SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 97
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Artificial Intelligence and Data Science

Course Name – 19AD601 – Natural Language Processing

III Year / VI Semester

Unit 1 – Introduction

Topic 4- Tokenization

Tokenization

Tokenization is splitting the raw text into small chunks of words or sentences, called tokens.

If the text is split into words, then its called as 'Word Tokenization' and if it's split into sentences then its called as 'Sentence Tokenization’.

Generally 'space' is used to perform the word tokenization and characters like 'periods, exclamation point and newline char are used for Sentence Tokenization.

Simple Approaches to Tokenization

The very simplest method for tokenizing text is to split on whitespace. The simplest way to tokenize text is to use whitespace within a string as the “delimiter” of words. This can be accomplished with Python’s split function, which is available on all string object instances as well as on the string built-in class itself.

Tokenization

Word Tokenization

Word Tokenization is the most commonly used tokenization algorithm. It splits a piece of text into individual words based on a certain delimiter. Depending upon delimiters, different word-level tokens are formed.

Example

```
text = """There are multiple ways we can perform tokenization on given text data. We can choose any method based on langauge, library and purpose of modeling."""
# Split text by whitespace
tokens = text.split()
print(tokens)
```

Output:

```
['There', 'are', 'multiple', 'ways', 'we', 'can', 'perform', 'tokenization', 'on', 'given', 'text', 'data.', 'We', 'can', 'choose', 'any', 'method', 'based', 'on', 'langauge,', 'library', 'and', 'purpose', 'of', 'modeling.']
```

Tokenization

NLTK Word Tokenize

Natural Language Toolkit (NLTK) is library written in python for natural language processing. NLTK has module word_tokenize() for word tokenization and sent_tokenize() for sentence tokenization.

Example

```
from nltk.tokenize import word_tokenize
text = """There are multiple ways we can perform tokenization on given text data. We can choose any method based on langauge, library and purpose of modeling."""
tokens = word_tokenize(text)
print(tokens)
```

Tokenization

Tokenization Using Regular Expressions(RegEx)

A regular expression is a sequence of characters that define a search pattern.Using RegEx we can match character combinations in string and perform word/sentence tokenization. We can use Python's re library for RegeEx related operations.

Example

```
import re
text = """There are multiple ways we can perform tokenization on given text data. We can choose any method based on langauge, library and purpose of modeling."""
tokens = re.findall("[\w]+", text)
print(tokens)
```

Tokenization

Sentence Tokenization

Sentence tokenization is the process of splitting text into individual sentences. Similar to word tokenization, sentence tokenization can performed by simple python library function split, NLTK sent_tokenize() module and Regular Expression.

Example

Simple Sentence Tokenization using split

```
text = """Characters like periods, exclamation point and newline char are used to separate the sentences. But one drawback with split() method, that we can only use one separator at a time! So sentence tonenization wont be foolproof with split() method."""
text.split(". ")
```

Tokenization

Sentence Tokenization using NLTK

sent_tokenize() module is used for sentence tokenization.

Example

from nltk.tokenize import sent_tokenize

text = """Characters like periods, exclamation point and newline char are used to separate the sentences. But one drawback with split() method, that we can only use one separator at a time! So sentence tonenization wont be foolproof with split() method."""

sent_tokenize(text)

Tokenization

Sentence Tokenization using RegEx

Example

```
import re

text = """Characters like periods, exclamation point and newline char are used to separate the sentences. But one drawback with split() method, that we can only use one separator at a time! So sentence tonenization wont be foolproof with split() method."""

tokens_sent = re.compile('[.!?] ').split(text)

tokens_sent
```

Output

```
['Characters like periods, exclamation point and newline char are used to separate the sentences',
 'But one drawback with split() method, that we can only use one separator at a time',
 'So sentence tonenization wont be foolproof with split() method.']
```

THANK YOU