This is AI4001

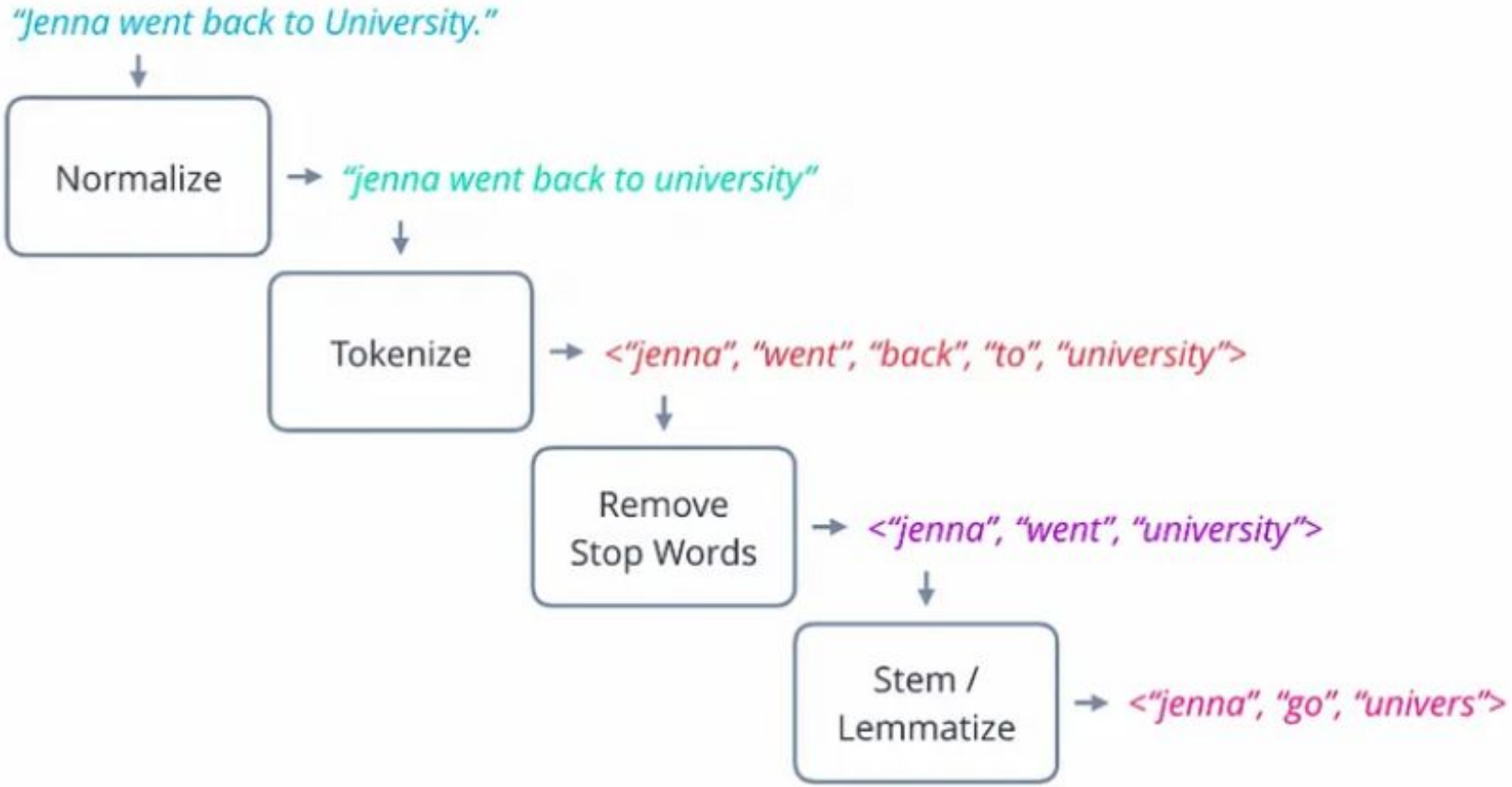
GCR : t37g47w

Text Normalization

Text Normalization

At least three tasks are commonly applied as part of any normalization process:

1. Segmenting/tokenizing words from running text
2. Normalizing word formats
3. Segmenting sentences in running text.


"Jenna went back to University."
Normalize
"jenna went back to university"
Tokenize
<"jenna", "went", "back", "to", "university">
Remove Stop Words
<"jenna", "went", "university">
Stem / Lemmatize
<"jenna", "go", "univers">

Tokenization: the task of segmenting running text into words

Normalization: the task of putting words/tokens in a standard format

Tokenization

Tokenization breaks the raw text into words, sentences called tokens.

These tokens help in understanding the context or developing the model for the NLP.

The tokenization helps in interpreting the meaning of the text by analyzing the sequence of the words.

Tokenization

As we now know, Tokenization helps split the original text into characters, words, sentences, etc. depending upon the problem at hand.

If you split the text data (or document) into words, it's called **Word Tokenization**.

If the document is split into sentences, then it is called **Sentence Tokenization**.

Similarly, splitting the document into individual characters is known as **Character Tokenization**.

Symbols Importance

Break off punctuation as a separate token; commas are a useful piece of information for parsers, periods help indicate sentence boundaries.

In examples like m.p.h,, Ph.D., AT&T, cap'n.

Special characters and numbers will need to be kept in prices ($45.55) and dates (01/02/06);

URLs (http://www.stanford.edu), Twitter hashtags (#nlproc), or email addresses (someone@cs.colorado.edu).

A clitic is a part of a word that can't stand on its own, and can only occur when it is attached to another word. e.g. converting what're to the two tokens what are, and we're to we are.

Tokenization

```
[3] # text corpus
    tweet = "Seek wealth, not money or status. Wealth is having assets \
            that earn while you sleep. Money is how we transfer time and wealth. \
            Status is your place in the social hierarchy."
    tweet       # preview
```

```
'Seek wealth, not money or status. Wealth is having assets that earn while you slee
p. Money is how we transfer time and wealth. Status is your place in the social hie
rarchy.'
```

Tokenization

```
[4] # text corpus
    text = "We are the best meal providers in U.S. and our meals starts from $10. \
           Feel free to reach out to us! You can tweet us on @example or \
           email us at email@example.com. Don't forget to \
           visit our website https://www.example.com/ 😊"
    text        # preview

    'We are the best meal providers in U.S. and our meals starts from $10. Feel free to
    reach out to us! You can tweet us on @example or email us at email@example.com. Do
    n't forget to visit our website https://www.example.com/ 😊'
```

In [3]:
```
# word tokenization
print(tweet.split())
print(f"# tokens: {len(tweet.split())}")
```

```
['Seek', 'wealth,', 'not', 'money', 'or', 'status.', 'Wealth', 'is', 'havin
g', 'assets', 'that', 'earn', 'while', 'you', 'sleep.', 'Money', 'is', 'how',
'we', 'transfer', 'time', 'and', 'wealth.', 'Status', 'is', 'your', 'place',
'in', 'the', 'social', 'hierarchy.']
# tokens: 31
```

In [4]:
```
# word tokenization
print(text.split())
print(f"# tokens: {len(text.split())}")
```

```
['We', 'are', 'the', 'best', 'meal', 'providers', 'in', 'U.S.', 'and', 'our',
'meals', 'starts', 'from', '$10.', 'Feel', 'free', 'to', 'reach', 'out', 't
o', 'us!', 'You', 'can', 'tweet', 'us', 'on', '@example', 'or', 'email', 'u
s', 'at', 'email@example.com.', "Don't", 'forget', 'to', 'visit', 'our', 'web
site', 'https://www.example.com/', '😊']
# tokens: 40
```

Word Tokenization

The punctuations have not been separated from the word, for example: “status.” and “us!”.

Also, notice how prefixes like “$” have not been separated from the token in “$10”.

Sentence Tokenization

```
# sentence tokenization
tweet.split(".")
```

```
['Seek wealth, not money or status',
 ' Wealth is having assets that earn while you sleep',
 ' Money is how we transfer time and wealth',
 ' Status is your place in the social hierarchy',
 '']
```

Sentence Tokenization

```
# sentence tokenization
text.split(".")
```

```
['We are the best meal providers in U',
 'S',
 ' and our meals starts from $10',
 ' Feel free to reach out to us! You can tweet us on @example or email us at email@e
 'com',
 " Don't forget to visit our website https://www",
 'example',
 'com/ 😊']
```

Word tokenization with NLTK

NLTK offers a bunch of different methods for word tokenization. We will explore the following:

word_tokenize()

TreebankWordTokenizer

WordPunctTokenizer

RegEx

Word Tokenization with NLTK

word_tokenize

In [4]:

```
# import
from nltk.tokenize import word_tokenize

# word tokenization
print(word_tokenize(tweet))
print(f"# tokens: {len(word_tokenize(tweet))}")
```

```
['Seek', 'wealth', ',', 'not', 'money', 'or', 'status', '.', 'Wealth', 'is',
'having', 'assets', 'that', 'earn', 'while', 'you', 'sleep', '.', 'Money', 'i
s', 'how', 'we', 'transfer', 'time', 'and', 'wealth', '.', 'Status', 'is', 'y
our', 'place', 'in', 'the', 'social', 'hierarchy', '.']
# tokens: 36
```

Word Tokenization with NLTK

TreebankWordTokenizer

In [ ]:

```
# import
from nltk.tokenize import TreebankWordTokenizer

# word tokenization
word_tokenizer = TreebankWordTokenizer()
print(word_tokenizer.tokenize(tweet))
print(f"# tokens: {len(word_tokenizer.tokenize(tweet))}")
```

```
['Seek', 'wealth', ',', 'not', 'money', 'or', 'status.', 'Wealth', 'is', 'hav
ing', 'assets', 'that', 'earn', 'while', 'you', 'sleep.', 'Money', 'is', 'ho
w', 'we', 'transfer', 'time', 'and', 'wealth.', 'Status', 'is', 'your', 'plac
e', 'in', 'the', 'social', 'hierarchy', '.']
# tokens: 33
```

Word Tokenization with NLTK

WordPunctTokenizer

In [ ]:

```
# import
from nltk.tokenize import WordPunctTokenizer

# word tokenization
word_tokenizer = WordPunctTokenizer()
print(word_tokenizer.tokenize(tweet))
print(f"# tokens: {len(word_tokenizer.tokenize(tweet))}")
```

```
['Seek', 'wealth', ',', 'not', 'money', 'or', 'status', '.', 'Wealth', 'is',
'having', 'assets', 'that', 'earn', 'while', 'you', 'sleep', '.', 'Money', 'i
s', 'how', 'we', 'transfer', 'time', 'and', 'wealth', '.', 'Status', 'is', 'y
our', 'place', 'in', 'the', 'social', 'hierarchy', '.']
# tokens: 36
```

Word Tokenization with NLTK

RegexpTokenizer

In [ ]:

```
# import
from nltk.tokenize import RegexpTokenizer

# word tokenization
word_tokenizer = RegexpTokenizer("[\w']+")
print(word_tokenizer.tokenize(tweet))
print(f"# tokens: {len(word_tokenizer.tokenize(tweet))}")
```

```
['Seek', 'wealth', 'not', 'money', 'or', 'status', 'Wealth', 'is', 'havin
g', 'assets', 'that', 'earn', 'while', 'you', 'sleep', 'Money', 'is', 'ho
w', 'we', 'transfer', 'time', 'and', 'wealth', 'Status', 'is', 'your', 'pla
ce', 'in', 'the', 'social', 'hierarchy']
# tokens: 31
```

Sentence tokenization with NLTK

Sentence tokenization is the process of splitting the text corpus into different sentences.

`sent_tokenize()`

`PunktSentenceTokenizer`

Sentence tokenization with NLTK

```
[ ]  # import
     from nltk.tokenize import sent_tokenize

     # sentence tokenization
     sent_tokenize(tweet)
```

```
['Seek wealth, not money or status.',
 'Wealth is having assets that earn while you sleep.',
 'Money is how we transfer time and wealth.',
 'Status is your place in the social hierarchy.']
```

```
[ ]  # number of tokens
     print(f"# tokens: {len(sent_tokenize(tweet))}")
```

```
# tokens: 4
```

Sentence tokenization with NLTK

```
[58] # import
     from nltk.tokenize import sent_tokenize

     # sentence tokenization
     sent_tokenize(text)
```

```
['We are the best meal providers in U.S. and our meals starts from $10.',
 'Feel free to reach out to us!',
 'You can tweet us on @example or email us at email@example.com.',
 "Don't forget to visit our website https://www.example.com/ 😊"]
```

```
[59] # number of tokens
     print(f"# tokens: {len(sent_tokenize(text))}")
```

```
# tokens: 4
```

Sentence tokenization with NLTK

```
[54] # import
     from nltk.tokenize import PunktSentenceTokenizer

     # sentence tokenization
     sent_tokenizer = PunktSentenceTokenizer()
     sent_tokenizer.tokenize(text)
```

```
['We are the best meal providers in U.S.',
 'and our meals starts from $10.',
 'Feel free to reach out to us!',
 'You can tweet us on @example or email us at email@example.com.',
 "Don't forget to visit our website https://www.example.com/ 😊"]
```

```
[60] len(sent_tokenizer.tokenize(text))
```

```
5
```

Sentence tokenization with NLTK

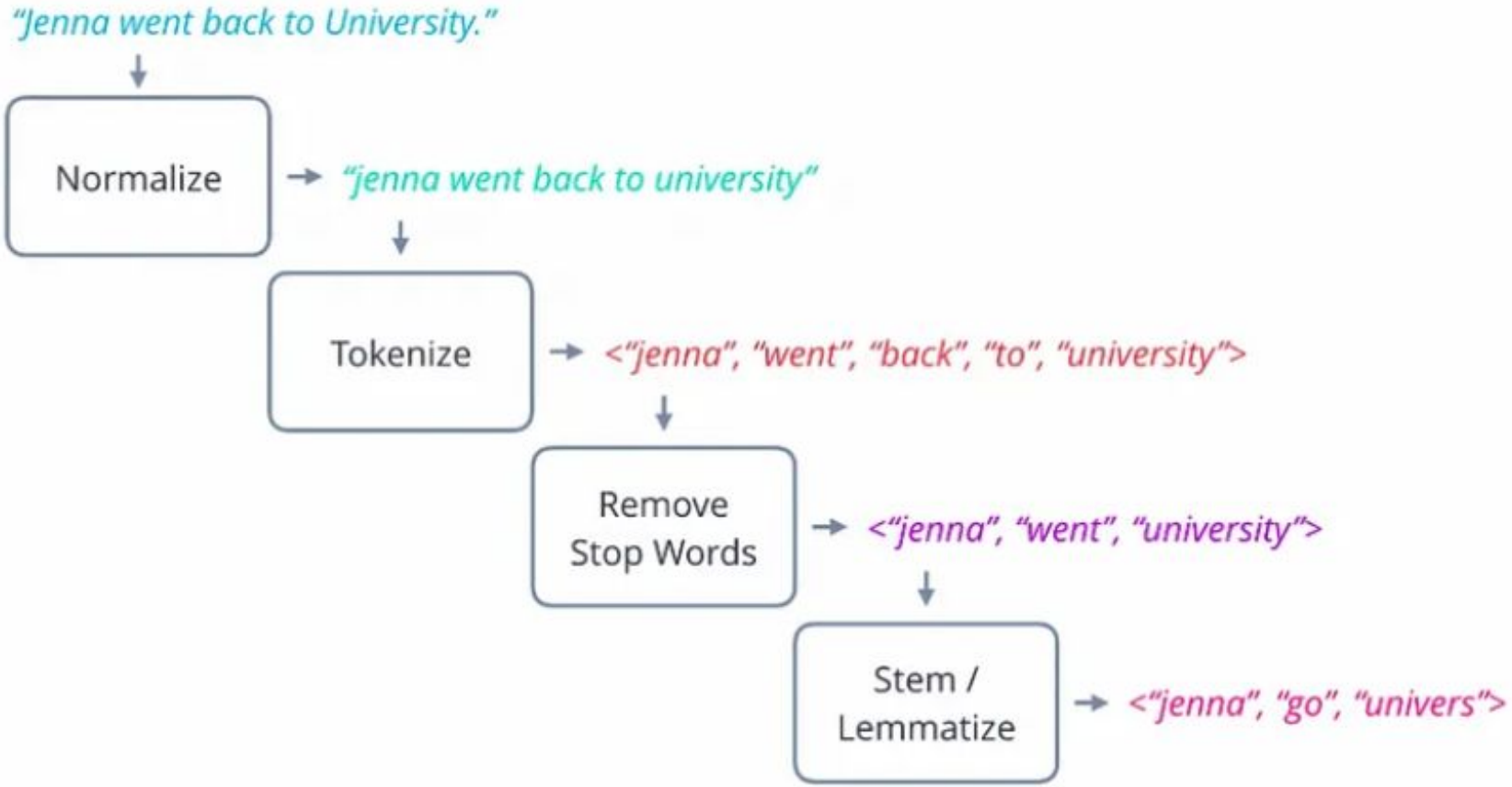
```
[54] # import
     from nltk.tokenize import PunktSentenceTokenizer

     # sentence tokenization
     sent_tokenizer = PunktSentenceTokenizer()
     sent_tokenizer.tokenize(text)
```

```
['We are the best meal providers in U.S.',
 'and our meals starts from $10.',
 'Feel free to reach out to us!',
 'You can tweet us on @example or email us at email@example.com.',
 "Don't forget to visit our website https://www.example.com/ 😊"]
```

```
[60] len(sent_tokenizer.tokenize(text))
```

```
5
```


"Jenna went back to University."
Normalize
"jenna went back to university"
Tokenize
<"jenna", "went", "back", "to", "university">
Remove Stop Words
<"jenna", "went", "university">
Stem / Lemmatize
<"jenna", "go", "univers">

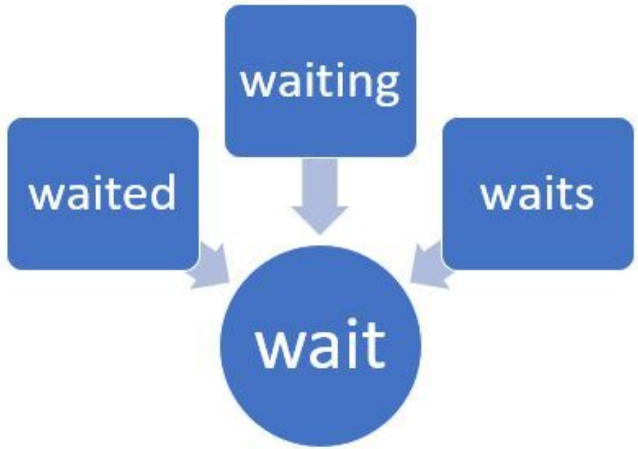
Case Folding

Lowercasing all alphabets

US vs us

Stemming

Stemming is a technique used to extract the base form of the words by removing affixes from them. It is just like cutting down the branches of a tree to its stems. For example, the stem of the words eating, eats, eaten is eat.



PorterStemmer class

```
import nltk

from nltk.stem import PorterStemmer

word_stemmer = PorterStemmer()

word_stemmer.stem('writing')
```

Porter Stemmer

This was not the map we found in Billy Bones's chest, but an accurate copy, complete in all things-names and heights and soundings-with the single exception of the red crosses and the written notes.

Thi wa not the map we found in Billi Bone s chest but an accur copi complet in all thing name and height and sound with the singl except of the red cross and the written note

Lancaster stemming class

```
Import nltk
from nltk.stem import LancasterStemmer
Lanc_stemmer = LancasterStemmer()
Lanc_stemmer.stem('eats')
```

RegexpStemmer class

```
import nltk
from nltk.stem import RegexpStemmer
Reg_stemmer = RegexpStemmer(‘ing’)
Reg_stemmer.stem('eating')
```

Problems in Stemming

Over-stemming: where a much larger part of a word is chopped off than what is required, which in turn leads to words being reduced to the same root word or stem incorrectly when they should have been reduced to more stem words. For example, the words “university” and “universe” that get reduced to “univers”.

Problems in Stemming

Under-stemming: occurs when two or more words could be wrongly reduced to more than one root word when they actually should be reduced to the same root word. For example, the words “data” and “datum” that get reduced to “dat” and “datu” respectively (instead of the same stem “dat”).

Lemmatization

The output we will get after lemmatization is called ‘lemma’, which is a root word rather than root stem.

WordNetLemmatizer class

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
lemmatizer.lemmatize('books')
```

Stemming Vs Lemmatization

Stemming	Lemmatization
Stemming is a process that stems or removes last few characters from a word, often leading to incorrect meanings and spelling.	**Lemmatization** considers the context and converts the word to its meaningful base form, which is called Lemma.
For instance, stemming the word '**Caring**' would return '**Car**'.	For instance, lemmatizing the word '**Caring**' would return '**Care**'.
Stemming is used in case of large dataset where performance is an issue.	Lemmatization is computationally expensive since it involves look-up tables and what not.

https://spotintelligence.com/2023/01/25/text-normalization-techniques-nlp/

https://towardsdatascience.com/text-normalization-for-natural-language-processing-nlp-70a314bfa646

https://medium.com/@utkarsh.kant/tokenization-a-complete-guide-3f2dd56c0682

https://www.tutorialspoint.com/natural_language_toolkit/natural_language_toolkit_stemming_lemmatization.htm