COMP 560 Retrospective

A semester of language model experiments, implementations, and research.

LLMs & NLP

Spring

2026

OVERVIEW

Semester at a Glance

01 **nanoGPT Foundation**

Forked Karpathy's nanoGPT as shared training infrastructure for all student experiments.

02 **Character-Level Experiments**

Alphabet sequences, tiny Shakespeare, count-by-twos — probing what small transformers learn.

03 **RAG Survey**

Reading Arslan et al. (2024) on RAG with LLMs led directly to the next paper and experiment.

04 **Latin Language Experiments**

Addition and verb conjugation in Latin — testing how non-English tokenization affects learning.

05 **DataFrame QA Paper**

Ye et al. (2024) introduced privacy-preserving tabular QA via Pandas query generation.

06 **DataFrame QA Implementation**

Built a working Python framework using the Claude API — zero-shot, sandboxed, pass@1 evaluated.

INFRASTRUCTURE · COMP560-NANOGPT

nanoGPT Fork

- Forked Andrej Karpathy's *nanoGPT* — a minimal GPT-2 reimplementation in PyTorch
- Added `NANOGPT_CONFIG` environment variable to decouple training scripts from the repository root
- Enables student repos to call `train.py` and `sample.py` from any working directory
- Shared checkpoint format supports training on Shakespeare, alphabet, Latin, and custom datasets

```
# Run training from any student repo

NANOGPT_CONFIG=../../comp560-
nanoGPT/configurator.py \
  python -u ../../comp560-nanoGPT/train.py \
  config/basic.py

# Sample from a trained checkpoint

NANOGPT_CONFIG=../../comp560-
nanoGPT/configurator.py \
  python -u ../../comp560-nanoGPT/sample.py \
  config/basic.py --num_samples=1
```

EXPERIMENTS · COMP560-JMAC & COMP560-COBYMCCLELLAN

Character-Level Experiments

COMP560-JMAC

Alphabet Models

Character-level models trained on alphabetic sequences. Four configs — *basic*, *mixed*, *reverse*, and *students* — probe what sequence patterns a small transformer can learn from scratch.

`basic.py` `mixed.py` `reverse.py` `students.py`

COMP560-JMAC

Tiny Shakespeare

Fine-tuned a character-level model on the classic Karpathy "tiny-shakespeare" dataset. Demonstrates how a small GPT can capture stylistic patterns of Elizabethan text at the character level.

`shakespeare` `char-level`

COMP560-COBYMCCLELLAN

Count by Twos

Trained a small model to learn the "count by twos" pattern — a minimal arithmetic sequence task. Tests whether character-level transformers can internalize simple numeric rules.

`arithmetic` `sequences`

PAPER 1 · ARSLAN ET AL. (2024)

A Survey on RAG with LLMs

Procedia Computer Science, Vol. 246 (2024)
KES 2024 · University of Burgundy, France

"A Survey on RAG with LLMs"

> *Reviews Retrieval-Augmented Generation as a solution to LLM hallucination and domain-specificity gaps. Covers both task-specific and discipline-specific RAG applications, with directions for future work.*

01

The Problem

LLMs struggle with domain-specific queries and can produce inaccurate outputs — especially on private or recent data not seen during training.

02

The Solution

RAG integrates external retrieval into the generation process — the model queries a vector store at inference time, grounding responses in real documents.

03

What Came Next

Reading about RAG's data-privacy limitations led directly to the DataFrame QA paper — a complementary approach for structured tabular data.

EXPERIMENT SERIES · COMP560-LATIN-ADDITION-

Latin Language Experiments

Exploring whether models trained on Latin — a morphologically rich language with a different character distribution — show different tokenization and generalization behavior.

COMP560-LATIN-ADDITION-

Two Latin Experiments

EXPERIMENT A

Latin Numeral Addition

Trained a small-scale character-level model on arithmetic expressed in Latin numerals. The goal: examine how a non-standard numeral system and its distinct token distribution affects a model's ability to learn addition.

- Roman numeral tokenization vs. Arabic digit tokenization
- Tests compositional generalization under character-level constraints
- Dataset generated as structured input-output pairs

EXPERIMENT B

Latin Verb Conjugations

Trained a model on Latin verb conjugation tables. Latin's rich inflectional morphology — six cases, multiple conjugation classes — provides a challenging test of whether small transformers can learn structured grammatical rules.

- Input: verb stem + person/number/tense markers
- Output: correctly inflected verb form
- Probes morphological rule learning in low-resource settings

PAPER 2 · YE, DU & WANG (2024)

DataFrame QA Framework

arXiv:2401.15463 · January 2024
New Jersey Institute of Technology

"DataFrame QA: A Universal LLM Framework on DataFrame Question Answering Without Data Exposure"

> *Proposes a framework that sends only column names and data types to the LLM — never actual data values — enabling privacy-preserving, zero-shot tabular question answering via generated Pandas queries.*

KEY RESULTS

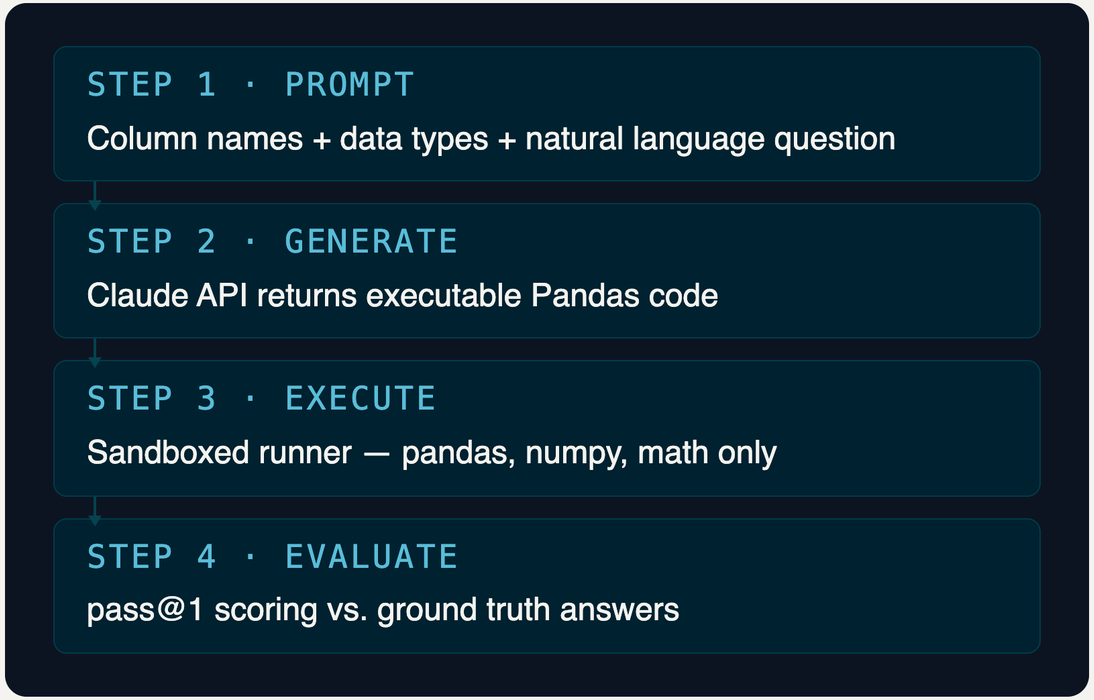
86% **pass@1 on WikiSQL**
GPT-4 zero-shot, using metadata only — no table values in the prompt

97% **pass@1 on UCI-DataFrameQA**
New dataset of complex analysis queries; GPT-4 nearly solves it

<250 **tokens per query**
Metadata-only prompts vs. full table embedding — ~90% token reduction

IMPLEMENTATION · LLM-DATAFRAME-QA

DataFrame QA in Python



- **Privacy-preserving** — raw data never sent to the API; only schema metadata
- **Zero-shot** — no fine-tuning required; works out-of-the-box with Claude Opus, Sonnet, or Haiku
- **Sandboxed execution** — generated code runs in a restricted environment; prevents arbitrary execution
- **8 error categories** — automatic failure classification matching the paper's taxonomy
- **Cross-domain generation** — training data generated across medical, automotive, and sports domains from schema alone
- **Token comparison tooling** — side-by-side benchmark of metadata-only vs. full-table prompts

What I Built This Semester

5

Active Repositories

nanoGPT fork, three student experiment repos, and a full DataFrame QA implementation — all interconnected through a shared training infrastructure.

6+

Trained Models

Alphabet (basic, mixed, reverse, students), tiny Shakespeare, count-by-twos, Latin numeral addition, and Latin verb conjugations.

2

Papers Implemented

Grounded our work in the DataFrame QA framework (Ye et al., 2024) and the RAG survey (Arslan et al., 2024) — bridging theory and hands-on code.