Mock Endterm Exam — Answers
Introduction to Applied Data Science

2026-05-17

Multiple Choice Answers

Question	Answer
1	B
2	C
3	B
4	B
5	B
6	B
7	B
8	B
9	C
10	C
11	B
12	B
13	C
14	B
15	C
16	C
17	B
18	B
19	B
20	B

Motivations:

1. **B** — TF-IDF weights a term by its frequency in the document (TF) multiplied by the inverse of how many documents contain it (IDF). “Quantitative easing” appears in only 1 of 5 documents,

giving IDF = log(5/1) ≈ 1.61. "Market" appears in all 5, giving IDF = log(5/5) = 0, so its TF-IDF score is zero regardless of frequency. Option A confuses raw frequency with TF-IDF weight; D is wrong because TF-IDF scores are non-negative.

2. **C** — The `beta` matrix from `tidy()` gives the probability that each term is generated by each topic — i.e., the word distribution over vocabulary for each topic. Option A describes the `gamma` matrix (document-topic probabilities); B describes raw corpus frequency; D describes the topic prior (Dirichlet hyperparameter α), not `beta`.

3. **B** — The CBOW hidden layer is the simple average of context word embeddings, which makes all orderings of the same context words produce the identical hidden vector. Word order within the context window is therefore thrown away — the defining characteristic of a "bag of words" representation. Option A is a plausible-sounding distractor: CBOW does weight words equally (no inverse-distance weighting), so order still does not matter — but the reason is the equal-weight average, not distance weighting. C confuses the level of aggregation; D describes neither CBOW nor any standard model.

4. **B** — The Key vector is what a token "offers" to other tokens for comparison. During self-attention, each token's Query vector is compared (via dot product) against all other tokens' Key vectors to compute attention scores. Option A describes the output (weighted sum of Value vectors); C describes the Query vector; D conflates positional encoding with the Key vector.

5. **B** — Self-attention processes all tokens in parallel, treating them as a set rather than a sequence. Without positional encoding, the model cannot distinguish word order, so "inflation rose then fell" and "inflation fell then rose" produce identical representations. Option A is false (softmax operates on unbounded scores); C is wrong (positional encoding adds to embeddings, not replaces Keys); D is fabricated.

6. **B** — `required = FALSE` signals that the LLM is permitted to omit the field if it cannot find the information in the input text. When omitted, `ellmer` returns `NA` in R rather than forcing the model to invent a value. Option A is wrong (no zero-default exists); C describes the hallucination that `required = FALSE` is designed to prevent; D is wrong — the call succeeds, returning `NA`.

7. **B** — `type_object(product = type_string(), price = type_number())` defines the schema for a single row (one product). When calling `parallel_chat_structured()` with a vector of prompts, each prompt produces one such object. Option A is syntactically invalid (mixed scalars in `type_array`); C wraps the row schema in `type_array`, which would be correct for extracting *multiple* items from a *single* prompt, not for per-prompt extraction; D is syntactically invalid.

8. **B** — In `ellmer`'s tool-calling workflow, the LLM does not execute code directly. It identifies that a tool is needed, emits a structured request (tool name + arguments), the R session executes the function, and returns the result as part of the conversation so the LLM can formulate its final answer. Option A is incorrect — LLMs have no code execution capability. C and D are fabricated.

9. **C** — `ragnar_store_build_index()` finalises the knowledge store by constructing the vector similarity search (VSS) index and BM25 text index over all inserted chunks. Without it, `ragnar_retrieve()` cannot efficiently search. Option A is wrong — embedding generation happens at `ragnar_store_insert()` time; B is wrong (no JSON conversion occurs); D is wrong (`ragnar_register_tool_retrieve()` handles tool registration).

10. **C** — `st_intersects(A, B, sparse = FALSE)` returns a logical matrix with `nrow(A)` rows and `nrow(B)` columns. With `hospitals` (30) and `districts` (12), the result is a 30 × 12 matrix. Option A describes a reduction to a vector (not what `sparse = FALSE` produces); B has the wrong dimensions; D describes the default sparse output (with `sparse = TRUE`).

11. **B** — EPSG:32119 is a projected CRS whose unit is metres. Therefore `dist = 5000` means exactly 5,000 metres (5 km). A geographic CRS such as EPSG:4326 uses degrees, and `dist = 5000` would be interpreted as 5,000 degrees — a nonsensical and incorrect buffer. Option A inverts the relationship; C invents a feet convention; D is false.

12. **B** — `st_join()` by default uses `left = TRUE` (a left spatial join), so all rows from the left-hand object (`firms`) are retained. Firms that do not fall inside any region receive `NA` for all region-derived attribute columns. Option A describes an inner join, not the default; C and D are incorrect descriptions of `st_join()` behaviour.

13. **C** — `st_touches()` returns `TRUE` when two geometries share boundary points but do not overlap in their interiors — exactly the definition of sharing a border without overlapping. This is the correct predicate for identifying contiguous spillover zones. `st_intersects()` (A) would also include overlapping geometries; `st_within()` (B) tests complete containment; `st_is_within_distance()` (D) tests proximity by a distance threshold, not border contact.

14. **B** — In a `stars` raster object, the `band` dimension stores the distinct layers of the array. For a Landsat satellite image, each band captures a different spectral wavelength (e.g., red, near-infrared, thermal). Option A is wrong — CRS is stored as metadata, not as a dimension; C is a common misconception (a temporal `time` dimension is possible but not the default label "band"); D confuses raster structure with vector polygons.

15. **C** — Cosine similarity close to 1 means the angle between the two vectors is near zero — they point in nearly the same direction. In the word embedding context, this means "bond" and "coupon" appear in very similar linguistic contexts in the training corpus. Option A confuses cosine similarity with co-occurrence count; B confuses direction with magnitude; D describes orthogonality (cosine = 0).

16. **C** — Temperature scales the logits before the softmax operation. A high temperature (e.g., 2.0) flattens the probability distribution, spreading probability mass more evenly across tokens, making rare tokens more likely and outputs more varied and less predictable. Low temperature (e.g., 0.2) sharpens the distribution, making the model more deterministic. Option A conflates temperature with `max_tokens`; B inverts the direction of the effect; D is fabricated.

17. **B** — `tidy(model, matrix = "gamma")` returns one row per `(document, topic)` combination. For a corpus of D documents and k topics, there are $D \times k$ rows. The rows for a specific

document collectively form its topic mixture distribution (the gamma vector sums to 1 across topics). Option A describes `beta` output; C mischaracterises the structure; D is wrong — all rows belong to the full distribution.

18. **B** — `getbb()` returns a 2 × 2 matrix with rows named `"x"` (longitude) and `"y"` (latitude), and columns `"min"` and `"max"`. To estimate the centre longitude, average the minimum and maximum longitude: `mean(bb["x", ])`. Option A uses the latitude row instead; C takes a single corner element; D mixes both dimensions and produces a meaningless average.

19. **B** — Spatial predicates (`st_within()`, `st_intersects()`, etc., used internally by `st_join()`) compare coordinate values directly. If `plants` is in EPSG:4326 (degrees) and `zones` is in EPSG:28992 (metres), the raw coordinate numbers are on entirely different scales and the join will produce silently wrong or empty results. The fix is to transform one dataset to match the other's CRS before joining. Option A is false; C inverts the argument convention (which is arbitrary); D is false.

20. **B** — This is a textbook example of hallucination. The LLM has no access to real-time or post-training statistics; it generates a number that sounds plausible based on patterns (e.g., typical Dutch unemployment ranges seen in training data). Option A is wrong — temperature affects randomness in token choice, not the accuracy of factual claims; C is a real phenomenon but not the primary cause here; D is not a meaningful LLM concept.

Open Question Answers

1. (3 pts) LDA `beta` and `gamma` interpretation.

The `beta` values in Topic 1 represent the **probability that each word in the vocabulary is generated by Topic 1**. They form a probability distribution over all terms: words with high `beta` for Topic 1 (e.g., "fiscal", "debt", "spending") are the words most characteristic of that topic. Since these top words relate to government budgets and public debt, Topic 1 most likely captures a **fiscal policy / public finance** theme.

The `gamma` values show the **document-topic mixture** — the probability that each document is associated with each topic. Document R1 has gamma ≈ [0.91, 0.09], meaning it is strongly associated with Topic 1 (fiscal/debt) and barely with Topic 2, which is consistent with its content ("fiscal deficit public debt government spending"). Document R3 has gamma ≈ [0.08, 0.92], meaning it is strongly associated with Topic 2 (monetary/inflation) and barely with Topic 1, consistent with its content ("inflation consumer prices monetary tightening interest rates"). Crucially, the gamma values reflect that each document is a **probabilistic mixture** of topics rather than belonging exclusively to one: a value of [0.91, 0.09] does not mean the document is entirely about Topic 1, only that Topic 1 generates most of its words.

Grading: 1 pt for correct definition of beta (probability of term given topic); 1 pt for identifying the economic theme of Topic 1; 1 pt for correct interpretation of gamma for both documents (mixture interpretation, not hard assignment).

2. (3 pts) Structured extraction with `ellmer`; local vs. cloud LLMs.

`required = FALSE` tells the LLM that it is permitted to omit the field when the information cannot be found. A plausible reason `basis_points` returns `NA` even when the information is present in the text is that the LLM **failed to extract or convert the numeric value correctly** — for example, it may not have parsed "25 basis points" as the integer 25, or its attention mechanism failed to associate the phrase with the `basis_points` field. With `required = FALSE`, the model responds with the field absent (yielding `NA`) rather than producing a potentially wrong integer. If `required = TRUE` were used, the model would be forced to return *some* integer, risking hallucination.

Regarding the colleague's suggestion: the primary privacy advantage of `chat_ollama()` is that the model runs entirely on the researcher's own machine via Ollama. No text is transmitted over the internet to an external server, so sensitive policy documents remain within the researcher's own infrastructure. The trade-off is capability: locally hosted open-source models are generally smaller and less capable than frontier cloud models such as GPT-4o. For complex extraction tasks requiring nuanced understanding of financial or legal language, the local model may produce lower accuracy or more extraction errors.

Grading: 1 pt for explaining why `NA` can appear (extraction failure or missing value, with `required = FALSE` permitting absence); 1 pt for privacy advantage (data stays local, no external transmission); 1 pt for capability trade-off (smaller local model vs. larger cloud model).

3. (3 pts) Self-attention and pronoun resolution.

The word with the highest pre-softmax score for "it" is **"subsidy"** (score 2.4). This indicates that the self-attention mechanism resolves "it" as referring to "the subsidy" — the entity that reduced input costs. The mechanism achieves this by computing high similarity between the Query vector of "it" and the Key vector of "subsidy", reflecting the learned contextual relationship: in this construction, "it" typically refers to the grammatical subject of the preceding clause.

The student's proximity claim is **incorrect**. "Farmers" (score 1.9) is positioned between "subsidy" and "it" — it is closer to "it" in word position than "subsidy" is. If proximity drove attention scores, "farmers" should score higher than "subsidy". Instead, the model assigns higher attention to "subsidy" because self-attention compares **Query and Key vectors** that encode semantic and syntactic roles learned during training, not raw positional distance. Word order information is incorporated via positional encoding added to the embeddings before attention is computed, but this does not make proximity a direct driver of attention scores.

Grading: 1 pt for identifying "subsidy" as the top-attended word and explaining pronoun resolution; 1 pt for correctly stating the proximity claim is false; 1 pt for the correct explanation (Query-Key dot products encode semantic/syntactic roles, not positional proximity).

4. (3 pts) Spatial analysis errors.

Error 1: CRS Mismatch in the Spatial Predicate

`schools` is in EPSG:25832, while `stations_buffered` inherited EPSG:4326 from the `stations` dataset. The `st_intersects()` function requires both `sf` objects to have the exact same Coordinate Reference System. Running this code will immediately throw a fatal CRS mismatch error.

Transform one dataset to match the other before running `st_intersects()`. (Transforming `stations` to the projected CRS before buffering is the best practice here).

- **Corrected Code:**

```
stations_projected <- st_transform(stations, crs = 25832)
stations_buffered <- st_buffer(stations_projected, dist = 500)
```

(Alternatively, accept `st_intersects(schools, st_transform(stations_buffered, 25832))`)

Error 2: Calling `sum()` directly on an `sgbp` list.

By default, `st_intersects()` does not return a logical vector or a simple dataframe; it returns a sparse geometry binary predicate list (an `sgbp` object). Each element corresponds to a school and contains a list of indices of the stations it intersects. Calling `sum()` directly on a list will throw an R error: `invalid 'type' (list) of argument`.

You must calculate the length of the list for each school to see if it intersects at least one station buffer, and then sum those logical values.

- **Corrected Code:**

```
# Option A: Using lengths() to count non-empty intersections
total_count <- sum(lengths(nearby_schools) > 0)
```

(Alternatively, accept Option B: using `st_filter` instead of `st_intersects`)

```
# Option B: Filtering the sf object directly
nearby_schools <- st_filter(schools, stations_buffered)
total_count <- nrow(nearby_schools)
```

5. (3 pts) RAG workflow explanation and limitations.

When `client$chat(...)` is called, the following sequence occurs: (1) The user's question is sent to the LLM (`llama3.1` via Ollama). (2) The LLM determines that to answer accurately it should use the retrieval tool registered via `ragnar_register_tool_retrieve()`. (3) The LLM emits a **tool call request** — a structured message naming the retrieval tool and passing the query (or a reformulation of it) as the search argument. (4) The R session executes `ragnar_retrieve()` against the DuckDB store, using vector similarity search (comparing the query embedding against stored chunk embeddings) and BM25 keyword matching to find the `top_k = 4` most relevant chunks. (5) The retrieved chunks are returned to the LLM as the tool's result, appended to the

conversation context. (6) The LLM synthesises a final answer drawing on the retrieved text rather than relying on its parametric memory alone. (7) The response is returned to the analyst.

Even with RAG, the model can still produce incorrect figures. One key reason is that **the relevant chunk may not have been retrieved**: if the correct figure appears in a passage that did not score in the top 4 (because the query embedding did not closely match the chunk's embedding, or the BM25 keyword search missed the passage), the LLM may fall back on its training data and hallucinate a plausible-sounding number. Alternatively, the LLM may misread or misquote a figure from the retrieved text, especially when the passage requires numerical interpretation.

Grading: 2 pts for the step-by-step explanation (must include: LLM decides to use tool, R session executes retrieval, retrieved chunks returned to LLM, LLM synthesises answer); 1 pt for a valid reason RAG still allows errors (missed retrieval, wrong chunk retrieved, or LLM misreads retrieved text).

End of Answers