# 07_llm_feature

November 20, 2025

## 1 07 · Feature Score de LLM Local Qwen

Me quedé pensando, esta clasificación suena como una tarea que haría muy bien un LLM. Medio que ya hacemos eso con los embeddings, pero capaz que puede aportar algo la feature del LLM score. Probemos!

[2]:
```
import json
import os
import time
from pathlib import Path

import pandas as pd
import requests

try:
    from tqdm.auto import tqdm
except ImportError:
    def tqdm(iterable, **kwargs):
        return iterable
```

[3]:
```
REPO_ROOT = Path.cwd()
if not (REPO_ROOT / 'data').exists():
    REPO_ROOT = REPO_ROOT.parent

DATA_DIR = REPO_ROOT / 'data' / 'nlp-getting-started'
TRAIN_PATH = DATA_DIR / 'train.csv'
TEST_PATH = DATA_DIR / 'test.csv'

OUTPUT_DIR = REPO_ROOT / 'resultados' / 'llm_features'
OUTPUT_DIR.mkdir(parents=True, exist_ok=True)

train_df = pd.read_csv(TRAIN_PATH)
test_df = pd.read_csv(TEST_PATH)
```

### 1.1 Prompt usado:

[4]:
```
PROMPT_TEMPLATE = """You are a classifier for a Kaggle competition.

Task:
Given the text of a tweet, estimate how likely it is that the tweet describes a␣
 ↪REAL disaster
(real fire, flood, earthquake, accident, explosion, etc.) rather than a␣
 ↪metaphor, joke, or
unrelated content.

Respond with a single number between 0 and 1 (inclusive):
- 0 means "definitely NOT about a real disaster"
- 1 means "definitely about a real disaster"

Examples:
Tweet: "Forest fire near La Ronge Sask. Canada"
Answer: 0.82

Tweet: "My phone is on fire from all these messages lol"
Answer: 0.03

Tweet: "Car accident on the highway blocking both lanes, emergency services on␣
 ↪scene"
Answer: 0.91

Tweet: "This party is so lit I'm dying  "
Answer: 0.01

Output ONLY the number, with no explanation.

Tweet:
{text}

Answer:
"""
```

### 1.2 Parámetros del prompt y Ollama

Me parece que lo mejor para esto es temp 0. Mi idea es que no se ponga muy creativo. Mismo concepto con top p. Repeat penalty lo dejo en ese default, y num predict bajo para que no se extienda mucho, quiero solo el score.

[5]:
```
import re
import requests

OLLAMA_BASE_URL = os.environ.get('OLLAMA_URL', 'http://localhost:11434')
OLLAMA_MODEL = os.environ.get('OLLAMA_MODEL', 'qwen2.5:7b-instruct')
```

```
OLLAMA_OPTIONS = {
    'temperature': 0.0,
    'top_p': 0.0,
    'repeat_penalty': 1.0,
    'num_predict': 8,
}

FLOAT_RE = re.compile(r'([01](?:\.\d+)?|\d?\.\d+)')

def llm_disaster_score(text: str) -> float:
    prompt = PROMPT_TEMPLATE.format(text=text)

    payload = {
        "model": OLLAMA_MODEL,
        "prompt": prompt,
        "stream": False,
        "options": OLLAMA_OPTIONS,
    }

    r = requests.post(f"{OLLAMA_BASE_URL}/api/generate", json=payload)
    r.raise_for_status()

    resp = r.json()["response"].strip()
    m = FLOAT_RE.search(resp)
    if not m:
        return 0.5

    try:
        val = float(m.group(1).strip('"'))
    except:
        val = 0.5

    return max(0.0, min(1.0, val))
```

Probemos el prompt con unos ejemplos:

[6]:
```
print(llm_disaster_score("Forest fire spotted in California. Evacuations␣
↪starting."))
```

```
0.95
```

Buenísimo, me gustó el score para ese tweet

[8]:
```
print(llm_disaster_score("Lol last night's the strokes concert was so fucking␣
↪lit, OMFG"))
```

```
0.01
```

Jajajaja le doy un aprobado personalmente. Ya soy un prompt engineer (? Bueno pasemosle todo el dataset entonces y guardemos el output en un JSON.

[9]:
```python
import json
from tqdm import tqdm
import math

def compute_split_scores(df: pd.DataFrame, split_name: str, output_dir: Path):
    scores = {}
    for _, row in tqdm(df.iterrows(), total=len(df), desc=f"LLM scores␣
 ↪{split_name}"):
        tweet_id = int(row["id"])
        text = row.get("text", "")
        try:
            score = llm_disaster_score(text if isinstance(text, str) else "")
        except Exception as e:
            print(f"[WARN] Error scoring id={tweet_id}: {e}")
            score = 0.5  # Si no anduvo le pongo 0.5, indefinido.

        if not isinstance(score, (int, float)) or not math.isfinite(score):
            score = 0.5
        score = max(0.0, min(1.0, float(score)))

        scores[str(tweet_id)] = score

    out_path = output_dir / f"llm_scores_{split_name}.json"
    with open(out_path, "w", encoding="utf-8") as f:
        json.dump(scores, f, ensure_ascii=False, indent=2)

    print(f"Saved {len(scores)} scores to {out_path}")


# === Ejecutar para train y test ===
compute_split_scores(train_df, "train", OUTPUT_DIR)
compute_split_scores(test_df, "test", OUTPUT_DIR)
```

```
LLM scores train: 100%|      | 7613/7613 [17:07<00:00,  7.41it/s]
```

```
Saved 7613 scores to /home/mate/FIUBA/ciencia-de-datos/tweet-
checker/resultados/llm_features/llm_scores_train.json
```

```
LLM scores test: 100%|      | 3263/3263 [07:18<00:00,  7.43it/s]
```

```
Saved 3263 scores to /home/mate/FIUBA/ciencia-de-datos/tweet-
checker/resultados/llm_features/llm_scores_test.json
```