Challenge 3: Predict Reactant Molecules

BENJAMIN POMMER

01525693

Overview

- Task description
- Categorization
- Model of choice: ReactionT5v2
- Loading of model and tokenizer
- Loading of the tokenizer function/data collector
- Seq2SeqTraining

Task description

- The goal of this challenge is to predict reactant molecules given product molecules.

  You are given a dataset of chemical reactions "[Reactants' SMILES]>>[Products' SMILES]" and you should train a machine learning model to predict the reactant SMILES starting from a new product SMILES*.

  *For simplicity, we don't care about atom mappings.

Catorization

- Template-based methods
  - Expert-based Knowledge
  - Classification
  - Retrieval
- **Template-free methods**
  - **(Sequence to Sequence) Translation**

Template … abstract dictionary for reaction rules

Model of choice: ReactionT5v2

- Based on Transformer T5
- Pretrained model with Open Reaction Database
- Possible applications
  - Forward synthesis
  - **Retrosynthesis**
  - Yield
- Model name for retrosynthesis: sagawa/ReactionT5v2-retrosynthesis
- Fine-Tuning with given dataset

Loading of model and tokenizer

```python
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

tokenizer = AutoTokenizer.from_pretrained("sagawa/ReactionT5v2-retrosynthesis")
model = AutoModelForSeq2SeqLM.from_pretrained("sagawa/ReactionT5v2-retrosynthesis")

inp = tokenizer('CCN(CC)CCNC(=S)NC1CCCc2cc(C)cnc21', return_tensors='pt')
output = model.generate(**inp, num_beams=1, num_return_sequences=1, return_dict_in_generate=True, output_scores=True)
output = tokenizer.decode(output['sequences'][0], skip_special_tokens=True).replace(' ', '').rstrip('.')
output # 'CCN(CC)CCN=C=S.Cc1cnc2c(c1)CCCC2N'
```

Loading of the tokenizer function/data collector

```
def tokenize_function(examples):
    model_inputs = tokenizer(examples["PRODUCT"], max_length=256, truncation=True, padding="max_length")
    labels = tokenizer(examples["REACTANT"], max_length=256, truncation=True, padding="max_length")
    model_inputs["labels"] = labels["input_ids"]
    return model_inputs

tokenized_datasets = dataset.map(tokenize_function, batched=True)
```

```
from transformers import DataCollatorForSeq2Seq

data_collator = DataCollatorForSeq2Seq(tokenizer=tokenizer, model=model)
```

Seq2SeqTraining

```python
from transformers import Seq2SeqTrainer, Seq2SeqTrainingArguments

training_args = Seq2SeqTrainingArguments(
    output_dir="./results",
    num_train_epochs=10,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    eval_strategy="epoch",
    save_strategy="epoch",
    logging_dir="./logs",
    report_to=[],
    predict_with_generate=True,
    fp16=True
)

trainer = Seq2SeqTrainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"],
    tokenizer=tokenizer,
    data_collator=data_collator
)
```