CSC401/2511 Assignment 1 Tutorial 1

Winston (Yuntao) Wu
University of Toronto

Overview

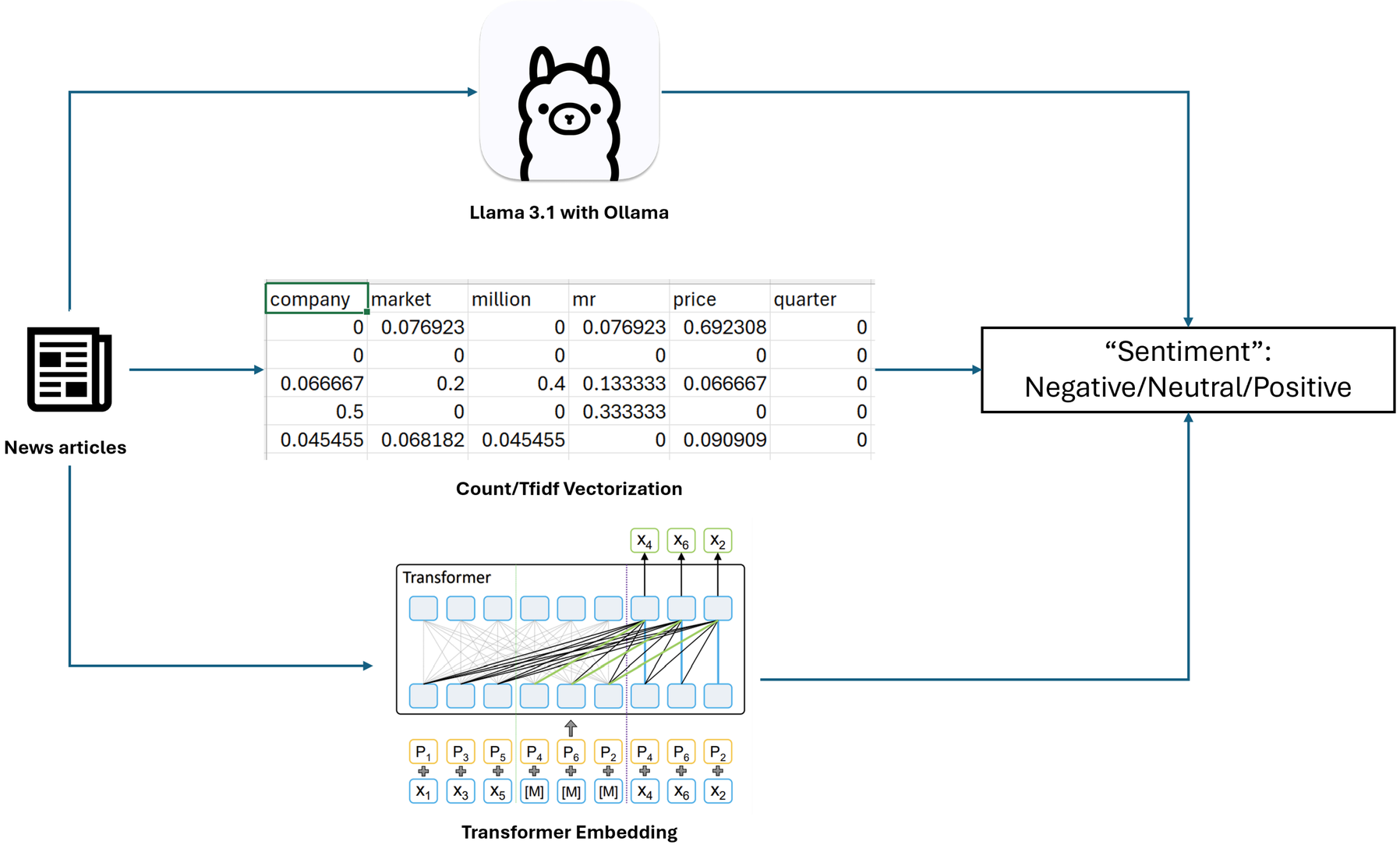


Image Source1 Image Source2

Dataset (FPB)

Financial Phrase Bank:

- Polar sentiment dataset of sentences from financial news
- 4k sentences from English language financial news rated by 5-8 annotators
- The dataset can be accessed from `/u/cs401/A1/data/fpb_dataset.parquet`

	text	label	label_numeric
0	According to Gran , the company has no plans t...	neutral	0
1	Technopolis plans to develop in stages an area...	neutral	0
2	The international electronic industry company ...	negative	-1
3	With the new production plant the company woul...	positive	1
4	According to the company 's updated strategy f...	positive	1

Dataset (WSJ89)

Wall Street Journal (1989):

- 2k articles from 1989 Wall Street Journal from the English Penn Treebank corpus. Dataset is at `/u/cs401/A1/data/wsj.gz`
- We tag each article by the sign of 5-day log return of S&P500 on previous day ($\log \frac{p_{t-1}}{p_{t-6}}$). Tags are save in `/u/cs401/A1/data/wsj89_labels.parquet`
- You do not need to worry about WSJ89 in first part.

```
wsj
├── 00
│   ├── wsj_0001.gz
│   └── wsj_0002.gz
│   ...
├── 01
│   ├── wsj_0100.gz
│   ├── wsj_0101.gz
│   ...
...
```

`wsj.gz` File Structure

```
.START

After a bad start, Treasury bonds were buoyed
by a late burst of buying to end modestly
higher.

"The market was pretty dull" for most of the
day, said Robert H. Chandross, vice president
at Lloyds Bank PLC.
He said some investors were reluctant to plunge
into the market ahead of several key economic
indicators due this week, especially Friday's
potentially market-moving employment report.
```

Sample article

	fn	label_numeric
0	wsj_0001.gz	-1.0
1	wsj_0002.gz	-1.0
2	wsj_0003.gz	-1.0
3	wsj_0004.gz	1.0
4	wsj_0005.gz	-1.0

`wsj89_labels.parquet`

Part 1 Llama 3.1 Analysis (due Sept 24)

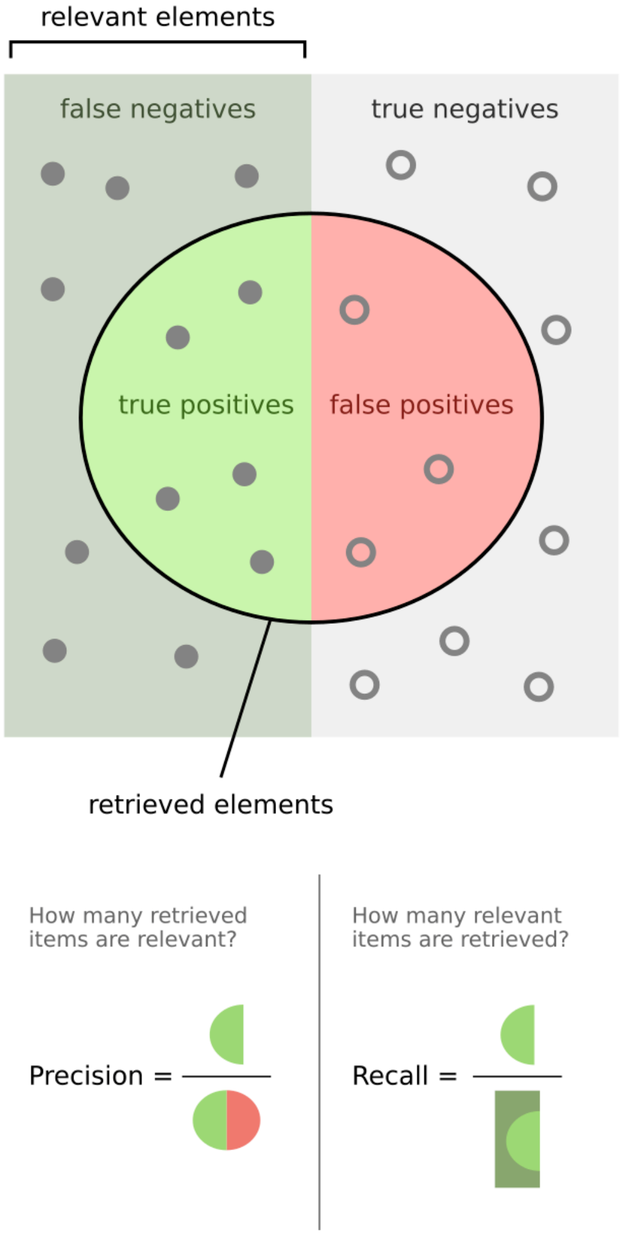
In this part, you are tasked with sentiment analysis using Llama 3.1. Submit on Markus (A1p1).

Deliverables:

- `a1_utils.py`: Functions for computing metrics (accuracy, precision, recall)
- `a1_llama3.py`: Select articles from FPB, query Llama 3.1, parse the response to get the sentiment scores, and compute metrics
- `a1_part1.txt` [Due Oct 8th with the full assignment]: Save the accuracy, precision, recall rate by the provided format string. Comment on the performance of Llama3.1, based on the accuracy, precision, recall rate, and samples of the generated explanation (at least one for correct and at least one for incorrect, with your comments)

Part 1: Metrics

- Accuracy: The total number of correctly classified instances over all classifications: $A = \frac{\sum_i c_{i,i}}{\sum_{i,j} c_{i,j}}$
- Recall: For each class κ, the fraction of cases that are truly class κ that were classified as κ, $R(\kappa) = \frac{c_{\kappa,\kappa}}{\sum_j c_{\kappa,j}}$.
- Precision: For each class κ, the fraction of cases classified as κ that truly are κ, $P(\kappa) = \frac{c_{\kappa,\kappa}}{\sum_i c_{i,\kappa}}$.
- You can use `sanity_check.py` to check your implementation.



Part 1: Llama 3.1

We have deployed a Llama 3.1 8B Instruct model on teach.cs server. The model is loaded with 4-bit quantization using Ollama.

The following is a replication code for running Llama 3.1 using pure huggingface transformer library:

```
# Load model
model_id = "./models/Meta-Llama-3.1-8B-Instruct"
tokenizer = AutoTokenizer.from_pretrained(model_id, trust_remote_code=True)
model = AutoModelForCausalLM.from_pretrained(
    model_id,
    quantization_config=BitsAndBytesConfig(load_in_4bit=True),
    device_map="auto",
    trust_remote_code=True
).eval()

# Tokenize chat input
messages = [
    {"role": "system", "content": "You will compute the sentiment score of articles. Start with the classification, capitalized. Ther
    {"role": "user", "content": input_text},
]
input_ids = tokenizer.apply_chat_template(
    messages,
    add_generation_prompt=True,
```

You don't have to worry too much about the parameter settings. You will learn more about `top_k` and `temperature` in this class.

If you want to change any parameter or the system prompt, you need to specify what you changed, and the results before/after the change in your written analysis (`a1_part1.txt` and `a1_compare.txt`)

Part 1: Server Setup

The Llama 3.1 is setup as a 4-bit quantized model on teach.cs server through Ollama.

It is wrapped by a queue-based (FIFO) request processing system that checks your user id. The logic of `send_request(user_id, input_text)` is as follows:

- Submit request to `http://csc401:8000/submit_request_llama3`. This will give you a `task_id` that's put onto the queue.

```
messages = [
    {"role": "system", "content": "You will compute the sentiment score of articles. Start with the classification, capitalized. Ther
    {"role": "user", "content": input_text},
]
req_body = {
    "user_id": user_id,
    "messages": messages,
    "max_new_tokens": 256,
    "temperature": 0.5,
    "top_p": 0.95,
    "top_k": 10,
}
res = requests.post("http://csc401:8000/submit_request_llama3", json=req_body, headers={"Content-Type": "application/json"})
```

- Then use this `task_id` to check your request status through:

```
res = requests.get(f"http://csc401:8000/check_request_result?user_id={user_id}&task_id={task_id}", headers={"Content-Type": "applicat
```

Part 1: Requesting the Server

Once the request is processed, the expected output by calling `send_request` in `a1_llama3.py` is:

```
{
    "model": "registry.ollama.ai/library/llama3:latest",
    "created_at": "2023-12-12T14:13:43.416799Z",
    "message": {
        "role": "assistant",
        "content": response
    },
    "done": true,
    "total_duration": time spent in nanoseconds generating the response,
    "load_duration": time spent in nanoseconds loading the model,
    "prompt_eval_count": number of tokens in the prompt,
    "prompt_eval_duration": time spent in nanoseconds evaluating the prompt,
    "eval_count": number of tokens in the response,
    "eval_duration": time in nanoseconds spent generating the response,
    "status": "done"
}
```

You need to parse the response by `parse_response` in `a1_llama3.py` to the following form:

```
{
    "label": extracted text label (POSTIVE, NEUTRAL, NEGATIVE) from the response,
    "label_numeric": 1/0/-1 for positive, neutral, negative,
    "raw_result": ["message"]["content"],
    "compute_time": total_duration in second
}
```

Part 1: Process Dataframe

You need to write the code for `process_df` in `a1_llama3.py`:

- Use `np.random.choice(len(df), sample_num, replace=False)`, with the specific seed we set using your UTORid
- Get the text, and label at the specific indices, submit the request using `send_request`. The request may not always be successful. Error handling is expected (e.g. resend the request, or save a failed value and try later).
- Parse the response using `parse_response`. You can assume that at least one of `POSTIVE, NEUTRAL, NEGATIVE` exist in Llama3.1 response. However, you can also manually check the response and edit in the result dataframe before computing the metrics.
- Any accuracy rate is expected, based on your student number. We are not looking for the best accuracy.
- You are expected to provide some examples where Llama predicts correctly and where Llama predicts incorrectly, with your comment (at least one for correct and at least one for incorrect)

Part 1: Submission

You need to submit the following files by Sept 24, which is 7/10 for part 1. If you do not submit them on time, they will not be graded in the final submission.

- `a1_utils.py`
- `a1_llama3.py`

You should resubmit the above two files and the following file for part 1 when the assignment is due on Oct 8.

- `a1_part1.txt`: accuracy, precision, recall rate by the provided format string. Comment on the performance of Llama3.1, based on the accuracy, precision, recall rate, and samples of the generated explanation (at least one for correct and at least one for incorrect, with your comments)

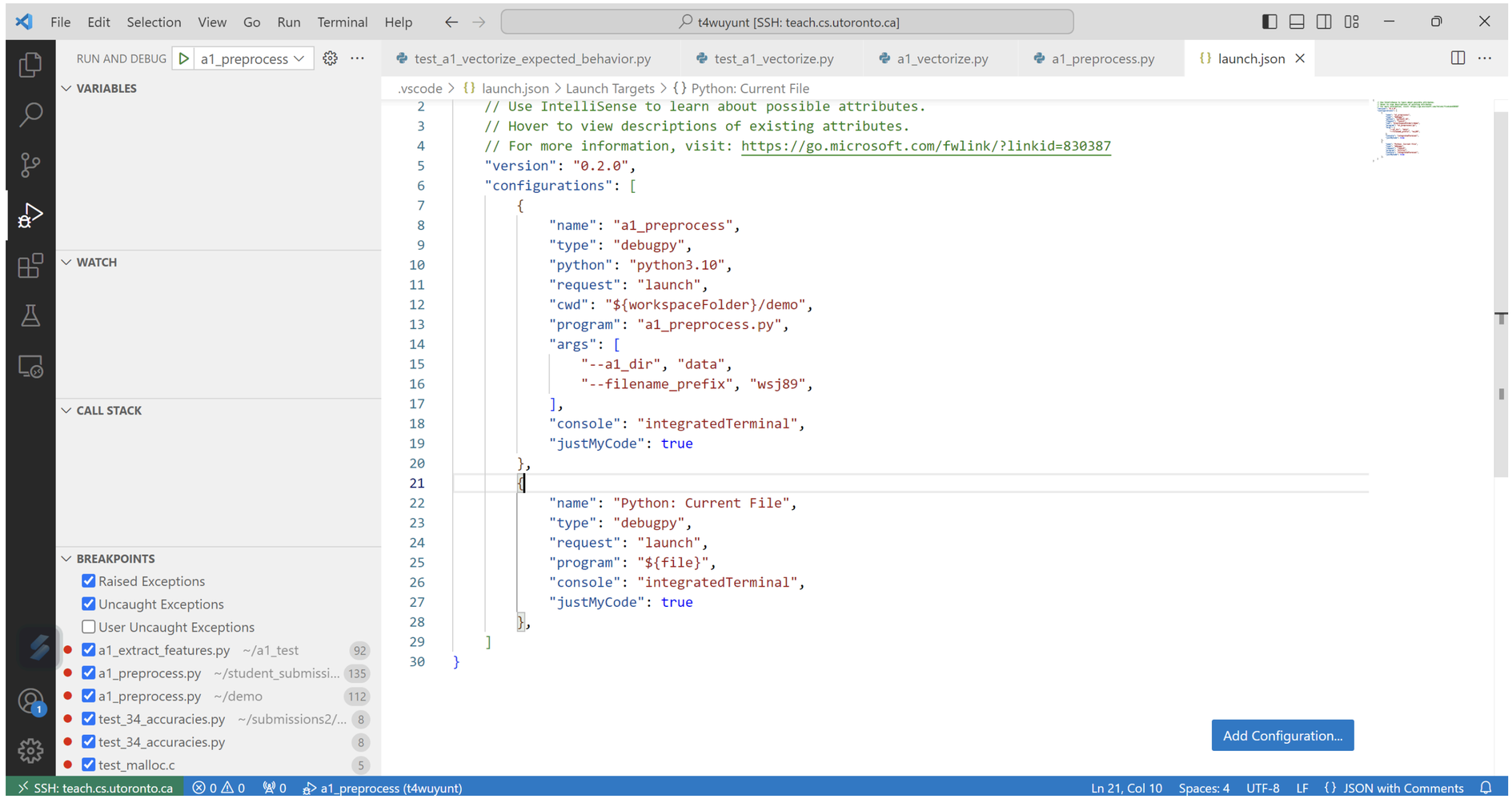
Part 1: Notes

- Computing sentiment scores using Llama 3.1 takes around 4 seconds per article. The server operates on a queuing system, so you may experience delays. Cache the results in a dataframe and save them locally to avoid repeated endpoint calls.
- Only one Llama 3.1 server is available for CSC401/2511 and CSC485. Start early to avoid long wait times in the request queue.
- Use the provided server exclusively for assignments, not personal projects. Each UTORid will be rate-limited.
- Use your own UTORid in your requests. Only requests from registered UTORids will be processed. We will track the total number of requests per UTORid as partial validation of your work.
- Llama 3.1 request may fail the first time, but should work properly afterwards. If Llama 3.1 request constantly fails, firstly try to increase the initial wait time `time.sleep(2)` to a longer time. If it does not help, contact the instructors on Piazza.
- Use `python3.10` instead of `python3` for consistent versioning with later parts.

VS Code Debugging

Install the Python extension for VSCode if you have not yet.

In Debug panel, choose `create a launch.json file' with Debug configuration set to Python File.



Debug and Watch Function

You can add breakpoints and start the debug function as highlighted in the image. Be sure to select a proper python interpreter (3.10.13 for A1) before you start.

Watch function helps you evaluate any expressions as you debug