Solution without a Dataset

```
import time

system = "Classify the text below as pro-Brexit or anti-Brexit. The answer
should be 'pro' or 'anti' depending on the stance.\n"

fewshot = """Below are a number of examples that show how this
classification task works.
Text: Brexit is bad. Immigrants make Britain great. They also cooked your
food The London restaurant causing a stir with anti-#Brexit messages on
your bill
Answer: 'anti'
Text: Britain's exit is a huge blow to the dream of a united Europe No. It
is the end of an anti-national, centralized, globalist, neoliberal and
authoritarian system and organism like the EU and its hegemonic power over
Europe.
Answer: 'pro'
"""

#fewshot = ""

pipe = pipeline(
   "text-generation",
   model=model,
   tokenizer=tokenizer,
   pad_token_id=tokenizer.eos_token_id,
   return_full_text=False,
   max_new_tokens=1, # We only need to generate 'pro' or 'anti'.
)

n_instances = 100
n_correct = 0

t0 = time.time()
for i, label, text in test_data.itertuples():

 prompt = f"<s>[INST] {system}\n{fewshot}\nText:\n{text}\n[/INST]
\nAnswer: '"
```

```
 pipe_output = pipe(prompt)[0]['generated_text']

 print(i, label, pipe_output, text)

 if pipe_output == label:
   n_correct += 1

 if i == n_instances-1:
   break
t1 = time.time()

print(f'{n_correct}/{n_instances} = {n_correct/n_instances:.4f}, time =
{t1-t0:.2f}')
```

Solution with a Dataset

```
fewshot = """Below are a number of examples that show how this
classification task works.
Text: Brexit is bad. Immigrants make Britain great. They also cooked your
food The London restaurant causing a stir with anti-#Brexit messages on
your bill
Answer: 'anti'
Text: Britain's exit is a huge blow to the dream of a united Europe No. It
is the end of an anti-national, centralized, globalist, neoliberal and
authoritarian system and organism like the EU and its hegemonic power over
Europe.
Answer: 'pro'
"""

n_instances = 100

import datasets
from tqdm import tqdm
from transformers.pipelines.pt_utils import KeyDataset

instance_prompts = {'text': [f"<s>[INST]
{system}\n{fewshot}\nText:\n{text}\n[/INST] \nAnswer: '" for text in
test_data.text[:n_instances]]}
test_dataset = datasets.Dataset.from_dict(instance_prompts)
```

```python
pipe = pipeline(
   "text-generation",
   model=model,
   tokenizer=tokenizer,
   return_full_text=False,
   pad_token_id=tokenizer.eos_token_id,
   do_sample=False,
   max_new_tokens=1,
   batch_size=8, # This is useful to make things a bit faster.
)

pipe.tokenizer.pad_token_id = model.config.eos_token_id
pipe_output = pipe(KeyDataset(test_dataset, 'text'))

n_correct = 0

t0 = time.time()
for res, label in tqdm(zip(pipe_output, test_data.label)):
 if res[0]['generated_text'] == label:
   n_correct += 1

t1 = time.time()

print(f'\n{n_correct}/{n_instances} = {n_correct/n_instances:.4f}, time =
{t1-t0:.2f}')
```