CSE 472 Project 1 - Content Analysis and Word Cloud Generation

```
# Install unsloth
!pip install unsloth -q
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.8/54.8 kB 4.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42.0/42.0 kB 3.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 314.6/314.6 kB 29.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 491.5/491.5 kB 37.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.3/11.3 MB 125.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 544.8/544.8 kB 40.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 233.9/233.9 kB 22.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 117.2/117.2 MB 8.1 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.3/61.3 MB 13.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 132.5/132.5 kB 12.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.1/3.1 MB 106.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 213.6/213.6 kB 21.0 MB/s eta 0:00:00
```

```
from unsloth import FastLanguageModel
import torch
max_seq_length = 2048
dtype = None # None for auto detection. Float16 for Tesla T4, V100, Bfloat16 for Ampere+
load_in_4bit = True # Use 4bit quantization to reduce memory usage.

model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "unsloth/llama-3-8b-bnb-4bit",
    max_seq_length = max_seq_length,
    dtype = dtype,
    load_in_4bit = load_in_4bit,
)
```

```
🦥 Unsloth: Will patch your computer to enable 2x faster free finetuning.
🦥 Unsloth Zoo will now patch everything to make training faster!
==((====))==  Unsloth 2025.9.7: Fast Llama patching. Transformers: 4.55.4.
   \\   /|    Tesla T4. Num GPUs = 1. Max memory: 14.741 GB. Platform: Linux.
O^O/ \_/ \    Torch: 2.8.0+cu126. CUDA: 7.5. CUDA Toolkit: 12.6. Triton: 3.4.0
\        /    Bfloat16 = FALSE. FA [Xformers = 0.0.32.post2. FA2 = False]
 "-____-"     Free license: http://github.com/unslothai/unsloth
Unsloth: Fast downloading is enabled - ignore downloading bars which are red colored!
model.safetensors: 100%                                                 5.70G/5.70G [01:09<00:00, 53.7MB/s]

generation_config.json: 100%                                                 198/198 [00:00<00:00, 13.6kB/s]

tokenizer_config.json:        50.6k/? [00:00<00:00, 4.97MB/s]

tokenizer.json:        9.09M/? [00:00<00:00, 117MB/s]

special_tokens_map.json: 100%                                                 350/350 [00:00<00:00, 41.1kB/s]
```

```
from transformers import GenerationConfig

FastLanguageModel.for_inference(model) # Enable native 2x faster inference

# Create a GenerationConfig instance with the desired settings
generation_config = GenerationConfig(
    bos_token_id=128000,
    eos_token_id=128001,
    max_length=4096,
    temperature=0.6,
    top_p=0.9
)

# Apply the generation configuration to the model
model.generation_config = generation_config

# Set pad_token_id
model.generation_config.pad_token_id = tokenizer.pad_token_id
```

```
prompt_template = """
System:
You are an information-extraction engine. Return valid JSON only.

Rules:
Extract up to 3 keywords that best represent the main topics of the post.
```

```
- Be concise, lowercase, no hashtags, no emojis or punctuation except hyphens.
- Output a maximum of 3 keywords.
- Return valid JSON only, matching the schema.


Schema (JSON):
{{
  "post_id": "<string>",
  "keywords": [{{"text":"<string>","confidence": <float>}}]
}}

Example:
Input post_id: "p_42"
Input text: "We fine-tuned a multilingual sentence transformer using hard negatives from click logs."
Output:
{{"post_id":"p_42","keywords":[
  {{"text":"sentence transformer","confidence":0.92}},
  {{"text":"hard negatives","confidence":0.84}},
  {{"text":"click logs","confidence":0.80}}
]}}

Input:
post_id: {post_id}
text: {post_text}

Output:
"""
```

```
import json, re, time, sys, shutil
from pathlib import Path

def render_progress(current: int, total: int, elapsed_s: float, last_dt_s: float, first: bool):
    cols = shutil.get_terminal_size((80, 20)).columns
    bar_width = max(20, min(50, cols - 30))

    filled = int(bar_width * current / max(1, total))
    bar = "█" * filled + "—" * (bar_width - filled)

    percent = (current / total) * 100
    line1 = f"elapsed {elapsed_s:6.1f}s | last {last_dt_s:5.2f}s".ljust(cols)
    line2 = f"[{bar}] {percent:5.1f}%".ljust(cols)

    if first:
        print(line1)
        print(line2, end="", flush=True)
    else:
        sys.stdout.write("\x1b[2F")
        sys.stdout.write("\r" + line1 + "\n")
        sys.stdout.write("\r" + line2)
        sys.stdout.flush()

def append_jsonl(path: Path, obj: dict):
    """Append one JSON object per line."""
    path.parent.mkdir(parents=True, exist_ok=True)
    with path.open("a", encoding="utf-8") as f:
        f.write(json.dumps(obj, ensure_ascii=False) + "\n")

def _safe_parse_json(raw_text: str, post_id: str):
    """
    Try to parse model output as JSON.
    Returns: (record_dict, parsed_ok: bool, reason_if_fail: str|None)
    - record_dict always conforms to {"post_id":..., "keywords":[...<=3]}
    - reason_if_fail is a descriptive string if parsing failed
    """
    candidate = raw_text.strip()
    # If the model added extra chatter, try grabbing the last {...}
    m = re.search(r"\{[\s\S]*\}$", candidate)
    if m:
        candidate = m.group(0)

    parse_error = None
    try:
        obj = json.loads(candidate)
        parsed_ok = True
    except Exception as e:
        obj = {"post_id": post_id, "keywords": []}
```

```
            parsed_ok = False
            # A compact, useful message; include a short preview of the model's output.
            preview = candidate[:160].replace("\n", " ")
            parse_error = f"JSON parse failed: {type(e).__name__}: {e}; output_preview='{preview}'"

        # Minimal sanitation + hard cap
        if obj.get("post_id") != post_id:
            obj["post_id"] = post_id

        kws = obj.get("keywords", [])
        if not isinstance(kws, list):
            kws = []

        norm = []
        for k in kws:
            if not isinstance(k, dict):
                continue
            text = str(k.get("text", "")).strip().lower()
            if not text:
                continue
            try:
                conf = float(k.get("confidence", 0.5))
            except Exception:
                conf = 0.5
            conf = max(0.0, min(1.0, conf))
            norm.append({"text": text, "confidence": conf})

        obj["keywords"] = norm[:3]
        return obj, parsed_ok, parse_error

    def pipeline(post_id: str, post_text: str):
        """
        Returns: (record_dict, raw_text)
        - record_dict is sanitized JSON-ready output
        - raw_text is the model's completion (no prompt)
        """
        prompt = prompt_template.format(post_id=post_id, post_text=post_text)
        inputs = tokenizer([prompt], return_tensors="pt").to("cuda")

        out_ids = model.generate(
            **inputs,
            max_new_tokens=256,
            temperature=0.2,
            top_p=0.9,
        )

        # Only the generated continuation (drop the prompt)
        gen_tokens = out_ids[:, inputs["input_ids"].shape[1]:]
        raw_text = tokenizer.decode(gen_tokens[0], skip_special_tokens=True)

        rec, _, _ = _safe_parse_json(raw_text, post_id)
        return rec, raw_text

    input_path  = Path("posts.json")
    output_json = Path("results.jsonl")
    error_log   = Path("error_log.jsonl")

    with input_path.open("r", encoding="utf-8") as f:
        posts = json.load(f)

    limit = 700
    total = min(limit, len(posts))
    total_start = time.perf_counter()

    for idx, post in enumerate(posts[:limit], start=1):
        t0 = time.perf_counter()

        post_id = post.get("id")
        post_uri = post.get("uri")
        post_text = post.get("content", "")
        is_reply = bool(post.get("in_reply_to_id"))

        try:
            rec, raw_text = pipeline(post_id, post_text)
            rec["is_reply"] = is_reply

            try:
```

```
                append_jsonl(output_json, rec)
            except Exception as e:
                raise Exception(f"Failed to save good record: {e}")

        except Exception as e:
            append_jsonl(error_log, {
                "post_id": post_id,
                "uri": post_uri,
                "reason": f"{type(e).__name__}: {e}"
            })

        dt = time.perf_counter() - t0
        elapsed = time.perf_counter() - total_start
        render_progress(idx, total, elapsed, dt, first=(idx == 1))

    sys.stdout.write("\n")
    print(f"Done. Processed {total} posts in {time.perf_counter() - total_start:.1f}s.")
```

```
elapsed   14.1s | last 14.15s
elapsed   18.8s | last  4.66s
elapsed   23.1s | last  4.31s
elapsed   28.0s | last  4.84s
elapsed   31.7s | last  3.73s
elapsed   51.1s | last 19.41s
elapsed   65.8s | last 14.63s
elapsed   68.6s | last  2.82s
elapsed   71.5s | last  2.94s
elapsed   74.6s | last  3.09s
elapsed   77.8s | last  3.19s
elapsed   92.5s | last 14.67s
elapsed   95.4s | last  2.99s
elapsed  110.3s | last 14.90s
elapsed  125.8s | last 15.47s
elapsed  129.0s | last  3.14s
elapsed  132.0s | last  3.00s
elapsed  135.0s | last  3.00s
elapsed  138.1s | last  3.09s
elapsed  143.2s | last  5.19s
elapsed  146.3s | last  3.05s
elapsed  150.0s | last  3.67s
elapsed  153.1s | last  3.15s
elapsed  157.5s | last  4.34s
elapsed  172.3s | last 14.84s
elapsed  175.3s | last  2.97s
elapsed  178.6s | last  3.30s
elapsed  181.4s | last  2.84s
elapsed  184.6s | last  3.23s
elapsed  187.8s | last  3.14s
elapsed  191.3s | last  3.51s
elapsed  194.3s | last  2.96s
elapsed  197.5s | last  3.25s
elapsed  212.4s | last 14.93s
elapsed  215.7s | last  3.27s
elapsed  218.8s | last  3.09s
elapsed  222.0s | last  3.22s
elapsed  225.2s | last  3.21s
elapsed  228.3s | last  3.08s
elapsed  231.4s | last  3.11s
elapsed  234.5s | last  3.10s
elapsed  237.5s | last  2.97s
elapsed  240.4s | last  2.94s
elapsed  242.3s | last  1.91s
elapsed  245.4s | last  3.08s
elapsed  249.2s | last  3.76s
elapsed  254.1s | last  4.91s
elapsed  257.7s | last  3.58s
elapsed  260.8s | last  3.08s
elapsed  264.0s | last  3.29s
elapsed  278.9s | last 14.81s
elapsed  293.8s | last 14.96s
elapsed  296.9s | last  3.12s
elapsed  300.0s | last  3.11s
elapsed  303.0s | last  2.93s
elapsed  306.6s | last  3.61s
elapsed  309.5s | last  2.93s
elapsed  312.6s | last  3.08s
```

```
Start coding or generate with AI.
```