Business Analytics: Sentiment Classification

Introduction

For this session, you will need access to a GPU to run the experiments efficiently. If you do not have a local GPU available, you can use an online environment such as **Google Colab**, which provides free GPU access for educational purposes.

You will work with Python notebooks that allow you to experiment interactively with both classical machine learning methods and large language models (LLMs). Make sure your environment supports installing Python packages and running GPU computations.

This session is divided into two parts:

- Part 1: Sentiment classification using a Support Vector Machine (SVM)
- Part 2: Sentiment classification using a Large Language Model (LLM)

Part 1 — Classification with an SVM

1.1 Loading and inspecting the data

Listing 1: Load the dataset

```
from datasets import load_dataset

train_url = "https://huggingface.co/datasets/stanfordnlp/sentiment140/resolve/refs%2
    ↪ Fconvert%2Fparquet/sentiment140/train/0000.parquet"
test_url = "https://huggingface.co/datasets/stanfordnlp/sentiment140/resolve/refs%2
    ↪ Fconvert%2Fparquet/sentiment140/test/0000.parquet"

dataset = load_dataset(
    "parquet",
    data_files={
        "train": train_url,
        "test": test_url
    }
)

train_dataset = dataset["train"]
test_dataset = dataset["test"]
```

a) What are the classes in this dataset?

b) How many samples are there for each class in the training set?

c) How many samples are there for each class in the test set?

1.2 Training an SVM

a) What do we need to train an SVM?

b) In our case, the data consists of text. How can we convert text into a format suitable for an SVM?

1.3 Evaluating the model

a) Compute the confusion matrix, recall, precision (for both positive and negative classes), and overall accuracy.

b) How can these metrics be interpreted?

c) Comment on the model's performance.

1.4 Comparison with a baseline model

Train a model that always predicts the majority class.
Hint: check the class `DummyClassifier` in `scikit-learn`.

a) Compute again the confusion matrix and performance metrics.

b) Compare the results with those from the SVM. What can you conclude?

Part 2 — Classification with a Large Language Model (LLM)

2.1 Loading the model

Choose one of the following models:

- `meta-llama/Llama-3.2-3B-Instruct`
- `amd/Instella-3B-Instruct`
- `mistralai/Mistral-7B-Instruct-v0.1`

Listing 2: Load the LLM model

```
from transformers import AutoTokenizer, AutoModelForCausalLM

model_name = [MODEL_NAME]

tokenizer = AutoTokenizer.from_pretrained(model_name)

model = AutoModelForCausalLM.from_pretrained(
    model_name,
    device_map="auto",
    dtype=torch.float32,
    trust_remote_code=True
)
```

a) Try to load the model in **FP32** and **FP16**. What happens? Why?

2.2 Quantization

Listing 3: Quantization example

```
from transformers import BitsAndBytesConfig

bnb_config = BitsAndBytesConfig(
    load_in_8bit=True
)

# bnb_config = BitsAndBytesConfig(
# load_in_4bit=True,
# bnb_4bit_use_double_quant=True,
# bnb_4bit_quant_type="nf4",
# bnb_4bit_compute_dtype=torch.float16
# )

model = AutoModelForCausalLM.from_pretrained(
    model_name,
    device_map="auto",
    quantization_config=bnb_config,
    trust_remote_code=True
)
```

a) Try loading the model in **INT8** and **INT4**. What happens?

b) Explain in your own words what quantization is and why it is useful.

2.3 Using the LLM for simple prompting

Listing 4: Example of simple prompting

```
prompt = "What is the capital of Belgium?"

pipe = pipeline("text-generation", tokenizer=tokenizer, model=model, dtype="auto",
    ↪ device_map="auto", pad_token_id=tokenizer.eos_token_id, return_full_text=False)

answer = pipe(prompt, max_new_tokens=500, eos_token_id=tokenizer.eos_token_id)
```

Test simple questions such as:

- What is the capital of Belgium?
- What is the capital of France?

a) What are the answers? Are they always the same?

b) Are you satisfied with the answers? Why or why not?

c) Try at least three different questions. Write down each full prompt and the corresponding answer.

2.4 Using the LLM for sentiment classification

We now want to use the LLM to classify the sentiment of our dataset directly.

a) Implement a loop that asks the LLM to classify each sample, with error handling and retries if needed.

b) Compute the confusion matrix and the performance metrics.

c) Compare the results with those obtained using the SVM.

2.5 Discussion

a) Which model did you choose among the three proposed? Why?

b) Compare the metrics with those from Part 1. How do they differ?

c) From a business perspective, what are the advantages and drawbacks of each approach?