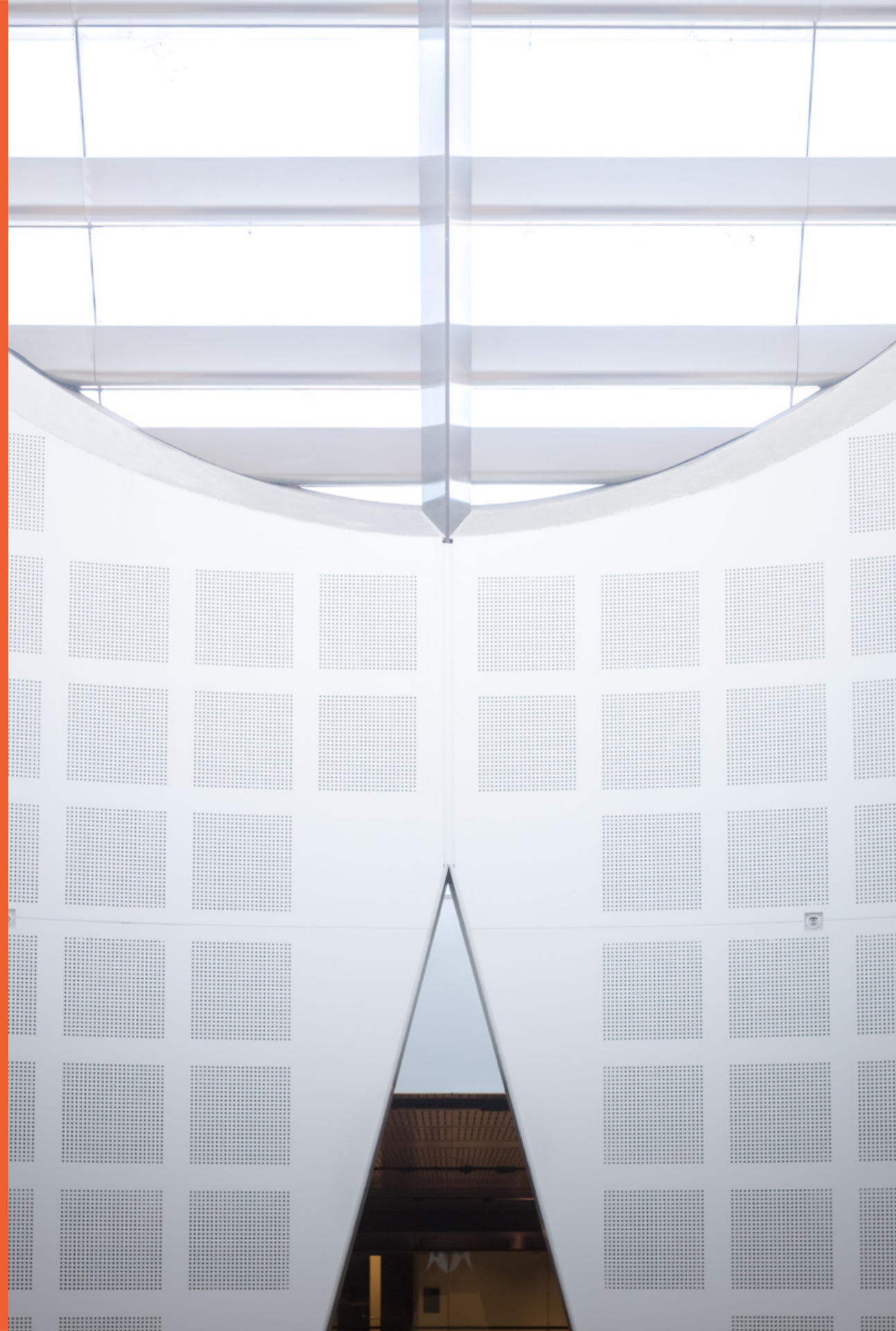
Natural Language Processing for Clinical Text



THE UNIVERSITY OF SYDNEY

Demo of NLP Tasks

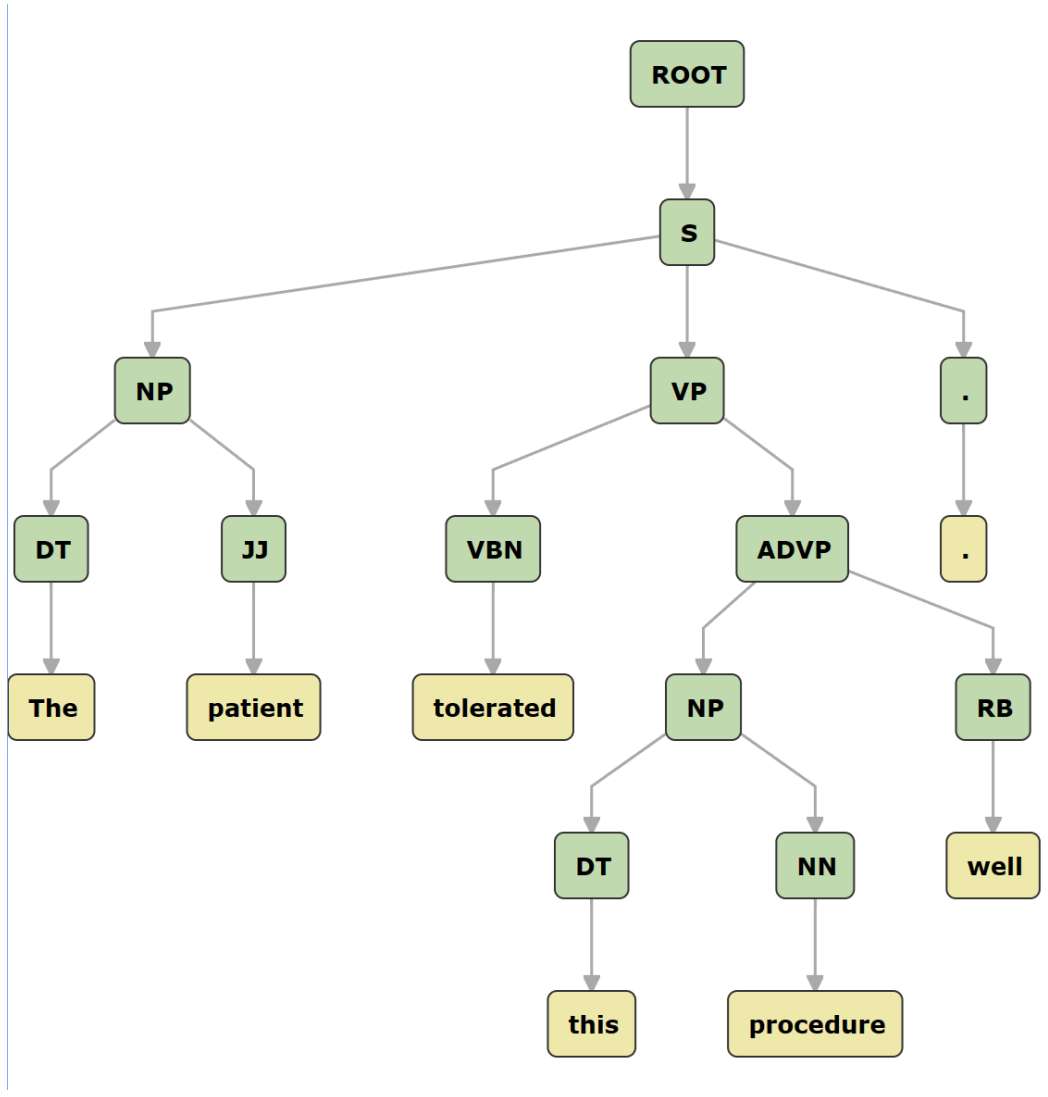
Part of speech

VBD: Verb, past tense
VBG: Verb, gerund or present participle
VBN: Verb, past participle
VBP: Verb, non-3rd person singular present
VBZ: Verb, 3rd person singular present
JJ: Adjective
RB: Adverb
POS: Possessive ending
DT: Determiner
PRP: Personal pronoun Phrase
PRP: Possessive pronoun Phrase
NN: Noun, singular or mass
NNS: Noun, plural
PP: Prerequisite Phrase
IN: Preposition or subordinating conjunction

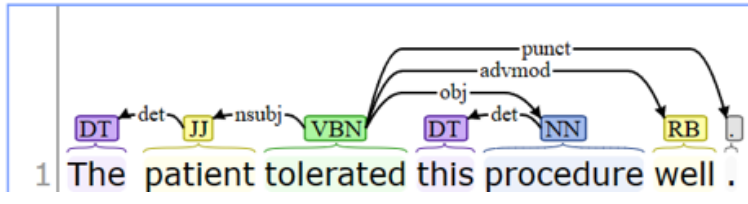
https://corenlp.run/

Demo of NLP Tasks

e.g. The patient tolerated this procedure well.



Basic Dependencies:

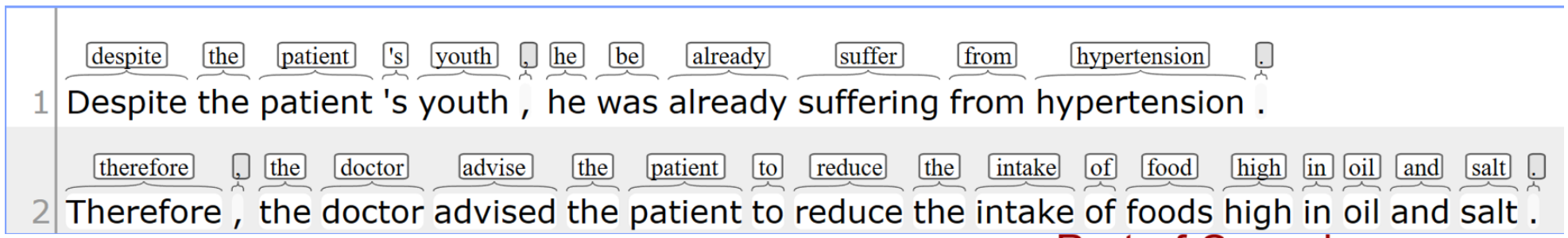


https://corenlp.run/

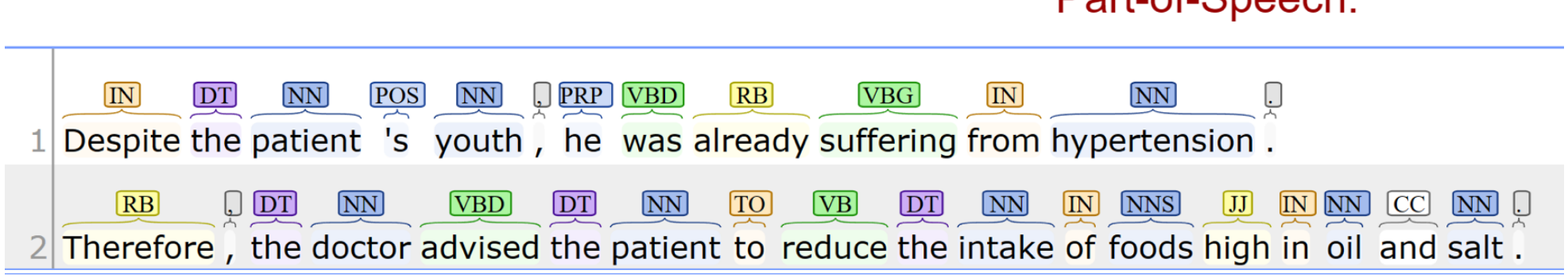
Demo of NLP Tasks

e.g. Despite the patient's youth, he was already suffering from hypertension. Therefore, the doctor advised the patient to reduce the intake of foods high in oil and salt.

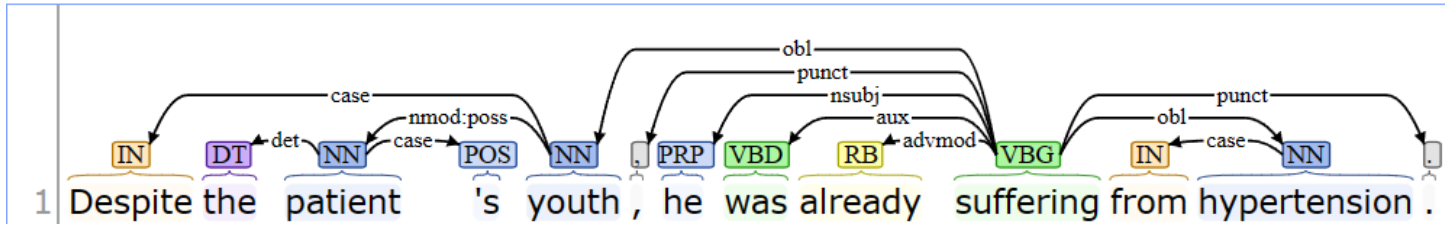
Lemmas:



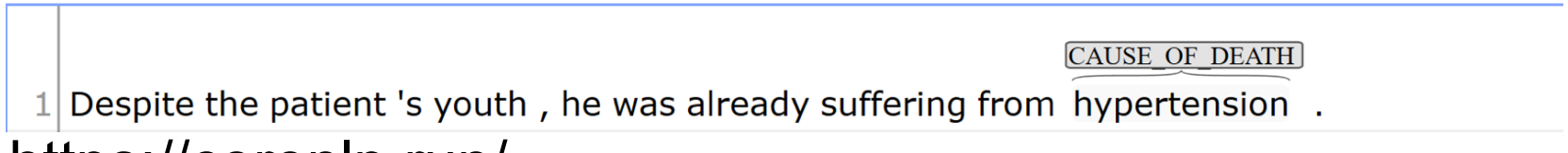
Part-of-Speech:



Basic Dependencies:



Named Entity Recognition:



https://corenlp.run/

Demo of NLP Tasks

e.g. Despite the patient's youth, he was already suffering from hypertension. Therefore, the doctor advised the patient to reduce the intake of foods high in oil and salt.

Open Information Extraction

Extractions for was :

Despite the patient 's youth , he **was** (V) already suffering from hypertension . Therefore , the doctor advised the patient to reduce the intake of foods high in oil and salt .

Extractions for suffering :

Despite the patient 's youth (ARGM-ADV) , he (ARG0) was already (ARGM-TMP) **suffering** (V) from hypertension . Therefore (ARG1) , the doctor advised the patient to reduce the intake of foods high in oil and salt .

Extractions for advised :

Despite the patient 's youth , he was already suffering from hypertension . Therefore , the doctor (ARG0) **advised** (V) the patient (ARG1) to reduce the intake of foods high in oil and salt (ARG2) .

Extractions for reduce :

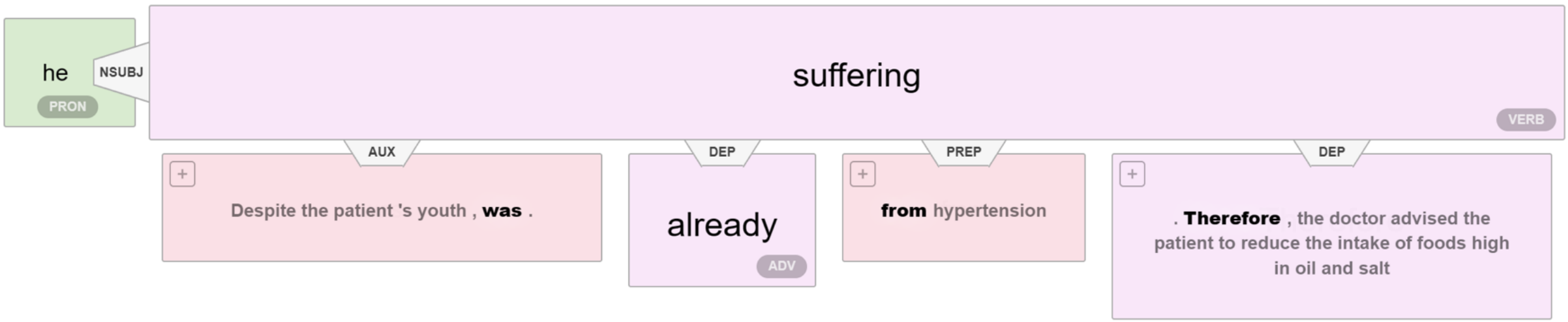
Despite the patient 's youth , he was already suffering from hypertension . Therefore , the doctor advised the patient (ARG0) to **reduce** (V) the intake of foods high in oil and salt (ARG1) .

https://demo.allennlp.org/sentiment-analysis/glove-sentiment-analysis

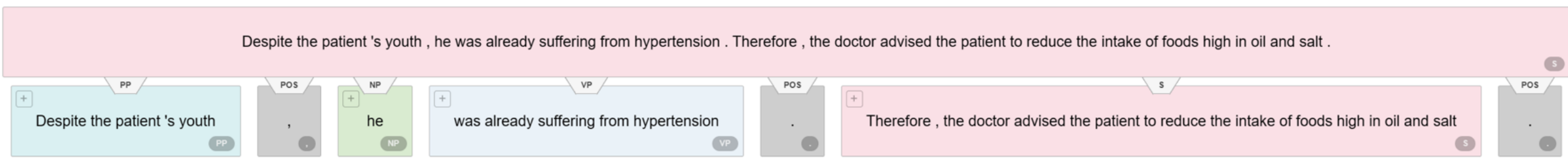
Demo of NLP Tasks

e.g. Despite the patient's youth, he was already suffering from hypertension. Therefore, the doctor advised the patient to reduce the intake of foods high in oil and salt.

Dependency Parsing



Constituency Parsing



https://demo.allennlp.org/sentiment-analysis/glove-sentiment-analysis

Demo of NLP Tasks

e.g. Despite the patient's youth, he was already suffering from hypertension. Therefore, the doctor advised the patient to reduce the intake of foods high in oil and salt.

Continue Writing (Language Modeling)

Sentence

Despite the patient's youth, he was already suffering from hypertension. Therefore, the doctor advised the patient to reduce the intake of foods high in oil and salt. ↲The patient's

Run Model

Model Output

Share

Prediction	Score
Despite the patient's youth, he was already suffering from hypertension. Therefore, the doctor advised the patient to reduce the intake of foods high in oil and salt. ↲The patient's **daily intake of the food ...**	99.1%
Despite the patient's youth, he was already suffering from hypertension. Therefore, the doctor advised the patient to reduce the intake of foods high in oil and salt. ↲The patient's **first half-life for ...**	0.7%
Despite the patient's youth, he was already suffering from hypertension. Therefore, the doctor advised the patient to reduce the intake of foods high in oil and salt. ↲The patient's **internal organs were keen to ...**	0.1%

https://demo.allennlp.org/sentiment-analysis/glove-sentiment-analysis

Demo of NLP Tasks

e.g. Despite the patient's youth, he was already suffering from hypertension. Therefore, the doctor advised the patient to reduce the intake of foods high in oil and salt.

Masked Language Modeling

Sentence

Despite the patient's youth, he was already suffering from hypertension. Therefore, the doctor advised the patient to reduce the [MASK] of foods high in oil and salt.

Run Model

Model Output

Share

Mask 1

Prediction	Score
Despite the patient ' s youth , he was already suffering from h ##yper ##tens ##ion . Therefore , the doctor advised the patient to reduce the **intake** of foods high in oil and salt .	51%
Despite the patient ' s youth , he was already suffering from h ##yper ##tens ##ion . Therefore , the doctor advised the patient to reduce the **consumption** of foods high in oil and salt .	36.2%
Despite the patient ' s youth , he was already suffering from h ##yper ##tens ##ion . Therefore , the doctor advised the patient to reduce the **use** of foods high in oil and salt .	3.2%
Despite the patient ' s youth , he was already suffering from h ##yper ##tens ##ion . Therefore , the doctor advised the patient to reduce the **amount** of foods high in oil and salt .	1.9%
Despite the patient ' s youth , he was already suffering from h ##yper ##tens ##ion . Therefore , the doctor advised the patient to reduce the **production** of foods high in oil and salt .	0.7%

https://demo.allennlp.org/sentiment-analysis/glove-sentiment-analysis

Demo of NLP Tasks

e.g. Despite the patient's youth, he was already suffering from hypertension. Therefore, the doctor advised the patient to reduce the intake of foods high in oil and salt.

Evaluate Reading Comprehension

Context

Despite the patient's youth, he was already suffering from hypertension. Therefore, the doctor advised the patient to reduce the intake of foods high in oil and salt.

Question

What should the patient do?

Reference

The patient should reduce the intake of foods high in oil and salt.

Candidate

The patient should eat less food high in oil and salt.

Run Model

Model Output

Predicted Score

0.7221695780754089

https://demo.allennlp.org/sentiment-analysis/glove-sentiment-analysis

import transformers

Hugging Face is a large open-source community that quickly became an enticing hub for pre-trained deep learning models, mainly aimed at NLP.

Install the package with: ***!pip install transformers***

https://huggingface.co/

Hugging Face

Pre-processing Text Sentences

transformers.AutoTokenizer

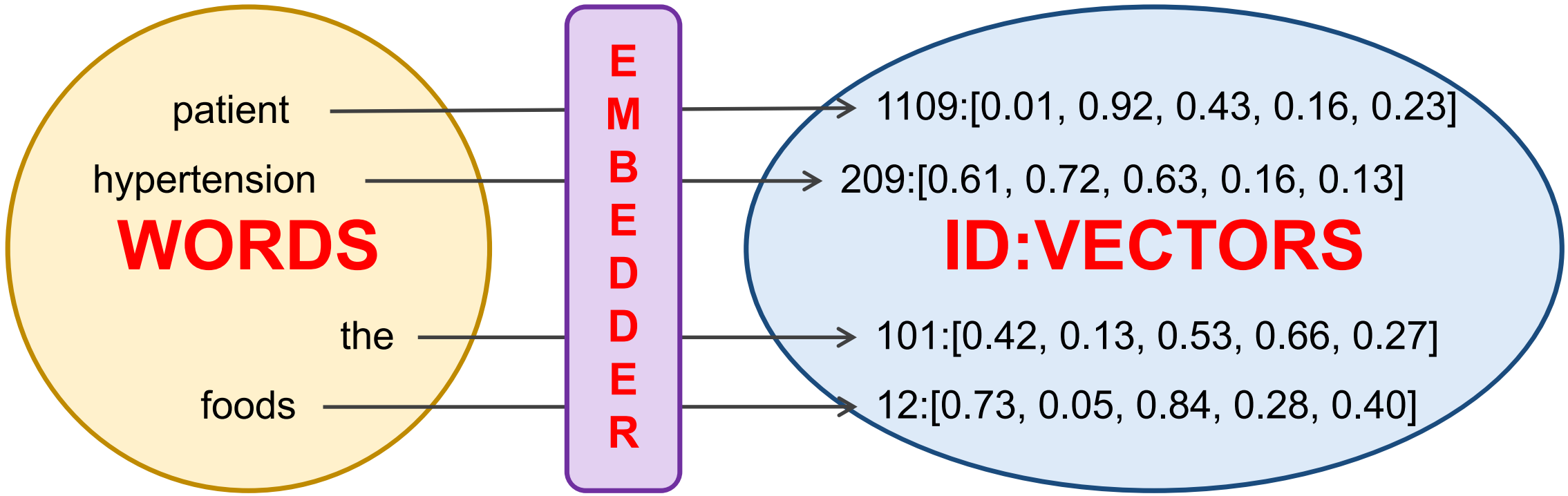
tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")

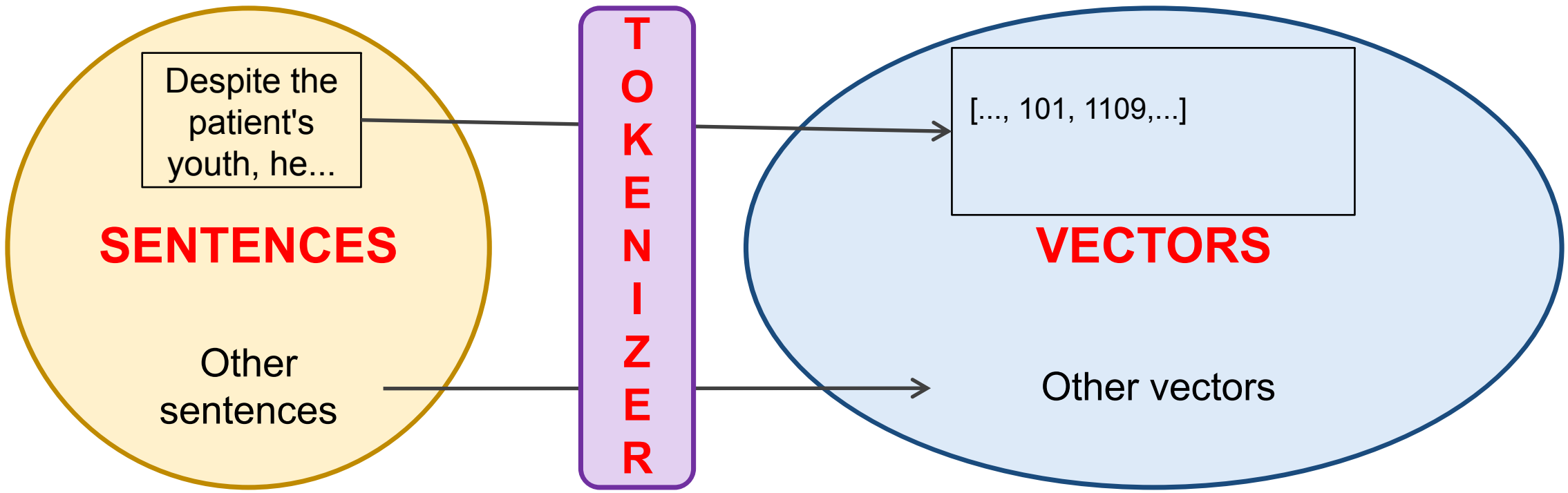
The main tool for processing textual data is a tokenizer. A tokenizer starts by splitting text into tokens according to a set of rules. The tokens are converted into numbers, which are used to build tensors as input to a model. Any additional inputs required by a model are also added by the tokenizer.

In our case, we get started quickly by loading a pretrained tokenizer with the AutoTokenizer class. This downloads the vocab used when a model is pretrained.

https://huggingface.co/docs/transformers/preprocessing

Tokenizer's Job





AutoTokenizer.tokenizer()

encoded_inputs = tokenizer("Input your sentence.")

Pass your sentence to the tokenizer:

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")

encoded_input = tokenizer("The patient tolerated this procedure well.")
print(encoded_input)
```

✓ 9.3s Python

```
{'input_ids': [101, 1109, 5351, 21073, 1181, 1142, 7791, 1218, 119, 102],
'token_type_ids': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 'attention_mask': [1, 1,
1, 1, 1, 1, 1, 1, 1, 1]}
```

https://huggingface.co/docs/transformers/preprocessing

AutoTokenizer.tokenizer()

encoded_inputs = tokenizer("Input your sentence.")

The output of tokenizer:

input_ids are the indices corresponding to each token in the sentence.

attention_mask indicates whether a token should be attended to or not.

token_type_ids identifies which sequence a token belongs to when there is more than one sequence.

https://huggingface.co/docs/transformers/preprocessing

tokenizer.decode()

encoded_inputs = tokenizer.decode(encoded_input["input_ids"])

You can decode the input_ids to return the original input:

```
tokenizer.decode(encoded_input["input_ids"])
✓ 0.6s                                                    Python

'[CLS] The patient tolerated this procedure well. [SEP]'
```

https://huggingface.co/docs/transformers/preprocessing

tokenizer.encode_plus()

How to pad, truncate, mask, build tensor with one function?

tokenizer.encode_plus()

```
inputs = tokenizer.encode_plus(
        "The patient tolerated this procedure well.",
        add_special_tokens=True,
        max_length=8, #   6 + [CLS] + [PAD]
        return_token_type_ids=True,
        padding="max_length",
        return_attention_mask=True,
        return_tensors='pt',
    )
    inputs
```

✓ 0.5s Python

```
{'input_ids': tensor([[  101,  1109,  5351, 21073,  1181,  1142,
7791,  1218,   119,   102]]), 'token_type_ids': tensor([[0, 0,
0, 0, 0, 0, 0, 0]]), 'attention_mask': tensor([[1, 1, 1,
1, 1, 1, 1, 1]])}
```

https://huggingface.co/docs/transformers/preprocessing

tokenizer.encode_plus()

Important Parameters:

text (str, List[str] or List[int]

text_pair (str, List[str] or List[int], optional) — Optional second sequence to be encoded.

add_special_tokens (bool, optional, defaults to True) — Whether or not to encode the sequences with the special tokens relative to their model.

padding (bool, str or PaddingStrategy, optional, defaults to False).

max_length: Pad to a maximum length specified with the argument max_length.

https://huggingface.co/docs/transformers/v4.20.1/en/internal/tokenization_utils#transformers.PreTrainedTokenizerBase.encode_plus

Other Tokenizers

Load any tokenizer based on the pre-trained model:

AutoTokenizer.from_pretrained("pretrained_model_name_or_path")

Parameters

- **pretrained_model_name_or_path** (`str` or `os.PathLike`) — Can be either:
  - A string, the *model id* of a pretrained model configuration hosted inside a model repo on huggingface.co. Valid model ids can be located at the root-level, like `bert-base-uncased`, or namespaced under a user or organization name, like `dbmdz/bert-base-german-cased`.
  - A path to a *directory* containing a configuration file saved using the save_pretrained() method, or the save_pretrained() method, e.g., `./my_model_directory/`.
  - A path or url to a saved configuration JSON *file*, e.g., `./my_model_directory/configuration.json`.

Tokenizers

Tokenizers class provides an implementation of today's most used tokenizers for NLP pre-processing. All tokenizers are built with the following functions:

- Train new vocabularies and tokenize, using today's most used tokenizers.
- Extremely fast (both training and tokenization), thanks to the Rust implementation.
- Easy to use, but also extremely versatile.
- Designed for research and production.
- Normalization comes with alignments tracking.
- Does all the pre-processing: Truncate, Pad, add the special tokens your model needs.

Build Model and Fine-Tuning

transformers.BertModel

How to load BERT model with specific type?

```
from transformers import BertTokenizer, BertModel
tokenizer = BertTokenizer.from_pretrained('bert-base-cased')
model = BertModel.from_pretrained("bert-base-cased")
```

In our case, we use the model type “`bert-base-cased`” as the demo. You can also try other model which pre-trained on the specific healthcare datasets. For example:

```
from transformers import BertTokenizer, BertModel
tokenizer = BertTokenizer.from_pretrained('biobert-v1.1')
model = BertModel.from_pretrained("biobert-v1.1")
```

https://huggingface.co/dmis-lab/biobert-v1.1

BertModel.config

How to check the info. of downloaded BERT?

BertModel.config

```
model.config
✓ 0.3s                                                    Python
BertConfig {
  "_name_or_path": "bert-base-cased",
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "transformers_version": "4.17.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 28996
}
```

https://huggingface.co/docs/transformers/model_doc/bert

BertModel.config

The Backbone of BERT shown in the BertModel.config:

- **vocab_size** (int, *optional*, defaults to 30522) — Vocabulary size of the BERT model. Defines the number of different tokens that can be represented by the inputs_ids passed when calling BertModel or TFBertModel.
- **hidden_size** (int, *optional*, defaults to 768) — Dimensionality of the encoder layers and the pooler layer.
- **num_hidden_layers** (int, *optional*, defaults to 12) — Number of hidden layers in the Transformer encoder.
- **num_attention_heads** (int, *optional*, defaults to 12) — Number of attention heads for each attention layer in the Transformer encoder.
- **intermediate_size** (int, *optional*, defaults to 3072) — Dimensionality of the "intermediate" (often named feed-forward) layer in the Transformer encoder.

https://huggingface.co/docs/transformers/model_doc/bert

Transformers.pipeline

How to directly apply the pre-trained model to the downstream tasks? Let's take a look at NER example.

Step1. Load tokenizer and model

```
tokenizer = AutoTokenizer.from_pretrained("dslim/bert-base-NER")
model = AutoModelForTokenClassification.from_pretrained("dslim/bert-base-NER")
```

Step2. Use the pipeline, input your tokenizer, model, and task description

```
nlp = pipeline("ner", model=model, tokenizer=tokenizer)
example = "He is presenting for revascularization ."
```

```
ner_results = nlp(text)
```

https://huggingface.co/docs/transformers/model_doc/bert

Transformers.pipeline

Output of NER Task.

```python
# from transformers import AutoTokenizer, AutoModelForTokenClassification
from transformers import pipeline

tokenizer = AutoTokenizer.from_pretrained("dslim/bert-base-NER")
model = AutoModelForTokenClassification.from_pretrained("dslim/bert-base-NER")

nlp = pipeline("ner", model=model, tokenizer=tokenizer)
example = "My name is Wolfgang and I live in Berlin"

ner_results = nlp(example)
print(ner_results)
```

Python

```
Downloading: 100%|██████████| 59.0/59.0 [00:00<00:00, 11.7kB/s]
Downloading: 100%|██████████| 829/829 [00:00<00:00, 205kB/s]
Downloading: 100%|██████████| 208k/208k [00:01<00:00, 177kB/s]
Downloading: 100%|██████████| 2.00/2.00 [00:00<00:00, 654B/s]
Downloading: 100%|██████████| 112/112 [00:00<00:00, 32.2kB/s]
Downloading: 100%|██████████| 413M/413M [07:35<00:00, 950kB/s]

[{'entity': 'B-PER', 'score': 0.9990139, 'index': 4, 'word': 'Wolfgang', 'start': 11, 'end': 19}, {'entity': 'B-
LOC', 'score': 0.999645, 'index': 9, 'word': 'Berlin', 'start': 34, 'end': 40}]
```

https://huggingface.co/docs/transformers/model_doc/bert

Transformers.pipeline

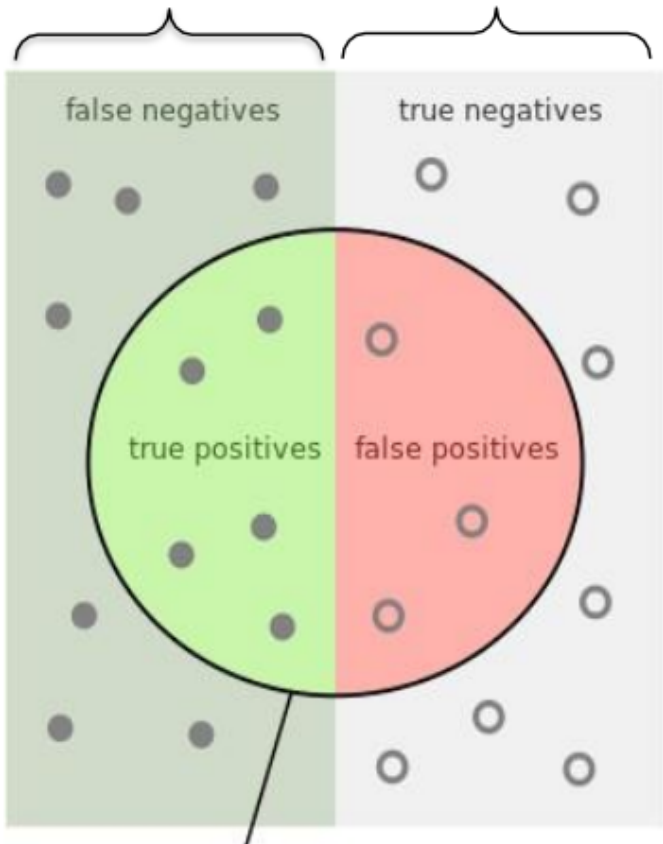
Visualization of NER Task.

https://huggingface.co/docs/transformers/model_doc/bert

Evaluation Metrics

datasets.load_metric

- Precision and recall
  - True positives(TP): The ‘Disease’s that the model detected as ‘Disease’
  - False positives(FP): The NOT ‘Disease’s that the model detected as ‘Disease’
  - False negatives(FN): The ‘Disease’s that the model detected as NOT ‘Disease’
  - True negatives(TN): The ‘NOT Disease’s that the model detected as NOT ‘Disease’



https://huggingface.co/docs/transformers/model_doc/bert

$$\text{Precision} = \frac{\text{Detected as ‘PERSON’ correctly}}{\text{Total number of detected as ‘PERSON’}}$$

$$\text{Recall} = \frac{\text{Detected as ‘PERSON’ correctly}}{\text{Total number of actual ‘PERSON’ entities}}$$

datasets.load_metric

- Precision and recall
  - True positives(TP): The ‘Disease’s that the model detected as ‘Disease’
  - False positives(FP): The NOT ‘Disease’s that the model detected as ‘Disease’
  - False negatives(FN): The ‘Disease’s that the model detected as NOT ‘Disease’
  - True negatives(TN): The ‘NOT Disease’s that the model detected as NOT ‘Disease’

GT： cat

label cat cat dog

GT： dog

label dog dog dog

Precision of cat: 100% **Recall** of cat: 66.7%

Precision of dog: 75% **Recall** of dog: 100%

datasets.load_metric

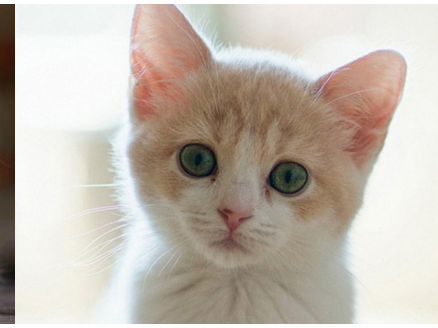
- Precision and recall
  - True positives(TP): The ‘Disease’s that the model detected as ‘Disease’
  - False positives(FP): The NOT ‘Disease’s that the model detected as ‘Disease’
  - False negatives(FN): The ‘Disease’s that the model detected as NOT ‘Disease’
  - True negatives(TN): The ‘NOT Disease’s that the model detected as NOT ‘Disease’

For class ‘cat’, the examples are:

GT： cat

label

cat
TP



cat
TP

dog
FN

GT： dog

label

dog
TN

dog
TN

dog
TN

Precision: 2/2 = 100%
Recall: 2/3 = 66.7%

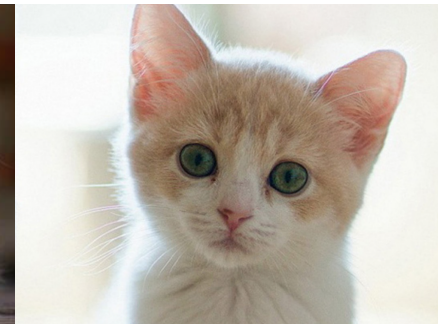
datasets.load_metric

- Precision and recall
  - True positives(TP): The ‘Disease’s that the model detected as ‘Disease’
  - False positives(FP): The NOT ‘Disease’s that the model detected as ‘Disease’
  - False negatives(FN): The ‘Disease’s that the model detected as NOT ‘Disease’
  - True negatives(TN): The ‘NOT Disease’s that the model detected as NOT ‘Disease’

For class ‘dog’, the examples are:

GT： cat

label: cat **TN**



label: cat **TN**

label: dog **FP**

GT： dog

label: dog **TP**

label: dog **TP**

label: dog **TP**

Precision: 3/4 = 75%
Recall: 3/3 = 100%

datasets.load_metric

```
from datasets import load_metric

metric_p = load_metric("precision")
#metric_r = load_metric("recall")

reference_sentence = "The patient tolerated this procedure well."
predicted_sentence = "The patient will tolerate this procedure."

metric_p.compute(predictions = predicted_sentence, references = reference_sentence)
```

https://huggingface.co/docs/datasets/v1.12.0/metrics.html

datasets.load_metric

- BLEU

BLEU (Bilingual Evaluation Understudy) is an algorithm for **evaluating** the **quality** of text which has been **machine-translated** from one natural language to another.

Quality is considered to be the **correspondence between** a **machine**'s output **and** that of a **human**.

"The closer a machine translation is to a professional human translation, the better it is" – this is the central idea behind BLEU.

BLEU was one of the first metrics to claim a high correlation with human judgements of quality, and remains one of the most popular automated and inexpensive metrics.

https://huggingface.co/docs/transformers/model_doc/bert

datasets.load_metric

```
from datasets import load_metric

metric = load_metric("bleu")

sentence1 = "The patient tolerated this procedure well."
sentence2 = "The patient will tolerate this procedure.”

metric.compute(predictions=sentence2, references=sentence1)
```

https://huggingface.co/docs/transformers/model_doc/bert