In [ ]:

```
 !!!!!!!!!!!WE ARE CHECKING WHICH CLAUSE IS ARBITRARY OPT OT
-------------------------------------------------------------

import spacy
from spacy.training.example import Example
import random
import numpy as np
from sklearn.model_selection import train_test_split
from transformers import BertTokenizer, TFBertModel
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_r

nlp = spacy.load("en_core_web_sm")

tokenizer = BertTokenizer.from_pretrained("bert-base-uncased
bert_model = TFBertModel.from_pretrained("bert-base-uncased"

# Example data: We need to replace this with our dataset
# 1 represents arbitration opt-out clauses, 0 represents non
data = [
    ("This is a sample non-arbitration text.", 0),
    ("If you wish to opt out of arbitration...", 1),
    # Add more data points as needed
]

X =  [item[0] for item in data]
y = [item[1] for item in data]

X_train, X_temp, y_train, y_temp = train_test_split(X, y, te
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_te


 def preprocess_text(text):
    doc = nlp(text)
    tokens = [token.text for token in doc]
    return " ".join(tokens)

 X_train = [preprocess_text(text) for text in X_train]
 X_val = [preprocess_text(text) for text in X_val]
 X_test = [preprocess_text(text) for text in X_test]

 X_train_tokens = [tokenizer(X_train, padding=True, truncati
```

```
 X_val_tokens = [tokenizer(X_val, padding=True, truncation=T
 X_test_tokens = [tokenizer(X_test, padding=True, truncation

 X_train_bert = [bert_model(X_train_tokens)for text in X_tra
 X_val_bert = [bert_model(X_val_tokens)for text in X_val_tok
 X_test_bert = [bert_model(X_test_tokens)for text in X_test_

 # Flatten the BERT embeddings
 X_train_bert = [X_train_bert[0].numpy()for text in X_train_
 X_val_bert = [X_val_bert[0].numpy()for text in X_val_bert]
 X_test_bert = [X_test_bert[0].numpy()for text in X_test_ber

classifier = MultinomialNB()
classifier.fit(X_train_bert, y_train)

y_val_pred = classifier.predict(X_val_bert)

accuracy = accuracy_score(y_val, y_val_pred)
print(f"Validation Accuracy: {accuracy:.2f}")
print(classification_report(y_val, y_val_pred, target_names=


!!!!!!!!!!!!!!!!!!!!!!!!CODE TO EXTRACT ARBITRARY OPT OUT CL
-----------------------------------------------------------

For example we have a license document. It will contains cla
and to extract arbitrary clauses


# Identify documents predicted as arbitration opt-out clause
predicted_arbitration_indices = [index for index, label in e

# Match predicted documents to original texts
arbitration_opt_out_texts = [X_input[index] for index in pre

# Extract the text content of the arbitration opt-out docume
for text in arbitration_opt_out_texts:
    print("Arbitration Opt-Out Document:")
    print(text)

#The text from above code will be used to extract the opt ou
```

```
!!!!!!!!!!!!!!!HERE WE ARE CREATING CUSTOM NER MODEL & EXTRA
-------------------------------------------------------------


nlp = spacy.blank("en")

ner = nlp.add_pipe("ner")
ner.add_label("DEADLINE")
ner.add_label("ACTION")
ner.add_label("CONTENT")
ner.add_label("TARGET")

# sample training data
training_data = [
    ("Deadline: 30, Action: MAIL, Content: John Doe 123 Main
    ("Target: Company Inc., 456 Oak St, Attn: Legal Departme
]

optimizer = nlp.begin_training()

for i in range(50):
    random.shuffle(training_data)

    for text, annotations in training_data:
        doc = nlp.make_doc(text)
        example = Example.from_dict(doc, annotations)
        nlp.update([example], drop=0.5)

# Save the trained model
nlp.to_disk("custom_ner_model")

# Load the trained model
nlp = spacy.load("custom_ner_model")

# Test the model on new text
text = "4.6 Acceptance of Arbitration and Right to Opt Out. \
"
doc = nlp(text)

for ent in doc.ents:
    print(ent.text, ent.label_)
```

In [ ]: