

Different NLP Libraries for Indian Languages

Vinay Verma¹, Shivam Gupta^{1,*}

Abstract

Humans are by nature varied and multilingual. While English holds the title of the most commonly spoken language globally, Hindi is also widely used by people worldwide. Natural language processing (NLP) is the discipline of artificial intelligence (AI) concerned with providing computers the capacity to interpret text and spoken language in the same manner that humans can. People nowadays utilize products that employ NLP as their foundation, such as Alexa or Siri. However, there are several ambiguities in NLP for Indian languages. Currently, NLP libraries such as iNLTK, Indic NLP, Stanford NLP, and others are utilized to process several Indian languages. This article contains information about the various NLP libraries, the processes supported by these libraries, and the accuracy of these models for Indian languages.

Keywords: iNLTK, tokenization, natural language processing (NLP), deep learning, multilingual NLP

INTRODUCTION

Natural language processing (NLP), a subset of computer science, notably artificial intelligence (AI), focuses on enabling computers to understand and analyze human language. NLP facilitates computers in intelligently analyzing, comprehending, and deriving meaning from human language [1]. To parse the text, numerous open python libraries are utilized. We deliver refined text in a specified language as an output by employing NLP. In several disciplines, this processed text is utilized while interacting with humans. The main objective of NLP is to improve the user experience of technologies like chatbots, Alexa, and Siri. NLP is utilized for a variety of tasks, including data analysis, predictive text, email filtering, search results, and language translation, in addition to smart assistance. In the rapidly developing discipline of NLP, texts may now be handled not just in English but also in Spanish, German, Arabic, Hindi, and other languages. Nevertheless, there are still a lot of issues, such text quality variations, insufficient data to interpret the text, and never 100% dependability. Errors might occur in both the forecast and the outcomes. Following are the steps that go into NLP [2].

Tokenization

This is the first step in NLP, where the input sentence is broken up into smaller words. After comprehending each word within the context of the sentence, it is then processed before proceeding to the subsequent step. This process is carried out to process a sentence because there are fully functional Python and Java libraries for NLP in the English language. Bigrams, Trigrams, and Ngrams—groups of two, three, or more words—can be formed from the words.

Stemming

After obtaining the individual words, the stemming procedure is used to normalize each word into its base or root form. However, since the prefix and suffixes are eliminated there, this might still be problematic. Thus, we proceed with the "Lemmatization" phase.

*Author for Correspondence

Shivam Gupta
E-mail: shivamgupta17600@gmail.com

¹Research Scholar, MCA Thakur Institute of Management Studies, Career Development & Research (TIMSCDR) Mumbai, India

Received Date: February 29, 2024
Accepted Date: March 22, 2024
Published Date: April 05, 2024

Citation: Vinay Verma, Shivam Gupta. Different NLP Libraries for Indian Languages. Journal of Open Source Developments. 2024; 11(1): 8–14p.

Lemmatization

In this case, each word is determined to have its root form from a pre-defined dictionary or dataset. One word with its correct root form is the result of this operation.

Part of Speech (POS) Tagging

This method scans the input text and returns the parts of speech that are found there.

Named Entity Recognition (NER)

This category includes terms such as geo-political entity, location, facility, organization, person, etc. This procedure indicates which term falls within which heading.

Chunking

In natural language processing, chunking refers to the procedure of combining discrete information into larger parts. Chunking is mostly used to create clusters of "noun phrases." By using regular expressions in conjunction with POS tagging, it gives the sentence structure.

NATURAL LANGUAGE PROCESSING FOR INDIAN LANGUAGES

Within the population of almost one billion people, just 10% speak English! Hindi is the primary language spoken by the majority of Indians, with Marathi, Telugu, Punjabi, etc. following [3].

English is not even a language that many residents of rural areas can understand or talk. Without introducing NLP, the vision of a Digital India that is inclusive cannot be achieved. study and use in India comparable to that of languages like English. When interacting with the language barrier might be a major barrier for NLP when it comes to smartphones and other technology. There are numerous areas where consumers can benefit from a native-language system [4].

NLP Applications for Indian Languages

The NLP applications for Indian languages include the following:

- Smartphones are compatible with intelligent virtual assistants like Siri, Google Assistant, and similar technologies. When these smart assistants cover numerous Indian languages, there would be a significant boost in user engagement at the local level.
- Digitization of Indian manuscripts in order to preserve the knowledge contained within them.
- Translation of signboards from local languages to aid travelers during their journeys.
- Indian script fonts for increasing the impact/readability of commercials, signboards, presentations, reports, and so on.

Problems of Working with Indian Languages

Since there are a lot of materials available, there are not many issues when working on English sentences. Online tools for learning NLP for Indian languages, however, are restricted, which causes a barrier. Other problems include a wide range of morphological variations, Limited availability of tools, languages, and annotated corpora (a sizable, structured collection of machine-readable texts) speaking distinct dialects with only little differences. Hence, proceeding with NLP tasks for Indian languages presents challenges. However, scientists are still researching in this area to discover novel approaches. For Indian languages, there are now three NLP libraries:

1. The Indian Natural Language Toolkit, or iNLTK
2. The Indic NLP Collection
3. The NLP Library at Stanford

We will go over each of these three libraries in-depth and learn about the features they offer.

iNLTK

The iNLTK library, as its name implies, is the equivalent of the well-known iNLTK Python package in Indian languages. A total of 13 Indian languages are supported by the iNLTK library. The purpose

of this library is to provide the functionalities that an application developer for NLP will require. Tokenization, sentence similarity, vector embedding generation, and other characteristics needed for modern NLP applications are all provided by iNLTK in an incredibly user-friendly application programming interface (API).

iNLTK, an open-source toolkit for Indian languages, provides pre-trained deep language models that significantly assist in NLP tasks such as text classification for Indian languages by enabling sharing. Additionally, textual similarity, tokenization, data augmentation, sentence and word embeddings, and text generation are all supported out-of-the-box by iNLTK. To make it easier for practitioners to do applied research and create products in Indic languages, iNLTK is made to be easy. This section covers a variety of NLP tasks under a single API that iNLTK supports out of the box.

Processes performed by iNLTK library:

- Tokenization
- Word Embeddings
- Text Completion
- Similarity of Sentences
- Data Augmentation

Indic NLP Library

Just like INLTK was designed for developers working with vernacular languages, this library is designed for scholars working in this field. As stated in the official documentation of Indic NLP, the Indic NLP library is designed to offer comprehensive support for a wide range of text processing and NLP functionalities tailored specifically for Indian languages. Indic NLP, inspired by the BERT (Bidirectional Encoder Representations from Transformers) approach, created IndicNLPSuite, a set of basic resources for language processing. Therefore, carrying out NLP tasks for Indian languages poses difficulties.

Indic NLP provides the following resources:

- *IndicCorp*: IndicCorp stands as one of the most extensive publicly accessible collections of corpora for Indian languages. It is a large-scale monolingual corpus with a total of 8.9 billion tokens that was created over several months by identifying and scraping thousands of web sources, primarily news, periodicals, and novels.
- *IndicBERT*: IndicBERT, a multilingual ALBERT model trained on the IndicCorp dataset, encompasses 12 significant Indian languages. It is capable of conducting various NLP tasks such as sentiment analysis, common-sense reasoning, and question answering in Indian languages. IndicBERT contains many less parameters than other public models such as
- mBERT and XLM-R while yet providing cutting-edge performance on a variety of tasks.
- *IndicFT*: IndicFT is a word embedding model created by pre-training fastText using the IndicCorp dataset.
- *IndicGLUE*: IndicGLUE is an initial effort to develop a reliable benchmark for natural language understanding (NLU) that can assess performance across several NLU capability parameters and on each of the 12 Indian languages.
- *IndicBART*: With an emphasis on Indic languages and English, IndicBART is a multilingual, sequence-to-sequence pre-trained model.
- *IndicTrans*: Trained on the Samanantar dataset, IndicTrans is a Transformer-XL model. There are two models that can translate both Indic to English and Indic to English. The model is capable of translating texts into Assamese, Bengali, Gujarati, Hindi, Kannada, Malayalam, Marathi, Oriya, Punjabi, Tamil, and Telugu, covering a total of eleven languages.

The following features are offered by this library:

- Text Normalization
- Script Information
- Tokenization
- Word Segmentation
- Script Conversion
- Romanization
- Indexing
- Transliteration
- Translation

NLP Library

The Stanford Natural Language Processing Library: The Stanford Natural Language Processing Research Group is the source of the NLP library known as StanfordNLP. This library's most notable feature is that it can parse text in about 53 different human languages. The Indian subcontinent's Hindi and Urdu are supported by StanfordNLP among these languages. StanfordNLP enables the generation of computational linguistic elements like dependency parsing, named entity recognition (NER), and POS tags.

StanfordNLP has processors built in to handle the following four fundamental NLP tasks:

1. Parts of Speech Tagging
2. Dependency Parsing
3. Lemmatization
4. Multi-Word Token Expansion

LITERATURE REVIEW

Manthan et al. [5] proposed various natural language processing tasks, such as transliteration, tokenization, and part-of-speech tagging, within the AMBER Chatbot framework and for detecting paraphrases in Devnagari. In AMBER, sentence generation is used. The Paraphrasing Detection Library is utilized. It also makes use of a number of frameworks to calculate semantic similarity. It is similar to DKPro. To determine alignment between idioms and phrasal chunks, the Word Alignment Algorithm is used. To obtain POS tags, the RDRPOST Tagger is utilized. To determine suffix, it employs a phonetic and stripping technique.

Kharate and Patil [6] conducted a survey on machine translation techniques and methods for translating Indian languages into English. They looked at approaches, including dictionary-based, rule-based, corpus-based, knowledge-based, and hybrid-based. This demonstrated the effort needed and how to formulate a response to a query in relation to the work of a shallow parser.

In 2019, Navalakha et al. [7] proposed a paper discussing a chatbot system intended for question-answering purposes. The user can communicate with the chatbot in Marathi. The portal has all the information needed to provide a response to the query. It is a computer program that replicates a conversation. The effectiveness of the interaction relies heavily on the dialogue exchanged with the user. The interface of a chatbot is notably user-friendly. It responds to queries posed by the user. An artificial intelligence technique called pattern matching is applied when creating a chatbot. A matched response is returned when the input is compared to the inputs stored in the database. When dealing with enormous data or file sizes, text recognition classification is used. Optical character recognition (OCR) technology simplifies the extraction of textual data from large files. The most interesting and challenging aspect of the extraction process is the correlation between the text detection, localization, and tracking modules. In his proposed paper, Arora [8] discusses the NLP library iNLTK in detail.

Besides reviewing all the features, he also discussed future prospects, such as integrating BERT into the supported model architectures and broadening the range of supported languages to encompass

additional Indic languages like Telugu and Maithili. Furthermore, he addressed coding for mixed languages such as Hinglish (a blend of Hindi and English), Manglish (a fusion of Malayalam and English), and Tanglish (a combination of Tamil and English).

METHODOLOGY

iNLTK Library

The popular NLTK Python package has an Indian language equivalent called the iNLTK library [9]. The objective of this library is to offer the necessary functionalities required by application developers in the field of NLP. With a very user-friendly and simple API, iNLTK offers the majority of the functionality needed for contemporary NLP activities, including tokenization, data augmentation, building a vector embedding for input text, and sentence similarity. Here, we have used the iNLTK library to carry out one functionality: tokenization.

Here we have tokenized a Hindi sentence using language code 'hi' for Hindi language (Figure 1). Again, we have passed a Gujarati sentence using code 'gu' for Gujarati language and we get small pieces (words) of a sentence for Gujarati language as shown in Figure 2.

Lastly, we tried to tokenize a Marathi sentence and got tokens as output as shown in Figure 3. Similarly, we can perform same task on other 11 Indian languages. Also, there are other packages present in iNLTK library to perform functions like word embedding, similarity in sentences, etc.

```
1 from inltk.inltk import tokenize
2
3 hindi_text = ""गाय हमें दूध देती है और घास खाती है""
4
5 # tokenize(input text, language code)
6 tokenize(hindi_text, "hi")
```

↳ ['_गाय', '_हमें', '_दूध', '_देती', '_है', '_और', '_घास', '_खाती', '_है']

Figure 1. Hindi sentence using language code 'hi'.

```
1 from inltk.inltk import tokenize
2
3 guj_text = ""ગાય આપણને દૂધ આપે છે અને ઘાસ ખાય છે""
4
5 # tokenize(input text, language code)
6 tokenize(guj_text, "gu")
```

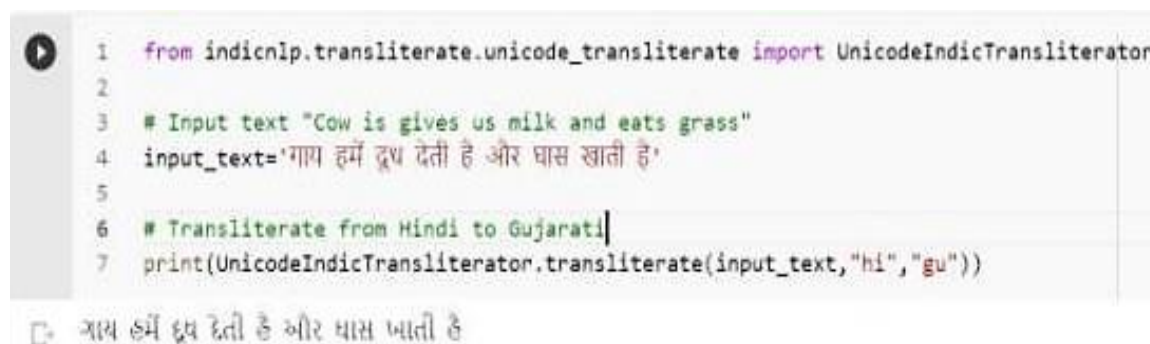
↳ ['_ગાય', '_આપણને', '_દૂધ', '_આપે', '_છે', '_અને', '_ઘાસ', '_ખાય', '_છે']

Figure 2. Gujarati sentence using code 'gu'.

```
1 from inltk.inltk import tokenize
2
3 marathi_text = ""गाय आपल्याला दूध देते आणि गवत खाते""
4
5 # tokenize(input text, language code)
6 tokenize(marathi_text, "mr")
```

↳ ['_गाय', '_आपल्याला', '_दूध', '_देते', '_आणि', '_गवत', '_खाते']

Figure 3. Marathi sentence and got tokens as output.



```

1 from indicnlp.transliterate.unicode_transliterate import UnicodeIndicTransliterator
2
3 # Input text "Cow is gives us milk and eats grass"
4 input_text='गाय हमें दूध देती है और घास खाती है'
5
6 # Transliterate from Hindi to Gujarati
7 print(UnicodeIndicTransliterator.transliterate(input_text,"hi","gu"))

```

ગાય હમેં દૂધ દેતી હે ઓર ઘાસ ખાતી હે

Figure 4. Hindi sentence into Gujarati sentence using Indic NLP library.

Indic NLP Library

Just like iNLTK, there are many common tasks between iNLTK and Indic NLP library performed by this respective library. I have demonstrated transliteration of a Hindi sentence into Gujarati sentence using Indic NLP library (Figure 4).

We may complete the remaining tasks by leveraging the various resources included in IndicNLPsuite, including as IndicCorp, IndicBERT, IndicTrans, and others [10]. Only two Indian subcontinental languages—Hindi and Urdu—are supported by the Stanford NLP library.

CONCLUSION

There is no denying that AI has advanced greatly. Moreover, due to supporting technologies like cloud computing, machine learning, deep learning, and the internet of things, people are becoming more reliant on technology. Nonetheless, if smart devices function (communicate) in the appropriate local language, people will be drawn to them more. NLP researchers are actively active in the subject, and iNLTK and Indic NLP are the two most often used libraries out of these three. It is fairly appropriate to consider the features of the iNLTK library, which is utilized in chatbots.

NLP research for the Indian language is still needed because there are still problems with distinct dialects that differ slightly, morphological variances, a lack of annotated corpora, and a lack of datasets.

Acknowledgements

Indeed, India has a wide variety of languages, and to accommodate these languages, a number of Natural Language Processing (NLP) libraries have been developed.

We would like to express our profound gratitude to the developers and collaborators of the NLP libraries designed specifically for Indian languages. Their dedication and creativity have greatly improved the accessibility and comprehension of various linguistic subtleties. We thank the following libraries for their essential contributions, among others:

The authors of the Indic NLP Library are acknowledged for their commitment to offering instruments and materials for processing texts in the Indian language, enabling cross-linguistic analysis and comprehension.

The Natural Language Toolkit (NLTK) for Indian Languages is acknowledged for the efforts of its contributors in expanding its functions to include Indian languages, hence promoting NLP research and development.

SpaCy for Languages in India: Indian language models are included in the Python NLP library SpaCy. These models can be applied to named entity recognition, dependency parsing, tokenization, and part-of-speech tagging.

REFERENCES

1. Megamsolutions. Levelling up NLP for Indian Languages. [Online]. 2021. Robert Bosch Center for Data Science and Artificial Intelligence. iitm.ac.in. Available at <https://rbcdsai.iitm.ac.in/blogs/leveling-up-nlp4-indian-langs/>
2. Chowdhary KR. Natural language processing. In: Fundamentals of Artificial Intelligence. New Delhi, India: Springer; 2020. pp. 603–649.
3. Sanad M. 3 Important NLP Libraries for Indian Languages You Should Try Out Today! [Online]. June 14, 2020. Analytics Vidhya. Available at <https://www.analyticsvidhya.com/blog/2020/01/3-important-nlp-libraries-indian-languages-python/>
4. Balaganur S. Top NLP Libraries & Datasets for Indian Languages. [Online]. Analytics India Magazine. February 7, 2020. Available at <https://analyticsindiamag.com/top-nlp-libraries-datasets-for-indian-languages/>
5. Manthan S, Kumar J, Mediratta A, Kundale A, Nangare SH. AMBER chatbot and detection of paraphrases for Devnagari. Vishwakarma J Eng Res. 2017; 1 (1): 1–5.
6. Kharate NG, Patil VH. Survey of machine translation for Indian languages to English and its approaches. Int J Sci Res Computer Sci Eng Inform Technol. 2018; 3 (1): 613–622.
7. Navalakha D, Pittule M, Mane R, Rathod A, Kharate NG. Review of chatbot system in Marathi language. Int Res J Eng Technol. 2019; 6 (11): 1814–1819.
8. Arora G. iNLTK: natural language toolkit for Indic languages. arXiv preprint arXiv:2009.12534. September 26, 2020. Available at <https://arxiv.org/abs/2009.12534>
9. Joseph J, Lalithsriram SR, Menon N. Applications and developments of NLP resources for text processing in Indian languages: shared multilingual corpora building and pre-trained models. In: Viola L, Spence P, editors. Multilingual Digital Humanities. London, UK: Taylor & Francis; 2023. Chapter 3. .
10. Aralikatte R, Cheng Z, Doddapaneni S, Cheung JC. Vārta: a large-scale headline-generation dataset for Indic languages. arXiv preprint arXiv:2305.05858. May 10, 2023. Available at <https://arxiv.org/abs/2305.05858>