# 3. NLP – Replacing and Correcting words

## Contents

# 3. NLP – Replacing and correcting words

## 1. Text conversions

- ✓ We can convert the text from lower to upper and upper to lower case

| | |
|---|---|
| Program Name | Converting lower case to upper case<br>demo1.py |

```
text ="hello good morning"
print(text.upper())
```

Output

```
HELLO GOOD MORNING
```

| | |
|---|---|
| Program Name | Converting upper case to lower case<br>demo2.py |

```
text ="HELLO GOOD MORNING"
print(text.lower())
```

Output

```
hello good morning
```

## 2. Removing numbers

- ✓ By using regular expression we can remove numbers from the text

Program Name: Removing numbers from the text
demo3.py

```
import re

myString = 'Box A has 4 red and 6 white balls, while Box B has 3 red and 5 blue balls.'

output = re.sub(r'\d+', '', myString)
print(output)
```

Output

```
Box A has red and white balls, while Box B has red and blue balls.
```

## 3. Removing punctuations

- ✓ By using regular expression we can remove the punctuations from the text

Program Name: Removing the punctuations from the text
demo4.py

```
import re

text = "Hello $@#$# Good !@#!@# morning #*#@&@#"

print("Text is:", text)

res = re.sub(r'[^\w\s]', '', text )

print("After punctuations:", res)
```

Output

```
Text is: Hello $@#$# Good !@#!@# morning #*#@&@#
After punctuations: Hello  Good  morning
```

## 4. Removing whitespaces

- ✓ We can remove the whitespaces in string by using strip() method.

Program Name: Removing whitespaces from text
demo5.py

```
text = "            a sample string      "

print(text)
res = text.strip()
print(res)
```

Output

```
            a sample string
a sample string
```

## 5. Part of Speech Tagging (POS)

- ✓ The goal of POS is to assign the various parts of a speech to every word of the provided text like nouns, adjectives, verbs, etc.
- ✓ This is normally done based on the definition and the context.
- ✓ Install textblob library,
    - pip install textblob

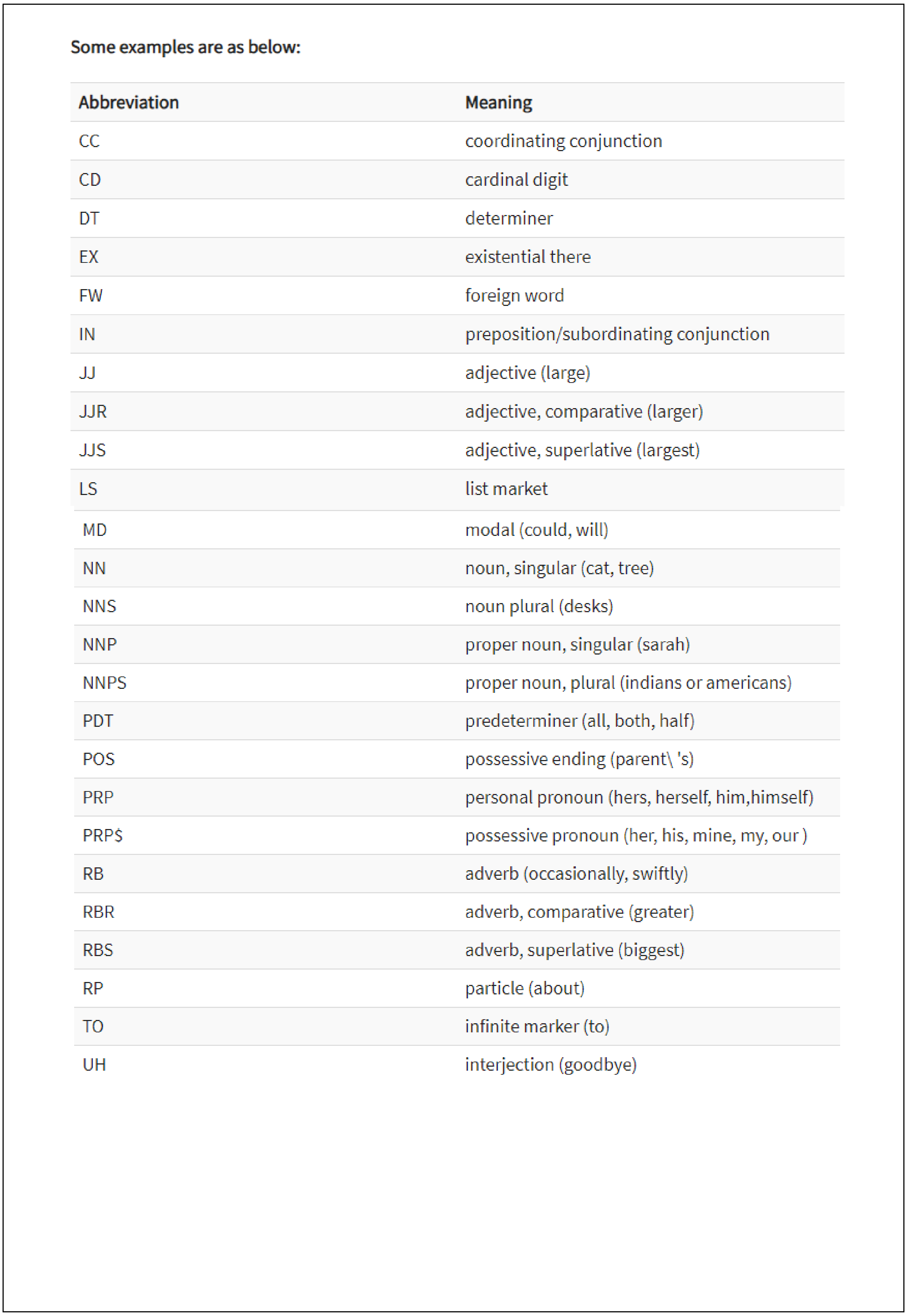
Program Name

Removing whitespaces from text
demo6.py

```
from textblob import TextBlob
import nltk

nltk.download('averaged_perceptron_tagger')

myString = "Parts of speech: an article, to run, fascinating, quickly, and, of"

output = TextBlob(myString)
print(output.tags)
```

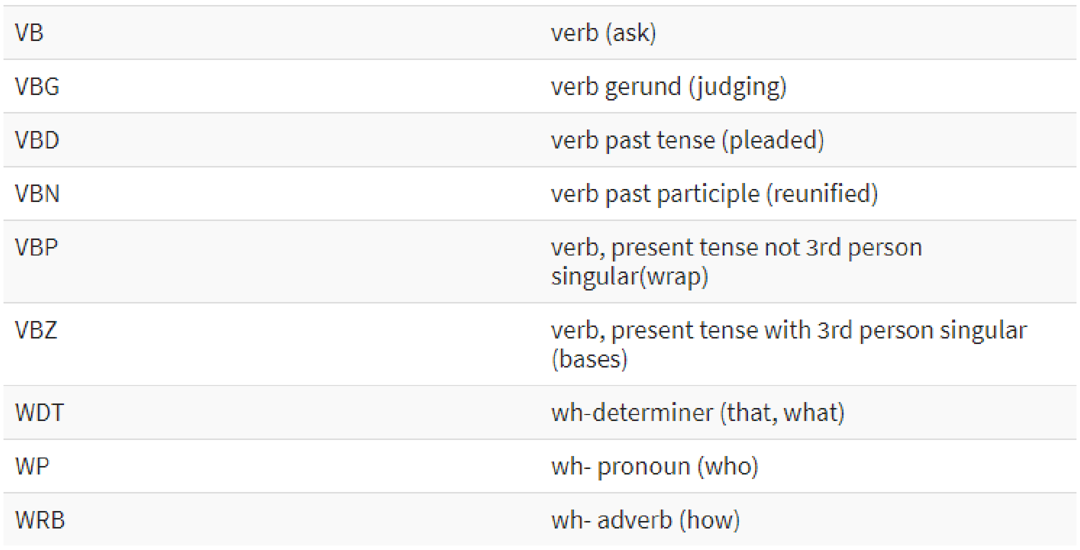
Output

```
[('Parts', 'NNS'), ('of', 'IN'), ('speech', 'NN'), ('an', 'DT'), ('article', 'NN'), ('to',
'TO'), ('run', 'VB'), ('fascinating', 'VBG'), ('quickly', 'RB'), ('and', 'CC'), ('of', 'IN')]
```



Some examples are as below:

| Abbreviation | Meaning |
|---|---|
| CC | coordinating conjunction |
| CD | cardinal digit |
| DT | determiner |
| EX | existential there |
| FW | foreign word |
| IN | preposition/subordinating conjunction |
| JJ | adjective (large) |
| JJR | adjective, comparative (larger) |
| JJS | adjective, superlative (largest) |
| LS | list market |
| MD | modal (could, will) |
| NN | noun, singular (cat, tree) |
| NNS | noun plural (desks) |
| NNP | proper noun, singular (sarah) |
| NNPS | proper noun, plural (indians or americans) |
| PDT | predeterminer (all, both, half) |
| POS | possessive ending (parent\ 's) |
| PRP | personal pronoun (hers, herself, him,himself) |
| PRP$ | possessive pronoun (her, his, mine, my, our ) |
| RB | adverb (occasionally, swiftly) |
| RBR | adverb, comparative (greater) |
| RBS | adverb, superlative (biggest) |
| RP | particle (about) |
| TO | infinite marker (to) |
| UH | interjection (goodbye) |



| | |
|---|---|
| VB | verb (ask) |
| VBG | verb gerund (judging) |
| VBD | verb past tense (pleaded) |
| VBN | verb past participle (reunified) |
| VBP | verb, present tense not 3rd person singular(wrap) |
| VBZ | verb, present tense with 3rd person singular (bases) |
| WDT | wh-determiner (that, what) |
| WP | wh- pronoun (who) |
| WRB | wh- adverb (how) |

Program Name: pos example demo7.py

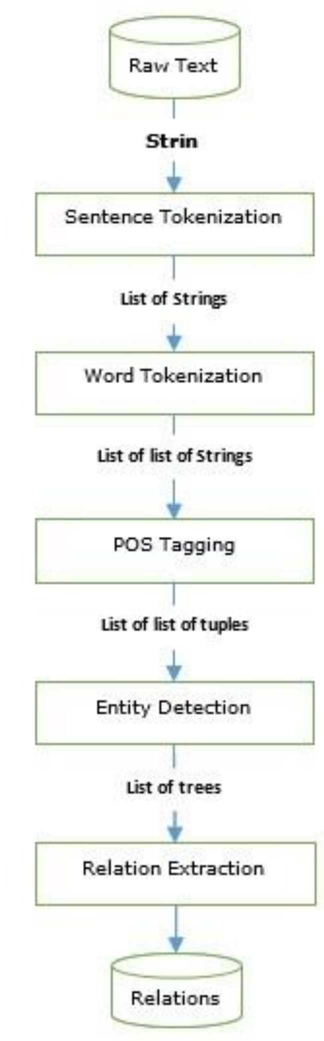
```
from nltk.corpus import wordnet

syn = wordnet.synsets('hello')[0]
print("Syn tag : ", syn.pos())

syn = wordnet.synsets('doing')[0]
print("Syn tag : ", syn.pos())

syn = wordnet.synsets('beautiful')[0]
print("Syn tag : ", syn.pos())
```

Output

```
Syn tag :  n
Syn tag :  v
Syn tag :  a
Syn tag :  r
```

## 6. Information Extraction

- ✓ We need to understand the tags and parsers to build information extraction engine.
- ✓ Let us see a basic information extraction pipeline



**7. Information extraction has many applications including**

- ✓ Business intelligence
- ✓ Resume harvesting
- ✓ Media analysis
- ✓ Sentiment detection
- ✓ Patent search
- ✓ Email scanning

## 8. Collocations: Bigrams and Trigrams

### What is Collocations?

- ✓ Collocations are the pairs of words occurring together many times in paragraphs.
- ✓ It is calculated by the number of those pair occurring together to the overall word count of the paragraph.
- ✓ We can say that finding collocations requires calculating the frequencies of words and their appearance in the context of other words.

### Bigrams and Trigrams

- ✓ Collocation can be categorized into two types,
  - Bigrams combination of two words
  - Trigrams combination of three words
- ✓ Bigrams and Trigrams provide more meaningful and useful features for the feature extraction stage.
- ✓ These are especially useful in text-based sentimental analysis.

Program Name: Bigram example
demo8.py

```
import nltk

text = "Data Science is a totally new kind of learning experience."
Tokens = nltk.word_tokenize(text)
output = list(nltk.bigrams(Tokens))

print(output)
```

Output

```
[('Data', 'Science'), ('Science', 'is'), ('is', 'a'), ('a', 'totally'), ('totally', 'new'),
('new', 'kind'), ('kind', 'of'), ('of', 'learning'), ('learning', 'experience'),
('experience', '.')]
```

| | |
|---|---|
| Program Name | Trigram example<br>demo9.py |

```
import nltk

text = "Data Science is a totally new kind of learning experience."
Tokens = nltk.word_tokenize(text)
output = list(nltk.trigrams(Tokens))

print(output)
```

Output

```
[('Data', 'Science', 'is'), ('Science', 'is', 'a'), ('is', 'a', 'totally'), ('a', 'totally',
'new'), ('totally', 'new', 'kind'), ('new', 'kind', 'of'), ('kind', 'of', 'learning'), ('of',
'learning', 'experience'), ('learning', 'experience', '.')]
```

## 9. Wordnet

- ✓ Wordnet is an NLTK lexical database for English.
- ✓ It can be used to find the meaning of words, synonym or antonym.

### synset

- ✓ Synset is a special kind of a simple interface that is present in NLTK to look up words in Wordnet.
- ✓ Synset instances are the groupings of synonymous words that express the same concept.

Program Name: wordnet example
demo10.py

```
from nltk.corpus import wordnet

syn = wordnet.synsets('hello')[0]

print ("Synset name :", syn.name())
print ("Synset meaning :", syn.definition())
print ("Synset example :", syn.examples())
```

Output

```
Synset name : hello.n.01
Synset meaning : an expression of greeting
Synset example : ['every morning they exchanged polite hellos']
```

Program Name

wordnet example
demo11.py

```
from nltk.corpus import wordnet

syn = wordnet.synsets('boy')[0]

print ("Synset name :", syn.name())
print ("Synset meaning :", syn.definition())
print ("Synset example :", syn.examples())
```

Output

```
Synset name : male_child.n.01
Synset meaning : a youthful male person
Synset example : ['the baby was a boy', 'she made the boy brush his teeth
every night', 'most soldiers are only boys in uniform']
```

Program Name

wordnet example
demo12.py

```
from nltk.corpus import wordnet

syn = wordnet.synsets('good')[0]

print ("Synset name :", syn.name())
print ("Synset meaning :", syn.definition())
print ("Synset example :", syn.examples())
```

Output

```
Synset name : good.n.01
Synset meaning : benefit
Synset example: ['for your own good', "what's the good of worrying?"]
```