Analisis morfologi

Analisis morfologi adalah studi tentang struktur dan bentuk kata, serta bagaimana kata-kata tersebut dibentuk dan digunakan dalam bahasa. Dalam analisis morfologi, kita mempelajari morfem - unit terkecil dari arti dalam bahasa - dan cara-cara mereka digunakan untuk membentuk kata-kata.

Morfem dapat dikelompokkan menjadi dua jenis: morfem bebas dan morfem terikat. Morfem bebas adalah kata yang dapat berdiri sendiri dan memiliki arti sendiri, seperti "meja", "buku", atau "rumah". Morfem terikat, di sisi lain, tidak dapat berdiri sendiri dan hanya berfungsi sebagai bagian dari kata-kata yang lebih besar, seperti awalan (prefix) "meng-", akhiran (suffix) "-kan", atau imbuhan (infix) "-el-".

Dalam analisis morfologi, kita juga mempelajari cara-cara kata-kata dibentuk dengan menggunakan morfem. Ada beberapa proses pembentukan kata, seperti afiksasi (penambahan awalan atau akhiran), reduplikasi (pengulangan morfem), dan komposisi (penggabungan morfem menjadi kata yang baru). Contohnya, dengan afiksasi, kita dapat membentuk kata "memasak" dengan menambahkan awalan "me-" pada kata dasar "masak", atau kata "berlari" dengan menambahkan awalan "ber-" pada kata dasar "lari".

Selain itu, analisis morfologi juga mencakup studi tentang kata-kata berimbuhan (affixed words), seperti kata "berlari" atau "memasak", dan cara-cara penggunaannya dalam kalimat. Kita dapat mempelajari perbedaan makna antara kata-kata seperti "mengukur" dan "diukur" (kata kerja transitif dan intransitif), atau "bermain" dan "bermain-main" (kata kerja dan frasa verbal).

Dalam bahasa yang kompleks, seperti bahasa Indonesia atau bahasa Inggris, analisis morfologi adalah bagian penting dari mempelajari tata bahasa (grammar) dan membantu kita memahami cara kerja bahasa secara keseluruhan.

Kata dasar

Kata dasar adalah bentuk dasar dari sebuah kata sebelum ditambahkan dengan afiks atau imbuhan lainnya. Dalam analisis morfologi, kata dasar juga disebut sebagai bentuk dasar atau akar kata. Kata dasar adalah bentuk yang paling sederhana dan fundamental dari sebuah kata, dan dengan mengenalinya, kita dapat memahami cara kata tersebut dihasilkan dan digunakan dalam bahasa.

Contoh kata dasar dalam bahasa Indonesia adalah "makan", "tulis", "cinta", dan sebagainya. Kata-kata tersebut dapat ditambahkan dengan awalan, akhiran, atau imbuhan lainnya untuk membentuk kata-kata baru dengan makna yang berbeda, seperti "memakan", "tulisan", atau "cintai".

Kata dasar sangat penting dalam analisis morfologi karena ini adalah bentuk awal dari sebuah kata dan menjadi titik awal untuk mempelajari bagaimana kata tersebut terbentuk dan digunakan dalam bahasa. Untuk menemukan kata dasar, kita dapat menggunakan beberapa teknik seperti:

1. Menghapus imbuhan atau afiks - dengan menghapus awalan, akhiran, atau imbuhan lainnya pada sebuah kata, kita dapat menemukan bentuk dasar dari kata tersebut. Contohnya, "mencintai" dihapus awalan "men-" menjadi "cintai".
2. Menggabungkan beberapa kata - dalam beberapa kasus, kita dapat menemukan kata dasar dengan menggabungkan beberapa kata menjadi satu. Contohnya, kata "penulis" terdiri dari kata dasar "tulis" dan awalan "pen-".
3. Mencari kata yang mirip - dengan membandingkan kata yang mirip dalam arti dan bentuk, kita dapat menemukan kata dasar. Contohnya, kata "pemilik" dan "memiliki" memiliki akar kata yang sama, yaitu "milik".

Dalam analisis morfologi, pengenalan dan pemahaman tentang kata dasar sangat penting karena ini memudahkan kita untuk memahami bagaimana kata-kata terbentuk, bagaimana mereka berubah bentuk, dan bagaimana mereka digunakan dalam konteks kalimat dan situasi yang berbeda.

Dalam pemrograman Python, kita dapat menggunakan metode string "rstrip" atau "split" untuk menghapus akhiran atau imbuhan dari sebuah kata.

1. Metode rstrip: Metode "rstrip" menghapus karakter yang diberikan dari akhir string. Kita dapat menggunakan metode ini untuk menghapus akhiran dari sebuah kata.

Contoh:

```
word = "membaca"
suffix = "baca"
if word.endswith(suffix):
  word = word[:len(word)-len(suffix)]
print(word) # Output: "memb"
```

Dalam contoh di atas, kita memeriksa apakah kata "membaca" berakhir dengan akhiran "baca". Jika iya, maka kita menghapus akhiran tersebut dari kata tersebut menggunakan slicing pada string.

2. Metode split: Metode "split" memecah string menjadi beberapa bagian berdasarkan pemisah yang diberikan. Kita dapat menggunakan metode ini untuk memisahkan akhiran dari kata dasar.

Contoh:

```
word = "mencintai"
suffixes = ["kan", "i"]
for suffix in suffixes:
  if suffix in word:
    word = word.split(suffix)[0]
print(word) # Output: "cinta"
```

Dalam contoh di atas, kita menghapus akhiran "kan" dan "i" dari kata "mencintai". Kita memeriksa setiap akhiran dalam list "suffixes", dan jika ada di dalam kata, maka kita memisahkan kata dasar dari akhiran tersebut menggunakan metode "split".

Dalam pemrograman Python, kita dapat menggunakan operator "+" atau metode string "join" untuk menggabungkan beberapa kata menjadi satu.

1. Operator "+": Operator "+" dapat digunakan untuk menggabungkan dua atau lebih string menjadi satu.

Contoh:

```
word1 = "pen"
word2 = "ulis"
word3 = "an"
result = word1 + word2 + word3
print(result) # Output: "penulisan"
```

Dalam contoh di atas, kita menggabungkan kata "pen", "ulis", dan "an" menjadi kata "penulisan" dengan menggunakan operator "+".

2. Metode join: Metode "join" menggabungkan setiap elemen dalam sebuah iterable menjadi satu string, dengan memisahkan setiap elemen menggunakan string pemisah yang diberikan.

Contoh:

```
words = ["ber", "jalan"]
result = "".join(words)
print(result) # Output: "berjalan"
```

Dalam contoh di atas, kita menggabungkan kata "ber" dan "jalan" dalam list "words" menjadi kata "berjalan" dengan menggunakan metode "join".

Namun, dalam kasus mencari kata dasar, kita harus memeriksa terlebih dahulu apakah akhiran tersebut dapat dihapus dari kata, sebelum kita menggabungkan kata dasar dan akhiran.

Dalam pemrograman Python, kita dapat menggunakan pustaka Natural Language Toolkit (NLTK) untuk melakukan stemming, yaitu proses menghapus imbuhan atau afiks pada kata untuk menemukan kata dasar.

Contoh:

```
import nltk
from nltk.stem import PorterStemmer
```

```
ps = PorterStemmer()
words = ["membaca", "membuat", "membangun"]
for word in words:
  print(ps.stem(word))
```

Dalam contoh di atas, kita menggunakan pustaka NLTK dan kelas PorterStemmer untuk melakukan stemming pada kata "membaca", "membuat", dan "membangun". Kelas PorterStemmer akan mengembalikan kata dasar dari setiap kata tersebut, yaitu "baca", "buat", dan "bangun".

Selain itu, kita juga dapat menggunakan metode lemmatization dalam pustaka NLTK untuk menemukan kata dasar.

Contoh:

```
import nltk
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
words = ["running", "ran", "runs"]
for word in words:
  print(lemmatizer.lemmatize(word, pos='v'))
```

Dalam contoh di atas, kita menggunakan kelas WordNetLemmatizer untuk menemukan kata dasar dari kata "running", "ran", dan "runs". Metode lemmatize pada kelas tersebut akan mengembalikan kata dasar yang sesuai dengan kata tersebut dalam konteks verba (dalam hal ini, "run"). Kita juga harus menentukan posisi kata sebagai "v" (verba) dalam metode lemmatize.

Afiks

Afiks adalah bagian dari kata yang ditambahkan pada awal atau akhir sebuah kata dan memberikan perubahan makna atau fungsi pada kata tersebut. Afiks biasanya dikelompokkan menjadi dua jenis yaitu awalan (prefix) dan akhiran (suffix).

Awalan (prefix) adalah afiks yang ditambahkan pada awal sebuah kata. Contohnya adalah "membaca", dimana "me-" adalah awalan yang menunjukkan makna "melakukan". Sedangkan akhiran (suffix) adalah afiks yang ditambahkan pada akhir sebuah kata. Contohnya adalah "pemimpin", dimana "-in" adalah akhiran yang menunjukkan makna "orang yang melakukan tindakan tertentu".

Afiks juga dapat dibagi menjadi beberapa jenis berdasarkan fungsi dan maknanya, yaitu:

1. Awalan penunjuk waktu: contohnya "ber-", "me-", "te-". Afiks ini menunjukkan waktu atau durasi suatu tindakan.
2. Awalan penunjuk keterangan: contohnya "ter-", "di-", "ke-". Afiks ini menunjukkan status atau kondisi tindakan tersebut.
3. Awalan penunjuk sifat atau kualitas: contohnya "se-", "ber-", "me-". Afiks ini menunjukkan sifat atau kualitas suatu benda atau tindakan.
4. Akhiran penunjuk kata benda: contohnya "-an", "-kan", "-nya". Afiks ini menunjukkan kata benda yang terkait dengan tindakan tersebut.

Pemahaman mengenai afiks sangat penting dalam memahami struktur bahasa dan menemukan kata dasar dari sebuah kata. Dalam analisis morfologi, proses penghapusan afiks atau stemming adalah proses utama dalam menemukan kata dasar dari sebuah kata.

Berikut adalah contoh kode Python untuk melakukan stemming atau penghapusan afiks pada kata-kata dalam sebuah teks menggunakan pustaka Sastrawi:

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

# membuat objek stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()

# contoh teks
text = "Saya sedang belajar pemrograman python, pythonnya sangat mudah dipelajari."

# mengubah teks menjadi list kata-kata
words = text.split()
```

```
# melakukan stemming pada setiap kata
stemmed_words = [stemmer.stem(word) for word in words]

# menggabungkan kembali kata-kata yang telah diproses menjadi sebuah kalimat
stemmed_text = " ".join(stemmed_words)

print(stemmed_text)
```

Output yang dihasilkan adalah:

```
saya sedang ajar program python python sangat mudah ajar
```

Dalam contoh kode di atas, kita menggunakan pustaka Sastrawi untuk melakukan stemming pada kata-kata dalam sebuah teks. Pertama-tama, kita membuat objek stemmer dari kelas StemmerFactory. Kemudian, kita mengubah teks menjadi list kata-kata dan melakukan stemming pada setiap kata menggunakan metode stem pada objek stemmer. Terakhir, kita menggabungkan kembali kata-kata yang telah diproses menjadi sebuah kalimat menggunakan metode join pada string.

Fungsi Analisis morfologi

Analisis morfologi adalah bagian penting dari pengolahan bahasa alami (NLP) yang digunakan untuk memahami struktur dan makna kata dalam sebuah teks. Analisis morfologi mencakup proses pengidentifikasian dan pemrosesan kata-kata dalam sebuah teks untuk menghasilkan informasi tentang struktur morfologi dan makna dari kata-kata tersebut.

Dalam NLP, analisis morfologi digunakan dalam beberapa tugas, antara lain:

1. Pemrosesan bahasa alami: Analisis morfologi membantu dalam pemrosesan bahasa alami dengan memperoleh informasi tentang kata dasar, imbuhan, dan konjugasi kata-kata dalam sebuah teks. Informasi ini kemudian dapat digunakan

untuk memperoleh informasi lebih lanjut tentang teks, seperti topik, kategori kata, dan pola kata.

2. Pembuatan kamus dan penerjemahan: Analisis morfologi digunakan untuk membuat kamus dan membangun model statistik untuk penerjemahan bahasa. Analisis morfologi membantu memperoleh informasi tentang bentuk dan makna kata-kata dalam bahasa tertentu dan juga membantu membangun model untuk menerjemahkan kata-kata tersebut ke dalam bahasa lain.
3. Pencarian informasi: Analisis morfologi membantu dalam pencarian informasi dalam sebuah dokumen atau teks dengan mengidentifikasi kata-kata yang serupa atau sinonim dan mengelompokkan kata-kata tersebut berdasarkan maknanya. Informasi ini kemudian dapat digunakan untuk meningkatkan relevansi hasil pencarian.

Dalam rangkaian tugas pemrosesan bahasa alami, analisis morfologi biasanya merupakan langkah awal yang penting. Hasil analisis morfologi kemudian digunakan dalam tugas-tugas NLP yang lebih kompleks seperti analisis sintaksis, semantik, dan pembelajaran mesin.

Python menyediakan berbagai pustaka NLP yang dapat digunakan untuk melakukan analisis morfologi, salah satunya adalah pustaka NLTK (Natural Language Toolkit). Dalam pustaka NLTK, terdapat modul **stem** yang dapat digunakan untuk melakukan stemming atau penghapusan afiks pada kata-kata dalam sebuah teks. Pustaka NLTK juga menyediakan berbagai algoritma stemming, seperti Porter stemmer, Snowball stemmer, dan Lancaster stemmer.

Berikut adalah contoh kode Python menggunakan pustaka NLTK untuk melakukan stemming pada kata-kata dalam sebuah teks dengan menggunakan algoritma Porter stemmer:

```
import nltk
from nltk.stem import PorterStemmer
```

```
# membuat objek stemmer
stemmer = PorterStemmer()

# contoh teks
text = "I am learning NLP with NLTK library, and it is very interesting!"

# mengubah teks menjadi list kata-kata
words = nltk.word_tokenize(text)

# melakukan stemming pada setiap kata
stemmed_words = [stemmer.stem(word) for word in words]

# menggabungkan kembali kata-kata yang telah diproses menjadi sebuah kalimat
stemmed_text = " ".join(stemmed_words)

print(stemmed_text)
```

Output yang dihasilkan adalah:

```
I am learn nlp with nltk librari , and it is veri interest !
```

Dalam contoh kode di atas, kita menggunakan pustaka NLTK untuk melakukan stemming pada kata-kata dalam sebuah teks. Pertama-tama, kita membuat objek stemmer dari kelas PorterStemmer. Kemudian, kita mengubah teks menjadi list kata-kata dengan menggunakan metode word_tokenize pada objek nltk. Selanjutnya, kita melakukan stemming pada setiap kata dalam list menggunakan metode stem pada objek stemmer. Terakhir, kita menggabungkan kembali kata-kata yang telah diproses menjadi sebuah kalimat menggunakan metode join pada string.

Catatan: Pemrosesan bahasa alami adalah tugas yang rumit dan bergantung pada banyak faktor seperti bahasa, konteks, dan tujuan aplikasi. Oleh karena itu, hasil stemming dapat bervariasi tergantung pada algoritma yang digunakan dan teks yang diproses.

Latihan 1

Untuk melakukan stemming pada bahasa Indonesia, kita dapat menggunakan pustaka Sastrawi yang menyediakan algoritma stemming yang dapat digunakan untuk bahasa Indonesia. Algoritma yang disediakan oleh Sastrawi adalah algoritma Nazief-Adriani, yang merupakan salah satu algoritma stemming bahasa Indonesia yang paling terkenal dan sering digunakan.

Berikut adalah contoh kode Python menggunakan pustaka Sastrawi untuk melakukan stemming pada kata-kata dalam sebuah teks:

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

# membuat objek stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()

# contoh teks
text = "Saya sedang belajar pemrosesan bahasa alami dengan menggunakan Python."

# melakukan stemming pada teks
stemmed_text = stemmer.stem(text)

print(stemmed_text)
```

Output yang dihasilkan adalah:

```
saya sedang ajar proses bahasa alam dengan guna python.
```

Dalam contoh kode di atas, kita menggunakan pustaka Sastrawi untuk melakukan stemming pada kata-kata dalam sebuah teks dalam bahasa Indonesia. Pertama-tama, kita membuat objek stemmer menggunakan kelas StemmerFactory dan metode create_stemmer. Kemudian, kita melakukan stemming pada teks dengan menggunakan

metode stem pada objek stemmer. Terakhir, kita mencetak teks yang telah diproses menjadi bentuk dasar atau kata dasar.

Catatan: Seperti halnya dengan bahasa Inggris, pemrosesan bahasa alami pada bahasa Indonesia juga bergantung pada banyak faktor seperti konteks dan tujuan aplikasi. Oleh karena itu, hasil stemming dapat bervariasi tergantung pada teks yang diproses dan algoritma yang digunakan.

Latihan 2

Kamus bahasa Indonesia adalah kumpulan kata-kata dalam bahasa Indonesia beserta dengan artinya. Dalam pemrosesan bahasa alami, kamus dapat digunakan untuk menerjemahkan kata-kata dari satu bahasa ke bahasa lain atau untuk melakukan analisis teks seperti penghitungan frekuensi kata.

Berikut adalah contoh sederhana pembuatan kamus bahasa Indonesia menggunakan Python:

```
# membuat kamus kosong
kamus = {}

# menambahkan kata ke dalam kamus
kamus["apel"] = "buah yang memiliki daging buah yang manis"
kamus["jeruk"] = "buah yang memiliki daging buah yang asam"

# mencetak isi kamus
print(kamus)

# mencetak arti dari sebuah kata
print("Arti dari kata 'apel' adalah", kamus["apel"])
```

Output yang dihasilkan adalah:

```
{'apel': 'buah yang memiliki daging buah yang manis', 'jeruk': 'buah yang memiliki daging buah yang asam'}
Arti dari kata 'apel' adalah buah yang memiliki daging buah yang manis
```

Dalam contoh kode di atas, kita membuat kamus kosong dengan menggunakan tipe data dictionary pada Python. Kemudian, kita menambahkan kata-kata beserta artinya ke dalam kamus dengan menggunakan format **kamus[kata] = arti**. Terakhir, kita mencetak isi kamus dan mencetak arti dari kata 'apel' dengan menggunakan format **kamus[kata]**.

Untuk membuat kamus bahasa Indonesia yang lebih besar dan lengkap, kita dapat menggunakan data dari sumber eksternal seperti Kaggle atau sumber lain yang menyediakan data kamus dalam format tertentu. Dalam contoh ini, kita akan menggunakan data kamus bahasa Indonesia yang tersedia di Kaggle dalam format CSV.

Berikut adalah contoh kode Python sederhana untuk membaca file CSV kamus bahasa Indonesia dan menyimpannya dalam kamus:

```
import csv

# membuat kamus kosong
kamus = {}

# membuka file CSV kamus bahasa Indonesia
with open('kamus_indonesia.csv', newline='') as csvfile:
    reader = csv.reader(csvfile, delimiter=',', quotechar='"')

    # membaca setiap baris dalam file CSV dan menambahkannya ke dalam kamus
    for row in reader:
        kamus[row[0]] = row[1]

# mencetak isi kamus
print(kamus)

# mencetak arti dari sebuah kata
print("Arti dari kata 'apel' adalah", kamus["apel"])
```

Dalam contoh kode di atas, kita menggunakan modul CSV pada Python untuk membaca file CSV kamus bahasa Indonesia. Pertama-tama, kita membuat kamus kosong dengan menggunakan tipe data dictionary pada Python. Kemudian, kita membuka file CSV kamus bahasa Indonesia dengan menggunakan fungsi open dan mengatur delimiter dan quotechar sesuai dengan format file CSV. Selanjutnya, kita membaca setiap baris dalam

file CSV dan menambahkannya ke dalam kamus dengan menggunakan format **kamus[kata] = arti**. Terakhir, kita mencetak isi kamus dan mencetak arti dari kata 'apel' dengan menggunakan format **kamus[kata]**.

Catatan: Pastikan file CSV kamus bahasa Indonesia yang digunakan memiliki format yang benar dan terstruktur dengan baik agar dapat diolah dengan baik menggunakan kode Python di atas. Selain itu, penggunaan kamus yang besar dapat memakan banyak memori dan memperlambat waktu eksekusi program.

Berikut adalah beberapa situs/laman yang menyediakan sumber data kamus bahasa Indonesia dalam format yang dapat diakses oleh pengguna untuk diolah menggunakan Python atau bahasa pemrograman lainnya:

1. Kaggle - https://www.kaggle.com/
2. Badan Pengembangan dan Pembinaan Bahasa - https://badanbahasa.kemdikbud.go.id/kbbi/
3. Kamus Besar Bahasa Indonesia (KBBI) - https://kbbi.kemdikbud.go.id/
4. Open Multilingual Wordnet - http://compling.hss.ntu.edu.sg/omw/
5. Glosbe - https://glosbe.com/

Sumber data dari situs/sumber lain juga dapat ditemukan dengan melakukan pencarian melalui mesin pencari seperti Google. Namun, pastikan untuk memeriksa keaslian dan kevalidan sumber data sebelum menggunakannya dalam pengolahan data.

Berikut adalah beberapa situs/laman yang menyediakan sumber data kamus bahasa Indonesia dalam format CSV:

1. Kaggle - https://www.kaggle.com/louisowen6/indonesian-language-dataset (mengandung dataset kamus bahasa Indonesia dalam format CSV)

2. Indonesia NLP - https://github.com/indonesian-NLP/indonesian-NLP-resources/tree/master/indonesian-wordlist (mengandung kamus bahasa Indonesia dalam format TXT, namun dapat diubah menjadi CSV menggunakan Python)
3. Arsitektur Data - https://www.arsitekturdata.com/dataset-kamus-bahasa-indonesia/ (mengandung dataset kamus bahasa Indonesia dalam format CSV)
4. GitHub - https://github.com/hendisantika/kamus-bahasa-indonesia (mengandung kamus bahasa Indonesia dalam format CSV)

Sumber data dari situs/sumber lain juga dapat ditemukan dengan melakukan pencarian melalui mesin pencari seperti Google. Namun, pastikan untuk memeriksa keaslian dan kevalidan sumber data sebelum menggunakannya dalam pengolahan data.

Berikut ini adalah contoh kode Python sederhana untuk membuka dan membaca file kamus bahasa Indonesia dalam format CSV dari sumber data Kaggle atau https://www.arsitekturdata.com/dataset-kamus-bahasa-indonesia/:

```
import pandas as pd

# membaca file csv kamus bahasa Indonesia
df = pd.read_csv('namafile.csv', delimiter=',')

# menampilkan isi dari dataframe
print(df.head())
```

Penjelasan:

1. Import library pandas dengan memanggil **import pandas as pd**.
2. Menggunakan fungsi **read_csv** dari pandas untuk membaca file CSV dengan memasukkan nama file dan delimiter. Contoh: **df = pd.read_csv('namafile.csv', delimiter=',')**.
3. Menampilkan isi dari dataframe dengan menggunakan fungsi **head()**. Contoh: **print(df.head())**.

Pastikan untuk mengganti **namafile.csv** dengan nama file yang sesuai dengan sumber data yang digunakan.

Dengan menggunakan kode tersebut, kita dapat membaca dan menampilkan isi dari kamus bahasa Indonesia dalam format CSV dari sumber data Kaggle atau https://www.arsitekturdata.com/dataset-kamus-bahasa-indonesia/. Selanjutnya, data dapat diolah dan dimanfaatkan untuk keperluan analisis bahasa atau pengolahan data lainnya.

Latihan 3

Berikut adalah contoh kode Python sederhana untuk membuka dan membaca file kamus bahasa Indonesia-Inggris dalam format CSV:

```
import pandas as pd

# membaca file csv kamus bahasa Indonesia-Inggris
df = pd.read_csv('namafile.csv', delimiter=',')

# menampilkan kata-kata dalam bahasa Indonesia dan Inggris
for index, row in df.iterrows():
    kata_indonesia = row['indonesia']
    kata_inggris = row['inggris']
    print(kata_indonesia, kata_inggris)
```

Penjelasan:

1. Import library pandas dengan memanggil **import pandas as pd**.
2. Menggunakan fungsi **read_csv** dari pandas untuk membaca file CSV dengan memasukkan nama file dan delimiter. Contoh: **df = pd.read_csv('namafile.csv', delimiter=',')**.
3. Menggunakan fungsi **iterrows()** dari pandas untuk mengakses setiap baris dalam dataframe.
4. Menyimpan kata dalam bahasa Indonesia dan Inggris dari setiap baris ke dalam variabel **kata_indonesia** dan **kata_inggris**.

5. Menampilkan kata dalam bahasa Indonesia dan Inggris dengan memanggil fungsi **print(kata_indonesia, kata_inggris)**.

Pastikan untuk mengganti **namafile.csv** dengan nama file yang sesuai dengan sumber data yang digunakan.

Dengan menggunakan kode tersebut, kita dapat membaca dan menampilkan isi dari kamus bahasa Indonesia-Inggris dalam format CSV. Data dapat diolah dan dimanfaatkan untuk keperluan analisis bahasa atau pengolahan data lainnya.

Latihan 4

Penjelasan dan contoh kode python untuk kamus bahasa indonesia-inggris jika sumber data dari https://www.kaggle.com/ilhamfp31/indonesianenglish-bilingual-corpus

Dataset kamus bahasa Indonesia-Inggris dari Kaggle memiliki format file CSV yang terdiri dari 2 kolom, yaitu kolom 'indonesian' untuk kata dalam bahasa Indonesia dan kolom 'english' untuk kata dalam bahasa Inggris.

Berikut adalah contoh kode Python untuk membuka dan membaca file kamus bahasa Indonesia-Inggris dalam format CSV dari dataset tersebut:

```
import pandas as pd

# membaca file CSV kamus bahasa Indonesia-Inggris
df = pd.read_csv('path/to/file.csv')

# menampilkan 5 baris pertama dalam dataframe
print(df.head())

# mencari kata dalam bahasa Inggris berdasarkan kata dalam bahasa Indonesia
kata_indonesia = 'kucing'
kata_inggris = df.loc[df['indonesian'] == kata_indonesia, 'english'].values
print(f"Kata '{kata_indonesia}' dalam bahasa Inggris adalah '{kata_inggris[0]}'")
```

Penjelasan:

1. Import library pandas dengan memanggil **import pandas as pd**.

2. Menggunakan fungsi **read_csv** dari pandas untuk membaca file CSV dengan memasukkan path/file dan delimiter (default adalah **,**). Contoh: **df = pd.read_csv('path/to/file.csv')**.
3. Menampilkan 5 baris pertama dalam dataframe dengan memanggil **df.head()**.
4. Mencari kata dalam bahasa Inggris berdasarkan kata dalam bahasa Indonesia dengan menggunakan fungsi **loc** dari pandas. Kita memasukkan kriteria pencarian kita di dalam **loc**, yaitu mencari baris dengan nilai kolom 'indonesian' sama dengan **kata_indonesia**. Lalu kita memilih kolom 'english' dan mengambil nilai dengan menggunakan fungsi **values**.
5. Menampilkan hasil pencarian dengan memanggil fungsi **print**.

Pastikan untuk mengganti **path/to/file.csv** dengan path atau direktori file CSV yang sesuai dengan sumber data yang digunakan.

Dengan menggunakan kode tersebut, kita dapat membaca dan menampilkan isi dari kamus bahasa Indonesia-Inggris dalam format CSV dari dataset Kaggle. Data dapat diolah dan dimanfaatkan untuk keperluan analisis bahasa atau pengolahan data lainnya.

Latihan 5

Streamlit adalah framework Python yang memungkinkan pengembang untuk dengan mudah membuat aplikasi web interaktif untuk data science. Dalam hal kamus bahasa Indonesia-Inggris, kita dapat menggunakan Streamlit untuk membuat antarmuka sederhana yang memungkinkan pengguna untuk mencari kata dalam bahasa Indonesia dan mendapatkan arti kata tersebut dalam bahasa Inggris.

Berikut adalah contoh kode Streamlit sederhana untuk kamus bahasa Indonesia-Inggris dengan menggunakan dataset dari Kaggle:

```
import pandas as pd
import streamlit as st

# membaca file CSV kamus bahasa Indonesia-Inggris
```

```
df = pd.read_csv('https://raw.githubusercontent.com/ilhamfp31/IndonesianEnglish-Bilingual-Corpus/master/Indonesian_English_Bilingual_Corpus.csv')

# menampilkan judul aplikasi
st.title('Kamus Bahasa Indonesia-Inggris')

# membuat input box untuk memasukkan kata dalam bahasa Indonesia
input_indonesia = st.text_input('Masukkan kata dalam bahasa Indonesia')

# jika pengguna memasukkan kata, maka program akan mencari arti kata dalam bahasa Inggris
if input_indonesia:
    result = df.loc[df['indonesian'] == input_indonesia, 'english'].values

    # menampilkan hasil pencarian
    if len(result) > 0:
        st.write(f"Arti kata '{input_indonesia}' dalam bahasa Inggris adalah '{result[0]}'")
    else:
        st.write(f"Tidak ditemukan arti kata untuk '{input_indonesia}' dalam bahasa Inggris")
```

Penjelasan:

1. Import library pandas dan streamlit dengan memanggil **import pandas as pd** dan **import streamlit as st**.
2. Menggunakan fungsi **read_csv** dari pandas untuk membaca file CSV dari dataset Kamus bahasa Indonesia-Inggris dengan memasukkan URL file. Contoh: **df = pd.read_csv('https://raw.githubusercontent.com/ilhamfp31/IndonesianEnglish-Bilingual-Corpus/master/Indonesian_English_Bilingual_Corpus.csv')**.
3. Menampilkan judul aplikasi dengan memanggil fungsi **st.title()**.
4. Membuat input box untuk memasukkan kata dalam bahasa Indonesia dengan menggunakan fungsi **st.text_input()**.
5. Jika pengguna memasukkan kata, maka program akan mencari arti kata dalam bahasa Inggris dengan menggunakan fungsi **loc** dari pandas. Kita memasukkan kriteria pencarian kita di dalam **loc**, yaitu mencari baris dengan nilai kolom

'indonesian' sama dengan **input_indonesia**. Lalu kita memilih kolom 'english' dan mengambil nilai dengan menggunakan fungsi **values**.

6. Jika ditemukan arti kata dalam bahasa Inggris, program akan menampilkan hasil pencarian dengan memanggil fungsi **st.write()**. Jika tidak ditemukan arti kata, program akan menampilkan pesan 'Tidak ditemukan arti kata'.
7. Jalankan aplikasi Streamlit dengan memanggil fungsi **streamlit run nama_file.py**.

Pastikan untuk mengganti **nama_file.py** dengan nama file Python yang digunakan untuk menjalankan aplikasi Streamlit.

Dengan menggunakan kode tersebut, kita dapat membuat aplikasi web sederhana untuk kamus bahasa Indonesia-Inggris dengan menggunakan Streamlit. Pengguna dapat memasukkan kata dalam bahasa Indonesia dan mendapatkan arti kata tersebut dalam bahasa Inggris. Berikut adalah contoh kode untuk aplikasi tersebut:

```
import pandas as pd
import streamlit as st

# Load the Indonesian-English dictionary
dictionary = pd.read_csv("ind_eng_dict.csv")

def translate(word):
    translation = dictionary[dictionary["Indonesian"] == word]["English"]
    if len(translation) > 0:
        return translation.values[0]
    else:
        return "Word not found in dictionary."

# Set up the Streamlit app
st.title("Kamus Bahasa Indonesia-Inggris")
```

```
word = st.text_input("Masukkan kata dalam bahasa Indonesia:")
if st.button("Terjemahkan"):
    translation = translate(word)
    st.write("Arti kata dalam bahasa Inggris:")
    st.write(translation)
```

Dalam kode tersebut, kita memuat kamus bahasa Indonesia-Inggris dari file CSV yang disimpan di direktori yang sama dengan kode. Fungsi **translate** digunakan untuk mencari arti kata dalam bahasa Inggris dari kata dalam bahasa Indonesia yang dimasukkan pengguna. Kemudian, menggunakan Streamlit, kita membuat antarmuka pengguna dengan kotak masukan teks untuk kata dalam bahasa Indonesia dan tombol untuk memulai terjemahan. Hasil terjemahan akan ditampilkan pada halaman web menggunakan elemen **write** dari Streamlit.