CS572 - Spring 25 Final Rubrics

Q1. Google's search engine started off being entirely keyword-based. Over the years, 'semantic' features have been added on a regular basis.

What are three examples of such semantics-based features [out of the many we covered]? Explain each, in a few (3 or 4) sentences.

Answer:
Three examples: 'People Also Ask', use of KGs [knowledge graphs] to present facts about people/place/things, question answering using BERT, etc. We can accept any three that Google does use [not imagined/non-existent/future ones].

Rubrics: +1 for each valid item.

Q2. Recommendation engines are obviously a useful info' retrieval tool. What three uses can you think of, in addition to the ones we covered (product recommendations, YouTube/TikTok, Lyft etc.)? Think in terms of domain areas and users in them [eg. product recommendations would fall under 'business' domain, users would be 'consumers'].

Solution:
Any three examples that are valid are acceptable as long as it is not discussed in class (YouTube/TikTok, Lyft, movies). These are some samples:

1. Recommend people (social media, dating)

Suggest potential friends or matches based on shared interests, mutual connections, and past interactions.

Domain: Social Networking / Dating

User: Individuals seeking social connections

2. Recommend news

Suggest articles or headlines by learning a reader's topic preferences and reading habits.

Domain: Media / Publishing

User: News readers

3. Recommend new games to play

Suggest games based on players' genre tastes, play history, and critic ratings.

Domain: Entertainment

User: Gamers

4. Code recommendation

Auto-suggest code snippets, libraries, or function calls based on a developer's current context and past code.

Domain: Software Development

User: Developers / Programmers

5. Job recommendation based on a person's CV

Matches job seekers with open positions by analyzing skills, experience, and employer requirements.

Domain: HR / Career Services

User: Job seekers

6. Recommend recipes given ingredients

Propose dishes you can cook with the items you already have, considering the dietary preferences and cooking skill level.

Domain: Food & Cooking

User: Home cooks

Rubrics:

- **+1 per valid example (max 3) that includes a brief description or title [listing the domain, and type of user is optional, ie. not required].**

Q3. Information can often be organized hierarchically. What are three examples we studied? Explain each in a few sentences.

Solution:

1. **WordNet:**
   A hierarchical lexical database that organizes English words into synonym sets (synsets) linked by semantic relationships.
2. **ImageNet:**
   An image database structured according to the WordNet hierarchy, used for visual object recognition research.
3. **Dewey Decimal System:**
   A hierarchical classification system used in libraries to organize books and related materials by subject.
4. **Yahoo/DMOZ:**
   A hierarchical directory that organizes web page links into categories and subcategories.

Partially correct answers – these are NOT hierarchies:

1. **Wikidata:**
   A collaborative, structured knowledge base representing entities and their properties, but not organized hierarchically.
2. **Google Knowledge Graph:**
   A large-scale knowledge base connecting entities and their relationships, but structured as a graph rather than a hierarchy.

Note: Explanations of taxonomy, ontology, or knowledge graph are not examples; taxonomy can be hierarchical, but ontology and knowledge graphs are not.

Rubrics:
- +1 point for each correct example. The explanation does not need to be detailed, but it does need to involve a hierarchy (a tree structure) [not a graph, or flat list, or set etc]
- +0.5 for each partly-correct (eg, Wikidata, Google knowledge graph etc)
- 0 for irrelevant/incorrect answers (eg explaining what hierarchical organization means)

Q4. For the first 3 HWs you used coding; for the last one, for Q2, you used LangFlow, which is a 'visual dataflow' tool. You could call the result in either case, a 'tool' (eg HW2 involved creating a web crawler tool, HW's Q2 involved creating a Q&A tool). Compare the two (coding vs visual dataflow) in terms of these three aspects [which are classic software dev ones]: creation, documentation, maintainability.

Answer:
Creation: coding involved explicit syntax [of Python, Java etc]. Dataflow on the other hand involved dragging and dropping nodes, wiring them up, and customizing node properties.

Documentation: in coding we document the code - modules, classes, functions/methods, and program logic [inside a function for ex]; in dataflow we document the dataflow graph - what connects to what, and why.

Maintainability: code-based tools are harder to maintain, since they require deep understanding of program logic expressed in them (code). In contrast, dataflow graphs are extremely easy to maintain (and adapt, or extend) because the program logic is made explicit via the nodes and their connections.

Rubrics: +1 for each correct part.

Q5. ‘Agentic AI’ is poised to up-end info’ retrieval in a big way, on account of its ability to ‘synthesize’ (generate) content, be it words, images, video or audio. What are three examples you can think of, where agentic AI can deliver way more than mere information search? For example, ‘agentic finance’ would be where the AI agent might function as a knowledgeable investment advisor.
Solution:
1. **Health Advising**
2. **Teaching / Tutoring**
3. **Web Navigation – Purchasing products for users**
4. **Legal Advising**
5. **Software development / code debugging**
6. **Event organizing / restaurant suggestions / itinerary planning**
7. **Fitness / Health Advising**
8. **Assembling items, eg. a bicycle, bookshelf, even cooking**
9. **Art, dance, music etc**

Rubrics:
+1 for each answer that cannot be accomplished by pure “search.” Specific tasks that require a form of generation / not using lookup.

Q6. What are three emerging standards, in the world of agentic AI? Explain each in a few lines.

Solution:

MCP: agent communication with tools (backend)
A2A: agent-to-agent communication

ADK: for developing agents [in Google's own words: 'Agent Development Kit (ADK) is a flexible and modular framework for developing and deploying AI agents. '

Rubrics:

- +1 for each correct answer

Q7. Use of a vector DB is one way to do 'RAG'. What is stored? How is it indexed (for quick retrieval)? How does retrieval work (on what basis)? Explain each, using (three) simple diagrams.

Solution:

What is stored (+1)

In a vector database, embeddings (dense vector representations) of documents, passages, or chunks of text are stored.
These embeddings are generated using models like BERT, Sentence-BERT, or OpenAI embeddings.
Along with embeddings, metadata (e.g., titles, source URLs, timestamps) is also stored.
IR Analogy: This is like storing document vectors in the vector space mo

How is it indexed (for quick retrieval)(+1)

Vector databases (e.g., FAISS, Pinecone, Weaviate) use **approximate nearest neighbor (ANN)** search algorithms like:

- **HNSW (Hierarchical Navigable Small World)**
- **IVF (Inverted File Index)**
- **PQ (Product Quantization)**

These allow **fast and scalable retrieval** even from millions of vectors.

This is similar in spirit to using an inverted index for fast lookup of keyword-document mappings.

How does retrieval work (on what basis)(+1)

When a query is received:

1. It is converted into a query embedding.
2. The system performs **vector similarity search** (usually cosine similarity or inner product).
3. The **top-k nearest vectors** (i.e., semantically similar chunks) are retrieved.
4. These chunks are passed to the language model to generate the final answer.

Instead of tf-idf or BM25 scores, we use **cosine similarity of vector embeddings** for ranking.

Rubrics:

- **+1 for each explanation with a relevant image (total 3)**
- **-1 if the explanations or the image are not related**

Q8. Before an LLM (or more generally, an FM, ie foundation model) is put to use (eg by making it be available on Hugging Face, or via a server (eg. at OpenAI or Google or Perplexity etc), it needs to undergo three preparatory stages. Explain the stages, using a few lines for each.

Solution (3 pts):

Pretraining:
The model is trained on a large-scale, diverse dataset (internet text, code, etc.) using self-supervised learning. The goal is to learn general-purpose language or representation capabilities without any task-specific supervision.

Finetuning:
The pretrained model is further trained (often with supervised or instruction-tuned datasets) to align it with more specific tasks or to make it more useful, robust, or aligned with user expectations. This may include RLHF or instruction tuning.

Evaluation and Safety Alignment:(eg. HFRL, GRPO etc)
Before deployment, the model undergoes rigorous testing for safety, fairness, bias, robustness, and factual accuracy. Guardrails (filters, moderation, prompt engineering) are often added, and external evaluations (red-teaming, audits) may be conducted.
NOTE: They might also say finetuning, evaluation, and deployment, where at the evaluation stage, the model's performance and weaknesses are measured, and in the deployment phase, the model is optimized for inference speed, resource availability, and safeguards are implemented. I think this answer would also be correct.

Rubrics:

- 1 pts per stage. If the explanation is wrong / missing -0.5 pts.

Q9. NER to 'graph RAG' - what is the connection (ie pipeline) that involves these two items? In other words, how would we start from plain text, and end up with a tool that helps us obtain high quality (non-hallucinatory) responses from an LLM? Explain the pipeline (steps).

Solution (3 points):

1. NER: Extract entities from the plain text.
2. KG Construction: Construct knowledge graph using extracted entities and their relations.
3. Graph Retrieval
   a. Query processing (link page 8, slide 20): Use NER to extract entities
   b. Subgraph Retrieval: Retrieve subgraphs relevant to these entities
4. RAG: Use LLM with retrieval augmented context (query + relevant graph).

The key part of the answer involves mentioning and discussing knowledge graphs.

Rubrics:

- +1: NER + KG Construction
  - Clearly states that entities are extracted from plain text using NER and used to construct a KG.

- +1: Graph Retrieval + Query Processing
  - Mentions using NER on the query to extract entities, then retrieving relevant subgraphs based on those entities (consistent with lecture).
  - *Alternatively, if embedding-based retrieval is proposed, full credit is given if the student clearly explains how embeddings are used to retrieve graph-relevant context (e.g., entity-linked or subgraph-level).*

- +1: RAG
    - Explains that the LLM generates responses using the query combined with the retrieved graph-based context.

Q10. Consider the following:

Ÿ user-user-user...
Ÿ item-item-item...
Ÿ content-content-content...

Explain the above, in terms of what we studied [as it relates to info' retrieval] :)

Possible Explanations:
These terms represent different approaches to retrieving relevant information, ie for use in recommendation engines:

- user-user-user: user-based collaborative filtering, where the recommendations find users similar to a target user.
- item-item-item: item-based collaborative filtering, where the recommendations find items similar to those a user liked or interacted with.
- content-content-content: content-based filtering, where the recommendations are based on items' features and match them to the user's preference.

Rubrics:

- +1 for each term explained (total 3).