Search projects

Help Sponsors Log in Register

bltk 1.2

Latest version

```
pip install bltk
```

Released: Mar 12, 2020

A lightweight but robust toolkit for Bengali Natural Language Processing.

Navigation

Project description

Release history

Download files

Project links

Homepage

Download

Statistics

GitHub statistics:

Stars: 12

Project description

BLTK: The Bengali Natural Language Processing Toolkit

A lightweight but robust toolkit for processing Bengali Language.

Open Source ❤ License MIT MADE WITH PYTHON

Overview

BLTK is a lightweight but exceptionally robust language processing toolkit for the Bengali Language. I, Mr. Saimon Hossain, along with my friend, Mr. Liton Shil, have conducted research as a part of our undergraduate thesis under the supervision of our respected sir, Mr. Sowmitra Das. This is the outcome of our 6 month long research & development project.

I have choosen this name after taking inspiration from the popular natural language processing toolkit - **NLTK**.

BLTK is still in its childhood. It's maturing everyday. It'll receive updates in the days to come.

If you want to contribute to BLTK's growth, please read the contribution section at the end of this page.

Supported Functionalities

- Word Tokenization
- Sentence Tokenization
- Sentence Splitting
- Stopwords Filtering
- Statistical Part-of-speech Tagging
- Phrase Chunking/Named-Entity Recognition
- Stemming

Installation

To get BLTK up and running, run the following command.

```
pip install bltk
```

Usage

1) The Bengali Characters

In **BLTK**, the ***banglachars*** module contains 7 lists of characters specific to the Bengali Language.

1. vowels
2. vowel signs
3. consonants
4. digits
5. punctuation marks
6. operators
7. others

Code

Forks: 4

Open issues/PRs: 1

View statistics for this project via Libraries.io, or by using our public dataset on Google BigQuery

Meta

License: MIT License (MIT)

Author: Saimon Hossain

Maintainer: Saimon Hossain

pos-tagger, pos tagger, phrase chunker, phrase-chunker, stemmer, bengali, natural language processing, Machine learning, NLP

Requires: Python >=3.6

Maintainers

saimoncse

Classifiers

Development Status

- 5 - Production/Stable

Intended Audience

- Developers
- Education

License

- OSI Approved :: MIT License

Natural Language

- Bengali
- English

Operating System

- OS Independent

Programming Language

- Python :: 3
- Python :: 3 :: Only
- Python :: 3.6
- Python :: 3.7
- Python :: 3.8

```
from bltk.langtools.banglachars import (vowels,
                                        vowel_signs,
                                        consonants,
                                        digits,
                                        operators,
                                        punctuations,
                                        others)
print(f'Vowels: {vowels}')
print(f'Vowel signs: {vowel_signs}')
print(f'Consonants: {consonants}')
print(f'Digits: {digits}')
print(f'Operators: {operators}')
print(f'Punctuation marks: {punctuations}')
print(f'Others: {others}')
```

Output

```
Vowels: ['অ', 'আ', 'ই', 'ঈ', 'উ', 'ঊ', 'ঋ', 'ঌ', 'এ'
Vowel signs: ['া', 'ি', 'ী', 'ু', 'ূ', 'ৃ', 'ৄ', 'ে', 'ৈ
Consonants: ['ক', 'খ', 'গ', 'ঘ', 'ঙ', 'চ', 'ছ', 'জ', '
Digits: ['০', '১', '২', '৩', '৪', '৫', '৬', '৭', '৮', '
Operators: ['=', '+', '-', '*', '/', '%', '<', '>', '×
Punctuation marks: ['।', ',', ';', ':', '?', '!', "'",
Others: ['৳', '৺', '্', '৭', '২', '#', '$']
```

2) Word Tokenization

In **BLTK**, the ***word_tokenizer(text: str)*** method of the Tokenizer class performs word tokenization. It takes a text string and returns a list of tokenized words. The Following code shows how it is done.

Code

```
from bltk.langtools import Tokenizer

# Sample text
text = "আমি জানি আমার এই লেখাটির জন্য আমাকে অনেক গালমন্দ শুনতে
       "লিখে খুব কাজ হয় সে রকম উদাহরণ আমার হাতে খুব বেশী নেই কিন্ত
       "যায় সেটাই আমার জন্যে অনেক।"
```

```
# Creating an instance
tokenizer = Tokenizer()

# Tokenizing words
print('TOKENIZED WORDS')
words = tokenizer.word_tokenizer(text)
print(words)
```

Output

```
TOKENIZED WORDS
['আমি', 'জানি', 'আমার', 'এই', 'লেখাটির', 'জন্য', 'আমাকে', '
```

3) Sentence Tokenization

In Bengali, most of the sentence delimiters are same as in English except full-stop. Statements and imperative sentences are terminated by । - the Devanagari Danda. Questions and exclamatory sentences are terminated by ? and ! respectively.

In **BLTK**, the ***sentence_tokenizer(text: str)*** method of the Tokenizer class performs sentence tokenization. It takes a text string and returns a list of tokenized sentences. The Following code shows how it is done.

Code

```
from bltk.langtools import Tokenizer

# Sample text
text = "আমি জানি আমার এই লেখাটির জন্য আমাকে অনেক গালমন্দ শুনতে
       "লিখে খুব কাজ হয় সে রকম উদাহরণ আমার হাতে খুব বেশী নেই কিন্ত
       "যায় সেটাই আমার জন্যে অনেক।"

# Creating an instance
tokenizer = Tokenizer()


# Tokenizing Sentences
print("TOKENIZED SENTENCES")
```

```
sentences = tokenizer.sentence_tokenizer(text)
print(sentences)
```

Output

```
TOKENIZED SENTENCES
['আমি জানি আমার এই লেখাটির জন্য আমাকে অনেক গালমন্দ শুনতে হবে, তা
```

4) Sentence Split

The ***sentence_splitter(sentence: list)*** method takes a list of tokened sentences and then splits them into their corresponding list of tokened words with the help of word_tokenizer() method. The return value is a list of tokened words lists.

Code

```
from bltk.langtools import Tokenizer

# Sample text
text = "আমি জানি আমার এই লেখাটির জন্য আমাকে অনেক গালমন্দ শুনতে
       "লিখে খুব কাজ হয় সে রকম উদাহরণ আমার হাতে খুব বেশী নেই কিন্তু
       "যায় সেটাই আমার জন্যে অনেক।"

# Creating an instance
tokenizer = Tokenizer()


# Tokenizing Sentences
sentences = tokenizer.sentence_tokenizer(text)

print("SPLIT SENTENCES")
sentence_list = tokenizer.sentence_splitter(sentences)
print(sentence_list)

print("INDIVIDUAL SENTENCE")
for i in sentence_list:
    print(i)
```

Output

```
SPLIT SENTENCES
[['আমি', 'জানি', 'আমার', 'এই', 'লেখাটির', 'জন্য', 'আমাকে',

INDIVIDUAL SENTENCE

['আমি', 'জানি', 'আমার', 'এই', 'লেখাটির', 'জন্য', 'আমাকে', '
['লিখে', 'খুব', 'কাজ', 'হয়', 'সে', 'রকম', 'উদাহরণ', 'আমার',
```

5) Stopwords Filtering

BLTK's *remove_stopwords(words: list, *, level: str = "soft")* function by default performs a soft stopwords elimination. It takes two parameters: **a list of words** and a **keyword argument** which can be either 'soft', 'moderate' or 'hard'. If no parameter is given, a soft elimination is performed.

Filtering stopwords is not always an ideal choice. In any language, there is no universal list of stop words, and sometimes different researchers use different methods for eliminating stopwords. If you are not sure about which level to use, use the default.

Code

```
from bltk.langtools import remove_stopwords
from bltk.langtools import Tokenizer

tokenizer = Tokenizer()

text = "আমি জানি আমার এই লেখাটির জন্য আমাকে অনেক গালমন্দ শুনতে
       "লিখে খুব কাজ হয় সে রকম উদাহরণ আমার হাতে খুব বেশী নেই কিন্ত
       "যায় সেটাই আমার জন্যে অনেক।"

tokened_words = tokenizer.word_tokenizer(text)

print(f"Len of words: {len(tokened_words)}")
print(f"After soft elimination: {(remove_stopwords(tok
print(f"Length after soft elimination: {len(remove_sto
print(f"After moderate elimination: {(remove_stopwords
print(f"Length after moderate elimination: {len(remove_
print(f"After hard elimination: {(remove_stopwords(tok
print(f"Length after hard elimination: {len(remove_sto
```

Output

```
Len of words: 40
After soft elimination: ['জানি', 'লেখাটির', 'অনেক', 'গালমন
Length after soft elimination: 20
After moderate elimination: ['জানি', 'লেখাটির', 'গালমন্দ', '
Length after moderate elimination: 15
After hard elimination: ['জানি', 'লেখাটির', 'গালমন্দ', 'শুনতে
Length after hard elimination: 13
```

6) Statistical Part-of-speech Tagging

BLTK includes a statistical POS tagger which has an overall system accuracy of 95.9%. The POS tagger works in sentence-level which means that instead of tagging a word individually, it tags words in a sentence or a phrase, taking features such as ***previous word*** and ***next word*** into consideration. It relies on the Logistic Regression classifier.

The BLTK's PosTagger class has a method ***pos_tag()*** which takes a list of split sentences and returns a list of tagged sentences. Each tagged sentence is a list of ***tuples of length 2*** each, where the first index of the tuple holds the word itself and the last index holds its corresponding tag.

Code

```
from bltk.langtools import PosTagger
from bltk.langtools import Tokenizer

pos_tagger = PosTagger()
tokenizer = Tokenizer()

text = "আমি জানি আমার এই লেখাটির জন্য আমাকে অনেক গালমন্দ শুনতে
       "লিখে খুব কাজ হয় সে রকম উদাহরণ আমার হাতে খুব বেশী নেই কিন্তু
       "যায় সেটাই আমার জন্যে অনেক।"

token_text = tokenizer.sentence_tokenizer(text)
```

```
pos_tags = []
for text in token_text:
    tokened = tokenizer.word_tokenizer(text)
    tagged = pos_tagger.pos_tag(tokened)
    pos_tags.append(tagged)
print(pos_tags)
```

Output

```
[[('আমি', 'PPR'), ('জানি', 'VM'), ('আমার', 'PPR'), ('এই
```

7) Phrase Chunking/Named-Entity Recognition

BLTK's phrase chunker can find out all the phrases in a given text as long as a correct grammatical syntax for that phrase is provided in the form of a regular expression. The performance of the chunker is unparalleled since it heavily relies on the BLTK's POS Tagger which has an outstanding accuracy and NLTK's Regular Expression Parser which is extremely powerful.

BLTK's Chunker class has method named ***chunk()*** that takes two parameters: ***a grammar*** in the form of regular expression, and ***a text*** from which phrases will be extracted.

This section explains how to create a noun phrase chunker using BLTK's Chunker class and a regular expression grammar. A noun phrase begins with an optional demonstrative, followed by zero or more adjectives/quantifiers and terminates with a noun. Some examples of Bangla noun phrases are given below:

NP: (NP গণতন্ত্র/NC) - a noun phrase with only one noun.

NP: (NP মানবিক/JJ বোধ/NC) - a noun phrase with an adjective followed by a noun.

NP: (NP এই/DAB সুন্দর/JJ লেখাটির/NC) - a noun phrase with a demonstrative, followed by an adjective and terminated by a noun.

The grammar for extracting Bangla noun phrases can be constructed with the following regular expression.

```
NP: {<DAB|DRL>?<JJ|JQ>*<N.>}
```

The tags used to construct the grammar as well as training the POS Tagger have been explained in the following table as specified by reseracher at **Microsof Research, India**. Grammars for verb phrases, named entities, etc. can also be constructed in the similar fashion.

Name	Tag	Example
COMMON NOUN	NC	মানুষ
PROPER NOUN	NP	রবীন্দ্রনাথ
VERBAL NOUN	NV	ঘটানো
SPATIO-TEMPORAL NOUN	NST	উপরে
MAIN VERB	VM	করছিলেন
AUXILIARY VERB	VA	এসে
PRONOMINAL PRONOUN	PPR	আমাদের
REFLEXIVE PRONOUN	PRF	নিজ
RECIPROCAL PRONOUN	PRC	পরস্পর
RELATIVE PRONOUN	PRL	যাহার
WH-PRONOUN	PWH	কেন
ADJECTIVE	JJ	গুরুত্বপূর্ণ
QUANTIFIER	JQ	কয়েকটি
ABSOLUTE DEMONSTRATIVE	DAB	এই
RELATIVE DEMONSTRATIVE	DRL	যে
WH-DEMONSTRATIVE	DWH	কী

Name	Tag	Example
ADVERB of MANNER	AMN	আবার
ADVERB of LOCATION	ALC	যখন
CONDITIONAL PARTICIPLE	LC	হলেই
VERBAL PARTICIPLE	LV	বইতে-বইতেই
POSTPOSITION	PP	জন্য
COORDINATING PARTICLE	CCD	এবং
SUBORDINATING PARTICLE	CSB	সুতরাং
CLASSIFIER PARTICLE	CCL	প্রমুখ
INTERJECTION	CIN	আরে
OTHER PARTICLE	CX	তাই
PUNCTUATION	PU	।
FOREIGN WORD	RDF	Schedule
SYMBOL	RDS	$
OTHER	RDX	৩৫৬

Like grammar for noun phrases, grammar for verb phrases, postpositional phrases, etc. can be constructed with valid regular expressions.

Code

```
from bltk.langtools import Tokenizer
from bltk.langtools import Chunker


grammar = r"""
  NP: {<DAB>?<JJ|JQ>*<N.>}
  """
text = "আমি জানি আমার এই লেখাটির জন্য আমাকে অনেক গালমন্দ শুনতে
```

```
        "লিখে খুব কাজ হয় সে রকম উদাহরণ আমার হাতে খুব বেশী নেই কিন্তু
        "যায় সেটাই আমার জন্যে অনেক।"

tokenizer = Tokenizer()
sentences = tokenizer.sentence_tokenizer(text)
tokened_text = [tokenizer.word_tokenizer(sentence) for

noun_phrases = []
for t in tokened_text:
    chunky = Chunker(grammar=grammar, tokened_text=t)
    chunk_tree = chunky.chunk()
    for i in chunk_tree.subtrees():
        if i.label() == "NP":
            print(i)
            noun_phrases.append(i)
```

Output

```
(NP এই/DAB লেখাটির/NC)
(NP অনেক/JQ গালমন্দ/NC)
(NP খুব/JQ কাজ/NC)
(NP রকম/NC)
(NP উদাহরণ/NC)
(NP হাতে/NC)
(NP ক্ষোভটুকু/NC)
(NP করা/NV)
```

Note: BLTK's Phrase Chunker relies on BLTK's POS Tagger and NLTK's RegexParser. For a complete documentation on ***NLTK's Tree class***, which has been used in its RegexParser, follow this link.

8) Stemming

BLTK currently supports one stemmer - the **Ugra stemmer**. It relies on some pre-arranged lists of suffixes and BLTK's POS Tagger for stemming Bangla words. The reason POS tagging is done before any stemming is even performed is that eliminating suffixes without determining part-of-speech of the words leads to serious miss-stemming issues.

The inflectional morpheme 'ও' or 'ই' modifies a word such as 'তারপরেও' . Ugra eliminates 'ও' or 'ই' from the end of the words and makes sure that after elimination of it, the lengths of the words are greater than or equal to two in terms of the number of characters. It should be noted that if 'ও' or 'ই' is an independent word, it's never removed.

Code

```
from bltk.langtools import UgraStemmer
from bltk.langtools import Tokenizer


text = "আমি জানি আমার এই লেখাটির জন্য আমাকে অনেক গালমন্দ শুনতে
       "লিখে খুব কাজ হয় সে রকম উদাহরণ আমার হাতে খুব বেশী নেই কিন
       "যায় সেটাই আমার জন্যে অনেক।"

stemmer = UgraStemmer()
tokenizer = Tokenizer()
tokenized_text = tokenizer.word_tokenizer(text)

stem = stemmer.stem(tokenized_text)

print(f"Before stemming: {tokenized_text}")
print(f'After stemming: {stem}')
```

Output

```
Before stemming: ['আমি', 'জানি', 'আমার', 'এই', 'লেখাটির',

After stemming: ['আমি', 'জানি', 'আমি', 'এই', 'লেখা', 'জন
```

Contribution

If you want to contribute, please make a pull request and wait for PR confirmation. You can also send me a mail to saimoncse19@gmail.com with the subject **Contributing to BLTK** specifying a little bit about what you are interested to contribute.

Contribution can also be made by adding issues.

Help

Installing packages
Uploading packages
User guide
FAQs

About PyPI

PyPI on Twitter
Infrastructure dashboard
Package index name retention
Our sponsors

Contributing to PyPI

Bugs and feedback
Contribute on GitHub
Translate PyPI
Development credits

Using PyPI

Code of conduct
Report security issue
Privacy policy
Terms of use

Status: All Systems Operational

Developed and maintained by the Python community, for the Python community.
Donate today!

Site map

Switch to desktop version

English español français 日本語 português (Brasil) українська Ελληνικά Deutsch 中文 (简体) 中文 (繁體) русский עברית esperanto