1. Các thư viện NLP

* Natural language toolkit (NLTK)
* Apache OpenNLP.
* Stanford NLP suite.
* Gate NLP library.

1. Giới thiệu NLTK

NLTK hay Natural Language Toolkit - Bộ công cụ ngôn ngữ tự nhiên, là một thư viện được viết bằng Python hỗ trợ xử lý ngôn ngữ tự nhiên. Bằng cách cung cấp các cơ chế và kỹ thuật xử lý ngôn ngữ phổ biến, nó giúp cho việc xử lý ngôn ngữ tự nhiên trở lên dễ dàng và nhanh chóng hơn. Được viết bởi Steven Bird và Edward Loper, làm việc tại Khoa Máy Tính, Đại Học Pennsylvania, Hoa Kỳ và năm 2001. Ngoài việc hỗ trợ xử lý ngôn ngữ, NLTK còn có các mô phỏng đồ hoạ và dữ liệu mẫu hữu ích. NLTK cung cấp các xử lý như classification, tokenization, stemming, tagging, parsing, và semantic reasoning... Những ứng dụng này chúng ta sẽ dần được tìm hiểu ở những bài viết sau. Ngoài việc phục vụ xử lý ngôn ngữ tự nhiên, NLTK còn được sử dụng trong Machine Learning với tác dụng làm sạch dữ liệu, xử lý dữ liệu đầu vào cho các thuật toán Machine Learning.

**Tokenize Text Using Pure Python (Mã hóa văn bản bằng python thuần túy (Làm sạch text))**

Mục đích bước này là loại bỏ noise trong data của bạn. Đa phần noise là các thẻ HTML, JavaScript, và đương nhiên nếu cứ để noise để tiến hành xử lý sẽ dẫn đến kết quả xử lý không tốt.

Sử dụng mô đun ***urllib*** để thu thập dữ liệu trang web

Sử dụng ***BeautifulShoup***để làm sạch văn bản (làm sạch các thẻ html)

Ví dụ:

**from** bs4 **import** BeautifulSoup  
  
**import** urllib.request  
  
response = urllib.request.urlopen(**'http://php.net/'**)  
  
html = response.read()  
  
soup = BeautifulSoup(html, **"html.parser"**)  
  
text = soup.get\_text(strip=**True**)  
  
tokens = [t **for** t **in** text.split()]  
  
print(tokens)

**Count Word Frequency (đếm tần số từ)**

Hàm chức năng ***FreqDist()***: tính toán phân phối tần số của các mã thông báo đó bằng Python NLTK.

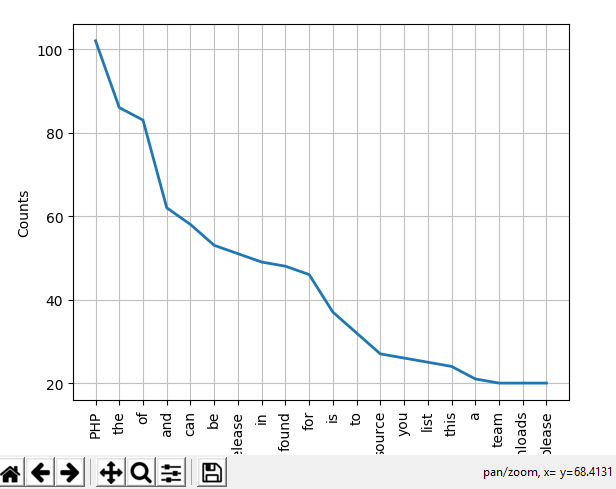
Ví dụ:

**from** bs4 **import** BeautifulSoup  
  
**import** urllib.request  
  
**import** nltk  
  
response = urllib.request.urlopen(**'http://php.net/'**)  
  
html = response.read()  
  
soup = BeautifulSoup(html, **"html.parser"**)  
  
text = soup.get\_text(strip=**True**)  
  
tokens = [t **for** t **in** text.split()]  
  
freq = nltk.FreqDist(tokens)  
  
**for** key, val **in** freq.items():  
 print(str(key) + **':'** + str(val))

freq.plot(20, cumulative=**False**)

Kết quả:

Có một số từ như The, Of, a, an, v.v. Những từ này là những từ dừng lại. Nói chung, dừng các từ cần được loại bỏ để ngăn chúng khỏi ảnh hưởng đến kết quả

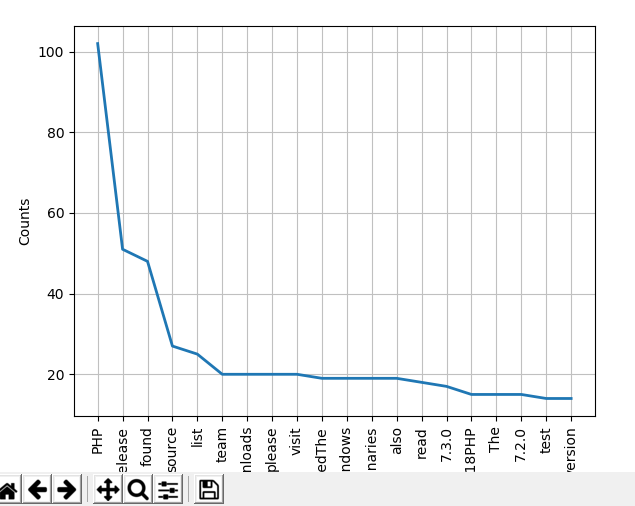


**Remove Stop Words Using NLTK**

Trong hầu hết các nhiệm vụ NLP và tìm kiếm thông tin, người ta thường loại bỏ các từ ngữ không cần thiết, các mạo từ, StopWords là những từ xuất hiện nhiều trong ngôn ngữ tự nhiên, tuy nhiên lại không mang nhiều ý nghĩa.

Ví dụ:

**from** bs4 **import** BeautifulSoup  
  
**import** urllib.request  
  
**import** nltk  
  
**from** nltk.corpus **import** stopwords  
  
response = urllib.request.urlopen(**'http://php.net/'**)  
  
html = response.read()  
  
soup = BeautifulSoup(html, **"html.parser"**)  
  
text = soup.get\_text(strip=**True**)  
  
tokens = [t **for** t **in** text.split()]  
  
clean\_tokens = tokens[:]  
  
sr = stopwords.words(**'english'**)  
  
**for** token **in** tokens:  
  
 **if** token **in** stopwords.words(**'english'**):  
 clean\_tokens.remove(token)  
  
freq = nltk.FreqDist(clean\_tokens)  
  
**for** key, val **in** freq.items():  
 print(str(key) + **':'** + str(val))  
   
freq.plot(20,cumulative=**False**)



So sánh với kết quả của hình phía trên ta thấy các từ: the, of, and, can, be... đã được lại bỏ

**Tokenize Text Using NLTK (Mã hóa văn bản bằng NLTK)**

Chúng ta đã thấy cách tách văn bản thành các thẻ bằng cách sử dụng chức năng chia tách, bây giờ chúng ta sẽ thấy cách mã hóa văn bản bằng cách sử dụng NLTK

Mã hóa văn bản quan trọng vì văn bản không thể được xử lý mà không cần mã thông báo. Quá trình mã hóa có nghĩa là chia nhỏ các phần lớn hơn thành các phần nhỏ.

Bạn có thể mã hóa các đoạn văn thành các câu và mã hóa các câu thành các từ theo nhu cầu của bạn. NLTK được gửi kèm với bộ mã thông báo và từ mã thông báo.

* Sent\_tokenize (tách câu):
* Một vài ứng dụng NLP yêu cầu việc tách một văn bản thô lớn vảo câu để lấy thông tin nhiều hơn. Câu là thành phần của một cuộc đối thoại. Khi nói đến máy tính, đây là nhiệm vụ khó hơn. Một sentence splitter điển hình đơn giản như việc tách các chuỗi dựa vào dấu chấm, phức tạp hơn như một bộ phân loại, xác định câu.

Ví dụ :

**from** nltk.tokenize **import** sent\_tokenize  
  
mytext = **"Hello Mr. Adam, how are you? I hope everything is going well. Today is a good day,see you dude."**print(sent\_tokenize(mytext))

Kết quả:

['Hello Mr. Adam, how are you?', 'I hope everything is going well.', 'Today is a good day, see you dude.']

* Sentence splitter trên phù hợp với 17 ngôn ngữ. Bạn chỉ cần xác định đối tượng tương ứng. NTLK sử dụng ***PunktSentenceTokenizer*** là một phần của mô-đun ***nltk.tokenize.punkt***
* Word\_tokenize(tách từ)

Ví dụ:

**from** nltk.tokenize **import** word\_tokenize  
  
mytext = **"Hello Mr. Adam, how are you? I hope everything is going well. Today is a good day, see you dude."**print(word\_tokenize(mytext))

Kết quả:

['Hello', 'Mr.', 'Adam', ',', 'how', 'are', 'you', '?', 'I', 'hope', 'everything', 'is', 'going', 'well', '.', 'Today', 'is', 'a', 'good', 'day', ',', 'see', 'you', 'dude', '.']

**Get Synonyms from WordNet(Lấy từ đồng nghĩa từ WordNet)**

WordNet là một cơ sở dữ liệu được xây dựng để xử lý ngôn ngữ tự nhiên. Nó bao gồm các nhóm từ đồng nghĩa và định nghĩa ngắn gọn

WordNet bao gồm rất nhiều định nghĩa

Bạn có thể sử dụng WordNet để nhận các từ đồng nghĩa như sau:

**import** nltk

**from** nltk.corpus **import** wordnet

synonyms = []  
**for** syn **in** wordnet.synsets(**'Computer'**):  
 **for** lemma **in** syn.lemmas():  
 synonyms.append(lemma.name())  
print(synonyms)

Kết quả:

['computer', 'computing\_machine', 'computing\_device', 'data\_processor', 'electronic\_computer', 'information\_processing\_system', 'calculator', 'reckoner', 'figurer', 'estimator', 'computer']

**Get Antonyms from WordNet (Lấy từ trái nghĩa từ WordNet)**

**NLTK Word Stemming**

Từ gốc có nghĩa là loại bỏ các từ ghép từ và trả về từ gốc. Ví dụ: Gốc của từ working => work

Ví dụ:

**from** nltk.stem **import** PorterStemmer  
  
stemmer = PorterStemmer()  
  
print(stemmer.stem(**'working'**))

Kết quả: work

**Lemmatizing Words Using WordNet**

lemmatizing tương tự như stemming, nhưng sự khác biệt là kết quả của sự lemmatizing là một từ thực

|  |  |
| --- | --- |
| **Stemming** | **Lemmatizing** |
| **from** nltk.stem **import** PorterStemmer  stemmer = PorterStemmer()  print(stemmer.stem(**'increases'**))  Kết quả: increas | **from** nltk.stem **import** WordNetLemmatizer  lemmatizer = WordNetLemmatizer()  print(lemmatizer.lemmatize(**'increases'**))  Kết quả: increase |

Kết quả có thể kết thúc với một từ đồng nghĩa hoặc một từ khác với cùng ý nghĩa.

Đôi khi, nếu bạn cố gắng lemmatize một từ như từ playing, nó sẽ kết thúc với cùng một từ.

Điều này là do phần mặc định của lời nói là danh từ. Để có động từ, bạn nên chỉ định nó như sau:

**from** nltk.stem **import** WordNetLemmatizer  
  
lemmatizer = WordNetLemmatizer()  
  
print(lemmatizer.lemmatize(**'playing'**, pos=**"v"**))  
  
print(lemmatizer.lemmatize(**'playing'**, pos=**"n"**))  
  
print(lemmatizer.lemmatize(**'playing'**, pos=**"a"**))  
  
print(lemmatizer.lemmatize(**'playing'**, pos=**"r"**))

Kết quả:

play

playing

playing

playing

**Stemming and Lemmatization Difference**

**from** nltk.stem **import** WordNetLemmatizer  
  
**from** nltk.stem **import** PorterStemmer  
  
stemmer = PorterStemmer()  
  
lemmatizer = WordNetLemmatizer()  
  
print(stemmer.stem(**'stones'**))  
  
print(stemmer.stem(**'speaking'**))  
  
print(stemmer.stem(**'bedroom'**))  
  
print(stemmer.stem(**'jokes'**))  
  
print(stemmer.stem(**'lisa'**))  
  
print(stemmer.stem(**'purple'**))  
  
print(**'----------------------'**)  
  
print(lemmatizer.lemmatize(**'stones'**))  
  
print(lemmatizer.lemmatize(**'speaking'**))  
  
print(lemmatizer.lemmatize(**'bedroom'**))  
  
print(lemmatizer.lemmatize(**'jokes'**))  
  
print(lemmatizer.lemmatize(**'lisa'**))  
  
print(lemmatizer.lemmatize(**'purple'**))

Kết quả:

stone

speak

bedroom

joke

lisa

purpl

----------------------

stone

speaking

bedroom

joke

lisa

purple

Stemming hoạt động trên các từ mà không biết bối cảnh của nó và đó là lý do tại sao gốc có độ chính xác thấp hơn và nhanh hơn so với sự lemmatization.

lemmatizing là tốt hơn so với stemming. Từ ngữ lemmatizing trả về một từ thực sự ngay cả khi nó không phải là cùng một từ, nó có thể là một từ đồng nghĩa, nhưng ít nhất đó là một từ thực sự.

Đôi khi bạn không quan tâm đến mức độ chính xác này và tất cả những gì bạn cần là tốc độ, trong trường hợp này, stemming tốt hơn.