iNLTK: Natural Language Toolkit for Indic Languages

Gaurav Arora
Jio Haptik
`gaurav@haptik.ai`

Abstract

We present iNLTK, an open-source NLP library consisting of pre-trained language models and out-of-the-box support for Data Augmentation, Textual Similarity, Sentence Embeddings, Word Embeddings, Tokenization and Text Generation in 13 Indic Languages. By using pre-trained models from iNLTK for text classification on publicly available datasets, we significantly outperform previously reported results. On these datasets, we also show that by using pre-trained models and data augmentation from iNLTK, we can achieve more than 95% of the previous best performance by using less than 10% of the training data. iNLTK is already being widely used by the community and has 40,000+ downloads, 600+ stars and 100+ forks on GitHub. The library is available at https://github.com/goru001/inltk.

1 Introduction

Deep learning offers a way to harness large amounts of computation and data with little engineering by hand (LeCun et al., 2015). With distributed representation, various deep models have become the new state-of-the-art methods for NLP problems. Pre-trained language models (Devlin et al., 2019) can model syntactic/semantic relations between words and reduce feature engineering. These pre-trained models are useful for initialization and/or transfer learning for NLP tasks. Pre-trained models are typically learned using unsupervised approaches from large, diverse monolingual corpora (Kunchukuttan et al., 2020). While we have seen exciting progress across many tasks in natural language processing over the last years, most such results have been achieved in English and a small set of other high-resource languages (Ruder, 2020).

Indic languages, widely spoken by more than a billion speakers, lack pre-trained deep language

models, trained on a large corpus, which can provide a headstart for downstream tasks using transfer learning. Availability of such models is critical to build a system that can achieve good results in "low-resource" settings - where labeled data is scarce and computation is expensive, which is the biggest challenge for working on NLP in Indic Languages. Additionally, there's lack of Indic languages support in NLP libraries like spacy[1], nltk[2] - creating a barrier to entry for working with Indic languages.

iNLTK, an open-source natural language toolkit for Indic languages, is designed to address these problems and to significantly lower barriers to doing NLP in Indic Languages by

- sharing pre-trained deep language models, which can then be fine-tuned and used for downstream tasks like text classification,

- providing out-of-the-box support for Data Augmentation, Textual Similarity, Sentence Embeddings, Word Embeddings, Tokenization and Text Generation built on top of pre-trained language models, lowering the barrier for doing applied research and building products in Indic languages

iNLTK library supports 13 Indic languages, including English, as shown in Table 2. GitHub repository[3] for the library contains source code, links to download pre-trained models, datasets and API documentation[4]. It includes reference implementations for reproducing text-classification results shown in Section 2.4, which can also be easily adapted to new data. The library has a permissive MIT License and is easy to download and install via pip or by cloning the GitHub repository.

---

[1]https://spacy.io/
[2]https://www.nltk.org/
[3]https://github.com/goru001/inltk
[4]https://inltk.readthedocs.io/

Language	# Wikipedia Articles		# Tokens	
	Train	Valid	Train	Valid
Hindi	137,823	34,456	43,434,685	10,930,403
Bengali	50,661	21,713	15,389,227	6,493,291
Gujarati	22,339	9,574	4,801,796	2,005,729
Malayalam	8,671	3,717	1,954,174	926,215
Marathi	59,875	25,662	7,777,419	3,302,837
Tamil	102,126	25,255	14,923,513	3,715,380
Punjabi	35,637	8,910	9,214,502	2,276,354
Kannada	26,397	6,600	11,450,264	3,110,983
Oriya	12,446	5,335	2,391,168	1,082,410
Sanskrit	18,812	6,682	11,683,360	4,274,479
Nepali	27,129	11,628	3,569,063	1,560,677
Urdu	107,669	46,145	15,421,652	6,773,909

Table 1: Statistics of Wikipedia Articles Dataset used for training Language Models

Language	Code	Language	Code
Hindi	hi	Marathi	mr
Punjabi	pa	Bengali	bn
Gujarati	gu	Tamil	ta
Kannada	kn	Urdu	ur
Malayalam	ml	Nepali	ne
Oriya	or	Sanskrit	sa
English	en		

Table 2: Languages supported in iNLTK

Language	Vocab size	Language	Vocab size
Hindi	30,000	Marathi	30,000
Punjabi	30,000	Bengali	30,000
Gujarati	20,000	Tamil	8,000
Kannada	25,000	Urdu	30,000
Malayalam	10,000	Nepali	15,000
Oriya	15,000	Sanskrit	20,000

Table 3: Vocab size for languages supported in iNLTK

2 iNLTK Pretrained Language Models

iNLTK has pre-trained ULMFiT (Howard and Ruder, 2018) and TransformerXL (Dai et al., 2019) language models for 13 Indic languages. All the language models (LMs) were trained from scratch using PyTorch (Paszke et al., 2017) and Fastai[5], except for English. Pre-trained LMs were then evaluated on downstream task of text classification on public datasets. Pre-trained LMs for English were borrowed from Fastai directly. This section describes training of language models and their evaluation.

2.1 Dataset preparation

We obtained a monolingual corpora for each one of the languages from Wikipedia for training LMs from scratch. We used the wiki extractor[6] tool and BeautifulSoup[7] for text extraction from Wikipedia. Wikipedia articles were then cleaned and split into train-validation sets. Table 1 shows statistics of

the monolingual Wikipedia articles dataset for each language. Hindi Wikipedia articles dataset is the largest one, while Malayalam and Oriya Wikipedia articles datasets have the least number of articles.

2.2 Tokenization

We create subword vocabulary for each one of the languages by training a SentencePiece[8] tokenization model on Wikipedia articles dataset, using unigram segmentation algorithm (Kudo and Richardson, 2018). An important property of SentencePiece tokenization, necessary for us to obtain a valid subword-based language model, is its reversibility. We do not use subword regularization as the available training dataset is large enough to avoid overfitting. Table 3 shows subword vocabulary size of the tokenization model for each one of the languages.

2.3 Language Model Training

Our model is based on the Fastai implementation of ULMFiT and TransformerXL. Hyperparameters

---

[5] https://github.com/fastai/fastai
[6] https://github.com/attardi/wikiextractor
[7] https://www.crummy.com/software/BeautifulSoup

[8] https://github.com/google/sentencepiece

Language	Dataset	FT-W	FT-WC	INLP	iNLTK
Hindi	BBC Articles	72.29	67.44	74.25	**78.75**
	IITP+Movie	41.61	44.52	45.81	**57.74**
	IITP Product	58.32	57.17	63.48	**75.71**
Bengali	Soham Articles	62.79	64.78	72.50	**90.71**
Gujarati		81.94	84.07	90.90	**91.05**
Malayalam	iNLTK	86.35	83.65	93.49	**95.56**
Marathi	Headlines	83.06	81.65	89.92	**92.40**
Tamil		90.88	89.09	93.57	**95.22**
Punjabi	IndicNLP News	94.23	94.87	96.79	**97.12**
Kannada	Category	96.13	96.50	97.20	**98.87**
Oriya		94.00	95.93	98.07	**98.83**

Table 4: Text classification accuracy on public datasets

Language	Perplexity	
	ULMFiT	TransformerXL
Hindi	34.0	26.0
Bengali	41.2	39.3
Gujarati	34.1	28.1
Malayalam	26.3	25.7
Marathi	17.9	17.4
Tamil	19.8	17.2
Punjabi	24.4	14.0
Kannada	70.1	61.9
Oriya	26.5	26.8
Sanskrit	5.5	2.7
Nepali	31.5	29.3
Urdu	13.1	12.5

Table 5: Perplexity on validation set of Language Models in iNLTK

Language	Dataset	N	# Examples	
			Train	Test
Hindi	BBC Articles	6	3467	866
	IITP+Movie	3	2480	310
	IITP Product	3	4182	523
Bengali	Soham Articles	6	11284	1411
Gujarati		3	5269	659
Malayalam	iNLTK	3	5036	630
Marathi	Headlines	3	9672	1210
Tamil		3	5346	669
Punjabi	IndicNLP	4	2496	312
Kannada	News	3	24000	3000
Oriya	Category	4	24000	3000

Table 6: Statistics of publicly available classification datasets (N is the number of classes)

of the final model are accessible from the GitHub repository of the library. Table 5 shows perplexity of language models on validation set. TransformerXL consistently performs better for all languages.

2.4 Text Classification Evaluation

We evaluated pre-trained ULMFiT language models on downstream task of text-classification using following publicly available datasets: (a) IIT-Patna Sentiment Analysis dataset (Akhtar et al., 2016), (b) BBC News Articles classification dataset[9], (c) iNLTK Headlines dataset[10], (d) Soham Bengali News classification dataset[11], (e) IndicNLP

News Category classification dataset (Kunchukuttan et al., 2020). Train and test splits, derived by the authors (Kunchukuttan et al., 2020) from the above mentioned corpora and used for benchmarking, are available on the IndicNLP corpus website[12]. Table 6 shows statistics of these datasets.

iNLTK results were compared against results reported in (Kunchukuttan et al., 2020) for pre-trained embeddings released by the Fast-Text project trained on Wikipedia (FT-W) (Bojanowski et al., 2016), Wiki+CommonCrawl (FT-WC) (Grave et al., 2018) and INLP embeddings (Kunchukuttan et al., 2020). Table 4 shows that iNLTK significantly outperforms other models across all languages and datasets[13].

Language	Dataset	# Training Examples		%age reduction	INLP Accuracy	iNLTK Accuracy		
		Full	Reduced		Full	Full	Reduced	
							Without Data Aug	With Data Aug
Hindi	IITP+Movie	2,480	496	80%	45.81	57.74	47.74	56.13
Bengali	Soham Articles	11,284	112	99%	72.50	90.71	69.88	74.06
Gujarati		5,269	526	90%	90.90	91.05	80.88	81.03
Malayalam	iNLTK	5,036	503	90%	93.49	95.56	82.38	84.29
Marathi	Headlines	9,672	483	95%	89.92	92.40	84.13	84.55
Tamil		5,346	267	95%	93.57	95.22	86.25	89.84
	Average	6514.5	397.8	91.5%	81.03	87.11	75.21	78.31

Table 7: Comparison of Accuracy on INLP trained on Full Training set vs Accuracy on iNLTK, using data augmentation, trained on reduced training set

3 iNLTK API

iNLTK is designed to be simple for practitioners in order to lower the barrier for doing applied research and building products in Indic languages. This section discusses various NLP tasks for which iNLTK provides out-of-the-box support, under a unified API.

Data Augmentation helps in improving the performance of NLP models (Duboue and Chu-Carroll, 2006; Marton et al., 2009). It is even more important in "low-resource" settings, where labeled data is scarce. iNLTK provides augmentations[14] for a sentence while preserving its semantics following a two step process. Firstly, it generates candidate paraphrases by replacing original sentence tokens with tokens which have closest embeddings from the embedding layer of pre-trained language model. And then, it chooses top paraphrases which are similar to original sentence, where similarity between sentences is calculated as the cosine similarity of sentence embeddings, obtained from pre-trained language model's encoder.

To evaluate the effectiveness of using data augmentation from iNLTK in low resource settings, we prepare[15] **reduced train sets** of publicly available text-classification datasets by picking first N examples from the full train set[16], where N is equal to size of reduced train set and compare accuracy of the classifier trained *with* vs *without* data augmen-

tation. Table 7 shows reduced dataset statistics and comparison of results obtained on full and reduced datasets using iNLTK. Using data augmentation from iNLTK gives significant increase in accuracy on Hindi, Bengali, Malayalam and Tamil dataset, and minor improvements in Gujarati and Marathi datasets. Additionally, Table 7 compares previously obtained best results on these datasets using INLP embeddings (Kunchukuttan et al., 2020) with results obtained using iNLTK pretrained models and iNLTK's data augmentation utility. On an average, with iNLTK we are able to achieve more than 95% of the previous accuracy using less than 10% of the training data[17].

Semantic Textual Similarity (STS) assesses the degree to which the underlying semantics of two segments of text are equivalent to each other (Agirre et al., 2016). iNLTK compares[18] sentence embeddings of the two segments of text, obtained from pre-trained language model's encoder, using a comparison function, to evaluate semantic textual similarity. Cosine similarity between sentence embeddings is used as the default comparison function.

Distributed representations are the cornerstone of modern NLP, which have led to significant advances in many NLP tasks. iNLTK provides utilities to obtain distributed representations for **words**[19], **sentences and documents**[20] obtained

---

[14]https://inltk.readthedocs.io/en/latest/api_docs.html#get-similar-sentences

[15]Notebooks to prepare reduced datasets are accessible from the GitHub repository of the library

[16]Labels in publicly available full train sets were not grouped together, instead were randomly shuffled

[17]Refer GitHub repository of the library for instructions to reproduce results on full and reduced dataset

[18]https://inltk.readthedocs.io/en/latest/api_docs.html#get-sentence-similarity

[19]https://inltk.readthedocs.io/en/latest/api_docs.html#get-embedding-vectors

[20]https://inltk.readthedocs.io/en/latest/api_docs.html#get-

from embedding layer and encoder output of pre-trained language models, respectively.

Additionally, iNLTK provides utilities to **generate text**[21] given a prompt, using pre-trained language models, **tokenize**[22] text using sentencepiece tokenization models described in Section 2.2, identify[23] which one of the supported Indic languages is given `text` in and remove tokens of a foreign language[24] from given `text`.

4 Related Work

NLP and ML communities have a strong culture of building open-source tools. There are lots of easy-to-use, user-facing libraries for general-purpose NLP like NLTK (Loper and Bird, 2002), Stanford CoreNLP (Manning et al., 2014), Spacy (Honnibal and Montani, 2017), AllenNLP (Gardner et al., 2018), Flair (Akbik et al., 2019), Stanza (Qi et al., 2020) and Huggingface Transformers (Wolf et al., 2019). But most of these libraries have limited or no support for Indic languages, creating a barrier to entry for working with Indic languages. Additionally, for many Indic languages word embeddings have been trained, but they still lack richer pre-trained representations from deep language models (Kunchukuttan et al., 2020). iNLTK tries to solve these problems by providing pre-trained language models and out-of-the-box support for a variety of NLP tasks in 13 Indic languages.

5 Conclusion and Future Work

iNLTK provides pre-trained language models and supports Data Augmentation, Textual Similarity, Sentence Embeddings, Word Embeddings, Tokenization and Text Generation in 13 Indic Languages. Our results significantly outperform other methods on text-classification benchmarks, using pre-trained models from iNLTK. These pre-trained models from iNLTK can be used as-is for a variety of NLP tasks, or can be fine-tuned on domain specific datasets. iNLTK is being widely[25] used[26] and

appreciated[27] by the community[28].

We are working on expanding the supported languages in iNLTK to include other Indic languages like Telugu, Maithili; code mixed languages like Hinglish (Hindi and English), Manglish (Malayalam and English) and Tanglish (Tamil and English); expanding supported model architectures to include BERT. Additionally, we want to mitigate any possible unwarranted biases which might exist in pre-trained language models (Lu et al., 2019), because of training data, which might propagate into downstream systems using these models. While these tasks are work in progress, we hope this library will accelerate NLP research and development in Indic languages.

Acknowledgments

References

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, San Diego, California. Association for Computational Linguistics.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics.

Md Shad Akhtar, Ayush Kumar, Asif Ekbal, and Pushpak Bhattacharyya. 2016. A hybrid deep learning architecture for sentiment analysis. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 482–493, Osaka, Japan. The COLING 2016 Organizing Committee.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR*, abs/1607.04606.

---

sentence-encoding

[21] https://inltk.readthedocs.io/en/latest/api_docs.html#predict-next-n-words

[22] https://inltk.readthedocs.io/en/latest/api_docs.html#tokenize

[23] https://inltk.readthedocs.io/en/latest/api_docs.html#identify-language

[24] https://inltk.readthedocs.io/en/latest/api_docs.html#remove-foreign-languages

[25] https://github.com/goru001/inltk/network/members

[26] https://pepy.tech/project/inltk

[27] https://github.com/goru001/inltk/stargazers

[28] https://github.com/goru001/inltk#inltks-appreciation

[29] https://github.com/anuragshas

[30] https://github.com/ravi-annaswamy

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Car-bonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language under-standing. 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. Associ-ation for Computational Linguistics.

Pablo Duboue and Jennifer Chu-Carroll. 2006. An-swering the question you wish they had asked: The impact of paraphrasing for question answering. In *Proceedings of the Human Language Technol-ogy Conference of the NAACL, Companion Volume: Short Papers*, pages 33–36.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Ar-mand Joulin, and Tomas Mikolov. 2018. Learn-ing word vectors for 157 languages. *CoRR*, abs/1802.06893.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embed-dings, convolutional neural networks and incremen-tal parsing. To appear.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tok-enizer and detokenizer for neural text processing. *CoRR*, abs/1808.06226.

Anoop Kunchukuttan, Divyanshu Kakwani, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. Ai4bharat-indicnlp corpus: Monolingual corpora and word embeddings for indic languages. *arXiv preprint arXiv:2005.00085*.

Yann LeCun, Y. Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521:436–44.

Edward Loper and Steven Bird. 2002. Nltk: the natural language toolkit. *CoRR*, cs.CL/0205028.

Kaiji Lu, Piotr Mardziel, Fangjing Wu, Preetam Aman-charla, and Anupam Datta. 2019. Gender bias in neural natural language processing.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language pro-cessing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Lin-guistics: System Demonstrations*, pages 55–60, Bal-timore, Maryland. Association for Computational Linguistics.

Yuval Marton, Chris Callison-Burch, and Philip Resnik. 2009. Improved statistical machine translation us-ing monolingually-derived paraphrases. In *Proceed-ings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 381–390.

Adam Paszke, S. Gross, Soumith Chintala, G. Chanan, E. Yang, Zachary Devito, Zeming Lin, Alban Des-maison, L. Antiga, and A. Lerer. 2017. Automatic differentiation in pytorch.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. *arXiv preprint arXiv:2003.07082*.

Sebastian Ruder. 2020. Why You Should Do NLP Beyond English. http://ruder.io/ nlp-beyond-english.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pier-ric Cistac, Tim Rault, R'emi Louf, Morgan Funtow-icz, and Jamie Brew. 2019. Huggingface's trans-formers: State-of-the-art natural language process-ing. *ArXiv*, abs/1910.03771.