Introduction to LLM

Practice Session 4

Word Representation I

Topics Covered Before LLMs

- LLMs implicitly learn classic NLP tasks (POS, parsing, disambiguation)
  - We teach them first to show what LLMs do implicitly, e.g., morphology, handling inflection, OOV.
  - These topics show what's happening under the hood.
- You learn how text becomes numbers and how we search and compare with it, key ideas behind modern LLM systems.
- Seeing pre-LLM methods teaches trade-offs and when simple tools are enough.
- Not every problem needs an LLM, always start with a baseline.
  - For small, well-defined, labeled task LLMs will be an overkill (e.g., 2-layer NN for tweet sentiment can hit $\gtrsim$ 90%).
- Evaluate fairly (simple models vs. LLM) to justify cost, latency, and privacy impacts.
- Takeaway: Understand the classics, then choose the right tool based on task complexity.
  - Use LLMs only when gains justify cost or complexity.

Clarifying Lesk Variants

- Original Lesk (1986)
  - Algorithm explained in Lecture
  - Compare glosses of all candidate senses with glosses of surrounding words' senses; score by overlap between definitions.
  - *(definitions ↔ definitions overlap)*
- Simplified Lesk (Kilgarriff & Rosenzweig (2000)):
  - Algorithm explained in PS & HW
  - Compare sentence context directly to each sense's gloss.
  - *(definitions ↔ sentence context overlap)*
- Simplified Lesk is the one you'll see used most often in practice (as a baseline and in variants):
  - Mainly because it's far cheaper computationally and tends to work better than the original on standard evaluations.

Difficulty of Exercise and HW

- It’s normal to feel a bit overwhelmed in PS: moving from theory to code isn’t straightforward.
- There’s a natural gap: you may understand the lecture but struggle in implementation
  - PS is designed to bridge that gap.
- This tries to mirror your first job: turning ideas/specs into working, efficient code for your problem.
- PS focus: review the concepts in lecture and see how can we implement a minimal working version.
  - After that we show you how you can use reliable libraries to avoid reinventing the wheel.
- Important that you have programming basics + ML foundations (we tried to review that in PS1).
  - We recommended external resources on Moodle, e.g. 3Blue1Brown YT channel for visual intuition.
- It’s okay if you don’t understand everything live.
  - Ideally you understand the main idea and purpose of the code, then you can go over the details step by step yourself.

Timeline

Python & ML Foundations

- Train/Dev/Test
- Hyperparameters
- Loss Fn.

CL/NLP Basics

- Morphology
- Syntax
- PoS
- Parsing
- Semantics
- Lemma
- Ambiguity

Word Representation I:

- Basics
- Evaluation
- Inverted Index
- TF-IDF
- BM25
- LSA
- nDCG

Word Representation II:

- Statistical Methods
- Cosine Similarity
- Word2Vec
- CBOW & Skip-gram
- GloVe

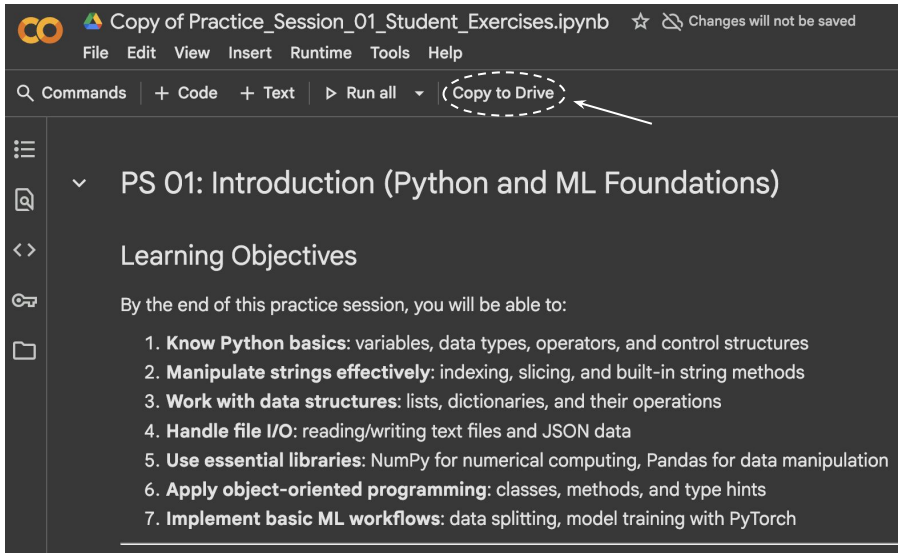
Dense Retrieval & Neural Reranking

- Dense & Hybrid
- S-BERT
- Cross-Encoders

Question Answering & RAG

- Query Decomposition
- Metadata Filtering
- Self-RAG

PS3: Colab Notebook (Available on Moodle)



- https://colab.research.google.com/drive/1IdxJCu6HOsoOJUXwnCJ6-jrF-ypjGL9G