

Experiment 4

Name: Raviraj Shinde

Roll. No.: 18102A0010

Title: To study Parts of Speech (POS) Tagging using Penn Tree Bank

Estimated time to complete this experiment: 2 hours

Objective: To understand importance of classification of words in Natural language and the concept of using tags to represent the class. To apply the concept on English language and hence tag lexicons with appropriate parts of speech.

Expected Outcome of Experiment: POS Tags for every word in a sentence.

References:

1. <https://www.nltk.org/book/ch05.html>
-

Pre Lab/ Prior Concepts: Programming language – Python or R

Brief description:

The process of classifying words into their parts of speech and labeling them accordingly is known as part-of-speech tagging, POS-tagging, or simply tagging. Parts of speech are also known as word classes or lexical categories. The collection of tags used for a particular task is known as a tagset.

New Concepts to be learned: N-Gram model

Requirements: PC with Python or R installed.

Flow Chart: -

Theory:

1. Using a Tagger
A part-of-speech tagger, or POS-tagger, processes a sequence of words, and attaches a part of speech tag to each word.
2. Tagged Corpora
 - i. Representing Tagged Tokens

By convention in NLTK, a tagged token is represented using a tuple consisting of the token and the tag.

ii. Reading Tagged Corpora

Several of the corpora included with NLTK have been tagged for their part-of-speech. Other corpora use a variety of formats for storing part-of-speech tags. NLTK's corpus readers provide a uniform interface so that the user doesn't have to be concerned with the different file formats.

iii. A Universal Part-of-Speech Tagset

A Universal Tagged corpora use many different conventions for tagging words.

iv. Nouns

Nouns generally refer to people, places, things, or concepts, e.g.: woman, Scotland, book, intelligence. Nouns can appear after determiners and adjectives, and can be the subject or object of the verbs.

v. Verbs

Verbs are words that describe events and actions, e.g. fall, eat. Verbs typically express a relation involving the referents of one or more noun phrases.

Tags:

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coord. conj.	<i>and, but, or</i>	NNP	proper noun, sing.	<i>IBM</i>	TO	"to"	<i>to</i>
CD	cardinal number	<i>one, two</i>	NNPS	proper noun, plu.	<i>Carolinas</i>	UH	interjection	<i>ah, oops</i>
DT	determiner	<i>a, the</i>	NNS	noun, plural	<i>llamas</i>	VB	verb base	<i>eat</i>
EX	existential 'there'	<i>there</i>	PDT	predeterminer	<i>all, both</i>	VBD	verb past tense	<i>ate</i>
FW	foreign word	<i>mea culpa</i>	POS	possessive ending	<i>'s</i>	VBG	verb gerund	<i>eating</i>
IN	preposition/ subordin-conj	<i>of, in, by</i>	PRP	personal pronoun	<i>I, you, he</i>	VBN	verb past partici- ple	<i>eaten</i>
JJ	adjective	<i>yellow</i>	PRP\$	possess. pronoun	<i>your, one's</i>	VBP	verb non-3sg-pr	<i>eat</i>
JJR	comparative adj	<i>bigger</i>	RB	adverb	<i>quickly</i>	VBZ	verb 3sg pres	<i>eats</i>
JJS	superlative adj	<i>wildest</i>	RBR	comparative adv	<i>faster</i>	WDT	wh-determ.	<i>which, that</i>
LS	list item marker	<i>1, 2, One</i>	RBS	superlatv. adv	<i>fastest</i>	WP	wh-pronoun	<i>what, who</i>
MD	modal	<i>can, should</i>	RP	particle	<i>up, off</i>	WP\$	wh-possess.	<i>whose</i>
NN	sing or mass noun	<i>llama</i>	SYM	symbol	<i>+, %, &</i>	WRB	wh-adverb	<i>how, where</i>

Figure 8.2 Penn Treebank part-of-speech tags.

Program:

1. Using a Tagger

i. Finding tags of words in a sentence:

```
text = word_tokenize("And now for something completely different")
nltk.pos_tag(text)
```

```
text = word_tokenize("They refuse to permit us to obtain the refuse permit")
nltk.pos_tag(text)
```

ii. Finding similar words:

Consider the following analysis involving woman (a noun), bought (a verb), over (a preposition), and the (a determiner). The `text.similar()` method takes a word `w`, finds all contexts `w1w w2`, then finds all words `w'` that appear in the same context, i.e. `w1w'w2`.

```
text = nltk.Text(word.lower() for word in nltk.corpus.brown.words())
text.similar('woman')
text.similar('bought')
text.similar('over')
text.similar('the')
```

2. Tagged Corpora

i. Representing Tagged Tokens

By convention in NLTK, a tagged token is represented using a tuple consisting of the token and the tag. We can create one of these special tuples from the standard string representation of a tagged token, using the function `str2tuple()`.

```
tagged_token = nltk.tag.str2tuple('fly/NN')
tagged_token
tagged_token[0]
tagged_token[1]
```

We can construct a list of tagged tokens directly from a string. The first step is to tokenize the string to access the individual word/tag strings, and then to convert each of these into a tuple (using `str2tuple()`).

```
sent = ''
.. The/AT grand/JJ jury/NN commented/VBD on/IN a/AT number/NN of/IN ...
```

other/AP topics/NNS ./, AMONG/IN them/PPO the/AT Atlanta/NP and/CC
 Fulton/NP-tl County/NN-tl purchasing/VBG departments/NNS which/WDT
 it/PPS... said/VBD ``/`` ARE/BER well/QL operated/VBN and/CC follow/VB
 generally/RB accepted/VBN practices/NNS which/WDT inure/VB to/IN the/AT
 best/JJT
 ... interest/NN of/IN both/ABX governments/NNS "/" ./.
 ... ""

```
[nltk.tag.str2tuple(t) for t in sent.split()]
```

ii. **Reading tagged corpora**

Other corpora use a variety of formats for storing part-of-speech tags. NLTK's corpus readers provide a uniform interface so that you don't have to be concerned with the different file formats. In contrast with the file fragment shown above, the corpus reader for the Brown Corpus represents the data as shown below.

```
nltk.corpus.brown.tagged_words()
nltk.corpus.brown.tagged_words(tagset='universal')
```

Whenever a corpus contains tagged text, the NLTK corpus interface will have a `tagged_words()` method.

```
print(nltk.corpus.nps_chat.tagged_words())
nltk.corpus.conll2000.tagged_words()
nltk.corpus.treebank.tagged_words()
```

Use a built-in mapping to the "Universal Tagset"

```
nltk.corpus.brown.tagged_words(tagset='universal')
nltk.corpus.treebank.tagged_words(tagset='universal')
nltk.corpus.indian.tagged_words()
```

iii. **A Universal Part-of-Speech Tagset**

The most common tags in the news category of the Brown corpus

```
from nltk.corpus import brown
brown_news_tagged = brown.tagged_words(categories='news',
tagset='universal')
tag_fd = nltk.FreqDist(tag for (word, tag) in brown_news_tagged)
```

```
tag_fd.most_common()
```

iv. **Nouns**

Find what parts of speech occur before a noun, with the most frequent ones first.

```
word_tag_pairs = nltk.bigrams(brown_news_tagged)
noun_preceders = [a[1] for (a, b) in word_tag_pairs if b[1] == 'NOUN']
fdist = nltk.FreqDist(noun_preceders)
[tag for (tag, _) in fdist.most_common()]
```

v. **Verbs**

Most common verbs in news text

```
wsj = nltk.corpus.treebank.tagged_words(tagset='universal')
word_tag_fd = nltk.FreqDist(wsj)
[wt[0] for (wt, _) in word_tag_fd.most_common() if wt[1] == 'VERB']
```

Initialize a conditional frequency distribution with a list of condition-event pairs. Find a frequency-ordered list of tags given a word.

```
cfd1 = nltk.ConditionalFreqDist(wsj)
cfd1['yield'].most_common()
cfd1['cut'].most_common()
```

Reverse the order of the pairs, so that the tags are the conditions, and the words are the events. Find likely words for a given tag.

```
wsj = nltk.corpus.treebank.tagged_words()
cfd2 = nltk.ConditionalFreqDist((tag, word) for (word, tag) in wsj)
list(cfd2['VBN'])
```

To clarify the distinction between VBD (past tense) and VBN (past participle), find words which can be both VBD and VBN,

```
[w for w in cfd1.conditions() if 'VBD' in cfd1[w] and 'VBN' in cfd1[w]]
idx1 = wsj.index(('kicked', 'VBD'))
wsj[idx1-4:idx1+1]
idx2 = wsj.index(('kicked', 'VBN'))
wsj[idx2-4:idx2+1]
```

vi. Exploring Tagged Corpora

To study the word often - see how it is used in text. Find the words that follow often

```
brown_learned_text = brown.words(categories='learned')
sorted(set(b for (a, b) in nltk.bigrams(brown_learned_text) if a == 'often'))
```

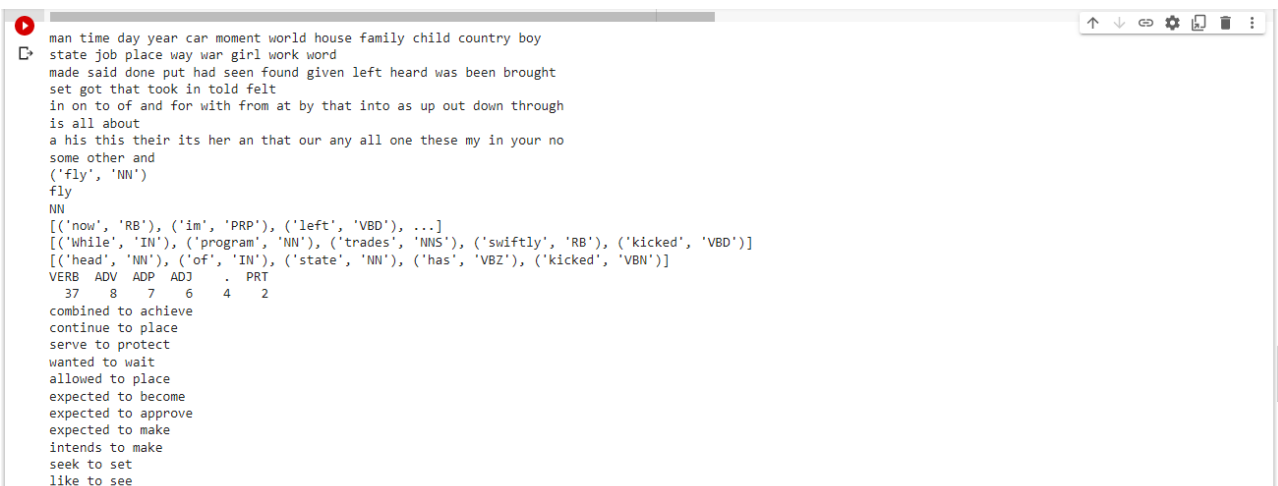
Using the tagged_words() method

```
brown_lrnd_tagged = brown.tagged_words(categories='learned',
tagset='universal')
tags = [b[1] for (a, b) in nltk.bigrams(brown_lrnd_tagged) if a[0] == 'often']
fd = nltk.FreqDist(tags)
fd.tabulate()
```

Find words involving particular sequences of tags and words (in this case "<Verb> to <Verb>")

```
from nltk.corpus import brown
def process(sentence):
    for (w1,t1), (w2,t2), (w3,t3) in nltk.trigrams(sentence): [1]
        if (t1.startswith('V') and t2 == 'TO' and t3.startswith('V')): [2]
            print(w1, w2, w3) [3]
for tagged_sent in brown.tagged_sents():
    process(tagged_sent)
```

Output:



```
man time day year car moment world house family child country boy
state job place way war girl work word
made said done put had seen found given left heard was been brought
set got that took in told felt
in on to of and for with from at by that into as up out down through
is all about
a his this their its her an that our any all one these my in your no
some other and
('fly', 'NN')
fly
NN
[['now', 'RB'], ('im', 'PRP'), ('left', 'VBD'), ...]
[['While', 'IN'], ('program', 'NN'), ('trades', 'NNS'), ('swiftly', 'RB'), ('kicked', 'VBD')]
[['head', 'NN'], ('of', 'IN'), ('state', 'NN'), ('has', 'VBZ'), ('kicked', 'VBN')]
VERB ADV ADP ADJ . PRT
37 8 7 6 4 2
combined to achieve
continue to place
serve to protect
wanted to wait
allowed to place
expected to become
expected to approve
expected to make
intends to make
seek to set
like to see
```

Conclusion:

- Words can be grouped into classes, such as nouns, verbs, adjectives, and adverbs. These classes are known as lexical categories or parts of speech. Parts of speech are assigned short labels, or tags, such as NN, VB,
- The process of automatically assigning parts of speech to words in text is called part-of-speech tagging, POS tagging, or just tagging.
- Automatic tagging is an important step in the NLP pipeline and is useful in a variety of situations including predicting the behaviour of previously unseen words, analysing word usage in corpora, and text-to-speech systems.
- Some linguistic corpora, such as the Brown Corpus, have been POS tagged.
- A variety of tagging methods are possible, e.g., default tagger, regular expression tagger, unigram tagger and n-gram taggers. These can be combined using a technique known as backoff.

Real Life Application:

- **Named Entity Recognition:** - Named-entity recognition is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc.
 - **Co-reference Resolution:** - Coreference resolution is the task of finding all expressions that refer to the same entity in a text. It is an important step for a lot of higher-level NLP tasks that involve natural language understanding such as document summarization, question answering, and information extraction.
 - **Speech Recognition**
-