v1 CS585 Final Rubrics

Total Marks	35 points
Late Penalty	< 30 mins : -2.5 points >= 30 mins: -5 points

General guidelines (Prof Saty's instructions note from the exam):

There are 8 questions below, each worth 5 points. You only need to answer any 7, for a total of 35. But if you like you can answer the 8th too - your total will be capped (ie. can't exceed) 35 points - "still", a great deal, omfg!

You can look up anything, but in your answers, do stick to what we covered in class. Answer in your own words, rather than copying and pasting verbatim from ChatGPT! Note that ChatGPT output, long answers (with needless detail) etc. will result in a 0 for each question where this happens. Translation: **use your own words, sticking to what we went over during the lectures**! So why even have this be an open test? No good reason :)

The exam is for an hour, with OSAS students allowed longer times.

And - please do NOT cheat in any way! Cheating will lead to your getting reported to OAI, and a 0 for the entire exam.

- If you find that student has **"literally"** copy-pasted the solution from ChatGPT / Bard or any other LLM, please give **-50% points** for that question
- If you find that a student has **"significantly"** answered **"more than what is asked"**, please provide **partial grading** written in each question's rubric.

Question 1:
a. What do spatial data and vector embeddings (used in genAI for ex) have in common, architecture-wise (at an internal level)? Explain using a few sentences.

b. What is an application where spatial data and genAI can be used together? Explain using a couple of sentences.

Solution:
a) Common architecture features in spatial data and vector embeddings (+3) :

- Multi-dimensional / high-dimensional / structured vector spaces, specialized / efficient indexing techniques using data indexing structures like R-trees, quadtrees, KD-trees, etc. **They both use a spatial hierarchy for indexing.**

b) Some possible applications of spatial data + genAI (simpler explanations are okay if they convey similar meaning) (+2):

- NeRFs : Neural radiance fields : synthesizing new views of complex scenes by optimizing an underlying continuous volumetric scene function using a sparse set of input photographs views
- Generate fictional realistic worlds (landscapes, environments) based on styles and spatial information
- Other spatial + generative applications like urban city planning / immersive experiences / city tours etc. are fine as long as correct explanation of using spatial data + generative AI (**eg. use of Google's Photorealistic Map Tiles (which is spatial data) along with genAI to render seamless drone-like flight paths**)

Deductions

# points deducted	Cause
-3	If no common architecture points are mentioned for similarities. (-1 for each missing point)
-1	If application mentioned, with no explanations at all
-2	No application or explanation of application mentioned

Question 2:
Las Vegas has a new 'dome':
https://en.wikipedia.org/wiki/Sphere_(venue)#/media/File:The_Las_Vegas_Sphere,_Nevada_(53349695459).jpg

a. What are two kinds of data that can be displayed on the outside of a dome (eg. outside looks like this: https://www.youtube.com/watch?v=5mzsxWAy1b4)

b. What are three forms of data viz [choosing from https://observablehq.com/@d3/gallery] that can be displayed on the inside of a dome? Inside looks like so [watch from 2:10: https://vimeo.com/209110112].

Solution:
a) 2 kinds of data displayed outside dome (+2):

- Video / Images / Artistic visuals / promotional content / information content / social content etc.
- - weather data
  - animal/bird migration patterns
  - …
  - any answer that 'makes sense' is acceptable [eg 'bar charts' don't quite make sense to be on a dome]

b) 3 forms of data displayed inside dome (+3):

- Projection imagery / maps / looping visuals etc.
- **Any form of viz from the gallery list, that has circular symmetry (eg sunburst sequence plot, zoomable circle packing)**

Deductions

# points deducted	Cause
-2	2 kinds of data for the outside of the dome not mentioned, -1 for each missing
-3	3 kinds of data for the inside of the dome not mentioned, -1 for each missing

Question 3:
Music can be synthesized by flowing signals (control, content) through modules, like so:

See/hear https://noisecraft.app/515 for another example :)

What is the connection between such 'visual dataflow graphs' [where 'data' can mean music, images, video, 3D CG elements...] and modern data 'pipelines'? Be descriptive, including providing an example. This question isn't vague or far-fetched - the connection has been mentioned many times during lectures.

Solution

They both have the last operation's output as input, do some fixed calculations to the data and send the output as input for the next operation in the pipeline.They are connected in sequence to achieve a certain outcome.
Example:
Visual data flow(Synthesizer): signals go through modules, each module takes the last module's output as input, tweaks the signal and sends the output as input for the next module, lastly we get the sound we want.
Data pipeline: data goes through a series processing steps, each step takes the last step's output as input, does some calculation and sends the output as input for the next step, lastly we get the data we want. **An example can be this: two tables get merged (unioned), then joined with a third table, then filtered and grouped via SELECT, and the results displayed visually.**

Deductions

# points deducted	Cause
-2	Description about the connection is totally wrong
-1.5	The example provided for visual data flow do not match with the description about the connection, or example totally wrong.(any kind of data type is acceptable)

-1.5	The example provided for the data pipeline do not match with the description about the connection, or example totally wrong.

Question 4:
How does GraphQL (a query LANGUAGE) relate to JSON (a data FORMAT)? In other words, describe how they are similar, how they are different? You can use any online example as-is (verbatim).

Solution

Similarities between JSON and GraphQL (+2):
- data representation / schema definition / hierarchical structure, data interchange formats / useful for web APIs, human readable, client-server communication, etc.
They both employ a curly braces syntax, and keys ('columns') that point to values (data).

Differences between JSON and GraphQL (+2):
- data exchange vs data query language, retrieving all data vs retrieving specific exact required data, less efficient vs more efficient in terms of what data is fetched, etc.
JSON has both keys and values; loosely, GraphQL specifies just the keys (and expects the query processor to return values for them, most commonly in JSON format).

*Example to illustrate **OR** Any additional non-repeated point made (+1):*
- (any valid example with some GraphQL / JSON syntax is fine, syntax need not be perfect) OR (any other point not already mentioned before is also fine)
- Example: For a user profile on Duolingo:

GraphQL:

```
query {
  user(id: "12345") {
    name
    email
    languages_learning {
      language
      level
      xp
    }
  }
}
```

JSON:

```
{
  "data": {
    "user": {
      "name": "John Doe",
      "email": "johndoe@example.com",
      "languages_learning": [
        {
          "language": "Spanish",
          "level": 5,
          "xp": 1500
        },
        {
          "language": "French",
          "level": 3,
          "xp": 800
        }
      ]
    }
  }
}
```

Deductions

# points deducted	Cause
-1	Similarities mentioned, but not described well
-2	No similarities mentioned at all, with no description
-1	Differences stated, but not described well
-2	No differences mentioned, with no description
-0.5	Example only mentioned, but no GraphQL / JSON specific structure pointed out to describe the example (unless self explanatory)
-1	No example/s / additional point mentioned at all
-5	If answer is not at all relevant anyhow to the question asked

Question 5:
Express the following, as VALID JSON. Note that there can be more than one way [so just choose one]!

a.
SELECT AccountID, AccountBalance FROM Customers
WHERE Country='US';

Note that there are 3 parts...

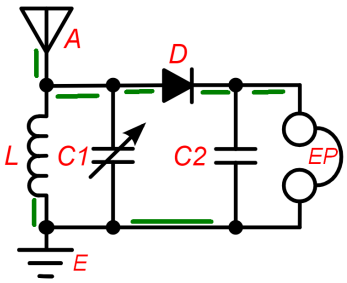
b.
int add2Nums(int a, int b) {

return (a+b)

}

Note that a function has 3 'pieces': return type, arg (param) list, body :)

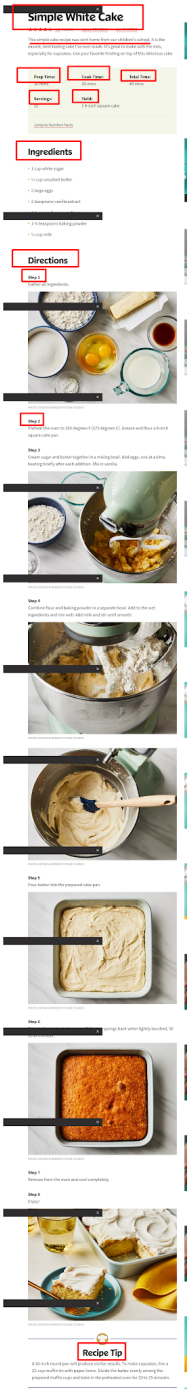
c.



Note - this is a circuit for a 'crystal radio' (https://en.wikipedia.org/wiki/Crystal_radio). The green marks I made are not part of the circuit [duh!] - you need to include just those connections.

d.
Note - the items in red boxes need to be in your JSON; no need for an exhaustive list of the ingredients, steps, or step descriptions, etc; you can use ... to leave out a lot!

e. What two features make JSON, better than a (relational) table for specifying (storing) data?



5.A Solution

Focus on the key value pairs. It is okay if only one json object is mentioned. **+1 Point**

```
{
  "SELECT": [
    "AccountID",
    "AccountBalance"
  ],
  "FROM": "Customers",
  "WHERE": [
    {
      "Country": "US"
    }
  ]
}
```

You can do a quick check at https://jsonformatter.org/json-parser

See if the answer is a json re-presentation of the query which should have “SELECT” , “FROM” and “WHERE”. Need not be an exact json array/string

5.A Deductions

# points deducted	Cause
-0.5	If invalid keys are used. The keys need to have SELECT, FROM and WHERE.
-0.1	Formatting issue. Issues like 1. Each json object needs to be wrapped with {} 2. If its a json array it needs to be wrapped with []. 3. Each json object and key-value pairs needs to be separated with a comma 4. Key and value pair should be “Key” : value (Make sure the key is within double quotes, and **:** is used as assigning symbol instead of = **Cap the deduction at -0.25**

5.B +1 Point

Solution

```
{
  "methodName": "add2Nums",
  "outputType": "int",
  "inputParams": [
    {
      "index": 0,
      "name": "a",
      "datatype": "int"
    },
    {
      "index": 1,
      "name": "b",
      "datatype": "int"
    }
  ],
  "body": "return (a+b)",
  "operator": "+"
}
```

Permissible answers for “inputParams” are

1. “inputPrams”: [“int a”, “int b”]
2. “inputPrams”: [“a”, “b”]

Mandatory keys are : outputType, inputParams , Body /Operator/Return Type/Return Function

Any logical interpretation of the algorithm in json format is valid. It need not be exactly the same as above. Keys and values vary based on assumptions. “Index” key is optional in inputParams, no deduction if missing.

5.B Deductions

# points deducted	Cause
-0.5	If invalid keys are used. The keys need to have outputType, inputParams , Body or Operator or Return Function
-0.1	Formatting issue. Issues like 1. Input params needs to be a json array 2. All the points included in 5.a “Formatting issue” section **Cap the deduction at -0.25**
-0.1	If used invalid datatypes . -0.1 for each violation **Cap the deduction at -0.25**

5.C Solution

```
{
  "A":["D","C1"],
  "D":["A","EP"],
  "L":["E"],
  "C1":["A","D","C2"],
  "C2":["C1","D","EP"],
  "EP":["D","C2"],
  "E":["L"]
}
```

The answer to this question should resemble like a unweighted graph, with the nodes as A,D,L,C1,C2,EP,E and the edges marked in Green line

5.C Deductions

# points deducted	Cause
-0.5	If all the nodes are not included. Cap at 0.5
-0.1	Formatting issue. Directions same as above **Cap the deduction at -0.25**
0	No deductions if few of the indirect links are missed/or assumed extra

5.D Solution

```
{
  "dishName": "Simple White Cake",
  "description": "The simple cake....school",
  "prep_time": "10 mins",
  "cook_time": "30mins",
  "total_time": "40 mins",
  "servings": 12,
  "yield": "1-9 inch sq cake",
  "Ingredients": [
    "1 cup of white sugar",
    "..",
    "milk"
  ],
  "Directions": {
    "Step_1": "Gather All Ingredients",
    "Step_2": "Preheat oven..."
  },
  "Reciepe Tip": "A 10 inch pan..20-25mins"
}
```

The answer will vary based on the interpretation. It is okay to make “Directions” and an array of numeric steps.

5.D Deductions

# points deducted	Cause
-0.1	If less than 5 keys are mentioned. -0.1 for each less than 5. If no keys are mentioned it is same as not attempting the question hence no marks awarded.

-0.1	Formatting issue. Directions same as above **Cap the deduction at -0.25**
-0.1	If Json array is not used at all or “Servings” is not int. It is important to explore all the datatypes available in json.

5.E What two features make JSON, better than a (relational) table for specifying (storing) data?

Solution

The answer should contain the advantages of JSON/NoSQL over RDBMS

1. Flexibility of having more/less attributes in different documents of the same collection(table) **[ie no need for a rigid, rectangular matrix]**
2. **Multi-valued attributes!**
3. Key and value pair provides better querying
4. As JSON doesn’t require a schema, it can evolve without any restrictions or dependency on other tables
5. Highly scalable and agile

Sample GPT answer.

JSON (JavaScript Object Notation) and relational tables each have their strengths and weaknesses, and the choice between them depends on the specific requirements of the application. Here are two features that make JSON preferable in certain situations compared to a relational table:

Flexibility and Hierarchical Structure:

JSON is inherently hierarchical and allows for nested structures. It supports arrays, objects, and key-value pairs, making it more flexible for representing complex and nested data structures.
Relational databases, on the other hand, are based on a tabular structure with predefined columns, and relationships between tables are typically established using keys. While this structure is well-suited for certain types of data with a clear schema, it can be less flexible when dealing with hierarchical or nested data.
Schema-less Nature:

JSON is schema-less, meaning that data can be added or removed without the need to alter a predefined schema. This flexibility is advantageous in scenarios where the data structure may evolve over time or where different parts of the application deal with different subsets of the data.
Relational databases rely on a predefined schema that describes the structure of the data, and any changes to the schema usually involve altering tables. This can be more cumbersome in situations where the data schema is subject to frequent changes.
It's important to note that the choice between JSON and a relational table depends on the specific use case, data structure, and requirements of the application. In scenarios where data relationships are well-defined and the structure is stable, a relational database might be a more suitable choice. Conversely, if flexibility, nested structures, and a schema-less approach are crucial, JSON could be a better fit. Many modern applications use a combination of both approaches, employing relational databases for structured data and JSON for more flexible and dynamic data.

Sample Bard answer.

While both JSON and relational tables have their strengths and weaknesses, JSON offers two distinct advantages over relational tables in certain situations:

1. Flexibility and Schema-lessness:

- Unlike relational tables with pre-defined columns and data types, JSON is inherently flexible and schema-less. This allows you to store data with varying structures without modifying the underlying database schema. This is particularly useful for:
    - Data with unpredictable or evolving structure: JSON can easily adapt to changing data requirements without requiring database schema changes, making it ideal for dynamic and evolving data.
    - Complex and nested data: JSON can efficiently represent complex data structures with nested objects and arrays, offering a more natural representation of hierarchical information than the flat structure of tables.

2. Ease of Use and Readability:

- JSON's human-readable syntax makes it easier to understand and work with data compared to the structured format of relational tables. This can be especially beneficial for:
    - Rapid prototyping and development: JSON's simplicity allows for faster iteration and prototyping, especially when working with complex data structures.
    - Data exchange and integration: JSON's widespread adoption in various technologies and platforms makes it a convenient format for data exchange and integration between different systems.

However, it's important to note that JSON also has limitations, such as:

- Performance: Relational databases generally offer better performance for large datasets and complex queries compared to JSON.
- Data integrity: Relational databases enforce data integrity through constraints and normalization, which can be lacking in JSON.
- Security: Relational databases provide more robust security features and access controls compared to JSON.

Therefore, the choice between JSON and relational tables depends on your specific requirements. JSON is better suited for situations where flexibility, schema-lessness, and ease of use are paramount, while relational databases excel in performance, data integrity, and security for structured data with well-defined relationships.

5.E Deductions

# points deducted	Cause
-0.5	If only one point is provided
-0.25	If the points are not properly justified, even a one liner is fine.

Question 6:
a. Why is unsupervised learning called that? Be specific!

b. Until "recently", unsupervised learning has been the ugly stepchild of AI - there has been just a single canonical DM algorithm in this category: which one?

c. Recently, what changed, ie. what caused the AI spotlight to turn on unsupervised ML? Be specific.

d. What is the 'data' connection to this new unsupervised learning algorithm/architecture?

Solution

a. During unsupervised learning, we train a model **without having any labeled data.** So, there is **no supervision during the training phase,** and the algorithm can learn patterns **without pre-defined ones**.

b. **K-means clustering algorithm**.

c. Focusing more on LLMs recently caused us to rely on unsupervised learning more. Because we can not and do not tag and label whole language for LLMs (Nouns, verbs, and so on). So, they need unsupervised learning methods (transformers) to learn and understand the language.

d. We need large amounts of data (textual data) for training transformers and LLMs. Cause they need to learn language patterns from unlabeled data.

Deductions

# points deducted	Cause
-1.5	Not mentioning the definition of unsupervised learning and how it corresponds with naming
-0.5	Not mentioning k-means algorithm
-1.5	Not mentioning the emergence of LLMs and GPT or transformers as a reason for the wide use of unsupervised learning
-1.5	Describe data connection with LLMs incorrectly

Question 7:
a. The usual categorization of DB types has been... what?

b. What new category does genAI add to the usual list?

c. What makes this new DB type be a fundamentally different one? In other words, describe the architecture in a few (eg 3 sentences), noting how it leads to the difference. Hint: "SS".

Solution

a) Some database categorizations (+2) :

- Relational vs non-relational : SQL vs NoSQL [**and in NoSQL, 4 types: k:v, document, columnar, graph**]
- Based on other parameters like location (centralized, distributed..), data currency (operational, analytical…), structure of contained data (unstructured, structured, semistructured…)
- Other valid categorizations also acceptable

b) GenAI could add any of the following (other valid ones are also acceptable) (+1):
Vector DBs

c) Ways in which genAI DB type/s are different (+2):

- Semantic Synthesis? store + retrieve data + synthesize data using learned patterns
- Vector embeddings - data points mapped to high-dimensional vectors, allowing vector search
- Vector indexing - create indexes on vector embeddings to enable faster similarity searches

The main diff is this: existing DBs are about exact search where the search params are specified (eg via SQL's WHERE conditions); vector DBs instead perform similarity search, based on a sample input (eg. query prompt, image, audio etc).

Deductions

# points deducted	Cause
-2	If no categorizations mentioned, -1 if unclear but somewhat coherent explanation
-1	If no answer to b) part, -0.5 if unclear but addresses the question
-2	If no differences mentioned, -1 for every repetitive unclear point

Question 8:
The EU 'AI ACT' [https://www.europarl.europa.eu/news/en/press-room/20231206IPR15699/artificial-intelligence-act-deal-on-comprehensive-rules-for-trustworthy-ai] is a set of regulations centered around modern AI advances (which in turn result from use of large volumes of... DATA).

Name five 'ills' from https://arxiv.org/abs/2310.00737 that could be addressed by these upcoming regulations. In other words, what five dangers listed in the paper, could the new regulations help mitigate? Just naming each in a few words should be fine.

Solution

1. Impersonating Celebrities or Public Figures (Propaganda)
2. Automated Propaganda Generation (Propaganda)
3. Creating Fake Historical Documents or Texts (Propaganda)
4. Generating Fake Product Reviews (Deception)
5. Generating Realistic but Fake Personal Stories or Testimonies (Deception)
6. Crafting Convincing Scam Emails (Deception)
7. Crafting Legal Documents with Hidden Clauses (Deception)
8. Automated Social Media Manipulation (Info. Manipulation)
9. Generating Fake Medical Advice or Information (Info. Manipulation)
10. Crafting Deceptive Advertisements (Info. Manipulation)
11. Creating Fake Financial Reports or Data (Financial Harm)
12. Generating Scripts for Scam Calls (Financial Harm)
13. Fake Personal Profiles and Identities (Personal and Identity Harm)
14. Automated Online Harassment (Personal and Identity Harm)
15. Generating Fake Evidence or Alibis (Personal and Identity Harm)
16. Fake Technical Support Scams (Tecno-social Harm)
17. Generating Biased or Prejudiced Content (Tecno-social Harm)

Deductions

# points deducted	Cause
-1 pt	need to provide at least five points, and 1 pt will be deducted for each missing point
-0.5 pt	only gives keywords and does not discuss it
-0.5 pt	the keyword is given, but the explanation does not match the keyword