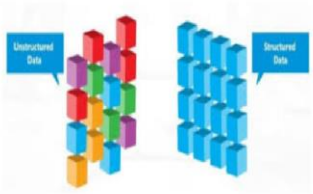
Penambangan Teks di Python: Preprocessing menggunakan NLTK

Dalam skenario hari ini, salah satu cara suksesnya orang-orang diidentifikasi dengan bagaimana mereka mengkomunikasikan dan membagikan informasi ke orang lain. Itulah dimana konsep bahasa berperan di dalam cerita. Namun, ada banyak bahasa di dunia ini. Masing-masing memiliki banyak standar dan huruf, dan kombinasi dari kata-kata ini diatur secara bermakna menghasilkan formasi kalimat. Setiap bahasa memiliki aturannya sendiri saat mengembangkan kalimat-kalimat ini dan sehimpunan aturan ini dikenal sebagai tata bahasa (*grammar*).



Di dunia saat ini, menurut perkiraan industri, hanya sekitar 20 persen dari data yang dihasilkan dalam format terstruktur saat kita berkomunikasi, seperti melalui tweet yang kita tulis, saat kita mengirim pesan menggunakan WhatsApp, Email, Facebook, Instagram atau pesan teks apa pun. Dan sebagian besar data ini ada dalam bentuk teks yang merupakan format yang sangat tidak terstruktur. Untuk menghasilkan wawasan yang bermakna dari data teks maka kita perlu mengikuti metode yang disebut Analisis Teks.

Apa itu Penambangan Teks?

Penambangan Teks adalah proses memperoleh (menurunkan) informasi yang bermakna dari teks bahasa alami.

Penambangan Teks adalah proses memperoleh informasi berkualitas tinggi dari teks.

Tujuan keseluruhannya adalah mengubah teks menjadi data untuk dianalisis melalui aplikasi Pengolahan Bahasa Alami (*Natural Language Processing*, NLP)

Apa itu NLP?

Natural Language Processing **(NLP) adalah bagian dari ilmu komputer dan kecerdasan buatan yang berhubungan dengan bahasa manusia.**

Dengan kata lain, NLP merupakan komponen penambangan teks yang melakukan jenis khusus dari analisis linguistik yang pada dasarnya membantu mesin "membaca" teks. NLP menggunakan metodologi yang berbeda untuk menguraikan ambiguitas dalam bahasa manusia, termasuk di antaranya adalah peringkasan otomatis (*summarization*), penandaan bagian-dari-ungkapan (*part of speech tagging*), disambiguasi, *chunking*, serta pengenalan dan pemahaman bahasa alami. Kita akan melihat semua proses tersebut secara bertahap menggunakan Python.

Pertama, kita perlu menginstal pustaka (*library*) NLTK yang merupakan toolkit bahasa alami untuk membangun program Python yang mampu bekerja dengan data bahasa manusia dan menyediakan antarmuka yang mudah digunakan oleh *programmer*.

Terminologi dalam NLP

Tokenisasi

Tokenisasi adalah langkah pertama dalam rangkaian proses NLP. Ini adalah proses memecah string menjadi token yang pada gilirannya merupakan struktur atau unit kecil. Tokenisasi melibatkan tiga langkah yang memecah kalimat kompleks menjadi kata-kata, memahami pentingnya setiap kata sehubungan dengan kalimat dan akhirnya menghasilkan deskripsi struktural pada suatu kalimat input.

Kode:

```
# Importing necessary library
import pandas as pd
import numpy as np
import nltk
import os
import nltk.corpus

# sample text for performing tokenization
text = "In Brazil they drive on the right-hand side of the road. Brazil has
a large coastline on the eastern side of South America"
```

```
# importing word_tokenize from nltk
from nltk.tokenize import word_tokenize

# Passing the string text into word tokenize for breaking the sentences
token = word_tokenize(text)
token
```

Output

```
['In','Brazil','they','drive', 'on','the', 'right-hand', 'side', 'of',
'the', 'road', '.', 'Brazil', 'has', 'a', 'large', 'coastline', 'on',
'the', 'eastern', 'side', 'of', 'South', 'America']
```

Dari output di atas, kita dapat melihat teks dibagi menjadi token-token. Kata, koma, tanda baca dianggap sebagai token.

Menemukan Frekuensi Kata dalam Teks

Kode 1

```
# finding the frequency distinct in the tokens
# Importing FreqDist library from nltk and passing token into FreqDist
from nltk.probability import FreqDist
fdist = FreqDist(token)
fdist
```

Output

```
FreqDist({'the': 3, 'Brazil': 2, 'on': 2, 'side': 2, 'of': 2, 'In': 1,
'they': 1, 'drive': 1, 'right-hand': 1, 'road': 1, ...})
```

Token ‘the’ ditemukan 3 kali di dalam teks, sedangkan token ‘Brazil’ ditemukan 2 kali di dalam teks, dst.

Kode 2

```
# To find the frequency of top 10 words
fdist1 = fdist.most_common(10)
fdist1
```

Output

```
[('the', 3),
 ('Brazil', 2),
 ('on', 2),
 ('side', 2),
 ('of', 2),
 ('In', 1),
 ('they', 1),
 ('drive', 1),
 ('right-hand', 1),
 ('road', 1)]
```

Stemming

Stemming biasanya merujuk pada proses menormalkan kata-kata menjadi bentuk dasarnya atau bentuk akarnya.

Di sini, kita mempunyai kata-kata “waited”, “waiting” dan “waits”. Kata akar dari ketiganya adalah ‘wait’. Ada dua metode terkenal dalam Stemming bernama Porter Stemming (menghilangkan akhiran morfologis dan infleksional umum dari kata-kata) dan Lancaster Stemming (algoritma stemming yang lebih agresif).

Kode 1

```
# Importing Porterstemmer from nltk library
# Checking for the word ‘giving’
from nltk.stem import PorterStemmer
pst = PorterStemmer()
pst.stem(“waiting”)
```

Output

```
'wait'
```

Kode 2

```
# Checking for the list of words
stm = ["waited", "waiting", "waits"]
for word in stm :
   print(word+ ":" +pst.stem(word))
```

Output

```
waited:wait
waiting:wait
waits:wait
```

Kode 3

```
# Importing LancasterStemmer from nltk
from nltk.stem import LancasterStemmer
lst = LancasterStemmer()
```

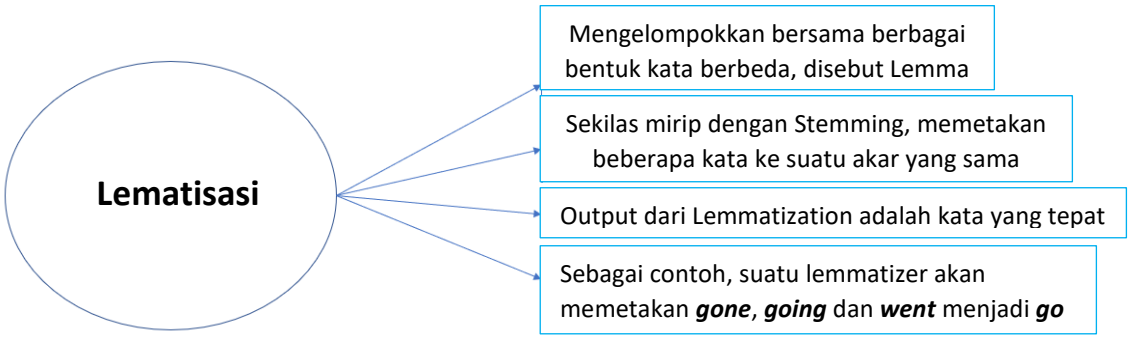
```
stm = ["giving", "given", "given", "gave"]
for word in stm :
 print(word+ ":" +lst.stem(word))
```

Output

```
giving:giv
given:giv
given:giv
gave:gav
```

Lancaster terlihat lebih agresif daripada Porter stemmer

Lematisasi



Dalam istilah lebih sederhana, ini termasuk proses mengkonversi suatu kata ke bentuk dasarnya. Perbedaan antara stemming dan lemmatization adalah lematisasi mempertimbangkan konteks dan mengkonversi kata tersebut ke bentuk dasar yang bermakna, sedangkan stemming hanya menghilangkan beberapa karakter akhiran, sering mengarah ke makna yang salah dan kesalahan ejaan.

Sebagai contoh, lematisasi akan dengan tepat mengidentifikasi bentuk dasar dari ‘caring’ yaitu ‘care’, sedangkan stemming akan memotong bagian ‘ing’ dan mengkonversinya menjadi “car”.

Lematisasi dapat diimplementasikan dalam Python menggunakan Wordnet Lemmatizer, Spacy Lemmatizer, TextBlob, atau Stanford CoreNLP. Sayangnya semua library ini masih belum tersedia untuk bahasa Indonesia.

Kode

```
# Importing Lemmatizer library from nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))
```

Output

```
rocks : rock
corpora : corpus
```

Stop Words

“Stop words” adalah kata-kata yang paling umum dalam suatu bahasaseperti “the”, “a”, “at”, “for”, “above”, “on”, “is”, “all”. Kata-kata ini tidak memberikan suatu makna dan biasanya dihapus dari dalam teks. Kita dapat menghapus stop word ini menggunakan pustaka nltk.

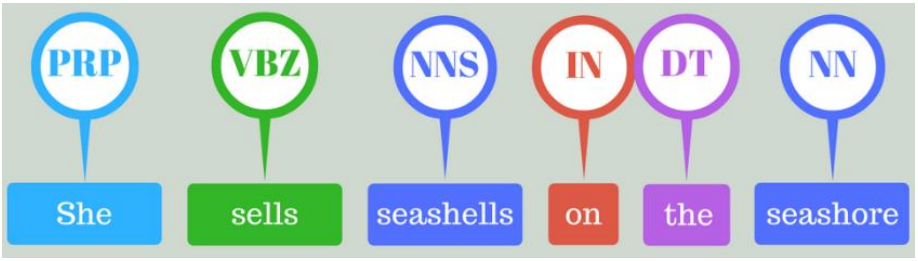
Kode

```
# importing stopwors from nltk library
from nltk import word_tokenize
from nltk.corpus import stopwords
a = set(stopwords.words('english'))
text = "Cristiano Ronaldo was born on February 5, 1985, in Funchal,
Madeira, Portugal."
text1 = word_tokenize(text.lower())
print(text1)
stopwords = [x for x in text1 if x not in a]
print(stopwords)
```

Output

```
Output of text:
['cristiano', 'ronaldo', 'was', 'born', 'on', 'february', '5', ',', '1985',
',', 'in', 'funchal', ',', 'madeira', ',', 'portugal', '.']
Output of stopwords:
['cristiano', 'ronaldo', 'born', 'february', '5', ',', '1985', ',',
'funchal', ',', 'madeira', ',', 'portugal', '.']
```

Part of Speech Tagging (POS)



Part-of-speech tagging (penandaan bagian dari ucapan) digunakan untuk menentukan bagian-bagian dari ucapan terhadap setiap kata dari suatu teks yang diberikan (seperti noun, verb, pronoun, adverb, conjunction, adjectives, interjection) berdasarkan oada definisinya dan konteksnya. Ada banyak tool tersedia untuk penanda ini (POS tagger) dan beberapa yang telah digunakan secara luas adalah NLTK, Spacy, TextBlob, Standford CoreNLP, dst.

Kode

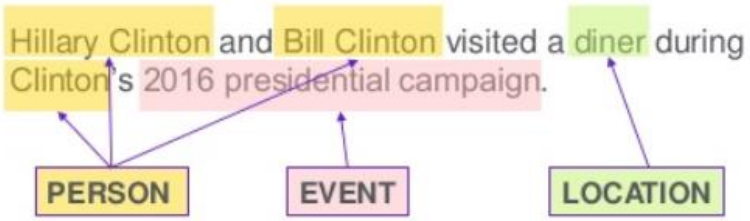
```
text = "vote to choose a particular man or a group (party) to represent
them in parliament"
#Tokenize the text
tex = word_tokenize(text)
for token in tex:
        print(nltk.pos_tag([token]))
```

Output

```
[('vote', 'NN')]
[('to', 'TO')]
[('choose', 'NN')]
[('a', 'DT')]
[('particular', 'JJ')]
[('man', 'NN')]
[('or', 'CC')]
[('a', 'DT')]
[('group', 'NN')]
[('(', '(')]
[('party', 'NN')]
[(')', ')')]
[('to', 'TO')]
[('represent', 'NN')]
[('them', 'PRP')]
[('in', 'IN')]
[('parliament', 'NN')]
```

Pengenalan Entitas Bernama

Ini adalah proses mendeteksi entitas yang mempunyai nama seperti nama orang, nama lokasi, nama perusahaan, kuantitas dan nilai moneter.



Kode

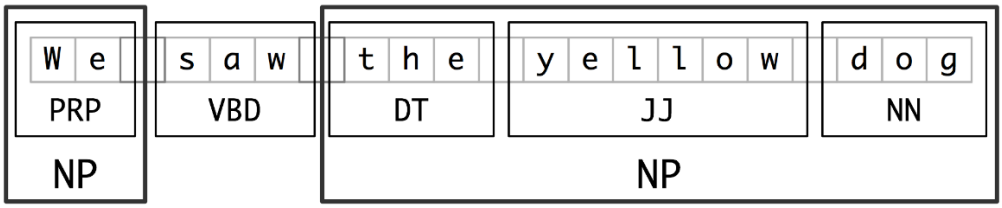
```
text = "Google's CEO Sundar Pichai introduced the new Pixel at Minnesota
Roi Centre Event"
#importing chunk library from nltk
from nltk import ne_chunk
# tokenize and POS Tagging before doing chunk
token = word_tokenize(text)
tags = nltk.pos_tag(token)
chunk = ne_chunk(tags)
chunk
```

Output

```
Tree('S', [Tree('GPE', [('Google', 'NNP')]), ("'s", 'POS'),
Tree('ORGANIZATION', [('CEO', 'NNP'), ('Sundar', 'NNP'), ('Pichai',
'NNP')]), ('introduced', 'VBD'), ('the', 'DT'), ('new', 'JJ'), ('Pixel',
'NNP'), ('at', 'IN'), Tree('ORGANIZATION', [('Minnesota', 'NNP'), ('Roi',
'NNP'), ('Centre', 'NNP')]), ('Event', 'NNP')])
```

Chunking

Chunking berarti mengambil potongan individu dari informasi dan mengelompokkannya menjadi potongan yang lebih besar. Dalam konteks NLP dan penambangan teks, chunking berarti mengelompokkan kata atau token ke dalam chunk-chunk.



Kode

```
text = "We saw the yellow dog"
token = word_tokenize(text)
tags = nltk.pos_tag(token)reg = "NP: {<DT>?<JJ>*<NN>}"
a = nltk.RegexpParser(reg)
result = a.parse(tags)
print(result)
```

Output

```
(S We/PRP saw/VBD (NP the/DT yellow/JJ dog/NN))
```

Tulisan singkat ini telah merangkum tahapan preprocessing terhadap teks dan menjelaskan bagaimana langkah-langkah pemanfaatan NLTK termasuk Tokenization, Stemming, Lemmatization, POS tagging, Named entity recognition dan Chunking.

Terimakasih telah membaca postingan ini. Terus belajar dan seringlah berkunjung ke husni.trunojoyo.ac.id!

Referensi:

1. https://www.expertsystem.com/natural-language-processing-and-text-mining/
2. https://www.nltk.org
3. https://www.edureka.co
4. https://www.geeksforgeeks.org/nlp-chunk-tree-to-text-and-chaining-chunk-transformation/
5. https://www.geeksforgeeks.org/part-speech-tagging-stop-words-using-nltk-python/