Lab 07: Machine Translation

Machine translation can be defines as a process in which a source input is converted into a "target". What we feed in the system is the source (can be any language) and the translation will convert the language. Example, hello converts to namaste.

Requirements:

1. Database/Corpus: this will have two fields, one will have all of the data (english) and corresponding to that in the second field we will have it in the target language(chinese). This kind is the tool based translation.

Ways:

1. Direct translation: Any text is converted to corresponding text based upon direct meaning from dictionary. Without preserving the meaning from the word
2. Syntactic transfer: based on the structure and rules
3. Semantic transfer: we will preserve the word meaning.
4. Interlingual: Large corpus needed to execute machine translation.

Types:

1. Stats based: using stats and prob, eg: ngram,tfidf
2. Rule based: two fields
3. Hybrid: stats+rule
4. Neural: no preset rules, find out relationships, using deep learning, based on relationships weights will be adjusted and w the weights we can make out the meaning of the senetence.

In translation, the meaning remains the same. In conversion, like closed captions in yt it converts direct language into words, sometimes there are errors. not updated.

In [1]:
```
!pip install transformers sentencepiece
```

```
Requirement already satisfied: transformers in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (4.33.3)
Collecting sentencepiece
  Downloading sentencepiece-0.2.0-cp311-cp311-win_amd64.whl.metadata (8.3 kB)
Requirement already satisfied: filelock in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (from transformers)
(3.12.4)
Requirement already satisfied: huggingface-hub<1.0,>=0.15.1 in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages
(from transformers) (0.17.3)
Requirement already satisfied: numpy>=1.17 in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (from transforme
rs) (1.24.1)
Requirement already satisfied: packaging>=20.0 in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (from transf
ormers) (23.0)
Requirement already satisfied: pyyaml>=5.1 in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (from transforme
rs) (6.0)
Requirement already satisfied: regex!=2019.12.17 in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (from tran
sformers) (2023.8.8)
Requirement already satisfied: requests in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (from transformers)
(2.28.2)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in c:\users\ahana\appdata\local\programs\python\python311\lib\site-pac
kages (from transformers) (0.13.3)
Requirement already satisfied: safetensors>=0.3.1 in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (from tra
nsformers) (0.3.3)
Requirement already satisfied: tqdm>=4.27 in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (from transformer
s) (4.65.0)
Requirement already satisfied: fsspec in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (from huggingface-hub
<1.0,>=0.15.1->transformers) (2023.9.2)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages
(from huggingface-hub<1.0,>=0.15.1->transformers) (4.8.0)
Requirement already satisfied: colorama in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (from tqdm>=4.27->t
ransformers) (0.4.6)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (fr
om requests->transformers) (3.0.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (from requests-
>transformers) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (from
requests->transformers) (1.26.14)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (from req
uests->transformers) (2022.12.7)
Downloading sentencepiece-0.2.0-cp311-cp311-win_amd64.whl (991 kB)
   ---------------------------------------- 0.0/991.5 kB ? eta -:--:--
   - -------------------------------------- 30.7/991.5 kB 660.6 kB/s eta 0:00:02
   - -------------------------------------- 41.0/991.5 kB 393.8 kB/s eta 0:00:03
   --- ------------------------------------ 92.2/991.5 kB 655.4 kB/s eta 0:00:02
   ------ --------------------------------- 163.8/991.5 kB 821.4 kB/s eta 0:00:02
   --------- ------------------------------ 225.3/991.5 kB 1.1 MB/s eta 0:00:01
   ---------------- ----------------------- 409.6/991.5 kB 1.5 MB/s eta 0:00:01
   ----------------------- ---------------- 573.4/991.5 kB 1.8 MB/s eta 0:00:01
   ---------------------------------- ------ 829.4/991.5 kB 2.2 MB/s eta 0:00:01
   ---------------------------------------- 991.5/991.5 kB 2.3 MB/s eta 0:00:00
Installing collected packages: sentencepiece
Successfully installed sentencepiece-0.2.0
```

In [23]:
```
from transformers import MarianMTModel, MarianTokenizer
import torch
```

In [24]:
```
#Model and tokenizer names for English to French translation
model_name = "Helsinki-NLP/opus-mt-en-fr"
tokenizer_name = model_name
```

In [25]:
```
model = MarianMTModel.from_pretrained(model_name)
tokenizer = MarianTokenizer.from_pretrained(tokenizer_name)
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
Cell In[25], line 1
----> 1 model = MarianMTModel.from_pretrained(model_name)
      2 tokenizer = MarianTokenizer.from_pretrained(tokenizer_name)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\transformers\utils\import_utils.py:1124, in DummyObject.__getattribute
__(cls, key)
   1122 if key.startswith("_") and key != "_from_config":
   1123     return super().__getattribute__(key)
-> 1124 requires_backends(cls, cls._backends)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\transformers\utils\import_utils.py:1103, in requires_backends(obj, bac
kends)
   1101 # Raise an error for users who might not realize that classes without "TF" are torch-only
   1102 if "torch" in backends and "tf" not in backends and not is_torch_available() and is_tf_available():
-> 1103     raise ImportError(PYTORCH_IMPORT_ERROR_WITH_TF.format(name))
   1105 # Raise the inverse error for PyTorch users trying to load TF classes
   1106 if "tf" in backends and "torch" not in backends and is_torch_available() and not is_tf_available():

ImportError:
MarianMTModel requires the PyTorch library but it was not found in your environment.
However, we were able to find a TensorFlow installation. TensorFlow classes begin
with "TF", but are otherwise identically named to our PyTorch classes. This
means that the TF equivalent of the class you tried to import would be "TFMarianMTModel".
If you want to use TensorFlow, please use TF classes instead!

If you really do want to use PyTorch please go to
https://pytorch.org/get-started/locally/ (https://pytorch.org/get-started/locally/) and follow the instructions that
match your environment.
```

In [10]:
```
text="How are you doing today?"

inputs=tokenizer(text,return_tensors="pt")

translated=model.generate(**inputs)

translation= tokenizer.batch_decode(translated,skip_special_tokens=True)[0]

print(f"Translated text ({model_name}): {translation}")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[10], line 3
      1 text="How are you doing today?"
----> 3 inputs=tokenizer(text,return_tensors="pt")
      5 translated=model.generate(**inputs)
      7 translation= tokenizer.batch_decode(translated,skip_special_tokens=True)[0]

NameError: name 'tokenizer' is not defined
```

Assignment 1 - 6 Marks

1. WAP to perform to POS Tag from given set of text file
2. WAP to perform and calculate TF-IDF of a given set of text file
3. WAP to implement n-grams model of text generation.

In [12]:
```
#Question 1: POS Tag
import nltk
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag
from nltk.corpus import stopwords

def pos_tagging(text):
    tokens = word_tokenize(text)
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [token for token in tokens if token.lower() not in stop_words]
    pos_tags = pos_tag(filtered_tokens)
    return pos_tags

text = input("Enter your text: ")
result = pos_tagging(text)
print("Parts of Speech Tags: ", result)
```

```
Enter your text: hi my name is ahana and i am studying natural language processing in the sixth semester of AI and DS
Parts of Speech Tags:  [('hi', 'NN'), ('name', 'NN'), ('ahana', 'IN'), ('studying', 'VBG'), ('natural', 'JJ'), ('language', 'NN'), ('pr
ocessing', 'NN'), ('sixth', 'JJ'), ('semester', 'NN'), ('AI', 'NNP'), ('DS', 'NNP')]
```

In [15]:
```python
#Question 2:
from sklearn.feature_extraction.text import TfidfVectorizer
def cal_tfidf(text):
    vectorizer = TfidfVectorizer(stop_words='english')
    tfidf_mat = vectorizer.fit_transform([text])
    fn = vectorizer.get_feature_names_out()
    return tfidf_mat,fn


user_input = input("Enter the text: ")
tfidf_mat, feature_names = cal_tfidf(user_input)
print("TF-IDF values:")
for i, feature in enumerate(feature_names):
    print(f"{feature}: {tfidf_mat[0, i]}")
```

```
Enter the text: hi my name is ahana my subject is nlp
TF-IDF values:
ahana: 0.5
hi: 0.5
nlp: 0.5
subject: 0.5
```

In [31]:
```python
#Question 3:
import random
def create_ngrams(text, n):
    words = text.split()
    ngrams = []
    for i in range(len(words) - n + 1):
        ngrams.append(words[i:i + n])
    return ngrams

def train_ngram_model(text, n):
    ngrams = create_ngrams(text, n)
    model = {}
    for ngram in ngrams:
        prefix = tuple(ngram[:-1])
        suffix = ngram[-1]
        if prefix not in model:
            model[prefix] = []
        model[prefix].append(suffix)
    return model

def generate_text(model, seed_text, max_words):
    output = seed_text.split()
    prefix = tuple(seed_text.split()[-(n - 1):])
    for _ in range(max_words):
        if prefix in model:
            suffix = random.choice(model[prefix])
            output.append(suffix)
            prefix = tuple(output[-(n - 1):])
        else:
            break
    return ' '.join(output)

text = """The quick brown fox jumps over the lazy dog. A quick brown dog outpaces a quick fox. The dog and the fox like to run in the for
n = 2

model = train_ngram_model(text, n)
seed_text = "fox"
max_words = 5
generated_text = generate_text(model, seed_text, max_words)
print(generated_text)
```

```
fox like to run in the
```