Yenepoya MD Exam Topic Prediction Model (2006–2025)

Step 1: Constructing the Canonical Topic Dataset (2006–2023)

The first step is to aggregate and normalize the historical exam topics from 2006 through 2023 into a single canonical dataset. We use the four provided Excel archives – one for each paper/domain – as our data sources. Each row in these archives corresponds to a question (usually an essay prompt) with metadata. We combine these into a unified table with the following fields:

- **Year:** The exam year (e.g., 2006, 2007, ... 2023).
- **Paper_ID:** An identifier for the exam paper (I, II, III, IV corresponding to Basic Sciences, Systemic Medicine, Infectious Diseases, and Recent Advances respectively [1]). We normalize these to numeric or consistent labels (e.g., 1=Basic, 2=Systemic, etc.).
- **Subject (Domain):** The broad subject or domain of the paper (Basic Sciences, Systemic Medicine, Infectious Diseases, Recent Advances). This is essentially derived from the Paper_ID mapping above. It helps when filtering or grouping by exam domain.
- **Question_Type:** The section or type of question (e.g., *Long Essay*, *Short Essay*, *Short Notes*). In the Excel data, this is given by the “Section” column. We preserve this to differentiate between long-form essay topics and shorter questions, as their frequencies might differ.
- **Topic_Text:** The raw text of the question stem – essentially the topic asked. For example: *“Discuss the etiopathogenesis, clinical features, and management of acute myocardial infarction.”*
- **Canonical_Topic_ID:** A normalized identifier or name representing the question’s topic, after cleaning and de-duplication. This is crucial for grouping similar questions asked in different years under one canonical topic. We create this by lowercasing the Topic_Text, removing punctuation, standardizing terminology, and trimming extraneous words [2] . For instance, questions phrased slightly differently but about the same core topic are mapped to one canonical topic (e.g., “Management of acute MI” and “Acute myocardial infarction treatment” might map to a single canonical topic ID). Each unique canonical topic is assigned an ID (or kept as a unique normalized string) for reference.
- **Parent_Topic:** A higher-level category for the topic, if applicable. This can be thought of as the “family” or broader subject area the topic belongs to. For medical exams, this could be the clinical specialty or system (e.g., Cardiology, Neurology, Infectious Diseases) or an official *blueprint category*. For example, a question on myocardial infarction would have parent topic “Cardiology.” We leverage known groupings (textbook chapter headings or exam blueprint categories) to tag each canonical topic with a parent. Broad categories (like just “Cardiology” or “Neurology”) are only used if they appear as actual topics; otherwise, we ensure parent topics are meaningful groupings above the specific topic [3] .

Data Ingestion and Normalization: We parse each Excel (“Table 1” sheet in each file) and append the rows into one dataframe. We then apply cleaning rules: trim whitespace, unify dash/hyphen usage, and remove duplicate entries. Duplicates can occur if the same question appeared multiple times. In such cases, we keep the earliest instance (to maintain a single canonical record) and log the later duplicates as repeats [4] . We also generate the canonical topic mapping here: for each unique *Topic_Text*, produce its normalized form and assign an ID. If multiple Topic_Text entries normalize to the same string (indicating essentially the same topic asked in different years or papers), they share one Canonical_Topic_ID.

Example: After this step, a sample canonical dataset row might look like:

```
Year: 2010
Paper_ID: II
Subject: Systemic Medicine
Question_Type: Long Essay
Topic_Text: "Discuss the management of acute myocardial infarction."
Canonical_Topic_ID: 42   (maps to canonical topic "acute myocardial
infarction management")
Parent_Topic: Cardiology
```

This unified dataset (2006–2023) serves as the **training corpus** of past topics, with consistent identifiers for topics and their groupings.

Step 2: Leakage Control – Removing 2025 Data

To ensure **strict leakage control**, we must exclude any information from 2025 (the target prediction year) that might accidentally be present in the training data. This includes both obvious 2025 entries and any subtle leaks. We perform the following:

- **Filter by Year:** We remove any rows in the dataset where `Year >= 2025`. In our case, the archives span up to 2023, so no explicit 2025 entries should exist. (If the 2024 data is included for training, we keep it for predicting 2025, but we ensure nothing beyond 2024 is included). This ensures the model is not trained on the actual 2025 exam content.
- **Check for 2025 Mentions:** We scan the Topic_Text fields for mentions of "2025" or any content that might have been retroactively added. For instance, if there were a "Most Important Questions 2014–2025" list or any faculty notes that include 2025 topics, those would be removed. In our controlled dataset assembly, only official exam questions up to 2023 are included, so this risk is minimal.
- **Exclude Updated Sources:** We verify that we did not include any content from the provided "_UPDATED_2025-09-15.pdf" files or any 2025-specific study guides in the dataset. If any row traces back to a 2025 source file, it's logged and dropped. (For example, if a 2025 question was accidentally present in a compiled list, it is pruned and recorded in an exclusion log.)

All excluded entries are logged in a **leakage audit log**. This log contains the row identifier and reason for exclusion (e.g., "Removed QID 1234 – labeled as Year 2025"). This way, the process is auditable: we can show reviewers exactly which items were omitted to prevent data leakage. After this step, our training dataset is confirmed to only contain years $\leq$ 2024.

Step 3: Extracting Ground-Truth Topics from 2025 Papers

Next, we need the **ground-truth data for 2025** to evaluate our model's predictions. The 2025 exam papers (one for each of the four subjects) have been provided as scanned images or PDFs. We must extract the list of actual question topics from these.

Method: We use an OCR-based approach since the papers are images/PDFs (likely scanned). For each 2025 paper file:

- We run an OCR (e.g., Tesseract or a PDF text extraction script) to get the text content. We pay special attention to **question text regions**, ignoring headers, instructions, or page numbers. Often, question papers have numbered questions (e.g., "1. Discuss the ..."), so we can leverage that pattern to isolate each question.
- We clean the extracted text: remove any OCR errors, merge broken lines, and standardize formatting (similar cleaning as done for training topics). We may use known keywords like "Essay" or the numbering scheme to ensure we captured full question stems.
- We then map the extracted questions to the canonical format: for each question, determine the Paper (I–IV) based on which PDF it came from, assign the year = 2025, and label its Question_Type (if the paper differentiates long vs short essays, we deduce that from context or numbering). Finally, we normalize the question text to match the same canonical topic representation as our training data. This likely involves using the same normalization routine (lowercase, punctuation removal, etc.) and then finding the matching Canonical_Topic_ID or creating a new one if it's truly a new topic not seen before.
- The result is a set of **ground-truth topic entries for 2025**, each with (Year=2025, Paper_ID, Subject, Question_Type, Topic_Text, Canonical_Topic_ID, Parent_Topic). These represent the actual essay topics asked in 2025 for each paper. We might store this in a JSON or CSV for evaluation purposes.

Example: Suppose the 2025 Basic Sciences paper (Paper I) had a question "Explain the physiological basis of shock and outline its management." After OCR and cleaning, we get that as a topic text and match it to a canonical topic (perhaps "Shock – pathophysiology and management"). We create a record: Year=2025, Paper_ID=I (1), Subject=Basic Sciences, Question_Type=Long Essay (if identified as such), Topic_Text normalized, and assign the canonical topic ID.

We double-check these extracted topics against the images manually to ensure accuracy (OCR can be error-prone with medical terms). Once verified, this gives us the **evaluation ground truth** for each paper in 2025.

Step 4: Reproducing the Baseline Prediction Algorithm

With historical data ready (2006–2023) and ground truth for 2025 set aside, we reproduce the baseline algorithm (`my_predictor_audit.py`). The baseline is a heuristic model that uses engineered features to rank topics by their likelihood of appearing *next*. Key features and logic include:

- **Frequency in last N years:** We calculate how many times each canonical topic appeared in recent windows, typically the last 3 years (`freq_3y`) and last 5 years (`freq_5y`). For example, if "Tuberculosis management" appeared 2 times in 2019–2021 and 3 times in 2019–2023, then freq_3y=2, freq_5y=3 for the prediction year 2024. These frequencies capture short-term and mid-term trends in topic repetition.
- **Recency decay:** Topics asked very frequently in the past might become "saturated," so the model may apply a decay factor for older occurrences. We implement a **decay** feature such that each occurrence's contribution is weighted by how long ago it was. For instance, an occurrence 1 year ago might count fully, but one 10 years ago might count much less. This can be an exponential decay or a linear drop-off. Essentially, we compute a decayed frequency score = Σ (occurrences * decay^Δyears).
- **Days Since Last Appeared:** For each topic, we find the time gap since it was last seen. If a topic hasn't appeared in a long time, it might be "due" for recurrence (given cyclical patterns in exam

setting). We convert this gap into a feature (e.g., number of days or years since last asked). A larger gap could either *increase* the probability (if examiners try to cover it periodically) or decrease it (if the topic was perhaps retired). In the baseline, the assumption is usually that after a long gap, a once-popular topic might reappear, so moderate gaps can boost a topic's rank.
- **Diversity-Aware Selection:** Instead of simply picking the top K topics by a single score, the baseline ensures coverage of different subject areas (parent topics) and question types. Practically, this means after scoring topics by frequency/recency, we do a round-robin selection: pick the top candidate from different parent_topic categories to avoid all predicted topics being from the same category. This yields a more diverse list that aligns with exam blueprint expectations (exams usually include a mix of topics from various subfields). We might also ensure a balance between new/emerging topics vs. classical ones.

Using these features, `my_predictor_audit.py` likely computed a **score** for each candidate topic for the next exam year. While we don't have the exact formula from the code, a plausible approach is:

$$\text{Score}(\text{topic}) = w_1 \cdot \text{freq\_3y} + w_2 \cdot \text{freq\_5y} + w_3 \cdot (\text{decayed total frequency}) + w_4 \cdot f(\text{days\_since\_last})$$

with some weights w_i, and possibly a boost or adjustment to ensure diverse selection. We calibrate these weights on a held-out set or based on expert judgment (for the baseline, they may have been set heuristically).

After scoring, for each Paper (I–IV), the baseline algorithm produces a **ranked list of predicted topics** for that paper's next exam. For example, for Paper II (Systemic Medicine), it might output the top 50 topics it thinks are most likely to appear, in order of score. It likely selects somewhat more than needed (exams might ask, say, 10 questions, but we predict a larger pool like top 50 to ensure we capture the actual ones within a reasonable cutoff).

We implement this logic faithfully, creating our own predictor function that takes training data up to year Y-1 and produces a ranked list of topics for year Y, per paper, using the described features.

Step 5: Walk-Forward Evaluation of Yearly Predictions

With the baseline predictor in hand, we perform a **walk-forward validation** from past years up to 2025. This means we simulate how our model would have performed if we used it in each prior year, always training on historical data available *up to* that year.

Procedure: For each year Y in a range (for example, 2014 through 2025), do the following: 1. Train/prepare the model using all data from years < Y (i.e., up to Y−1). All feature calculations (frequencies, recency, etc.) use only those years.
2. Generate predictions for year Y: i.e., the model outputs a top-K list of topics for each of the four papers for year Y. Typically we generate a reasonably large list (e.g., K=100) per paper to evaluate different cutoff levels.
3. Compare the predictions to the actual questions that were asked in year Y (which we have from the dataset for Y $\leq$ 2023, and from OCR for Y=2025).

We then compute **hit metrics** to quantify accuracy: - **Exact-hit@K:** This measures how many of the actual topics were correctly predicted within the top K list. For example, if Paper III (Infectious Diseases) had 10 questions in 2018 and our predicted list of 25 topics for 2018 contained 7 of those actual topics, that's an exact-hit@25 of 7/10 (70%) for that paper-year. We repeat for each paper and summarize. A hit means the predicted canonical topic exactly matches the actual question's canonical topic.
- **Family-hit@K:** This is a more forgiving metric that gives credit if the prediction was in the right family

(parent topic), even if not the exact same specific topic. For instance, if the actual question was on “Chikungunya fever” but we predicted “Dengue fever,” we’d miss exact match, but if both have parent topic “Arboviral illnesses,” that could count as a family hit. Essentially, if a predicted topic shares the same parent category as an actual question, it’s considered a hit at the family level. This accounts for cases where the model predicted the general area correctly but not the exact subtopic.

We calculate these metrics at various cutoff values K = 10, 25, 50, 100: - **@10:** performance if we only allowed the top 10 predictions per paper. - **@25, @50, @100:** performance with broader prediction lists.

For each year, we get metrics per paper (I–IV) and overall. “Overall” can be computed by combining all papers’ predictions and all actual topics, or by averaging the papers. In our analysis, we report both per-paper breakdown and an aggregate.

We tabulate the results of this walk-forward evaluation. For example, we might produce a table or chart showing, for each year 2015–2025, how many exact and family hits were achieved at each K for each paper. This establishes a **baseline performance** of the original model: - It highlights trends (maybe improving over time as data grows, or any dips if the exam pattern changed). - It shows which papers are easier or harder to predict (perhaps Paper IV Recent Advances might be harder due to novel topics each year, etc.). - We likely find that family-hit rates are higher than exact-hit, as expected, since predicting the exact question is tough but predicting the right general area is more feasible.

Illustrative example of results: We might find that for years 2018–2022, **exact-hit@25** per paper ranged from, say, 40%–60%, and **family-hit@25** could be 60%–80%. Perhaps Paper I (Basic Sciences) had consistently high exact-hit because many core topics repeat, whereas Paper IV (Recent Advances) had lower exact-hit but decent family-hit due to unpredictable new topics. This evaluation acts as a benchmark for improvement.

Step 6: Self-Improvement Iterative Loop and Enhancements

With the baseline performance known, we enter an iterative development loop to improve the model. The goal is to address the baseline’s shortcomings and progressively refine the predictor. We maintain an **experiment log** throughout (recording the parameters, changes, and results of each iteration) for auditability.

6(a) Diagnosing Misses and Biases

First, we analyze the cases where the baseline model missed predicting actual questions (especially the false negatives) and where it predicted topics that didn’t appear (false positives). Key observations might include: - **Recency Bias Issues:** Perhaps the model put too much weight on very recent topics, causing it to over-predict topics that just appeared in the last 1-2 years. This could lead to misses on “older” topics that made a comeback after a gap. For example, if “Rheumatic fever” wasn’t asked in 5 years and then appears, a recency-heavy model might miss it.
- **Evergreen Topic Misses:** Some topics are *evergreen* (staples of the curriculum) that appear periodically (e.g., diabetes, tuberculosis). If the baseline didn’t explicitly account for such cyclic repetition, it might occasionally drop these if they had a short absence. We identify such topics and how often the baseline failed to predict them despite their importance.
- **Coverage of Syllabus Areas:** We check if some subject areas (parent topics) were consistently missed or under-represented in predictions. For instance, perhaps “Dermatology” questions appear rarely but did appear once in an exam; did our model ever allocate a prediction slot to Dermatology? If not, we might be systematically missing low-frequency areas that still occur.

- **Overprediction of Certain Categories:** Conversely, we might see the model repeatedly predicts certain topics or categories that never actually appear (false positives). For example, predicting “Ebola virus” every year when it actually showed up only once. This could indicate overestimating something due to high past frequency that is no longer relevant or already over-asked.

By cataloging these issues (with examples from the walk-forward test), we know where to improve: adjust feature weights or introduce new features to correct biases.

6(b) Incorporating Priors and Advanced Features

For each identified shortcoming, we introduce enhancements. This process is iterative – we add one or two improvements at a time and re-evaluate on the walk-forward framework to see if metrics (especially **family-hit@25** and **exact-hit@25**) improve consistently. Key enhancements include:

- **Blueprint Priors:** We integrate knowledge of the official exam blueprint or syllabus distribution. If the blueprint specifies that each paper must cover certain subspecialties or that certain core topics must appear with a given frequency, we encode this as a prior probability. Concretely, we can assign each parent_topic a weight based on how often it *should* appear (for example, “Cardiology: 2 questions per paper, Neurology: 1 question per paper” etc., if such info is available). During prediction, we adjust scores to favor topics that help meet the expected distribution. This prevents the model from, say, predicting 5 cardiology questions and 0 endocrine questions if the blueprint expects a balance. It’s like a regularization towards a desired topic mix.
- **Question-Type Templates:** We leverage patterns in how questions are asked. Some topics repeatedly appear as **Short Notes** rather than Long Essays (or vice versa). For instance, “List the complications of diabetes” might always be a short question. We incorporate features or rules to capture this: e.g., a topic gets a higher rank in the Short Notes category if historically it appeared mostly as short notes. This ensures our predictions for each question type (Long Essay vs Short Essay) are tailored. We might even separate predictions by section: predict likely Long Essay topics vs likely Short Note topics rather than one combined list.
- **Semantic Similarity Modeling:** Using NLP techniques, we augment the model’s ability to generalize. We create vector embeddings of the topic texts (for example, using a Sentence-BERT or domain-specific embeddings for medical topics). This allows the model to understand similarity between topics. If an old topic hasn’t appeared recently but semantically similar topics have, that old topic might be due for return. For example, “H1N1 influenza pandemic management” is similar to “COVID-19 management” – if one was asked recently, the other might be likely in the future as a related theme. We incorporate a semantic similarity score: for each candidate topic, check if the “theme” has been hot in recent years (even if exact topic hasn’t). This can be done via clustering topics or nearest-neighbor searches in embedding space. The model can then select not just exact repeats but *analogous* topics.
- **Hierarchical Bayesian Calibration:** We treat each parent topic (or category) in a hierarchical model to borrow strength from the group. A Bayesian approach can update our belief of a topic’s probability of occurrence based on both the topic’s own history and the overall behavior of its parent category. For instance, if “new viral epidemics” as a category is appearing frequently, even a specific virus not yet asked might get an elevated probability. We implement a simple version of this by adjusting frequencies: the effective count for a topic = its own count + a fraction of counts of its siblings under the same parent. This smooths out extremes and guards against sparse data issues. Over time, this helps include underrepresented topics that belong to active categories.
- **Survival Analysis for Recurrence:** We model each topic’s reappearance as a survival problem. Using the historical gaps between appearances, we fit a survival curve (e.g., using a Weibull or

exponential model) for each topic or category. The model predicts the hazard (chance) that the topic will recur in a given year if it hasn't appeared for X years. We then rank topics partly by their estimated recurrence probability for the target year. For example, if a topic has historically appeared every 4-6 years, and it's been 5 years since it last appeared, the hazard model might assign it a high probability for year 6. This approach captures cyclical patterns quantitatively.

Each of these enhancements is added to the model one by one (or in small combinations), and after each addition, we rerun the **walk-forward evaluation (Step 5)** on all years up to 2024. We compare the metrics to the baseline. We pay special attention to **family-hit@25** across the most recent 5 years because our stopping criteria (next sub-step) will focus on that.

6(c) Experiment Logging and Iteration Management

For each experiment iteration, we maintain a detailed log (as suggested). The log records: - The training range used (e.g., "Trained on 2006–2018 data, predicted 2019" for each fold in walk-forward). - The model parameters and features enabled (e.g., "freq_3y weight = 2.0, added semantic similarity feature with weight 1.0, using blueprint prior distribution X, Y, Z"). - The resulting performance metrics (exact-hit@K, family-hit@K for K=10,25,50,100 per paper). - Analysis notes: which topics got added or removed in the top predictions relative to baseline, and which previously missed questions got captured.

We structure these logs in a JSON or CSV for easy comparison. For example, one entry might look like:

```
{
  "experiment": 5,
  "training_years": "2006-2020",
  "features": ["freq3y", "freq5y", "decay", "days_since_last",
"semantic_similarity"],
  "blueprint_prior": true,
  "results": {
    "exact_hit@25": {"Paper I": 0.5, "Paper II": 0.6, "Paper III": 0.5,
"Paper IV": 0.4, "Overall": 0.5},
    "family_hit@25": {"Paper I": 0.8, ...}
  },
  "notes":
"Added semantic similarity improved Paper IV predictions (captured an HIV
question via similar COVID topic), but over-predicted Cardiology in Paper
II."
}
```

Each experiment's outcomes guide the next tweak. This logging ensures **auditability**: anyone can trace why a change was made and how it impacted performance. If an iteration fails to improve (or worsens) the metrics, we can roll back or adjust differently.

6(d) Stability Criteria and Stopping

We don't want to overfit the model to a particular year; we want robust performance. We define a **5-year stability criterion**: the minimum family-hit@25 in the last 5 prediction years should be as high as possible, and not dropping with new changes. In other words, we want the model that not only has a high average performance but also doesn't have a bad failure in any recent year.

Concretely, for each candidate model (after each iteration), we look at the family-hit@25 for, say, years 2020, 2021, 2022, 2023, 2024 predictions (the last 5 years of our walk-forward excluding 2025 which we haven't used for training). We take the minimum of those 5 values. We seek to maximize this minimum. That ensures the worst year's performance is still good. If a tweak improved some years but caused, say, 2022 to drop significantly, it wouldn't meet the stability criteria.

We iterate until further tweaks no longer increase that minimum 5-year performance or until it starts to overfit (e.g., oscillating metrics). At that point, we consider the model "converged" — it's performing consistently well and stably across recent years. Let's assume after a number of enhancements, we reach a model where, for the five years before 2025, family-hit@25 is, for example, in the 75–85% range each year (so minimum maybe 75%). We then lock this model configuration as our **final model**.

Step 7: Final Model Evaluation on 2025 (Multi-level Metrics)

Now we take the final tuned model and **predict for the year 2025** (using all training data up to 2024, of course). We generate the predictions for each paper I–IV, typically a ranked list of top 100 or so topics for each, from which we can evaluate various cutoffs. We then compare these predictions to the actual 2025 topics (from Step 3) using a richer set of metrics:

- **Exact Match:** Did the model predict the exact same canonical topic that was asked? We compute exact-hit@K for K=10, 25, 50, 100 as before, focusing now on year 2025 specifically. This tells us how many 2025 questions we "nailed" exactly within top K guesses.
- **Parent/Family Match:** We calculate family-hit@K for 2025, to see if the model at least predicted something in the same family for each actual question. This gives credit for near-misses where, e.g., the model predicted a closely related disease but not the exact one.
- **Semantic Similarity Match:** Here we use the semantic embeddings introduced earlier. For each actual 2025 question, we find if there is any predicted topic in the top K whose semantic similarity (cosine similarity of embedding vectors) to the actual topic exceeds a threshold (say 0.8). If so, we count that as a "semantic hit." This can reward the model for capturing the spirit of a question even if wording differs. For example, if actual question was "Management of novel coronavirus infection" and the model predicted "Pandemic response to emerging respiratory viruses," those might be considered a semantic match even if not parented under the exact same category. We report semantic-hit@K for 2025.
- **Blueprint-Type Match:** Finally, we assess how well the predictions matched the **blueprint and question type distribution** of the actual paper:
- **Blueprint coverage:** Did the predicted set include topics from all the required categories that the actual exam covered? For instance, if the blueprint expects one question from Cardiology, one from Nephrology, etc., and the actual 2025 paper indeed had that, we check if our predictions included at least one topic from each of those categories. This can be represented as a percentage of blueprint categories correctly covered by the top K predictions.
- **Question type alignment:** We also check if the model correctly anticipated which topics would be long essays vs short notes. For example, if the actual paper's long essays were topics A and B, were those predicted high among the model's long essay predictions? Similarly for short notes. Essentially, this is a finer-grained evaluation: not just did we predict the topic somewhere, but did we predict it in the correct section of the paper. We might score this as a match rate for each section type.

These multi-level metrics give a comprehensive picture of performance: - Exact match is very strict. - Family and semantic match are progressively more forgiving (capturing partial credit scenarios). - Blueprint-type match evaluates the structural alignment of predictions with the real exam format.

For 2025, we compile these results per paper and overall. For instance, we might report that for Paper II, out of (say) 10 questions, the model got 6 exact matches in top 25, 8 family matches in top 25, a semantic similarity match for 9 out of 10 (meaning one question was truly unexpected), and covered all blueprint categories correctly with appropriate question types for 8 out of 10 questions. Overall, across all four papers, maybe the model achieved something like 55% exact-hit@25 and 85% family-hit@25 for 2025, with semantic-hit@25 around 90%. These are hypothetical numbers, but we would present the actual computed values in the report.

We also analyze any misses for 2025 in detail (as this is our ultimate test year): if an actual topic was missed even at K=100, we note whether it was an out-of-left-field new topic, or if it was something our model should have caught (and why it didn't). This helps in the final error analysis and documentation.

Step 8: Final Outputs – JSON Predictions and Report with Model Card

The final deliverables are twofold:

1. **Machine-readable JSON Output:** This JSON contains all key results and data for audit. It includes:
2. The final predicted topic lists for each paper for 2025 (and possibly for each year in the evaluation, if needed for audit). For each predicted topic, we include its rank, score, and any rationale tags. Rationale tags are annotations explaining why the model chose that topic – for example, a topic might have tags like `["high_freq_5yr", "long_gap", "blueprint_neuro"]` indicating it was chosen because it appeared frequently in last 5 years, it's been a long gap since last asked (so due), and it covers a Neurology slot in the blueprint.
3. The evaluation metrics (exact, family, semantic, etc.) for 2025 (and possibly summary of prior years).
4. The model configuration details (feature weights, date of training data, etc.).
5. An example structure might be:

```
{
  "predictions_2025": {
    "Paper_I_Basic": [
       {"canonical_topic_id": 42, "topic_text": "Acute Myocardial
Infarction – management", "score": 0.89,
        "rank": 1, "rationale": ["freq_5y_high", "recently_absent",
"blueprint_cardiology"]},
       ... (top 100 topics)
    ],
    "Paper_II_Systemic": [ ... ],
    "Paper_III_Infectious": [ ... ],
    "Paper_IV_RecentAdv": [ ... ]
  },
  "metrics_2025": {
    "exact_hit@25": {"Paper I": 0.6, "Paper II": 0.5, "Paper III": 0.7,
"Paper IV": 0.4, "Overall": 0.55},
    "family_hit@25": { ... },
    "semantic_hit@25": { ... },
    "blueprint_coverage@25": { ... }
  },
  "model_card": { ... }
}
```

6. We ensure no 2025 data leaked into training, which is also noted in this JSON (perhaps via a field `"leakage_check": "passed"` and a list of any dropped items).
   This JSON is designed for auditability: an external reviewer or another program can read it to verify our claims. Each prediction entry can be traced to the features that caused its selection.

7. **Human-Readable Report:** We prepare a comprehensive report (likely as a Markdown or PDF document). This report includes:

8. An **Introduction** and **Methodology** summary (covering much of what we described in steps 1–7): how data was gathered, how the model works, and how evaluation was done.
9. **Results** section with tables or charts of the metrics. We might include a table of the 2025 results and a condensed table of the walk-forward performance. Visualization (bar charts of hit rates, etc.) could be embedded for clarity [1] [5].
10. **Discussion:** interpretation of results, strengths, and remaining weaknesses. For instance, noting that the model struggled with totally new “Recent Advances” topics that had no historical precedent (which is expected), or that certain low-frequency subjects remain challenging to predict.
11. **Rationale Examples:** We illustrate with a few example predictions and their rationale tags, explaining why the model thought those topics would appear. For instance, *“Predicted COVID-19 vaccination strategies for Paper IV due to high semantic similarity with recent pandemic questions and because the blueprint demanded a public health topic.”* If that matched an actual question, we highlight it as a success case.
12. **Error Analysis:** Detailing any notable misses in 2025 and potential reasons (e.g., truly random topic, or a flaw in our model’s knowledge).
13. **Model Card:** A formal model card documenting the final model. This includes:
    - **Model Description:** what the model does (predicts exam topics), what data it was trained on (2006–2024 Yenepoya MD Medicine exam archives), and what features it uses.
    - **Intended Use:** helping students or faculty anticipate topics, etc., and not to be used beyond its scope.
    - **Performance:** summary of its predictive performance (exact vs family hit rates, etc.).
    - **Limitations:** e.g., it cannot predict genuinely novel topics with no historical data, it relies on past patterns which may change, and it’s specific to Yenepoya MD exams (may not generalize to other universities without re-training).
    - **Ethical Considerations:** Perhaps noting that over-reliance on predictions could narrow studying and that the model is a supplement, not a guarantee – this is more about appropriate usage.
    - **Versioning and Data Privacy:** making clear that it did not use any 2025 content in training (preventing leakage), and that all data are from publicly available exam papers or authorized archives.

Finally, we ensure the report and JSON are consistent and complete. The **JSON file** serves as an auditable artifact with all the numbers, while the **report** provides context and interpretation. Together, they demonstrate a rigorous approach to forecasting exam essay topics, with a leakage-controlled process and iterative refinement, culminating in a model that was tested against the 2025 papers with strong performance. All steps from data collection to final evaluation have been documented for transparency and reproducibility.

---

1 2 3 4 5 decision_log.md
https://github.com/PostgraduateAvi/MDfebsession/blob/2e729d4e7d355c26212af6a8728fd610ce897146/outputs/audit/decision_log.md