1b-Intro.to.NLP.using.TextBlob

November 13, 2024

1 TextBlob

TextBlob is a powerful NLP Python library. It can be used to perform a variety of NLP tasks. Documentation for TextBlob can be found here.

[8]:
```
%%capture
# Install textblob
!pip install -U textblob
```

[9]:
```
from textblob import TextBlob
```

1.1 Corpora

[10]:
```
%%capture
# Download corpora
!python -m textblob.download_corpora
```

[11]:
```
import nltk
# nltk.download('omw-1.4')
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger_eng')
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data…
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
```

[11]:
```
True
```

1.2 TextBlobs

[12]:
```
my_blob = TextBlob("There is more than one way to skin a cat.")
```

[13]:
```
my_blob
```

[13]:
```
TextBlob("There is more than one way to skin a cat.")
```

1.3 Tagging Parts of Speech

A list of the different parts of speech tags can be found here.

code	meaning	example
CC	coordinating conjunction	
CD	cardinal digit	
DT	determiner	
EX	existential there	(like: “there is” … think of it like “there exists”)
FW	foreign word	
IN	preposition/subordinating conjunction	
JJ	adjective	‘big’
JJR	adjective, comparative	‘bigger’
JJS	adjective, superlative	‘biggest’
LS	list marker	1)
MD	modal could,	will
NN	noun, singular	‘desk’
NNS	noun plural	‘desks’
NNP	proper noun, singular	‘Harrison’
NNPS	proper noun, plural	‘Americans’
PDT	predeterminer	‘all the kids’
POS	possessive ending	parent‘s
PRP	personal pronoun	I, he, she
PRP$	possessive pronoun	my, his, hers
RB	adverb	very, silently,
RBR	adverb, comparative	better
RBS	adverb, superlative	best
RP	particle	give up
TO	to go	’to‘ the store.
UH	interjection	errrrrrrrm
VB	verb, base form	take
VBD	verb, past tense	took
VBG	verb, gerund/present participle	taking
VBN	verb, past participle	taken
VBP	verb, sing. present, non-3d	take
VBZ	verb, 3rd person sing. present	takes
WDT	wh-determiner	which
WP	wh-pronoun	who, what
WP$	possessive wh-pronoun	whose
WRB	wh-adverb	where, when

[14]:
```
# Use the .tags attribute to see parts of speech
my_blob.tags
```

[14]:
```
[('There', 'EX'),
 ('is', 'VBZ'),
 ('more', 'JJR'),
 ('than', 'IN'),
 ('one', 'CD'),
 ('way', 'NN'),
 ('to', 'TO'),
 ('skin', 'VB'),
 ('a', 'DT'),
 ('cat', 'NN')]
```

1.4 Sentiment Analysis

Sentiment analysis can be used to understand the feeling or emotion tied to the text. The sentiment attribute in TextBlob will return two values: 1. The **polarity score** (a float between -1.0 and 1.0). -1 is negative, 1 is positive. 2. The **subjectivity** (a float between 0.0 and 1.0). 0 is very objective, while 1 is very subjective.

[15]:
```
neg_blob = TextBlob("I am so tired. Today was a long, hard day.")
neg_blob.sentiment
```

[15]:
```
Sentiment(polarity=-0.24722222222222223, subjectivity=0.5472222222222222)
```

[16]:
```
pos_blob = TextBlob("Today was a great day. I am so happy.")
pos_blob.sentiment
```

[16]:
```
Sentiment(polarity=0.8, subjectivity=0.875)
```

[17]:
```
obj_blob = TextBlob("The cat is gray.")
obj_blob.sentiment
```

[17]:
```
Sentiment(polarity=0.0, subjectivity=0.0)
```

[18]:
```
subj_blob = TextBlob("The cat is so cute and sweet.")
print(subj_blob.sentiment)
print(subj_blob.sentiment.subjectivity) # Only get the subjectivity
```

```
Sentiment(polarity=0.425, subjectivity=0.825)
0.825
```

Sentiment analysis of multiple sentences

[19]:
```
my_poem = TextBlob('''
  Python is a great language to learn.
  You can easily do NLP; it's fab.
  It might take some getting used to.
  But it's definitely more gooder than Matlab.
''')
```

[20]:
```
my_poem
```

[20]:
```
TextBlob("
  Python is a great language to learn.
  You can easily do NLP; it's fab.
  It might take some getting used to.
  But it's definitely more gooder than Matlab.
")
```

[21]:
```
my_poem.sentiment
```

[21]:
```
Sentiment(polarity=0.5777777777777778, subjectivity=0.6944444444444445)
```

[22]:
```
my_poem.sentences
```

[22]:
```
[Sentence("
   Python is a great language to learn."),
 Sentence("You can easily do NLP; it's fab."),
 Sentence("It might take some getting used to."),
 Sentence("But it's definitely more gooder than Matlab.")]
```

[23]:
```
for sentence in my_poem.sentences:
  print(sentence.sentiment)
```

```
Sentiment(polarity=0.8, subjectivity=0.75)
Sentiment(polarity=0.43333333333333335, subjectivity=0.8333333333333334)
Sentiment(polarity=0.0, subjectivity=0.0)
Sentiment(polarity=0.5, subjectivity=0.5)
```

1.4.1 Your Turn

Create three TextBlobs with the following sentiments: 1. Negative, subjective 2. Positive, objective 3. Neutral

[24]:
```
# Solution 1
text_ns = "It's a cruddy day."
neg_sub = TextBlob(text_ns)
neg_sub.sentiment
```

[24]:
```
Sentiment(polarity=-0.9, subjectivity=0.9)
```

[25]:
```
# Solution 1
text_ns = "Hitler was a terrible man."
neg_sub = TextBlob(text_ns)
neg_sub.sentiment
```

[25]:
```
Sentiment(polarity=-1.0, subjectivity=1.0)
```

[26]:
```
# Solution 2
text_po = "Bill is a nice guy. He won the race."
pos_obj = TextBlob(text_po)
pos_obj.sentiment
# no luck
```

[26]:
```
Sentiment(polarity=0.6, subjectivity=1.0)
```

[27]:
```
# Solution 2
text_po = "My best friend had a baby boy."
pos_obj = TextBlob(text_po)
pos_obj.sentiment
# no luck
```

[27]:
```
Sentiment(polarity=1.0, subjectivity=0.3)
```

[28]:
```
# Solution 3
text_n = "One plus one is two."
neut = TextBlob(text_n)
neut.sentiment
```

[28]:
```
Sentiment(polarity=0.0, subjectivity=0.0)
```

1.5 Tokenization

Tokenization is the process of splitting long strings of text into small pieces (tokens).

[29]:
```
my_poem.sentences
```

[29]:
```
[Sentence("
   Python is a great language to learn."),
 Sentence("You can easily do NLP; it's fab."),
 Sentence("It might take some getting used to."),
 Sentence("But it's definitely more gooder than Matlab.")]
```

[30]:
```
my_poem.sentences[0].words
```

[30]:
```
WordList(['Python', 'is', 'a', 'great', 'language', 'to', 'learn'])
```

[31]:
```
my_poem.words
```

[31]:
```
WordList(['Python', 'is', 'a', 'great', 'language', 'to', 'learn', 'You', 'can',
'easily', 'do', 'NLP', 'it', "'s", 'fab', 'It', 'might', 'take', 'some',
'getting', 'used', 'to', 'But', 'it', "'s", 'definitely', 'more', 'gooder',
'than', 'Matlab'])
```

[32]:
```
sorted(my_poem.word_counts.items(), key = lambda x: x[1], reverse=True)
```

[32]:
```
[('it', 3),
 ('to', 2),
 ('s', 2),
 ('python', 1),
 ('is', 1),
 ('a', 1),
 ('great', 1),
 ('language', 1),
 ('learn', 1),
 ('you', 1),
 ('can', 1),
 ('easily', 1),
 ('do', 1),
 ('nlp', 1),
 ('fab', 1),
 ('might', 1),
 ('take', 1),
 ('some', 1),
 ('getting', 1),
 ('used', 1),
 ('but', 1),
 ('definitely', 1),
 ('more', 1),
 ('gooder', 1),
 ('than', 1),
 ('matlab', 1)]
```

1.6 Singular & Plural

[33]:
```
my_sent = TextBlob("The octopi went swimming in the dark ocean waters.")
```

[34]:
```
my_sent.words
```

[34]:
```
WordList(['The', 'octopi', 'went', 'swimming', 'in', 'the', 'dark', 'ocean',
'waters'])
```

[35]:
```
my_sent.words[0]
```

[35]:
```
'The'
```

[36]:
```
# Singularize
my_sent.words[-1].singularize()
```

[36]:
```
'water'
```

[37]:
```
my_sent.words[1].singularize()
```

[37]:
```
'octopus'
```

[38]:
```
# Pluralize
my_sent.words[-2].pluralize()
```

[38]:
```
'oceans'
```

[39]:
```
foo = my_sent.words[-2]
foo == foo.singularize()
```

[39]:
```
True
```

[40]:
```
TextBlob("corpus").words.singularize(), TextBlob("corpus").words.pluralize()
```

[40]:
```
(WordList(['corpu']), WordList(['corpora']))
```

[41]:
```
my_sent.words[2:5]
```

[41]:
```
WordList(['went', 'swimming', 'in'])
```

[42]:
```
import numpy as np

np.array(my_sent.words)
```

[42]:
```
array(['The', 'octopi', 'went', 'swimming', 'in', 'the', 'dark', 'ocean',
       'waters'], dtype='<U8')
```

1.7 Stemming & Lemmatization

Stemming is the process of deleting prefixes and suffixes from a word, leaving on the word "stem". Lemmatization is similar to stemming, but lemmatization is able to capture the underlying meaning of the word.

[43]:
```
my_sent
```

[43]:
```
TextBlob("The octopi went swimming in the dark ocean waters.")
```

[44]:
```
# Find the index of 'swimming'
my_sent.words.index('swimming')
```

[44]:
```
3
```

[45]:
```
# Stemming
print(my_sent.words[3].stem())
print(my_sent.words[1].stem())
```

```
swim
octopi
```

[46]:
```
# Lemmatization
print(my_sent.words[3].lemmatize())
```

```
print(my_sent.words[1].lemmatize())
```

```
swimming
octopus
```

[47]:
```
care = TextBlob("caring")

(
  care.words.stem(),
  care.words.lemmatize()
)
```

[47]:
```
(WordList(['care']), WordList(['caring']))
```

1.8 WordNet

[48]:
```
my_sent
```

[48]:
```
TextBlob("The octopi went swimming in the dark ocean waters.")
```

[49]:
```
{ my_sent.words[-2] : my_sent.words[-2].definitions }
```

[49]:
```
{'ocean': ['a large body of water constituting a principal part of the
hydrosphere',
  'anything apparently limitless in quantity or volume']}
```

[50]:
```
{"swimming", "tennis"} - set(my_sent.words)
```

[50]:
```
{'tennis'}
```

1.9 Spelling (correcting)

[51]:
```
my_bad_spelling = TextBlob('Helllo, today is my birfday.')
my_bad_spelling.correct()
```

[51]:
```
TextBlob("Hello, today is my birthday.")
```

1.10 Counting Words

[52]:
```
my_cheer = TextBlob('Data science is the best, data science is the coolest.')
my_cheer.words.count('data')
```

[52]:
```
2
```

[53]:
```
my_cheer.word_counts
```

[53]:
```
defaultdict(int,
            {'data': 2,
```

```
'science': 2,
'is': 2,
'the': 2,
'best': 1,
'coolest': 1})
```

1.10.1 Your Turn

1. Create a TextBlob called `message` and set it equal to `Good morning, todayy is going to be a fantastic day!`.
2. Correct the spelling in your TextBlob and set it equal to a new variable called `message_sp`.
3. Find the index of the word `fantastic`.
4. Look up the definition of the word `fantastic`.
5. Stem and lemmatize the word `fantastic`.

[54]:
```
# Solution 1
message = TextBlob("Good morning, todayy is going to be a fantastic day!.")
message
```

[54]:
```
TextBlob("Good morning, todayy is going to be a fantastic day!.")
```

[55]:
```
# Solution 2
message_sp = message.correct()
message_sp
```

[55]:
```
TextBlob("Good morning, today is going to be a fantastic day!.")
```

[56]:
```
list(zip(message.words, message_sp.words ))
```

[56]:
```
[('Good', 'Good'),
 ('morning', 'morning'),
 ('todayy', 'today'),
 ('is', 'is'),
 ('going', 'going'),
 ('to', 'to'),
 ('be', 'be'),
 ('a', 'a'),
 ('fantastic', 'fantastic'),
 ('day', 'day')]
```

[57]:
```
[ (i,t) for i, t in enumerate(zip(message.words, message_sp.words )) if t[0] !=␣
 ↪t[1] ]
```

[57]:
```
[(2, ('todayy', 'today'))]
```

[58]:
```
# Solution 3
(
    message.index("fantastic"),
```

```
    message.words.index("fantastic")
)
```

[58]: (38, 8)

[59]:
```
# Solution 4
TextBlob("fantastic").words[0].definitions
message.words[ message.words.index("fantastic") ].definitions
```

[59]:
```
['ludicrously odd',
 'extraordinarily good or great ; used especially as intensifiers',
 'fanciful and unrealistic; foolish',
 'existing in fancy only; - Nathaniel Hawthorne',
 'extravagantly fanciful in design, construction, appearance']
```

[60]:
```
# Solution 5
fan = message.words[ message.words.index("fantastic") ]
(
  fan.stem(),
  fan.lemmatize()
)
```

[60]: ('fantast', 'fantastic')

1.11 TextBlobs as Strings

TextBlobs act as strings, meaning you can use all of the normal string methods and you can index them as you would a string.

[61]:
```
my_cheer
```

[61]: TextBlob("Data science is the best, data science is the coolest.")

[62]:
```
my_cheer[0:6]
```

[62]: TextBlob("Data s")

[63]:
```
my_cheer.upper()
```

[63]: TextBlob("DATA SCIENCE IS THE BEST, DATA SCIENCE IS THE COOLEST.")

[64]:
```
my_cheer.lower()
```

[64]: TextBlob("data science is the best, data science is the coolest.")

1.12 n-grams

Overlapping lists of words.

[65]:
```
my_cheer
```

[65]:
```
TextBlob("Data science is the best, data science is the coolest.")
```

[66]:
```
my_cheer.words
```

[66]:
```
WordList(['Data', 'science', 'is', 'the', 'best', 'data', 'science', 'is',
'the', 'coolest'])
```

[67]:
```
my_cheer.ngrams(n=3)
```

[67]:
```
[WordList(['Data', 'science', 'is']),
 WordList(['science', 'is', 'the']),
 WordList(['is', 'the', 'best']),
 WordList(['the', 'best', 'data']),
 WordList(['best', 'data', 'science']),
 WordList(['data', 'science', 'is']),
 WordList(['science', 'is', 'the']),
 WordList(['is', 'the', 'coolest'])]
```

[68]:
```
[ " ".join(i) for i in my_cheer.ngrams(n=3) ]
```

[68]:
```
['Data science is',
 'science is the',
 'is the best',
 'the best data',
 'best data science',
 'data science is',
 'science is the',
 'is the coolest']
```

[69]:
```
my_cheer.split(",")
```

[69]:
```
WordList(['Data science is the best', ' data science is the coolest.'])
```

[70]:
```
my_cheer.words
```

[70]:
```
WordList(['Data', 'science', 'is', 'the', 'best', 'data', 'science', 'is',
'the', 'coolest'])
```

[71]:
```
[ " ".join(i) for i in TextBlob("italian pop rock").ngrams(n=2) ]
```

[71]:
```
['italian pop', 'pop rock']
```

1.13 Translation

[72]:
```
# %%capture
!pip install googletrans==3.1.0a0 transformers sacremoses
```

```
Collecting googletrans==3.1.0a0
  Downloading googletrans-3.1.0a0.tar.gz (19 kB)
  Preparing metadata (setup.py) … done
Collecting transformers
  Downloading transformers-4.46.2-py3-none-any.whl.metadata (44 kB)
Collecting sacremoses
  Downloading sacremoses-0.1.1-py3-none-any.whl.metadata (8.3 kB)
Collecting httpx==0.13.3 (from googletrans==3.1.0a0)
  Downloading httpx-0.13.3-py3-none-any.whl.metadata (25 kB)
Requirement already satisfied: certifi in /usr/local/lib/python3.12/site-
packages (from httpx==0.13.3->googletrans==3.1.0a0) (2024.8.30)
Collecting hstspreload (from httpx==0.13.3->googletrans==3.1.0a0)
  Downloading hstspreload-2024.11.1-py3-none-any.whl.metadata (2.1 kB)
Requirement already satisfied: sniffio in /usr/local/lib/python3.12/site-
packages (from httpx==0.13.3->googletrans==3.1.0a0) (1.3.1)
Collecting chardet==3.* (from httpx==0.13.3->googletrans==3.1.0a0)
  Downloading chardet-3.0.4-py2.py3-none-any.whl.metadata (3.2 kB)
Collecting idna==2.* (from httpx==0.13.3->googletrans==3.1.0a0)
  Downloading idna-2.10-py2.py3-none-any.whl.metadata (9.1 kB)
Collecting rfc3986<2,>=1.3 (from httpx==0.13.3->googletrans==3.1.0a0)
  Downloading rfc3986-1.5.0-py2.py3-none-any.whl.metadata (6.5 kB)
Collecting httpcore==0.9.* (from httpx==0.13.3->googletrans==3.1.0a0)
  Downloading httpcore-0.9.1-py3-none-any.whl.metadata (4.6 kB)
Collecting h11<0.10,>=0.8 (from
httpcore==0.9.*->httpx==0.13.3->googletrans==3.1.0a0)
  Downloading h11-0.9.0-py2.py3-none-any.whl.metadata (8.1 kB)
Collecting h2==3.* (from httpcore==0.9.*->httpx==0.13.3->googletrans==3.1.0a0)
  Downloading h2-3.2.0-py2.py3-none-any.whl.metadata (32 kB)
Collecting hyperframe<6,>=5.2.0 (from
h2==3.*->httpcore==0.9.*->httpx==0.13.3->googletrans==3.1.0a0)
  Downloading hyperframe-5.2.0-py2.py3-none-any.whl.metadata (7.2 kB)
Collecting hpack<4,>=3.0 (from
h2==3.*->httpcore==0.9.*->httpx==0.13.3->googletrans==3.1.0a0)
  Downloading hpack-3.0.0-py2.py3-none-any.whl.metadata (7.0 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/site-
packages (from transformers) (3.16.1)
Collecting huggingface-hub<1.0,>=0.23.2 (from transformers)
  Downloading huggingface_hub-0.26.2-py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/site-
packages (from transformers) (2.0.2)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.12/site-packages (from transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/site-
```

```
packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.12/site-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.12/site-
packages (from transformers) (2.32.3)
Collecting safetensors>=0.4.1 (from transformers)
  Downloading safetensors-0.4.5-cp312-cp312-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.8 kB)
Collecting tokenizers<0.21,>=0.20 (from transformers)
  Downloading tokenizers-0.20.3-cp312-cp312-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.7 kB)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.12/site-
packages (from transformers) (4.67.0)
Requirement already satisfied: click in /usr/local/lib/python3.12/site-packages
(from sacremoses) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.12/site-packages
(from sacremoses) (1.4.2)
Requirement already satisfied: fsspec>=2023.5.0 in
/usr/local/lib/python3.12/site-packages (from huggingface-
hub<1.0,>=0.23.2->transformers) (2024.10.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.12/site-packages (from huggingface-
hub<1.0,>=0.23.2->transformers) (4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.12/site-packages (from requests->transformers) (3.4.0)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.12/site-packages (from requests->transformers) (2.2.3)
Downloading httpx-0.13.3-py3-none-any.whl (55 kB)
Downloading chardet-3.0.4-py2.py3-none-any.whl (133 kB)
Downloading httpcore-0.9.1-py3-none-any.whl (42 kB)
Downloading idna-2.10-py2.py3-none-any.whl (58 kB)
Downloading h2-3.2.0-py2.py3-none-any.whl (65 kB)
Downloading transformers-4.46.2-py3-none-any.whl (10.0 MB)
                         10.0/10.0 MB
7.7 MB/s eta 0:00:00 0:00:01
Downloading sacremoses-0.1.1-py3-none-any.whl (897 kB)
                         897.5/897.5 kB
3.9 MB/s eta 0:00:00a 0:00:01
Downloading huggingface_hub-0.26.2-py3-none-any.whl (447 kB)
Downloading
safetensors-0.4.5-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(434 kB)
Downloading
tokenizers-0.20.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(3.0 MB)
                         3.0/3.0 MB
11.8 MB/s eta 0:00:00a 0:00:01
Downloading rfc3986-1.5.0-py2.py3-none-any.whl (31 kB)
```

```
Downloading hstspreload-2024.11.1-py3-none-any.whl (1.2 MB)
                           1.2/1.2 MB
12.1 MB/s eta 0:00:00
Downloading h11-0.9.0-py2.py3-none-any.whl (53 kB)
Downloading hpack-3.0.0-py2.py3-none-any.whl (38 kB)
Downloading hyperframe-5.2.0-py2.py3-none-any.whl (12 kB)
Building wheels for collected packages: googletrans
  Building wheel for googletrans (setup.py) … done
  Created wheel for googletrans: filename=googletrans-3.1.0a0-py3-none-
any.whl size=16353
sha256=673b3bf11953a6f3c4b04123016aa64e2ab429467d3855966531f5c9aa2196ec
  Stored in directory: /root/.cache/pip/wheels/96/ac/bd/9df9eab356c0576896e97147
425987f6f45e9e46456c978d18
Successfully built googletrans
Installing collected packages: rfc3986, hyperframe, hpack, h11, chardet,
safetensors, sacremoses, idna, hstspreload, h2, httpcore, huggingface-hub,
httpx, tokenizers, googletrans, transformers
  Attempting uninstall: h11
    Found existing installation: h11 0.14.0
    Uninstalling h11-0.14.0:
      Successfully uninstalled h11-0.14.0
  Attempting uninstall: idna
    Found existing installation: idna 3.10
    Uninstalling idna-3.10:
      Successfully uninstalled idna-3.10
  Attempting uninstall: httpcore
    Found existing installation: httpcore 1.0.6
    Uninstalling httpcore-1.0.6:
      Successfully uninstalled httpcore-1.0.6
  Attempting uninstall: httpx
    Found existing installation: httpx 0.27.2
    Uninstalling httpx-0.27.2:
      Successfully uninstalled httpx-0.27.2
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
jupyterlab 4.3.0 requires httpx>=0.25.0, but you have httpx 0.13.3 which is
incompatible.
Successfully installed chardet-3.0.4 googletrans-3.1.0a0 h11-0.9.0 h2-3.2.0
hpack-3.0.0 hstspreload-2024.11.1 httpcore-0.9.1 httpx-0.13.3 huggingface-
hub-0.26.2 hyperframe-5.2.0 idna-2.10 rfc3986-1.5.0 sacremoses-0.1.1
safetensors-0.4.5 tokenizers-0.20.3 transformers-4.46.2
```

```
WARNING: Running pip as the 'root' user can result in broken permissions
and conflicting behaviour with the system package manager, possibly rendering
your system unusable.It is recommended to use a virtual environment instead:
https://pip.pypa.io/warnings/venv. Use the --root-user-action option if you know
what you are doing and want to suppress this warning.
```

1.13.1 Google translate

[73]:
```
from googletrans import Translator
```

[74]:
```
translator = Translator()
```

[75]:
```
result = translator.translate(
    'Hello, how are you?',
    src='en',
    dest='es',
)
print(result.text)
```

```
¿Hola, cómo estás?
```

[76]:
```
result = translator.translate(
    'Hello, how are you?',
    src='en',
    dest='fr',
)
print(result.text)
```

```
Bonjour comment allez-vous?
```

[77]:
```
result = translator.translate(
    'Hello, how are you?',
    src='en',
    dest='ar',
)
print(result.text)
```

:[78]
```
result = translator.translate(
    'Hello, how are you?',
    src='en',
    dest='de',
)
print(result.text)
```

```
Hallo, wie geht es dir?
```

All in one …

[79]:
```
langs = 'es fr ar de'.split()

for lang in langs:
  result = translator.translate(
      'Hello, how are you?',
      src='en',
      dest=lang,
  )
  print(result.text)
```

```
¿Hola, cómo estás?
Bonjour comment allez-vous?

Hallo, wie geht es dir?
```

1.13.2 Hugging Face Transformers (via pre-trained models)

[80]:
```
from transformers import MarianMTModel, MarianTokenizer
```

```
2024-11-13 05:23:00.337959: I tensorflow/core/util/port.cc:153] oneDNN custom
operations are on. You may see slightly different numerical results due to
floating-point round-off errors from different computation orders. To turn them
off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-11-13 05:23:00.737994: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to register
cuFFT factory: Attempting to register factory for plugin cuFFT when one has
already been registered
WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
E0000 00:00:1731475380.887693     806 cuda_dnn.cc:8310] Unable to register cuDNN
factory: Attempting to register factory for plugin cuDNN when one has already
been registered
E0000 00:00:1731475380.929343     806 cuda_blas.cc:1418] Unable to register
cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has
already been registered
2024-11-13 05:23:01.279020: I tensorflow/core/platform/cpu_feature_guard.cc:210]
This TensorFlow binary is optimized to use available CPU instructions in
performance-critical operations.
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations,
rebuild TensorFlow with the appropriate compiler flags.
```

[ ]:
```
# Load pre-trained MarianMT model
model_name = 'Helsinki-NLP/opus-mt-en-es'
model = MarianMTModel.from_pretrained(model_name)
```

```
tokenizer = MarianTokenizer.from_pretrained(model_name,␣
 ↪clean_up_tokenization_spaces=True )

text = "Hello, how are you?"
translated = model.generate(**tokenizer(text, return_tensors="pt",␣
 ↪padding=True))
result = tokenizer.decode(translated[0], skip_special_tokens=True)

print(result)
```

```
config.json:   0%|          | 0.00/1.47k [00:00<?, ?B/s]
pytorch_model.bin:   0%|          | 0.00/312M [00:00<?, ?B/s]
generation_config.json:   0%|          | 0.00/293 [00:00<?, ?B/s]
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
Cell In[81], line 4
      2 model_name = 'Helsinki-NLP/opus-mt-en-es'
      3 model = MarianMTModel.from_pretrained(model_name)
----> 4 tokenizer = MarianTokenizer.from_pretrained(model_name,␣
 ↪clean_up_tokenization_spaces=True )
      6 text = "Hello, how are you?"
      7 translated = model.generate(**tokenizer(text, return_tensors="pt",␣
 ↪padding=True))

File /usr/local/lib/python3.12/site-packages/transformers/utils/import_utils.py
 ↪1651, in DummyObject.__getattribute__(cls, key)
   1649 if key.startswith("_") and key != "_from_config":
   1650     return super().__getattribute__(key)
-> 1651 requires_backends(cls, cls._backends)

File /usr/local/lib/python3.12/site-packages/transformers/utils/import_utils.py
 ↪1639, in requires_backends(obj, backends)
   1637 failed = [msg.format(name) for available, msg in checks if not␣
 ↪available()]
   1638 if failed:
-> 1639     raise ImportError("".join(failed))

ImportError:
MarianTokenizer requires the SentencePiece library but it was not found in your
 ↪environment. Checkout the instructions on the
installation page of its repo: https://github.com/google/
 ↪sentencepiece#installation and follow the ones
that match your environment. Please note that you may need to restart your␣
 ↪runtime after installation.
```

```
model.safetensors:   0%|          | 0.00/312M [00:00<?, ?B/s]
```

[ ]:
```
# Load pre-trained MarianMT model
model_name = 'Helsinki-NLP/opus-mt-en-fr'
model = MarianMTModel.from_pretrained(model_name)
tokenizer = MarianTokenizer.from_pretrained(model_name,␣
 ↪clean_up_tokenization_spaces=True)

text = "Hello, how are you?"
translated = model.generate(**tokenizer(text, return_tensors="pt",␣
 ↪padding=True))
result = tokenizer.decode(translated[0], skip_special_tokens=True)

print(result)
```

```
config.json:   0%|          | 0.00/1.42k [00:00<?, ?B/s]
pytorch_model.bin:   0%|          | 0.00/301M [00:00<?, ?B/s]
generation_config.json:   0%|          | 0.00/293 [00:00<?, ?B/s]
tokenizer_config.json:   0%|          | 0.00/42.0 [00:00<?, ?B/s]
source.spm:   0%|          | 0.00/778k [00:00<?, ?B/s]
target.spm:   0%|          | 0.00/802k [00:00<?, ?B/s]
vocab.json:   0%|          | 0.00/1.34M [00:00<?, ?B/s]
Bonjour, comment allez-vous?
```

[ ]:
```
# Load pre-trained MarianMT model
model_name = 'Helsinki-NLP/opus-mt-en-ar'
model = MarianMTModel.from_pretrained(model_name)
tokenizer = MarianTokenizer.from_pretrained(model_name,␣
 ↪clean_up_tokenization_spaces=True)

text = "Hello, how are you?"
translated = model.generate(**tokenizer(text, return_tensors="pt",␣
 ↪padding=True))
result = tokenizer.decode(translated[0], skip_special_tokens=True)

print(result)
```

```
config.json:   0%|          | 0.00/1.39k [00:00<?, ?B/s]
pytorch_model.bin:   0%|          | 0.00/308M [00:00<?, ?B/s]
generation_config.json:   0%|          | 0.00/293 [00:00<?, ?B/s]
tokenizer_config.json:   0%|          | 0.00/44.0 [00:00<?, ?B/s]
```

```
source.spm:   0%|          | 0.00/801k [00:00<?, ?B/s]
target.spm:   0%|          | 0.00/917k [00:00<?, ?B/s]
vocab.json:   0%|          | 0.00/2.12M [00:00<?, ?B/s]
```

]]:
```
# Load pre-trained MarianMT model
model_name = 'Helsinki-NLP/opus-mt-en-de'
model = MarianMTModel.from_pretrained(model_name)
tokenizer = MarianTokenizer.from_pretrained(model_name,␣
 ↪clean_up_tokenization_spaces=True)

text = "Hello, how are you?"
translated = model.generate(**tokenizer(text, return_tensors="pt",␣
 ↪padding=True))
result = tokenizer.decode(translated[0], skip_special_tokens=True)

print(result)
```

```
config.json:   0%|          | 0.00/1.33k [00:00<?, ?B/s]
pytorch_model.bin:   0%|          | 0.00/298M [00:00<?, ?B/s]
generation_config.json:   0%|          | 0.00/293 [00:00<?, ?B/s]
tokenizer_config.json:   0%|          | 0.00/42.0 [00:00<?, ?B/s]
source.spm:   0%|          | 0.00/768k [00:00<?, ?B/s]
target.spm:   0%|          | 0.00/797k [00:00<?, ?B/s]
vocab.json:   0%|          | 0.00/1.27M [00:00<?, ?B/s]
Hallo, wie geht's?
```

[ ]: