peters et al. (2018) and Lan et al. (2020) for polysemous words indicating the same sense of meaning forms a cluster for each sense of meaning (Section 6.3). This is indeed a serendipitous outcome for such data-hungry models when trained on negligibly small amount of corpus available for Sanskrit.

There are many limitations to this study. While in this work, we restrict our study to intrinsic evaluation tasks for the standard word embedding approaches with their default setting. We plan to extend this work on extrinsic evaluation tasks as well. Moreover, to get substantial empirical evidence on linguistics phenomenons captured by these models, we plan to evaluate them for context-sensitive tasks similar to Pilehvar and Camacho-Collados (2019). We find that all the models perform poorly on relatedness and similarity tasks since most of the words present in these evaluation test data are OOV words. However, these models perform relatively well for the words present in model vocabulary. This observation suggests investigating: What is the effect of the corpus size? We currently train our models with around 5 million tokens, which is negligible compared to the 840 billion tokens used by the resource-rich language models. We plan to investigate on applicability of the existing multi-lingual pretrained models for Sanskrit.

Acknowledgements

We are grateful to Amba Kulkarni for the discussion related to evaluation task decisions. We thank A V S D S Mahesh for helping us with the data preparation step. We thank the anonymous reviewers for their constructive feedback towards improving this work. The TCS Fellowship supports the work of the first author under the Project Number TCS/EE/2011191P.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and WordNet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27, Boulder, Colorado, June. Association for Computational Linguistics.

Amir Bakarov. 2018. A survey of word embeddings evaluation methods.

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland, June. Association for Computational Linguistics.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. A framework for the construction of monolingual and cross-lingual word similarity datasets. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 1–7, Beijing, China, July. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 160–167, New York, NY, USA. Association for Computing Machinery.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12(null):2493–2537, November.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Pawan Goyal and Gérard Huet. 2016. Design and analysis of a lean interface for sanskrit corpus annotation. *Journal of Language Modelling*, 4:145, 10.

Pawan Goyal, Gérard Huet, Amba Kulkarni, Peter Scharf, and Ralph Bunker. 2012. A distributed platform for Sanskrit processing. In *Proceedings of COLING 2012*, pages 1011–1028, Mumbai, India, December. The COLING 2012 Organizing Committee.

Ashim Gupta, Amrith Krishna, Pawan Goyal, and Oliver Hellwig. 2020. Evaluating neural morphological taggers for Sanskrit. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 198–203, Online, July. Association for Computational Linguistics.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Samer Hassan and Rada Mihalcea. 2011. Semantic relatedness using salient semantic analysis. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, page 884–889. AAAI Press.

Benjamin Heinzerling and Michael Strube. 2018. BPEmb: Tokenization-free pre-trained subword embeddings in 275 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May. European Language Resources Association (ELRA).

Oliver Hellwig and Sebastian Nehrdich. 2018. Sanskrit word segmentation using character-level recurrent and convolutional neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2754–2763, Brussels, Belgium, October-November. Association for Computational Linguistics.

Oliver Hellwig, Salvatore Scarlata, Elia Ackermann, and Paul Widmer. 2020. The treebank of vedic Sanskrit. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5137–5146, Marseille, France, May. European Language Resources Association.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July. Association for Computational Linguistics.

Gérard Huet and Pawan Goyal. 2013. Design of a lean interface for sanskrit corpus annotation. 12.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online, July. Association for Computational Linguistics.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2741–2749. AAAI Press.

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)*, pages 1–10, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Amrith Krishna, Pavan Kumar Satuluri, and Pawan Goyal. 2017. A dataset for Sanskrit word segmentation. In *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 105–114, Vancouver, Canada, August. Association for Computational Linguistics.

Amrith Krishna, Bishal Santra, Sasi Prasanth Bandaru, Gaurav Sahu, Vishnu Dutt Sharma, Pavankumar Satuluri, and Pawan Goyal. 2018. Free as in free word order: An energy based model for word segmentation and morphological tagging in Sanskrit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2550–2561, Brussels, Belgium, October-November. Association for Computational Linguistics.

Amrith Krishna, Vishnu Sharma, Bishal Santra, Aishik Chakraborty, Pavankumar Satuluri, and Pawan Goyal. 2019. Poetry to prose conversion in Sanskrit as a linearisation task: A case for low-resource languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1160–1166, Florence, Italy, July. Association for Computational Linguistics.

Amrith Krishna, Ashim Gupta, Deepak Garasangi, Jivnesh Sandhan, Pavankumar Satuluri, and Pawan Goyal. 2020a. Neural approaches for data driven dependency parsing in sanskrit.

Amrith Krishna, Ashim Gupta, Deepak Garasangi, Pavankumar Satuluri, and Pawan Goyal. 2020b. Keep it surprisingly simple: A simple first order graph based parsing model for joint morphosyntactic parsing in Sanskrit. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4791–4797, Online, November. Association for Computational Linguistics.

Amrith Krishna, Bishal Santra, Ashim Gupta, Pavankumar Satuluri, and Pawan Goyal. 2020c. A graph based framework for structured prediction tasks in sanskrit. *Computational Linguistics*, 46(4):1–63, December.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November. Association for Computational Linguistics.

Malhar Kulkarni, 2017. *Sanskrit WordNet at Indian Institute of Technology (IITB) Mumbai*, pages 231–241. Springer Singapore, Singapore.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26:3111–3119.

Sivaja Nair and Amba Kulkarni. 2010. The knowledge structure in amarakosa. pages 173–189, 01.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June. Association for Computational Linguistics.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2020. Embeddings in natural language processing: Theory and advances in vector representations of meaning. *Synthesis Lectures on Human Language Technologies*, 13(4):1–175.

Vikas Reddy, Amrith Krishna, Vishnu Sharma, Prateek Gupta, Vineeth M R, and Pawan Goyal. 2018. Building a word segmenter for Sanskrit overnight. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May. European Language Resources Association (ELRA).

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633, October.

Jivnesh Sandhan, Amrith Krishna, Pawan Goyal, and Laxmidhar Behera. 2019. Revisiting the role of feature engineering for compound type identification in Sanskrit. In *Proceedings of the 6th International Sanskrit Computational Linguistics Symposium*, pages 28–44, IIT Kharagpur, India, October. Association for Computational Linguistics.

Jivnesh Sandhan, Amrith Krishna, Ashim Gupta, Laxmidhar Behera, and Pawan Goyal. 2021. A little pretraining goes a long way: A case study on dependency parsing task for low-resource morphologically rich languages.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515, Austin, Texas, November. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.