

Nama : Muhamad Faqih Azhar

NIM : 0110221092

Kelas : 5TI03 / Senin Malam

Repository

<https://github.com/FaqihAzh/nlp-task/tree/main/week-4>

Pertanyaan

1. Lakukan pra-proses terhadap data teksnya terlebih dahulu
 - a. Hapus simbol, angka, dan token yang tidak umum,
 - b. Ubah semua teks menjadi lowercase

Jawab:

```
from bs4 import BeautifulSoup as BS
import re

with open('artikel1k.txt', 'r') as artikelFile:
    artikelData = artikelFile.read()

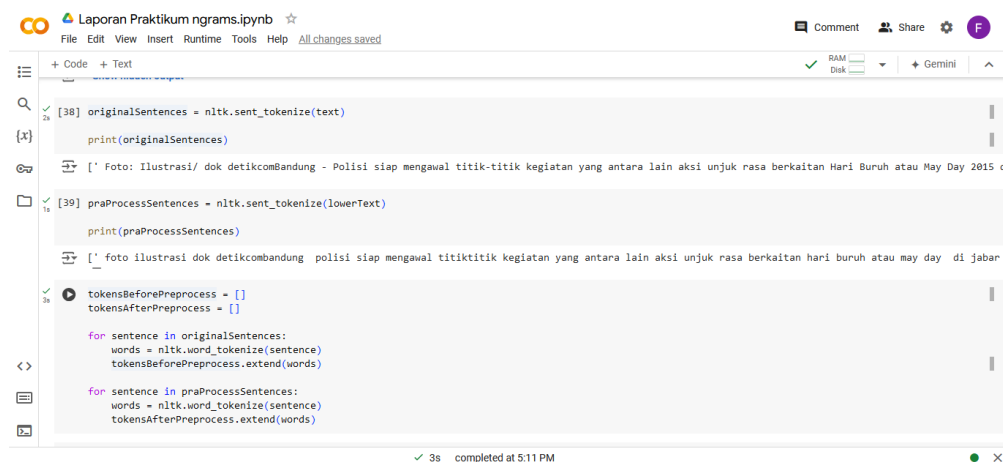
soup = BS(artikelData, 'html.parser')

allText = soup.findAll("div", {"class": "detail_text"})
text = "".join([x.getText() for x in allText])

# Lakukan pra-proses terhadap data teksnya terlebih dahulu
# Hapus simbol, angka, dan token yang tidak umum
cleanedText = re.sub(r'\s+', ' ', text) # Menghilangkan spasi/jarak berlebih pada text
cleanedText = re.sub(r'[^\s-a-zA-Z\s]', '', text) # Menghilangkan semua karakter simbol kecuali huruf dan spasi pada text

# Ubah semua teks menjadi lowercase
lowerText = cleanedText.lower()
```

2. Hitung berapa banyak token unigram, bigram, dan trigramnya sekarang kemudian bandingkan dengan kondisi awal yang sebelum pra-proses!



The screenshot shows a Jupyter Notebook titled "Laporan Praktikum ngrams.ipynb". The code in the notebook performs the following steps:

- Line 38: `originalSentences = nltk.sent_tokenize(text)` and `print(originalSentences)`. The output shows a list of sentences, including "Foto: Ilustrasi/ dok detikcomBandung - Polisi siap mengawal titik-titik kegiatan yang antara lain aksi unjuk rasa berkaitan Hari Buruh atau May Day 2015".
- Line 39: `praProcessSentences = nltk.sent_tokenize(lowerText)` and `print(praProcessSentences)`. The output shows the same sentences converted to lowercase.
- Line 39 (continued): Initialization of `tokensBeforePreprocess = []` and `tokensAfterPreprocess = []`.
- Line 40: A loop `for sentence in originalSentences:` that tokenizes each sentence with `nltk.word_tokenize(sentence)` and appends the tokens to `tokensBeforePreprocess`.
- Line 41: A loop `for sentence in praProcessSentences:` that tokenizes each sentence with `nltk.word_tokenize(sentence)` and appends the tokens to `tokensAfterPreprocess`.

The notebook interface shows the code, the output of the print statements, and the state of the token lists. The status bar at the bottom indicates "3s completed at 5:11 PM".

Laporan Praktikum ngrams.ipynb

```

# Unigram
unigramsBeforePreprocess = tokensBeforePreprocess
print("Unigram: ", unigramsBeforePreprocess, "\n")

# Bigram
bigramsBeforePreprocess = list(ngrams(tokensBeforePreprocess, 2))
print("Bigram: ", bigramsBeforePreprocess, "\n")

# Trigram
trigramsBeforePreprocess = list(ngrams(tokensBeforePreprocess, 3))
print("Trigram: ", trigramsBeforePreprocess, "\n")

# Unigram
unigramsAfterPreprocess = tokensAfterPreprocess
print("Unigram: ", unigramsAfterPreprocess, "\n")

# Bigram
bigramsAfterPreprocess = list(ngrams(tokensAfterPreprocess, 2))
print("Bigram: ", bigramsAfterPreprocess, "\n")

# Trigram
trigramsAfterPreprocess = list(ngrams(tokensAfterPreprocess, 3))
print("Trigram: ", trigramsAfterPreprocess, "\n")

```

completed at 5:11 PM

Laporan Praktikum ngrams.ipynb

```

data = {
    'Unigram': [len(unigramsBeforePreprocess), len(unigramsAfterPreprocess)],
    'Bigram': [len(bigramsBeforePreprocess), len(bigramsAfterPreprocess)],
    'Trigram': [len(trigramsBeforePreprocess), len(trigramsAfterPreprocess)]
}

comparisonDf = pd.DataFrame(data, index=['Before', 'After'])
print("Tabel Perbandingan Unigram, Bigram, dan Trigram:", "\n")
print(comparisonDf)

Tabel Perbandingan Unigram, Bigram, dan Trigram:

```

	Unigram	Bigram	Trigram
Before	282313	282312	282311
After	228282	228281	228280

```

unigram = ngrams(allToken, 1)
unigramFd = FreqDist(unigram)

```

completed at 5:11 PM

3. Hitung ulang peluang dari kedua kalimat yang diberikan (kal1 dan kal2), mana yang lebih mungkin untuk terjadi?

Laporan Praktikum ngrams.ipynb

- Membandingkan dua kalimat yang paling benar berdasarkan probabilitasnya
- Prediksi kata berdasarkan probabilitas kata sebelumnya
- Membuat kalimat yang mendekati bahasa tersebut

Membandingkan dua kalimat yang paling benar berdasarkan probabilitasnya

```

kal1 = "saya minta itu dicopot untuk pidato budaya"
kal2 = "saya minta itu untuk pidato budaya"

# Function untuk menghitung probabilitas kalimat
def search_sentence_prob(sentence, model):
    token_test = nltk.word_tokenize(sentence)
    p = 1
    for kata in token_test:
        pkata = model.freq((kata,))
        print("p(' %s' ) \t= %f"%(kata, pkata))
        p=p*pkata
    print("p(' %s' ) \t= %f"(sentence),p)
    print()
    return p

search_sentence_prob(kal1, unigramFd)

```

```

p('saya') = 0.003539
p('minta') = 0.000245
p('itu') = 0.009528
p('dicopot') = 0.000001

```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

completed at 7:44 AM

Laporan Praktikum ngrams.ipynb

File Edit View Insert Runtime Tools Help Saving failed since 7:27 AM

+ Code + Text

return p

[43] search_sentence_prob(kal1, unigramFd)

```
p('saya') = 0.003539
p('minta') = 0.000245
p('itu') = 0.009528
p('dicopot') = 0.000061
p('untuk') = 0.007499
p('pidato') = 0.000061
p('budaya') = 0.000123
p(saya minta itu dicopot untuk pidato budaya) = 2.8620342759062594e-23
2.8620342759062594e-23
```

search_sentence_prob(kal2, unigramFd)

```
p('saya') = 0.003539
p('minta') = 0.000245
p('itu') = 0.009528
p('untuk') = 0.007499
p('pidato') = 0.000061
p('budaya') = 0.000123
p(saya minta itu untuk pidato budaya) = 4.666792204088805e-19
4.666792204088805e-19
```

<> ▾ Prediksi kata berdasarkan probabilitas kata sebelumnya

[35] def predict_word_trigram(sentence, trigram_fn):

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

0s completed at 7:44 AM

4. *Generate 5 buah kalimat dari corpus berita yang diberikan! Diperbolehkan memanfaatkan bigram, trigram, atau ngram di atasnya.*

Laporan Praktikum ngrams.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[117] bigramCfd = ConditionalFreqDist(listBigram)

import random

```
def generateBigram(bigramCfd, batas):
    kalimat = []
    kata = "<s>"
    selesai = False

    while not selesai:
        r = random.random()
        acc = 0.0

        # Cek jika kata ada di dalam model bigramCfd
        if kata not in bigramCfd:
            break

        # Pilih kata berikutnya berdasarkan probabilitas
        for key in bigramCfd[kata]:
            acc += bigramCfd[kata].freq(key)
            if acc >= r:
                kata = key
                if kata != "</s>": # Abaikan simbol akhir kalimat
                    kalimat.append(kata)
                break

        if kata == "</s>" or batas <= 0:
            selesai = True
        else:
            batas -= 1
```

0s completed at 9:27 PM

Laporan Praktikum ngrams.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

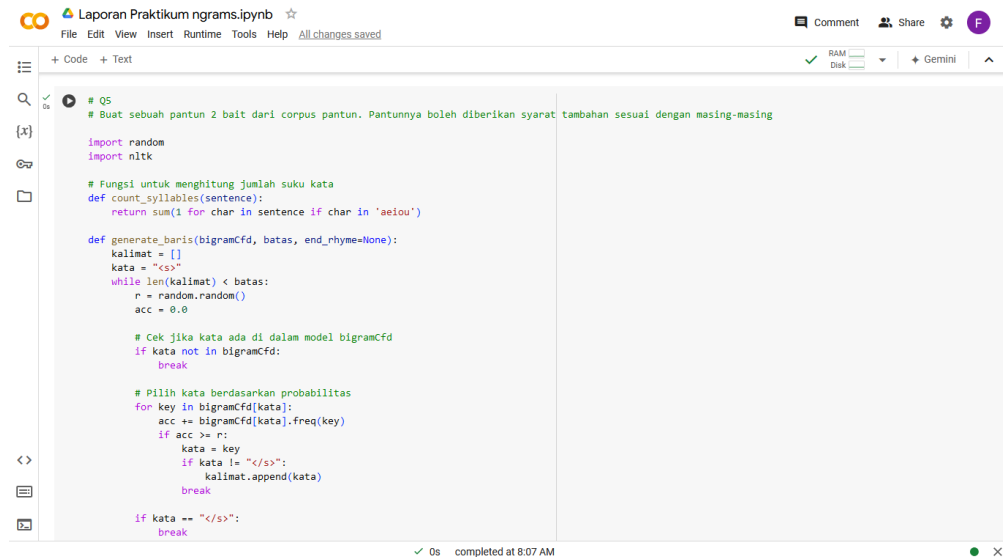
[137] return ' '.join(kalimat)

for i in range(5):

```
print(f"Kalimat Bigram {i+1}: {generate_bigram(bigramCfd, 10)}")
```

Kalimat Bigram 1: foto marty adalah dua wni asal china dan unit mesin kemassatu
 Kalimat Bigram 2: foto dok detikcom menyebutkan bahwa kuba dan sudah melakukan penelitian untuk
 Kalimat Bigram 3: foto diambil namun ia mengimbau untuk meningkatkan kerjasama ini pesawat sebelum
 Kalimat Bigram 4: foto asrar yusuf saat dihubungi detikcom erdvan jakarta utara menuju bandara
 Kalimat Bigram 5: foto asrar yusufruulul foto ray jordan foto enggran eko budiantomojokerto tepat

5. Buat sebuah pantun 2 bait dari corpus pantun dan diperbolehkan memberikan syarat tambahan sesuai dengan masing-masing!



```
# 05
# Buat sebuah pantun 2 bait dari corpus pantun. Pantunnya boleh diberikan syarat tambahan sesuai dengan masing-masing

import random
import nltk

# Fungsi untuk menghitung jumlah suku kata
def count_syllables(sentence):
    return sum(1 for char in sentence if char in 'aeiou')

def generate_baris(bigramCfd, batas, end_rhyme=None):
    kalimat = []
    kata = "<s>"
    while len(kalimat) < batas:
        r = random.random()
        acc = 0.0

        # Cek jika kata ada di dalam model bigramCfd
        if kata not in bigramCfd:
            break

        # Pilih kata berdasarkan probabilitas
        for key in bigramCfd[kata]:
            acc += bigramCfd[kata].freq(key)
            if acc >= r:
                kata = key
                if kata != "</s>":
                    kalimat.append(kata)
                    break

        if kata == "</s>":
            break
```



```
        if kata == "</s>":
            break

        # Lanjutkan jika ada kalimat yang dihasilkan
        if len(kalimat) == 0:
            # Jika kosong, ulang
            return generate_baris(bigramCfd, batas, end_rhyme)
        sentence = " ".join(kalimat)

        # Cek jumlah suku kata
        if not (8 <= count_syllables(sentence) <= 12):
            # Jika tidak sesuai jumlah suku kata, ulang
            return generate_baris(bigramCfd, batas, end_rhyme)

        # Cek rima
        if end_rhyme:
            if sentence.split()[-1][-len(end_rhyme):] != end_rhyme:
                return generate_baris(bigramCfd, batas, end_rhyme) # Ga sesuai ulang

        return sentence

# Fungsi buat pantun 2 bait
def generate_pantun(bigramCfd):
    bait1 = [generate_baris(bigramCfd, 6, end_rhyme='a') for _ in range(2)]
    bait2 = [generate_baris(bigramCfd, 6, end_rhyme='i') for _ in range(2)]

    output = "Bait 1:\n" + "\n".join(bait1) + "\n\n" + "Bait 2:\n" + "\n".join(bait2)
    return output

pantun = generate_pantun(bigramCfd)
print(pantun)
```

Laporan Praktikum ngrams.ipynb

File Edit View Insert Runtime Tools Help All changes saved

RAM 100% Disk 100%

+ Gemini

[61] return output

▶

pantun = generate_pantun(bigramCfd)

print(pantun)

Bait 1:
sambil menikmati indahnya
jangan suka bermain bersama

Bait 2:
ada bebek berenang bermain judi
anak ayam turun membasahi bumi

<>

0s completed at 8:07 AM

● X