```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```python
import spacy
from spacy import displacy

nlp = spacy.load('en_core_web_sm')

text = """
Python is an interpreted, high-level, general-purpose programming
language.
Its design philosophy emphasizes code readability with its notable use
of indentation.
Guido van Rossum created Python, and it was first released in 1991.
Google and Microsoft use Python for various applications.
"""

doc = nlp(text)

print("Named Entities, Phrases, and Concepts:")
for ent in doc.ents:
    print(f"{ent.text} ({ent.label_})")
```

```
Named Entities, Phrases, and Concepts:
Guido van Rossum (PERSON)
first (ORDINAL)
1991 (DATE)
Google (ORG)
Microsoft (ORG)
```

```python
displacy.render(doc, style='ent', jupyter=True)
```

```
<IPython.core.display.HTML object>
```

```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```python
import spacy
from spacy import displacy

nlp = spacy.load('en_core_web_sm')

text = """
Python is an interpreted, high-level, general-purpose programming
language.
Its design philosophy emphasizes code readability with its notable use
of indentation.
Guido van Rossum created Python, and it was first released in 1991.
Google and Microsoft use Python for various applications.
"""

doc = nlp(text)

print("Named Entities, Phrases, and Concepts:")
for ent in doc.ents:
    print(f"{ent.text} ({ent.label_})")
```

```
Named Entities, Phrases, and Concepts:
Guido van Rossum (PERSON)
first (ORDINAL)
1991 (DATE)
Google (ORG)
Microsoft (ORG)
```

```python
displacy.render(doc, style='ent', jupyter=True)
```

```
<IPython.core.display.HTML object>
```

```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```python
import spacy
from spacy import displacy

nlp = spacy.load('en_core_web_sm')

text = """
Python is an interpreted, high-level, general-purpose programming
language.
Its design philosophy emphasizes code readability with its notable use
of indentation.
Guido van Rossum created Python, and it was first released in 1991.
Google and Microsoft use Python for various applications.
"""

doc = nlp(text)

print("Named Entities, Phrases, and Concepts:")
for ent in doc.ents:
    print(f"{ent.text} ({ent.label_})")
```

```
Named Entities, Phrases, and Concepts:
Guido van Rossum (PERSON)
first (ORDINAL)
1991 (DATE)
Google (ORG)
Microsoft (ORG)
```

```python
displacy.render(doc, style='ent', jupyter=True)
```

```
<IPython.core.display.HTML object>
```

```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```python
import spacy
from spacy import displacy

nlp = spacy.load('en_core_web_sm')

text = """
Python is an interpreted, high-level, general-purpose programming
language.
Its design philosophy emphasizes code readability with its notable use
of indentation.
Guido van Rossum created Python, and it was first released in 1991.
Google and Microsoft use Python for various applications.
"""

doc = nlp(text)

print("Named Entities, Phrases, and Concepts:")
for ent in doc.ents:
    print(f"{ent.text} ({ent.label_})")
```

```
Named Entities, Phrases, and Concepts:
Guido van Rossum (PERSON)
first (ORDINAL)
1991 (DATE)
Google (ORG)
Microsoft (ORG)
```

```python
displacy.render(doc, style='ent', jupyter=True)
```

```
<IPython.core.display.HTML object>
```

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
```

```
True
```

```
sia = SentimentIntensityAnalyzer()
```

```
text = "I love this product! It's absolutely amazing. The quality is
top-notch."
```

```
sentiment_score = sia.polarity_scores(text)
```

```
print("Sentiment Analysis Results:")
print(f"Positive: {sentiment_score['pos']}")
print(f"Neutral: {sentiment_score['neu']}")
print(f"Negative: {sentiment_score['neg']}")
print(f"Overall Compound Score: {sentiment_score['compound']}")
```

```
Sentiment Analysis Results:
Positive: 0.518
Neutral: 0.482
Negative: 0.0
Overall Compound Score: 0.862
```

```
if sentiment_score['compound'] >= 0.05:
    print("Overall Sentiment: Positive")
elif sentiment_score['compound'] <= -0.05:
    print("Overall Sentiment: Negative")
else:
    print("Overall Sentiment: Neutral")
```

```
Overall Sentiment: Positive
```

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
```

```
True
```

```
sia = SentimentIntensityAnalyzer()
```

```
text = "I love this product! It's absolutely amazing. The quality is
top-notch."
```

```
sentiment_score = sia.polarity_scores(text)
```

```
print("Sentiment Analysis Results:")
print(f"Positive: {sentiment_score['pos']}")
print(f"Neutral: {sentiment_score['neu']}")
print(f"Negative: {sentiment_score['neg']}")
print(f"Overall Compound Score: {sentiment_score['compound']}")
```

```
Sentiment Analysis Results:
Positive: 0.518
Neutral: 0.482
Negative: 0.0
Overall Compound Score: 0.862
```

```
if sentiment_score['compound'] >= 0.05:
    print("Overall Sentiment: Positive")
elif sentiment_score['compound'] <= -0.05:
    print("Overall Sentiment: Negative")
else:
    print("Overall Sentiment: Neutral")
```

```
Overall Sentiment: Positive
```

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
```

```
True
```

```
sia = SentimentIntensityAnalyzer()
```

```
text = "I love this product! It's absolutely amazing. The quality is
top-notch."
```

```
sentiment_score = sia.polarity_scores(text)
```

```
print("Sentiment Analysis Results:")
print(f"Positive: {sentiment_score['pos']}")
print(f"Neutral: {sentiment_score['neu']}")
print(f"Negative: {sentiment_score['neg']}")
print(f"Overall Compound Score: {sentiment_score['compound']}")
```

```
Sentiment Analysis Results:
Positive: 0.518
Neutral: 0.482
Negative: 0.0
Overall Compound Score: 0.862
```

```
if sentiment_score['compound'] >= 0.05:
    print("Overall Sentiment: Positive")
elif sentiment_score['compound'] <= -0.05:
    print("Overall Sentiment: Negative")
else:
    print("Overall Sentiment: Neutral")
```

```
Overall Sentiment: Positive
```

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
```

```
True
```

```
sia = SentimentIntensityAnalyzer()
```

```
text = "I love this product! It's absolutely amazing. The quality is
top-notch."
```

```
sentiment_score = sia.polarity_scores(text)
```

```
print("Sentiment Analysis Results:")
print(f"Positive: {sentiment_score['pos']}")
print(f"Neutral: {sentiment_score['neu']}")
print(f"Negative: {sentiment_score['neg']}")
print(f"Overall Compound Score: {sentiment_score['compound']}")
```

```
Sentiment Analysis Results:
Positive: 0.518
Neutral: 0.482
Negative: 0.0
Overall Compound Score: 0.862
```

```
if sentiment_score['compound'] >= 0.05:
    print("Overall Sentiment: Positive")
elif sentiment_score['compound'] <= -0.05:
    print("Overall Sentiment: Negative")
else:
    print("Overall Sentiment: Neutral")
```

```
Overall Sentiment: Positive
```

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
import nltk
import numpy as np
import networkx as nx
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.tokenize import sent_tokenize

nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
True
```

```
import nltk
nltk.download('punkt_tab')
```

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

```
True
```

```
def preprocess_text(text):
    """Tokenizes and cleans text"""
    sentences = sent_tokenize(text)
    return sentences

def build_similarity_matrix(sentences):
    """Computes sentence similarity matrix using TF-IDF"""
    vectorizer = TfidfVectorizer()
    sentence_vectors = vectorizer.fit_transform(sentences)

    similarity_matrix = np.dot(sentence_vectors,
sentence_vectors.T).toarray()

    return similarity_matrix

def text_rank_summary(text, num_sentences=3):
    """Applies TextRank algorithm for summarization"""
    sentences = preprocess_text(text)
    if len(sentences) <= num_sentences:
        return " ".join(sentences)

    similarity_matrix = build_similarity_matrix(sentences)

    graph = nx.from_numpy_array(similarity_matrix)
    scores = nx.pagerank(graph)
```

```
    ranked_sentences = sorted(((scores[i], s) for i, s in
enumerate(sentences)), reverse=True)

    summary = " ".join([sent for _, sent in
ranked_sentences[:num_sentences]])

    return summary

text = """Natural Language Processing (NLP) is a field of artificial
intelligence that focuses on
the interaction between computers and humans using natural language.
The ultimate goal of NLP is
to enable computers to understand, interpret, and generate human
language in a way that is valuable.
Some common NLP tasks include machine translation, sentiment analysis,
text summarization, and speech recognition."""

summary = text_rank_summary(text, num_sentences=2)
print("Summary:\n", summary)
```

```
Summary:
 The ultimate goal of NLP is
to enable computers to understand, interpret, and generate human
language in a way that is valuable. Natural Language Processing (NLP)
is a field of artificial intelligence that focuses on
the interaction between computers and humans using natural language.
```

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
import nltk
import numpy as np
import networkx as nx
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.tokenize import sent_tokenize

nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
True
```

```
import nltk
nltk.download('punkt_tab')
```

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

```
True
```

```
def preprocess_text(text):
    """Tokenizes and cleans text"""
    sentences = sent_tokenize(text)
    return sentences

def build_similarity_matrix(sentences):
    """Computes sentence similarity matrix using TF-IDF"""
    vectorizer = TfidfVectorizer()
    sentence_vectors = vectorizer.fit_transform(sentences)

    similarity_matrix = np.dot(sentence_vectors,
sentence_vectors.T).toarray()

    return similarity_matrix

def text_rank_summary(text, num_sentences=3):
    """Applies TextRank algorithm for summarization"""
    sentences = preprocess_text(text)
    if len(sentences) <= num_sentences:
        return " ".join(sentences)

    similarity_matrix = build_similarity_matrix(sentences)

    graph = nx.from_numpy_array(similarity_matrix)
    scores = nx.pagerank(graph)
```

```
    ranked_sentences = sorted(((scores[i], s) for i, s in
enumerate(sentences)), reverse=True)

    summary = " ".join([sent for _, sent in
ranked_sentences[:num_sentences]])

    return summary

text = """Natural Language Processing (NLP) is a field of artificial
intelligence that focuses on
the interaction between computers and humans using natural language.
The ultimate goal of NLP is
to enable computers to understand, interpret, and generate human
language in a way that is valuable.
Some common NLP tasks include machine translation, sentiment analysis,
text summarization, and speech recognition."""

summary = text_rank_summary(text, num_sentences=2)
print("Summary:\n", summary)
```

```
Summary:
 The ultimate goal of NLP is
to enable computers to understand, interpret, and generate human
language in a way that is valuable. Natural Language Processing (NLP)
is a field of artificial intelligence that focuses on
the interaction between computers and humans using natural language.
```

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
import nltk
import numpy as np
import networkx as nx
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.tokenize import sent_tokenize

nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
True
```

```
import nltk
nltk.download('punkt_tab')
```

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

```
True
```

```
def preprocess_text(text):
    """Tokenizes and cleans text"""
    sentences = sent_tokenize(text)
    return sentences

def build_similarity_matrix(sentences):
    """Computes sentence similarity matrix using TF-IDF"""
    vectorizer = TfidfVectorizer()
    sentence_vectors = vectorizer.fit_transform(sentences)

    similarity_matrix = np.dot(sentence_vectors,
sentence_vectors.T).toarray()

    return similarity_matrix

def text_rank_summary(text, num_sentences=3):
    """Applies TextRank algorithm for summarization"""
    sentences = preprocess_text(text)
    if len(sentences) <= num_sentences:
        return " ".join(sentences)

    similarity_matrix = build_similarity_matrix(sentences)

    graph = nx.from_numpy_array(similarity_matrix)
    scores = nx.pagerank(graph)
```

```
    ranked_sentences = sorted(((scores[i], s) for i, s in
enumerate(sentences)), reverse=True)

    summary = " ".join([sent for _, sent in
ranked_sentences[:num_sentences]])

    return summary

text = """Natural Language Processing (NLP) is a field of artificial
intelligence that focuses on
the interaction between computers and humans using natural language.
The ultimate goal of NLP is
to enable computers to understand, interpret, and generate human
language in a way that is valuable.
Some common NLP tasks include machine translation, sentiment analysis,
text summarization, and speech recognition."""

summary = text_rank_summary(text, num_sentences=2)
print("Summary:\n", summary)
```

```
Summary:
 The ultimate goal of NLP is
to enable computers to understand, interpret, and generate human
language in a way that is valuable. Natural Language Processing (NLP)
is a field of artificial intelligence that focuses on
the interaction between computers and humans using natural language.
```

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
import nltk
import numpy as np
import networkx as nx
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.tokenize import sent_tokenize

nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
True
```

```
import nltk
nltk.download('punkt_tab')
```

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

```
True
```

```
def preprocess_text(text):
    """Tokenizes and cleans text"""
    sentences = sent_tokenize(text)
    return sentences

def build_similarity_matrix(sentences):
    """Computes sentence similarity matrix using TF-IDF"""
    vectorizer = TfidfVectorizer()
    sentence_vectors = vectorizer.fit_transform(sentences)

    similarity_matrix = np.dot(sentence_vectors,
sentence_vectors.T).toarray()

    return similarity_matrix

def text_rank_summary(text, num_sentences=3):
    """Applies TextRank algorithm for summarization"""
    sentences = preprocess_text(text)
    if len(sentences) <= num_sentences:
        return " ".join(sentences)

    similarity_matrix = build_similarity_matrix(sentences)

    graph = nx.from_numpy_array(similarity_matrix)
    scores = nx.pagerank(graph)
```

```
    ranked_sentences = sorted(((scores[i], s) for i, s in
enumerate(sentences)), reverse=True)

    summary = " ".join([sent for _, sent in
ranked_sentences[:num_sentences]])

    return summary

text = """Natural Language Processing (NLP) is a field of artificial
intelligence that focuses on
the interaction between computers and humans using natural language.
The ultimate goal of NLP is
to enable computers to understand, interpret, and generate human
language in a way that is valuable.
Some common NLP tasks include machine translation, sentiment analysis,
text summarization, and speech recognition."""

summary = text_rank_summary(text, num_sentences=2)
print("Summary:\n", summary)
```

```
Summary:
 The ultimate goal of NLP is
to enable computers to understand, interpret, and generate human
language in a way that is valuable. Natural Language Processing (NLP)
is a field of artificial intelligence that focuses on
the interaction between computers and humans using natural language.
```

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
import nltk
import numpy as np
import networkx as nx
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.tokenize import sent_tokenize

nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
True
```

```
import nltk
nltk.download('punkt_tab')
```

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

```
True
```

```
def preprocess_text(text):
    """Tokenizes and cleans text"""
    sentences = sent_tokenize(text)
    return sentences

def build_similarity_matrix(sentences):
    """Computes sentence similarity matrix using TF-IDF"""
    vectorizer = TfidfVectorizer()
    sentence_vectors = vectorizer.fit_transform(sentences)

    similarity_matrix = np.dot(sentence_vectors,
sentence_vectors.T).toarray()

    return similarity_matrix

def text_rank_summary(text, num_sentences=3):
    """Applies TextRank algorithm for summarization"""
    sentences = preprocess_text(text)
    if len(sentences) <= num_sentences:
        return " ".join(sentences)

    similarity_matrix = build_similarity_matrix(sentences)

    graph = nx.from_numpy_array(similarity_matrix)
    scores = nx.pagerank(graph)
```

```
    ranked_sentences = sorted(((scores[i], s) for i, s in
enumerate(sentences)), reverse=True)

    summary = " ".join([sent for _, sent in
ranked_sentences[:num_sentences]])

    return summary

text = """Natural Language Processing (NLP) is a field of artificial
intelligence that focuses on
the interaction between computers and humans using natural language.
The ultimate goal of NLP is
to enable computers to understand, interpret, and generate human
language in a way that is valuable.
Some common NLP tasks include machine translation, sentiment analysis,
text summarization, and speech recognition."""

summary = text_rank_summary(text, num_sentences=2)
print("Summary:\n", summary)
```

```
Summary:
 The ultimate goal of NLP is
to enable computers to understand, interpret, and generate human
language in a way that is valuable. Natural Language Processing (NLP)
is a field of artificial intelligence that focuses on
the interaction between computers and humans using natural language.
```

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
from transformers import MarianMTModel, MarianTokenizer

def translate_text(text, src_lang="en", tgt_lang="fr"):
    """Translates text using a pre-trained MarianMT model."""

    model_name = f"Helsinki-NLP/opus-mt-{src_lang}-{tgt_lang}"
    tokenizer = MarianTokenizer.from_pretrained(model_name)
    model = MarianMTModel.from_pretrained(model_name)

    inputs = tokenizer(text, return_tensors="pt", padding=True,
truncation=True)

    translated_tokens = model.generate(**inputs)
    translated_text = tokenizer.decode(translated_tokens[0],
skip_special_tokens=True)

    return translated_text

text = "Machine translation enables seamless communication between
different languages."
translated = translate_text(text)
print("Translated Text:", translated)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/
_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your
settings tab (https://huggingface.co/settings/tokens), set it as
secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to
access public models or datasets.
  warnings.warn(
```


```
{"model_id":"7c1ea29b7a3e4a099c174a769e653cba","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"f5f4b0d2433e40a99b0df5db39ec089f","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"9384f967a86144fa9b5cf366185f783f","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"9585a8ccaf794a4a86f6cf7513e49433","version_major":2,"vers
ion_minor":0}
```

{"model_id":"0d5d51e309254e81abe1eab843e49f34","version_major":2,"version_minor":0}

```
/usr/local/lib/python3.11/dist-packages/transformers/models/marian/tokenization_marian.py:175: UserWarning: Recommended: pip install sacremoses.
  warnings.warn("Recommended: pip install sacremoses.")
```

{"model_id":"4e1efb0caca14cbfae5c531ad85e474f","version_major":2,"version_minor":0}

{"model_id":"d61176ba7c654e9293388ad1e515c95b","version_major":2,"version_minor":0}

{"model_id":"0417950d019e4d8d89acd06bc2ac388f","version_major":2,"version_minor":0}

```
Translated Text: La traduction automatique permet une communication transparente entre les différentes langues.
```

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
from transformers import MarianMTModel, MarianTokenizer

def translate_text(text, src_lang="en", tgt_lang="fr"):
    """Translates text using a pre-trained MarianMT model."""

    model_name = f"Helsinki-NLP/opus-mt-{src_lang}-{tgt_lang}"
    tokenizer = MarianTokenizer.from_pretrained(model_name)
    model = MarianMTModel.from_pretrained(model_name)

    inputs = tokenizer(text, return_tensors="pt", padding=True,
truncation=True)

    translated_tokens = model.generate(**inputs)
    translated_text = tokenizer.decode(translated_tokens[0],
skip_special_tokens=True)

    return translated_text

text = "Machine translation enables seamless communication between
different languages."
translated = translate_text(text)
print("Translated Text:", translated)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/
_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your
settings tab (https://huggingface.co/settings/tokens), set it as
secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to
access public models or datasets.
  warnings.warn(
```


```
{"model_id":"7c1ea29b7a3e4a099c174a769e653cba","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"f5f4b0d2433e40a99b0df5db39ec089f","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"9384f967a86144fa9b5cf366185f783f","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"9585a8ccaf794a4a86f6cf7513e49433","version_major":2,"vers
ion_minor":0}
```

{"model_id":"0d5d51e309254e81abe1eab843e49f34","version_major":2,"version_minor":0}

```
/usr/local/lib/python3.11/dist-packages/transformers/models/marian/tokenization_marian.py:175: UserWarning: Recommended: pip install sacremoses.
  warnings.warn("Recommended: pip install sacremoses.")
```

{"model_id":"4e1efb0caca14cbfae5c531ad85e474f","version_major":2,"version_minor":0}

{"model_id":"d61176ba7c654e9293388ad1e515c95b","version_major":2,"version_minor":0}

{"model_id":"0417950d019e4d8d89acd06bc2ac388f","version_major":2,"version_minor":0}

```
Translated Text: La traduction automatique permet une communication transparente entre les différentes langues.
```

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
from transformers import MarianMTModel, MarianTokenizer

def translate_text(text, src_lang="en", tgt_lang="fr"):
    """Translates text using a pre-trained MarianMT model."""

    model_name = f"Helsinki-NLP/opus-mt-{src_lang}-{tgt_lang}"
    tokenizer = MarianTokenizer.from_pretrained(model_name)
    model = MarianMTModel.from_pretrained(model_name)

    inputs = tokenizer(text, return_tensors="pt", padding=True,
truncation=True)

    translated_tokens = model.generate(**inputs)
    translated_text = tokenizer.decode(translated_tokens[0],
skip_special_tokens=True)

    return translated_text

text = "Machine translation enables seamless communication between
different languages."
translated = translate_text(text)
print("Translated Text:", translated)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/
_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your
settings tab (https://huggingface.co/settings/tokens), set it as
secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to
access public models or datasets.
  warnings.warn(
```


```
{"model_id":"7c1ea29b7a3e4a099c174a769e653cba","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"f5f4b0d2433e40a99b0df5db39ec089f","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"9384f967a86144fa9b5cf366185f783f","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"9585a8ccaf794a4a86f6cf7513e49433","version_major":2,"vers
ion_minor":0}
```

{"model_id":"0d5d51e309254e81abe1eab843e49f34","version_major":2,"version_minor":0}

```
/usr/local/lib/python3.11/dist-packages/transformers/models/marian/tokenization_marian.py:175: UserWarning: Recommended: pip install sacremoses.
  warnings.warn("Recommended: pip install sacremoses.")
```

{"model_id":"4e1efb0caca14cbfae5c531ad85e474f","version_major":2,"version_minor":0}

{"model_id":"d61176ba7c654e9293388ad1e515c95b","version_major":2,"version_minor":0}

{"model_id":"0417950d019e4d8d89acd06bc2ac388f","version_major":2,"version_minor":0}

```
Translated Text: La traduction automatique permet une communication transparente entre les différentes langues.
```

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
from transformers import MarianMTModel, MarianTokenizer

def translate_text(text, src_lang="en", tgt_lang="fr"):
    """Translates text using a pre-trained MarianMT model."""

    model_name = f"Helsinki-NLP/opus-mt-{src_lang}-{tgt_lang}"
    tokenizer = MarianTokenizer.from_pretrained(model_name)
    model = MarianMTModel.from_pretrained(model_name)

    inputs = tokenizer(text, return_tensors="pt", padding=True,
truncation=True)

    translated_tokens = model.generate(**inputs)
    translated_text = tokenizer.decode(translated_tokens[0],
skip_special_tokens=True)

    return translated_text

text = "Machine translation enables seamless communication between
different languages."
translated = translate_text(text)
print("Translated Text:", translated)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/
_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your
settings tab (https://huggingface.co/settings/tokens), set it as
secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to
access public models or datasets.
  warnings.warn(
```


```
{"model_id":"7c1ea29b7a3e4a099c174a769e653cba","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"f5f4b0d2433e40a99b0df5db39ec089f","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"9384f967a86144fa9b5cf366185f783f","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"9585a8ccaf794a4a86f6cf7513e49433","version_major":2,"vers
ion_minor":0}
```

{"model_id":"0d5d51e309254e81abe1eab843e49f34","version_major":2,"version_minor":0}

```
/usr/local/lib/python3.11/dist-packages/transformers/models/marian/tokenization_marian.py:175: UserWarning: Recommended: pip install sacremoses.
  warnings.warn("Recommended: pip install sacremoses.")
```

{"model_id":"4e1efb0caca14cbfae5c531ad85e474f","version_major":2,"version_minor":0}

{"model_id":"d61176ba7c654e9293388ad1e515c95b","version_major":2,"version_minor":0}

{"model_id":"0417950d019e4d8d89acd06bc2ac388f","version_major":2,"version_minor":0}

```
Translated Text: La traduction automatique permet une communication transparente entre les différentes langues.
```

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
from transformers import MarianMTModel, MarianTokenizer

def translate_text(text, src_lang="en", tgt_lang="fr"):
    """Translates text using a pre-trained MarianMT model."""

    model_name = f"Helsinki-NLP/opus-mt-{src_lang}-{tgt_lang}"
    tokenizer = MarianTokenizer.from_pretrained(model_name)
    model = MarianMTModel.from_pretrained(model_name)

    inputs = tokenizer(text, return_tensors="pt", padding=True,
truncation=True)

    translated_tokens = model.generate(**inputs)
    translated_text = tokenizer.decode(translated_tokens[0],
skip_special_tokens=True)

    return translated_text

text = "Machine translation enables seamless communication between
different languages."
translated = translate_text(text)
print("Translated Text:", translated)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/
_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your
settings tab (https://huggingface.co/settings/tokens), set it as
secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to
access public models or datasets.
  warnings.warn(
```


```
{"model_id":"7c1ea29b7a3e4a099c174a769e653cba","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"f5f4b0d2433e40a99b0df5db39ec089f","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"9384f967a86144fa9b5cf366185f783f","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"9585a8ccaf794a4a86f6cf7513e49433","version_major":2,"vers
ion_minor":0}
```

{"model_id":"0d5d51e309254e81abe1eab843e49f34","version_major":2,"version_minor":0}

```
/usr/local/lib/python3.11/dist-packages/transformers/models/marian/tokenization_marian.py:175: UserWarning: Recommended: pip install sacremoses.
  warnings.warn("Recommended: pip install sacremoses.")
```

{"model_id":"4e1efb0caca14cbfae5c531ad85e474f","version_major":2,"version_minor":0}

{"model_id":"d61176ba7c654e9293388ad1e515c95b","version_major":2,"version_minor":0}

{"model_id":"0417950d019e4d8d89acd06bc2ac388f","version_major":2,"version_minor":0}

```
Translated Text: La traduction automatique permet une communication transparente entre les différentes langues.
```