In [1]:

```
# Named Entity Recognition (ner) using Transformers developed by Hugging Face, Inc.
# Note: Transformers were installed with 'python -m pip install transformers' in pytorch environment (pytorchenv active)
```

In [2]:

```
# import necessary transformers classes and torch

from transformers import AutoModelForTokenClassification, AutoTokenizer
import torch
```

In [3]:

```
# instantiate pretrained tokenizer and model

# uses fine-tuned model on CoNLL-2003, fine-tuned by @stefan-it from dbmdz
model = AutoModelForTokenClassification.from_pretrained("dbmdz/bert-large-cased-finetuned-conll03-english")
tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
```

In [4]:

```
# define the label list with which the model was trained on

label_list = [
    "O",       # Outside of a named entity
    "B-MISC",  # Beginning of a miscellaneous entity right after another miscellaneous entity
    "I-MISC",  # Miscellaneous entity
    "B-PER",   # Beginning of a person's name right after another person's name
    "I-PER",   # Person's name
    "B-ORG",   # Beginning of an organisation right after another organisation
    "I-ORG",   # Organisation
    "B-LOC",   # Beginning of a location right after another location
    "I-LOC"    # Location
]
```

In [5]:

```
# define a sequence with named entities

sequence = "Google was founded in September 1998 by Larry Page and Sergey Brin while " \
           "they were Ph.D. students at Stanford University in California."
```

In [6]:

```
# split words into tokens so that they can be mapped to the predictions
tokens = tokenizer.tokenize(tokenizer.decode(tokenizer.encode(sequence)))

# encode that sequence into IDs
inputs = tokenizer.encode(sequence, return_tensors = "pt")
```

In [7]:

```
# retrieve the predictions by passing the input to the model

outputs = model(inputs)[0]
predictions = torch.argmax(outputs, dim = 2)
```

In [8]:

```
# zip together each token with its prediction and print it

print([(token, label_list[prediction]) for token, prediction in zip(tokens, predictions[0].tolist())])
```

```
[('[CLS]', 'O'), ('Google', 'I-ORG'), ('was', 'O'), ('founded', 'O'), ('in',
'O'), ('September', 'O'), ('1998', 'O'), ('by', 'O'), ('Larry', 'I-PER'), ('P
age', 'I-PER'), ('and', 'O'), ('Sergey', 'I-PER'), ('B', 'I-PER'), ('##rin',
'I-PER'), ('while', 'O'), ('they', 'O'), ('were', 'O'), ('Ph', 'O'), ('.',
'O'), ('D', 'O'), ('.', 'O'), ('students', 'O'), ('at', 'O'), ('Stanford', 'I
-ORG'), ('University', 'I-ORG'), ('in', 'O'), ('California', 'I-LOC'), ('.',
'O'), ('[SEP]', 'O')]
```

In [9]:

```
# print only predictions for Person, Organization and Location --> label in ["I-PER", "I-ORG", "I-LOC"]

print([(token, label_list[prediction]) for token, prediction in zip(tokens, predictions[0].tolist())
       if prediction in [4, 6, 8]])
```

```
[('Google', 'I-ORG'), ('Larry', 'I-PER'), ('Page', 'I-PER'), ('Sergey', 'I-PE
R'), ('B', 'I-PER'), ('##rin', 'I-PER'), ('Stanford', 'I-ORG'), ('Universit
y', 'I-ORG'), ('California', 'I-LOC')]
```

In [10]:

```
# All named entities have been accurately recognized
```