Creation of a Digital Rig Vedic Index (Anukramani) for Computational Linguistic Tasks

A V S D S Mahesh and **Arnab Bhattacharya**
Dept. of Computer Science and Engineering
Indian Institute of Technology Kanpur
maheshak@cse.iitk.ac.in, arnabb@cse.iitk.ac.in

Abstract

Rig Veda, the oldest text in Vedic Sanskrit, is a collection of hymns addressed to various *devatā*s. We present an index of Rig Vedic verses along with the respective *devatā*, *ṛṣi* and *chandas* which is, in short, a digitized form of the well known Rig Vedic *Anukramaṇī*. We then build several deep learning based language models from the corpus. The models are tested on a downstream classification task that predicts the *devatā* associated with a given verse. The six classifiers we test are based on FastText, ELMo, ALBERT, Word2Vec, GloVe and RoBERTa language models pretrained on the GRETIL data. We find that RoBERTa outperforms the other models on this task and achieves an accuracy of 77.3%.

***Keywords*—** Vedic Sanskrit, Rig Vedic Anukramani, Text Classification

1 Introduction

The *Anukramaṇī* is an index of the Rig Vedic hymns; it contains the details about the *author* (**ṛṣi**), the *divinity* that a hymn is dedicated to (**devatā**), and the *meter* (**chandas**) for each hymn. The widely found *Anukramaṇī* is that of *Kātyāyana* and is known as the *Sarvānukramaṇī* (Macdonell, 1885). The details present are considered to be accurate to a good extent (Jamison and Brereton, 2014). In particular, the *devatā* addressed can easily be cross-checked from the content of the hymns. To the nest of our knowledge, we are the first to present this data in a machine-readable format. One may find in the text form of Griffith's Rig Veda (Griffith, 1889) an index of hymns each labeled with the *devatā*. However, these labels are not verse-wise and the *devatā* addressed is often not the same for the entire hymn. Hence, to our consensus, the data we provide is not available before. We also referred the Maharishi International University's Vedic Reserve's[1] materials in the process of annotation. We use this information about the *devatā*s as a *text classification* dataset with the verses as input texts and their respective *devatā*s as labels.

1.1 Motivation

To be able to benchmark the performance of computational natural language models, it is necessary to judge their realiability in downstream tasks. The same applies to Vedic Sanskrit of Rig Veda, which is an ancient form of Sanskrit. We present one such benchmark dataset, the Rig Vedic *Anukramaṇī*. Although it has been available from historical times, it was not in any machine-readable form.

Thus, upon this dataset, we run six classifiers that respectively use the embeddings from FastText (Joulin et al., 2016; Bojanowski et al., 2016), ELMo (Peters et al., 2018), ALBERT (Lan et al., 2019), Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and RoBERTa (Liu et al., 2019) pretrained language models. We carried out the pretraining of FastText, ELMo, ALBERT and RoBERTa based models on the GRETIL data[2] and whereas pretraining of Word2Vec and GloVe are carried out on lemmatized text from Digital Corpus of Sanskrit (DCS) (Hellwig, 2010–2021). The results thus obtained show that the transformer (Vaswani et al., 2017) based RoBERTa model outperforms the other ones in this classification task.

1.2 Contributions

Our contributions in this work are two-fold:

[1] https://vedicreserve.miu.edu/
[2] http://gretil.sub.uni-goettingen.de/gretil.html

Proceedings of the Computational Sanskrit and Digital Humanities, 18^{th} *World Sanskrit Conference 2023*, pages 89–96

1. We present a digital Rig Vedic index (*Anukramaṇī*) dataset consisting of *devatā*, *ṛṣi* and *chandas* information, labeled for each verse. This acts as a benchmark dataset in Vedic Sanskrit.
2. We apply six classifiers based on pretrained ALBERT, FastText, ELMo, Word2Vec, GloVe and RoBERTa embeddings on the above dataset.

The dataset and the models are available from https://github.com/mahesh-ak/WSC2023.

1.3 Organization

The rest of the paper is organized as follows. The details about the classification data are given in Section 2, while the information about the pre-training and training of the various classifiers applied are described in Section 3. The results obtained on the classifiers are discussed in Section 4 and, finally, the paper is concluded in Section 6.

2 Dataset

2.1 Rig Vedic *Anukramaṇī*

There are a total of 10,552 verses in the Rig Veda. Each verse is assigned with a *devatā* label. Although most hymns are associated with a single *devatā*, for example, *Agni*, there are many hymns where a verse has more than one *devatā* labels. Some have two *devatā*s addressed in dual tense, for example, *Indrāgnī*, i.e., *Indra* and *Agni*. Some have a special designated group of *devatā*s, for example, *Viśvedevā*s or the *ādityā*s. Some again may just be a case where multiple *devatā*s are addressed together, for example, *Iḷā-Sarasvatī-Bhāratī*. In these cases, we make no attempt to present these labels as a mixture of the compositional *devatā*s. Rather, we present them as they are. For example, a dual like *Mitrāvaruṇau* is not decomposed into *Mitra* and *Varuṇa*, but is treated as a single label only. In future, it will be worthwhile to present them in a decomposed format. Similary, a verse generally has only a single *ṛṣi*, but can have multiple *ṛṣi*s rarely. There is always a unique *chandas* of a hymn by definition of a meter. An example line from *Anukramaṇī* from first *Maṇḍala* and its annotation is demonstrated as follows:

(27) saptaviṃśaṃ sūktam
(1-13) trayodaśarcasyāsya sūktasyājīgartiḥ śunaḥśepa ṛṣiḥ ।
(1-12) prathamādidvādaśarcām agniḥ (13) trayodaśyāśca devā devatāḥ ।
(1-12) prathamādidvādaśarcāṃ gāyatrī (13) trayodaśyāśca triṣṭup chandasī

The annotation for the above is:

27.13.ājīgartiḥ śunaḥśepaḥ.(1-12)agniḥ,(13)devāḥ.(1-12)gāyatrī,(13)triṣṭup

The fields of hymn, verse, *ṛṣi*, *devatā* and *chandas* are separated by a period, whereas muliple entries within a field are seperated by a comma. This yields a format that can be easily processed in an automated fashion.

It can be seen that the original *anukramaṇī*, even if made available in a machine-readable format, requires manual annotation since the sentences are in a natural language, whose processing would, in turn, require a gold standard.

2.2 Classification Task

For the classification task, the focus is on a single label classification, that is, a single *devatā* prediction for a verse. We attempt to group *devatā*s that are fundamentally the same under one single label. For example, *Rakṣoghna Agni* is labeled as only *Agni*. However, although the classes labeled *Soma* and *Pavamāna Soma* may look like one one, we leave them unmerged since there is a subtle difference between the two classes. The hymns of *Pavamāna Soma* almost always invoke the purifying nature of *Soma* like *Somaḥ pavate* which is not the case with the class labeled just *Soma*. Additionally, the several *Dānastuti*s or the praises of patrons are labeled as just *Dānastuti* without the names of the patrons like *Paijavana Sudās*. This produces an overall list of 216 labels.

From this data, we consider the labels with at least 30 instances or verses for the classification task. This brings the label count down to 28. The number of instances corresponding to these 28 labels is

9,496, which is still a fairly large size. We keep aside 15% of the data consisting of 1,424 hymns as the *test* set. The rest of the data is split using a 90:10 ratio into *training* (7,265 hymns) and *validation* (807 hymns) sets. The splits are done in a *stratified* manner.

3 Training the Classifiers

As mentioned earlier, we evaluate on this task six classification models based on FastText, ELMo, ALBERT, Word2Vec, GloVe and RoBERTa. Training these models involve pre-training as a first step and then fine-tuning for the classification task at hand. These two steps are next discussed separately.

3.1 Pre-training

The models, except for Word2Vec and GloVe, are pre-trained on the GRETIL data[3] that consists of over 2000 Sanskrit texts. These are cleaned to remove pieces of text that are not in Sanskrit as much as possible to produce over 450 MB of textual data with about 45 million words. Note that a word here can refer to a compound word (*sandhi* or *samāsa* or *both*) as well. In case of Word2Vec and GloVe, the pretraining was done on lemmatized text available from the Digital Corpus of Sanskrit (DCS)(Hellwig, 2010–2021) corpus, consisting about 34 MB of text data and 5 million words. The pre-training is carried on an RTX 2080 GPU machine with 11 GB memory for ELMo, ALBERT and RoBERTa and on a 12-core CPU with 32 GB RAM for FastText, Word2Vec and GloVe. The details are discussed next.

1. **FastText** (Bojanowski et al., 2016) is a skip-gram based model that learns continuous word representations from character n-grams. It can be trained fast on a CPU unlike the deep neural network models that require GPU. In our case, it was trained in less than an hour on a 12-core CPU. The FastText software provided on the official website[4] is used.

2. **ELMo** (Peters et al., 2018) is a deep learning model consisting of CNN embedders of input characters and bidirectional LSTM encoders that train on the bidirectional language modeling task. ELMo demonstrated significantly the power of transfer learning in the realm of NLP. We use the small model with about 13 million parameters. It still took about 3 days to pre-train on the GPU. We used the official TensorFlow implementation of ELMo[5].

3. **ALBERT** (Lan et al., 2019) is a variant of the transformer (Vaswani et al., 2017) based BERT model (Devlin et al., 2018) with smaller number of parameters. It trains on a masked language modeling task. We use unigram language model tokenization (Kudo, 2018) with a vocabulary size of 5,000. Our model has about 8 million parameters (albert-base with vocabulary size 5,000) and the training converged in about 3 days on the GPU. We used the Huggingface (Wolf et al., 2019) library here.

4. **Word2Vec** (Mikolov et al., 2013) is a skip-gram based model that learns distributed word representations. In implementation, this is special case of FastText with window size parameter set to 0. So, the same FastText software is used. The training details are same as that of FastText.

5. **GloVe** (Pennington et al., 2014) is an unsupervised algorithm for learing word vector representations based on word-word co-occurrence statistics. Training details including hardware and time taken is similar to that of FastText. The GloVe software provided on the official website[6] is used.

6. **RoBERTa** (Lan et al., 2019) is a variant of the transformer (Vaswani et al., 2017) based BERT model (Devlin et al., 2018) with robust training method. It trains on a masked language modeling task like ALBERT. We use byte-pair encoding (BPE) tokenization (Sennrich et al., 2016) with variable vocabulary sizes of 2,500, 5,000, 10,000 and 20,000. The best-performing model has about 90 million parameters (roberta-base with vocabulary size 20,000) and the training converged in about 3 days on the GPU. We used the Huggingface (Wolf et al., 2019) library here.

[3] http://gretil.sub.uni-goettingen.de/gretil.html
[4] https://fasttext.cc
[5] https://github.com/allenai/bilm-tf
[6] https://nlp.stanford.edu/projects/glove/

3.2 Fine-tuning

We use the training classification data mentioned in Section 2 to fine-tune the pretrained models for the classification task. As previously mentioned, we consider only 28 labels for this task. The details for each the models including the architecture and algorithms are next discussed. The fine-tuning time for this task is always less than 15 minutes on respective hardwares.

- **FastText** classifier involves averaging the input word representations to form the hidden layer which is fed foward into a linear classifier (Joulin et al., 2016). Again the software provided in the official website is used here.
- **ELMo** embeddings of the input words learned previously, similar to above, are summed up to form the hidden layer which is fed forward into a linear classifier. This simple architecture is known to often outperform the complex ones (Perone et al., 2018). Here, we use the AllenNLP library (Gardner et al., 1803).
- **ALBERT** classifier takes as input the ALBERT final hidden representation layer of the CLS token and feeds it to a linear classifier. We use the implementation provided in the Huggingface library (Wolf et al., 2019).
- **Word2Vec, GloVe**: FastText classifier is used by loading pretrained vectors from Word2Vec and GloVe respectively. GloVe vectors are converted into Word2Vec format using the Gensim (Řehůřek and Sojka, 2010) library.
- **RoBERTa**: The details are similar to those of ALBERT.

4 Results

The F1 scores for each of the 28 classes is reported in Table 1. From overall macro-averaged F1 scores, it can be seen that RoBERTa performs the best. The RoBERTa here refers to the one with vocabulary size of 20K, which is the one that performs the best. The comparison of results of RoBERTa for different vocabulary sizes is given in Table 2.

We also report in Table 1, the performance of only the *under-represented* classes or in other words, the classes which see very few instances relative to the other classes. Here, we consider the under-represented classes as those which have less than overall 70 instances. That gives about 20-50 instances for training. The macro and weighted average F1 scores for only these under-represented classes are reported in the last two rows of Table 1. RoBERTa performs the best in the classification of under-represented classes as well except in case of the classes *Dānastuti*, *Aśva* and *Āpas*. Word2Vec outperforms RoBERTa in these classes. These results are discussed further in the following section.

It should be noted from Table 1 and Table 2 that RoBERTa outperforms ALBERT while having same vocabulary size of 5,000 as well. This is expected since ALBERT shares parameters across the layers and hence has small number of parameters to train.

From the confusion matrix of RoBERTa predictions provided in Figure 1, which is normalized across rows (i.e., true labels), one can see the patterns in wrong predictions. For instance, one can observe that various other labels are being predicted instead of the *Viśvedevā*s (index 3 on y-axis); this happens since *Viśvedevā*s is like an umbrella term for 'many' or rather 'all *devatā*s'. For instance, the following verse (RV 6.055.03) labeled as *Viśvedevā*s invokes *Aditi* and *Uṣāsānaktā*. The predicted label is *Uṣas* which is indeed close.

prá pastyà(3)m áditiṃ síndhum arkaíḥ svastím ī́le sakhyā́ya devī́m |
ubhé yáthā no áhanī nipā́ta uṣā́sānáktā karatām ádabdhe ||

Note that the accents are provided only for the purpose of reading. They are not considered while training and testing.

Also, one can observe that the class *Soma* (index 13 on y-axis) is often wrongly being predicted as *Pavamāna Soma* (index 2 on x-axis) which can be understandable in the sense that both are almost the

Class	Label	Count	ALBERT	FastText	ELMo	Word2Vec	GloVe	RoBERTa
1	Indra	2887	0.855	0.844	0.836	0.778	0.810	**0.858**
2	Agni	2000	0.789	0.804	0.785	0.667	0.756	**0.825**
3	Pavamāna Soma	1087	0.799	0.834	**0.890**	0.750	0.793	0.824
4	Viśvedevās	815	0.435	0.426	0.404	0.364	0.414	**0.519**
5	Aśvinau	631	0.840	0.807	0.819	0.783	0.791	**0.864**
6	Marutas	425	0.656	0.641	0.656	**0.823**	0.589	0.651
7	Mitrāvaruṇau	183	0.638	0.429	0.378	0.403	0.439	**0.653**
8	Uṣas	182	0.576	**0.739**	0.585	0.222	0.622	0.655
9	Indrāgnī	117	0.737	0.684	0.789	0.606	0.718	**0.829**
10	Ādityās	100	0.609	0.400	0.286	**0.762**	0.333	0.538
11	Varuṇa	99	**0.600**	0.545	0.222	0.222	0.471	0.522
12	Ṛbhavas	96	0.690	**0.833**	0.545	0.759	0.500	0.733
13	Savitṛ	83	0.714	0.545	0.250	**0.727**	0.222	0.667
14	Soma	80	0.308	0.417	0.444	0.300	0.316	**0.588**
15	Pūṣan	77	0.552	0.385	0.167	0.286	0.385	**0.625**
16	Bṛhaspati	74	**0.923**	0.600	0.800	0.667	0.857	0.833
17	Indrāvaruṇau	70	**0.923**	0.667	0.870	0.593	0.667	**0.923**
18	Dānastuti	65	0.286	0.250	0	**0.667**	0.154	0.167
19	Sūrya	62	0.500	0.316	0.471	0.417	0.353	**0.600**
20	Vāyu	53	0.667	0.588	0.182	0.556	0.667	**0.778**
21	Dyāvāpṛthivyau	48	0.444	0.400	0.286	0.524	0.500	**0.545**
22	Brahmaṇaspati	48	0.545	0.444	0.500	0.560	**0.667**	**0.667**
23	Āpas	45	0.286	0.364	0.333	**0.500**	0.286	0.286
24	Rudra	38	**0.667**	0.333	0	0.261	0.250	**0.667**
25	Aśva	35	0	0	0	**0.333**	0	0.200
26	Indravāyū	33	**1**	0.667	0.500	0.727	0.750	**1**
27	Sarasvatī	32	0.714	0.364	0	0.333	0.400	**0.750**
28	Viṣṇu	31	0.444	0.545	0	0.800	0.286	**1**
	Overall (Weighted)		*0.745*	*0.730*	*0.710*	*0.668*	*0.693*	***0.773***
	Overall (Macro)		*0.614*	*0.531*	*0.429*	*0.550*	*0.500*	***0.670***
	Few-instance classes (Weighted)		*0.490*	*0.382*	*0.221*	*0.519*	*0.392*	***0.573***
	Few-instance classes (Macro)		*0.505*	*0.388*	*0.207*	*0.516*	*0.392*	***0.605***

Table 1: F1 scores for the classification task

Vocabulary Size	2.5K	5K	10K	20K
Overall (Weighted)	0.774	0.766	**0.775**	0.773
Overall (Macro)	0.639	0.638	0.634	**0.670**
Few-instances (Weighted)	0.458	0.485	0.483	**0.573**
Few-instances (Macro)	0.488	0.513	0.515	**0.605**

Table 2: Performance of RoBERTa for different vocabulary sizes

same. Note that, by our assumptions, in this case, *Soma* and *Pavamāna Soma* should have been clubbed into a single class in the first place. However, this is not done in order to observe how the model would get confused of either label. Their difference is better captured by ELMo (see in Table 1).

Another interesting case is the class that is labeled as *Aśva*. This occurs in a single hymn (RV 1.162) which talks about a sacrifice of a horse. Except for Word2Vec and RoBERTa, none of the other models appear to have learned anything of this class. From the confusion matrix (Figure 1), the label *Aśva* (index 24 on y-axis) is wrongly predicted as *Agni* often. The following is one such instance (RV 1.162.15):

mā́ tvāgnír dhvanayīd dhūmágandhir mókhā bhrā́janty abhí vikta jághriḥ ।
iṣṭáṃ vītám abhígūrtaṃ váṣaṭkṛtaṃ táṃ devā́saḥ práti gṛbhṇanty áśvam ॥

This is not surprising since the verse not only talks about *Agni*, which is the Sacred Fire, but also invokes many concepts and terms related to the Fire such as the word *vaṣaṭkṛta*, an exclamation made while making an offering to the Fire.

The class *Āpas* or the Waters is also poorly predicted by RoBERTa. A possible factor attributing to this

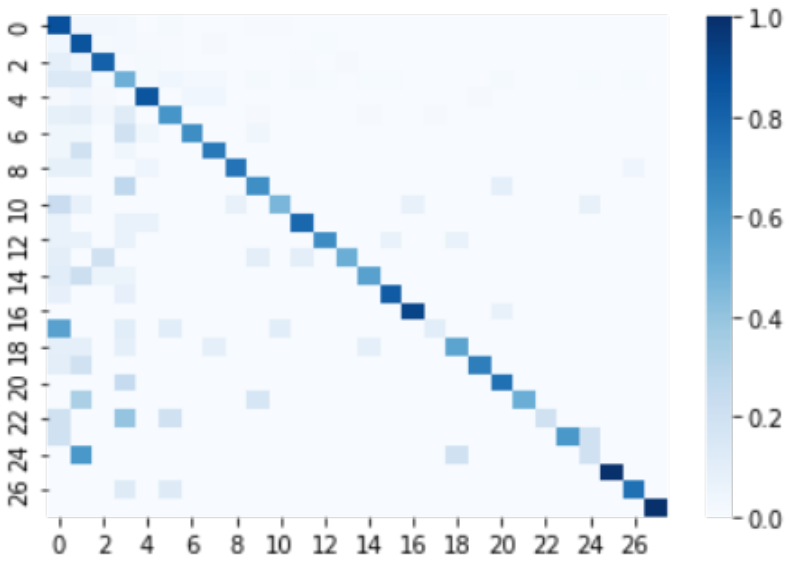

Figure 1: Confusion matrix with label indices for the best model, i.e., RoBERTa (see Table 1)

result could be that compared to other few-instance classes like *Rudra*, *Viṣṇu* etc., the word *āpaḥ* which means water appear quite regulary throughout the Rig Veda and this is not the case with aforementioned classes. Apart from these classes, the class of *Dānastuti* hymns, which are dedicated to some patron describing their generous donations, are also poorly predicted by RoBERTa. This is not surprising since this class denotes an abstract meaning that is not limited to a single word. Word2Vec surprising learns these classes to a good extent.

It should also be noted that Word2Vec outperforms RoBERTa with vocabulary sizes other than 20K in the case of under-estimated classes. This is likely due to the fact that Word2Vec has an advantage of being trained and tested on a lemmatized dataset. The behaviour of Word2Vec on *Dānastuti*, a semantically difficult class is noteworthy and a potential topic of future exploration. ELMo, probably owing to that fact that it starts from the character level, is outperformed by all others in case of few-instance classes. On the other hand, this does not happen with FastText since it considers a higher window of size 2-5 in our case.

5 Related Work

A transformer based architecture was tested for Sentence segmentation in (Hellwig and Nehrdich, 2018) probably for the first time for Sanskrit. Classification task related to compounds were seen in (Krishna et al., 2016; Sandhan et al., 2019) where use of neural network based models like Word2Vec, GloVe and LSTM are found. Various neural embeddings are explored for Classical Sanskrit in (Sandhan et al., 2021).

6 Conclusions

In this paper, we have produced a machine readable form of Rig Vedic *Anukramaṇī* that can be used as a benchmark dataset in Vedic Sanskrit Compuational Linguistics. Also, we have benchmarked six neural network based models namely, FastText, ELMo, ALBERT, Word2Vec, GloVe and RoBERTa on this dataset and found that the transformer-based RoBERTa outperforms the other models, especially in case of classification of under-represented classes. It is desirable to benchmark the BERT based models on more such tasks. Improving the classification scores can be taken up as a future task. As it was pointed out earlier, study of semantically difficult classes like *Dānastuti* can also be a potential future task.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 1803. Allennlp: a deep semantic natural language processing platform (2018). *arXiv preprint arXiv:1803.07640*.

Ralph Thomas Hotchkin Griffith. 1889. *The Hymns of the Rig Veda*, volume 1-4. Benares : E.J. Lazarus.

Oliver Hellwig. 2010–2021. *The Digital Corpus of Sanskrit (DCS)*.

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. Association for Computational Linguistics.

Stephanie W Jamison and Joel P Brereton. 2014. *The Rigveda: the earliest religious poetry of India*, volume 1. South Asia Research.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

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. The COLING 2016 Organizing Committee.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Arthur Anthony Macdonell. 1885. *Die Sarvanukramani des Katyayana zum Rigveda, zum ersten mal mit kritischen anmerkungen hrsg. von Arthur Macdonell.* Ph.D. thesis, Leipzig.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Neural and Information Processing System (NIPS)*.

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. Association for Computational Linguistics.

Christian S. Perone, Roberto Silveira, and Thomas S. Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks.

Matthew E. 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. Association for Computational Linguistics.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Jivnesh Sandhan, Om Adideva, Digumarthi Komal, Laxmidhar Behera, and Pawan Goyal. 2021. Evaluating neural word embeddings for sanskrit. *arXiv preprint arXiv:2104.00270*.

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. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.