JSONFormer

JSONFormer is a library that wraps local HuggingFace pipeline models for structured decoding of a subset of the JSON Schema.

It works by filling in the structure tokens and then sampling the content tokens from the model.

Warning - this module is still experimental

```
pip install --upgrade jsonformer > /dev/null
```

HuggingFace Baseline

First, let's establish a qualitative baseline by checking the output of the model without structured decoding.

```
import logging

logging.basicConfig(level=logging.ERROR)
```

```
from typing import Optional
from langchain.tools import tool
import os
import json
import requests
```

```
HF_TOKEN = os.environ.get("HUGGINGFACE_API_KEY")


@tool
def ask_star_coder(query: str, temperature: float = 1.0, max_new_tokens: float = 250):
    """Query the BigCode StarCoder model about coding questions."""
    url = "https://api-inference.huggingface.co/models/bigcode/starcoder"
    headers = {
        "Authorization": f"Bearer {HF_TOKEN}",
        "content-type": "application/json",
    }
    payload = {
        "inputs": f"{query}\n\nAnswer:",
        "temperature": temperature,
        "max_new_tokens": int(max_new_tokens),
    }
    response = requests.post(url, headers=headers, data=json.dumps(payload))
    response.raise_for_status()
    return json.loads(response.content.decode("utf-8"))
```

```
prompt = """You must respond using JSON format, with a single action and single action input.
You may 'ask_star_coder' for help on coding problems.

{arg_schema}

EXAMPLES
----
Human: "So what's all this about a GIL?"
AI Assistant:{{
  "action": "ask_star_coder",
  "action_input": {{"query": "What is a GIL?", "temperature": 0.0, "max_new_tokens": 100}}"
}}
```

```
Observation: "The GIL is python's Global Interpreter Lock"
Human: "Could you please write a calculator program in LISP?"
AI Assistant:{{
  "action": "ask_star_coder",
  "action_input": {{"query": "Write a calculator program in LISP", "temperature": 0.0, "max_new_tokens": 
250}}
}}
Observation: "(defun add (x y) (+ x y))\n(defun sub (x y) (- x y ))"
Human: "What's the difference between an SVM and an LLM?"
AI Assistant:{{
  "action": "ask_star_coder",
  "action_input": {{"query": "What's the difference between SGD and an SVM?", "temperature": 1.0, 
"max_new_tokens": 250}}
}}
Observation: "SGD stands for stochastic gradient descent, while an SVM is a Support Vector Machine."

BEGIN! Answer the Human's question as best as you are able.
------
Human: 'What's the difference between an iterator and an iterable?'
AI Assistant:""".format(
    arg_schema=ask_star_coder.args
)
```

```
from transformers import pipeline
from langchain.llms import HuggingFacePipeline

hf_model = pipeline(
    "text-generation", model="cerebras/Cerebras-GPT-590M", max_new_tokens=200
)

original_model = HuggingFacePipeline(pipeline=hf_model)
```

```
generated = original_model.predict(prompt, stop=["Observation:", "Human:"])
print(generated)
```

```
    Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.


     'What's the difference between an iterator and an iterable?'
```

That's not so impressive, is it? It didn't follow the JSON format at all! Let's try with the structured decoder.

JSONFormer LLM Wrapper

Let's try that again, now providing a the Action input's JSON Schema to the model.

```
decoder_schema = {
    "title": "Decoding Schema",
    "type": "object",
    "properties": {
        "action": {"type": "string", "default": ask_star_coder.name},
        "action_input": {
            "type": "object",
            "properties": ask_star_coder.args,
        },
    },
}
```

```python
from langchain.experimental.llms import JsonFormer

json_former = JsonFormer(json_schema=decoder_schema, pipeline=hf_model)
```

```python
results = json_former.predict(prompt, stop=["Observation:", "Human:"])
print(results)
```

```
    {"action": "ask_star_coder", "action_input": {"query": "What's the difference between an iterator and an iter", "temperature": 0.0, "max_new_tokens": 50.0}}
```

Voila! Free of parsing errors.