Intro to deep learning

Dr. Janoś Gabler, University of Bonn

Lecture 3: First experience with the Hugging Face

What you will learn in this lecture

- How to use the Hugging Face models library
- How to define specific tasks
- How to do first predictions with pretrained models

Example input

```
text = """Dear Amazon, last week I ordered an Optimus
Prime action figure from your online store in Germany.
Unfortunately, when I opened the package, I discovered
to my horror that I had been sent an action figure of
Megatron instead! As a lifelong enemy of the Decepticons,
I hope you can understand my dilemma. To resolve the
issue, I demand an exchange of Megatron for the Optimus
Prime figure I ordered. Enclosed are copies of my records
concerning this purchase. I expect to hear from you soon.
Sincerely, Bumblebee."""
```

- Does the text have a negative or positive connotation?
- What people/objects are named in the text?
- What are main points of the text?
- What if we need a translation of the text?

Text classification (sentiment analysis)

```
>>> from transformers import pipeline
>>> classifier = pipeline(task="text-classification")
>>> sentiments = classifier(text)
>>> sentiments
[{'label': 'NEGATIVE', 'score': 0.9015460014343262}]
```

- Call a `` `pipeline` `` with the task `` `"text-classification"` ``
- A task-specific default model is used, but you can define your model!
- The results are a list of predictions with confidence scores.

Named entity recognition (ner)

```
>>> ner_tagger = pipeline(
... task="ner", aggregation_strategy="simple"
  )
>>> entities = ner_tagger(text)
>>> entities # truncated output
[{'entity_group': 'ORG',
  'score': 0.9411546,
  'word': 'amazon',
  'start': 5,
  'end': 11},
 {'entity_group': 'MISC',
  'score': 0.9315696,
  'word': 'optimus prime',
  'start': 36,
  'end': 49},]
```

- The task key is `` `"ner"` ``
- Additional arguments to define the pipeline
- The output is a list of dictionaries with recognized entities

Question answering

```
>>> reader = pipeline(task="question-answering")
>>> question = "What does the customer want?"
>>> answers = reader(question=question, context=text)
>>> answers
{'score': 0.6312923431396484,
 'start': 335,
 'end': 358,
 'answer': 'an exchange of Megatron'}
```

- The task key is `` `"question-answering"` ``
- Define your question(s) as a string (or list of strings)
- Pass your question and the text as a context to the pipeline
- The output is a list of dictionaries with answers to questions

Summarization

```
>>> summarizer = pipeline(task="summarization")
>>> summaries = summarizer(
...  text,
...  max_length=45,
...  min_length=45,
...  clean_up_tokenization_spaces=True
... )
>>> summaries
[{"summary_text": "Amazon's Optimus Prime action figure\
wassent to a German online store in Germany. Bumblebee\
asks for an exchange of Megatron for the Optimus Prime\
figure he ordered. He expects to hear from you soon."}]
```

- The task key is `` `"summarization"` ``
- Summarize the text using additional arguments
- The output is a list of dictionaries with short summaries of the input text

Translation

```
>>> translator = pipeline(task="translation_en_to_de")
>>> translations = translator(
... text,
... clean_up_tokenization_spaces=True,
... min_lenghth=50
... )
>>> translations
[{"translation_text": "Sehr geehrter Amazon,\
letzte Woche habe ich eine Optimus Prime Action\
Figur aus Ihrem Online-Shop in Deutschland bestellt.\
Leider, als ich das Paket öffnete, entdeckte ich zu\
meinem Entsetzen, dass ich stattdessen eine Action\
Figur von Megatron geschickt worden war! Als\
lebenslanger Feind der Decepticons, Ich hoffe,\
Sie können mein Dilemma verstehen. Um das Problem\
zu lösen, Ich fordere einen Austausch von Megatron\
für die Optimus Prime Figur habe ich bestellt."}]
```

- The task key is `` `"translation_xx_to_yy"` ``
- Translate your text using additional arguments
- The output is a list of dictionaries with the translated text