**“POS tagging using Unigram Tagger Technique”**

DESCRIPTION OF DATASET:

NLTK contains a collection of tagged corpora, arranged as convenient Python ob-

jects.The nltk.corpus package defines a collection of corpus reader classes, which can

be used to access the contents of a diverse set of corpora. Each corpus reader class

is specialized to handle a specific corpus format.

Here I Used the news articles provided by the Marathi Corpus by the NLTK

library.I trained a Unigram POS Tagger provided by the NLTK library to train on

the Marathi corpus.

* Marathi corpus:

NLTK provides support for various NLP applications in Indic languages. The

languages supported are Hindi (hi), Punjabi (pa), Sanskrit (sa), Gujarati (gu),

Kannada (kn), Malayalam (ml), Nepali (ne), Odia (or), Marathi (mr), Bengali

(bn), Tamil (ta), Urdu (ur), English (en).

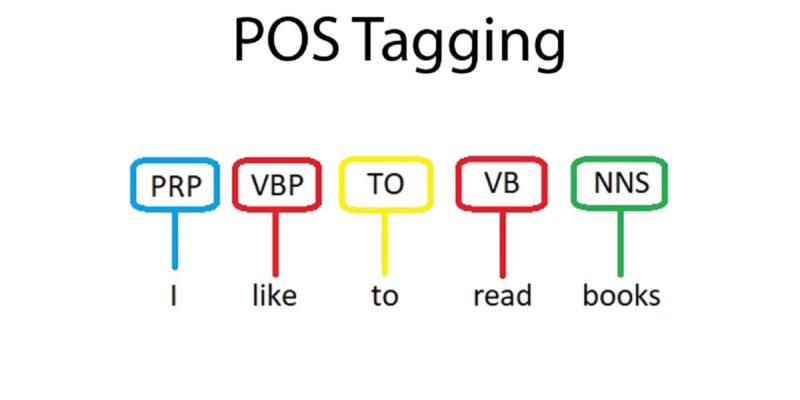
POS TAGGING WITH NLTK:

The primary target of Part-of-Speech(POS) tagging is to identify the grammatical

group of a given word. Whether it is a NOUN, PRONOUN, ADJECTIVE, VERB,

ADVERBS, etc. based on the context. POS Tagging looks for relationships within

the sentence and assigns a corresponding tag to the word.



Let’s learn with a NLTK Part of Speech example:

Input: Everything to permit us.

Output: [(’Everything’, NN),(’to’, TO), (’permit’, VB), (’us’, PRP)]

Basically, the goal of a POS tagger is to assign linguistic (mostly grammatical) information to sub-sentential units. Such units are called tokens and, most of the time, correspond to words and symbols (e.g. punctuation).

UNIGRAM TAGGER:

As the name implies, unigram tagger is a tagger that only uses a single word as

its context for determining the POS(Part-of-Speech) tag. In simple words, Unigram

Tagger is a context-based tagger whose context is a single word, i.e., Unigram.

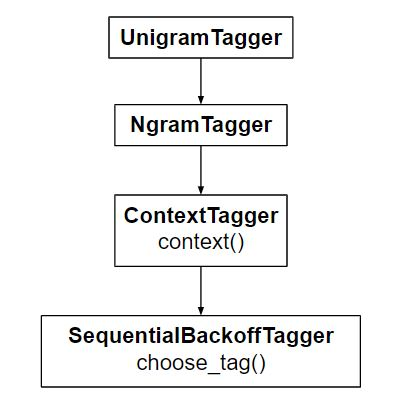
The Unigram tagger (n-gram, n = 1) is a simple statistical tagging algorithm. For

each token, it assigns the tag that is most likely for that token. For example, it will

assign the tag ‘adj’ to any occurrence of the word ‘frequent’, since ‘frequent’ is used

as an adjective (e.g. a frequent word) more often than it is used as a verb (e.g. I

frequent this cafe).



IMPLEMENTATION:

* How does the code work?

UnigramTagger builds a context model from the list of tagged sentences. Because Uni-

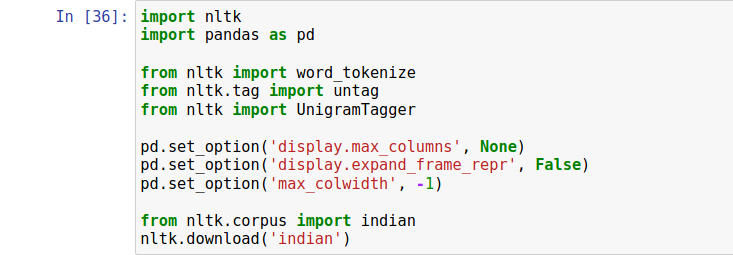
gramTagger inherits from ContextTagger, instead of providing a choose tag() method,

it must implement a context() method, which takes the same three arguments a

choose tag(). The context token is used to create the model, and also to look up

the best tag once the model is created. This is explained graphically in the above

diagram also.



* Training a Unigram Tagger

NLTK’s Unigram Tagger can be trained by providing a list of tagged sentences at the

time of initialization. In the example below, we are going to use the tagged sentences

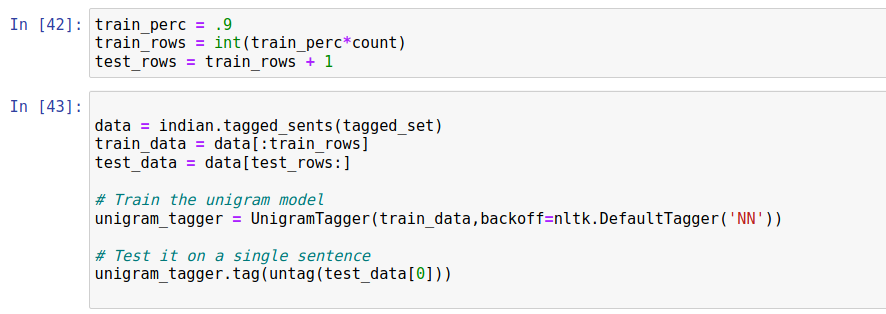
of the marathi corpus.

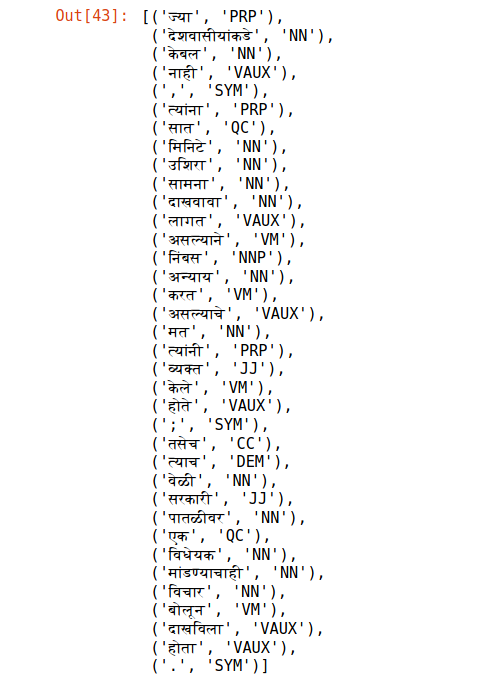
Generally unigram method for calculating part of speech are based on simple

statistical model i.e. basic idea behind that,is calculation of unigram probability. For

this reason,unigram tagger is also called 1-gram tagger. In this method for each

word,assigns the tag that is most likely for that particular word.





RESULTS:

* Accuracy evaluation

First off, we would need some data that is marked up with POS tags, then we can

test. This is usually referred to as a train/test split, since some of the data we use for

training the POS tagger, and some is used for testing or evaluating it’s performance.

The evaluate() method takes a list of tagged tokens as a gold standard to evaluate

the tagger. Evaluation is shown in below fig. Testing accuracy achieved was 72% on

the Marathi test data.



* Testing Result

Following is the POS tagging for the above news article:

