Task-B Enhancement Plan

Healthcare Named Entity & Event Extraction

---

1. Caching Mechanisms for Processed Documents

Objective

Reduce processing latency and computational cost by reusing results for documents that have been previously processed, either in full or in part.

Approach

- Implement a **two-tier caching system**:
    - **In-Memory LRU Cache** for hot results.
    - **Persistent Redis/SQLite Store** for warm/cold data across sessions.
- Use **content hashing (SHA-256)** of normalized text combined with `rule_version` and `model_version` as the cache key.
- Enable **partial-document caching** by segmenting documents into logical sections (e.g., *History*, *Medications*, *Labs*).
- Set **TTL expiry** and allow **manual purge** via an admin endpoint.

Implementation

- Create a `CacheService` integrated into `/api/extract` and `/api/upload`.
- On cache hit → return stored JSON results immediately.
- On cache miss → process normally, store results with metadata (created_at, expires_at).
- Integrate Prometheus metrics:
    - `cache_hit_rate`, `cache_miss_rate`, `avg_latency_saved_ms`.

Quantified Benefits

- **Latency Reduction**: 50–70% for repeat/overlapping documents.
- **CPU-Hour Reduction**: ~40% for workloads with ≥30% repetition.
- **Target Cache Hit Rate**: ≥60% for batch ingestion scenarios.

2. Pre-trained Domain-Specific Language Models (Hybrid Pipeline)

Objective

Enhance extraction accuracy by combining the precision of rule-based methods with the contextual understanding of transformer-based language models trained on biomedical corpora.

Approach

- Integrate **ClinicalBERT**, **BioBERT**, or distilled equivalents for NER and event extraction.
- Apply **hybrid fusion**:
    - **Rule-First**: Accept high-confidence rule matches, then expand or correct with ML model predictions.
    - **Confidence Fusion**: Merge outputs using calibrated thresholds (`thresholds.json`).
- Use **ONNX export** and **INT8 quantization** for efficient CPU inference.

Implementation

- Add `HybridPipeline` wrapper to execute rules and model in parallel.
- Store model weights in `/models/` directory; load at app startup.
- Enable or disable via config flag: `enable_ml_model = true|false`.
- Maintain a calibration set for periodic threshold tuning.

Quantified Benefits

- **NER Macro-F1 Improvement**: +8–12 percentage points for MEDICATION, DISEASE, SYMPTOM.
- **Event Trigger Recall**: +10–15% with <5% precision drop.
- **Latency Target**: ≤250 ms/doc on CPU after quantization.

---

3. Distributed Processing for Large Collections

Objective

Scale the system horizontally to handle millions of documents efficiently.

Approach

- Use **Celery + RabbitMQ** (or Redis Queue) for distributed task execution.
- Partition workloads at document or section level.
- Ensure **idempotency** by using content-hash job keys.

Implementation

- Bulk upload handler enqueues file references to a message broker.
- Worker processes load the same extraction pipeline as the real-time API.
- Persist results to the DB or object store; index for search and analytics.
- Implement fault tolerance: retries with exponential backoff, dead-letter queue.

Quantified Benefits

- **Throughput**: 100–150 docs/min per worker (2 KB average).
- **Scalability**: Linear up to 50 workers with <5% coordination overhead.
- **SLA Compliance**: 95% of jobs within batch processing window.

---

4. Relation Extraction Between Domain Entities

Objective

Identify clinically relevant relationships between extracted entities (e.g., *MEDICATION—treats→DISEASE*).

Approach

- **Pattern-Based Candidate Generation** using domain-specific lexical cues (e.g., “treated with”).
- **ML-Assisted Classification** for relation validation using entity type, context embeddings, and dependency parsing features.

Implementation

- Add `relation_extractor.py` after NER & event extraction.
- Output graph-ready JSON: `{ subject_id, predicate, object_id, evidence, confidence }`.

- Provide `/api/relations` endpoint for UI consumption.

Quantified Benefits

- **Precision**: ≥80% for top 5 relation types.
- **Recall**: ≥70% on curated evaluation set.
- **Analyst Time Saved**: ~30% less manual chart review.

---

5. Temporal Reasoning for Event Sequencing

Objective

Accurately order and normalize events to produce patient timelines for longitudinal analysis.

Approach

- Normalize both **absolute** and **relative** time expressions (e.g., “post-op day 2”).
- Handle event intervals (start–end) and instantaneous events (admission, surgery).

Implementation

- Add `temporal_normalizer.py` after event extraction.
- Return enriched event objects with `t_start`, `t_end`, `confidence`.
- Provide `/api/timeline` endpoint; render via `vis.js` timeline component.

Quantified Benefits

- **Correct Sequencing**: ≥90% accuracy in test timelines.
- **Ambiguity Resolution**: ≥85% for relative time expressions.

---

6. Visualization Tools for Entities, Relations & Timelines

Objective

Improve user insight and trust via interactive visualizations.

Approach

- **Entity–Relation Graph** using D3.js.
- **Event Timeline** with zoom and filter controls.
- **Analytics Dashboard** with co-occurrence heatmaps, latency charts, cache hit rates.

Implementation

- Extend `/api/export` to output graph/timeline JSON.
- Add PHI-safe evidence tooltips in UI.
- Implement role-based access control for sensitive views.

Quantified Benefits

- **Finding Recall**: +15–20% compared to text-only review.
- **Interpretation Time Reduction**: 25–40% in user tests.

---

7. Phased Roadmap

Phase 0 (Week 0–1) – Baseline & Caching

- Implement caching layer, integrate metrics, measure baseline KPIs.

Phase 1 (Week 2–4) – Hybrid Model Integration

- Add transformer model, calibrate fusion thresholds, run A/B tests.

Phase 2 (Week 5–7) – Distributed Processing

- Deploy Celery workers, tune throughput, enhance observability.

Phase 3 (Week 8–10) – Advanced Analysis & Visualization

- Deploy relation extraction, temporal reasoning, and rich visual dashboards.

---

8. KPIs & Acceptance Criteria

Quality

- +8–12 pp Macro-F1 for MEDICATION/DISEASE/SYMPTOM.
- ≥0.80 F1 for event triggers.

Performance

- p95 latency ≤600 ms for short notes.
- Cache hit rate ≥60%.
- Batch throughput ≥100 docs/min/worker.

User Experience

- ≥4/5 satisfaction score for clarity of highlights, graphs, and timelines.

---

9. Risks & Mitigations

- **Model Drift** → Schedule evaluations, enable quick rollback via config flags.
- **PHI Exposure** → Redaction pipeline, encrypted storage, RBAC.
- **Cost Spikes** → Autoscaling caps, off-peak batch scheduling.
- **Integration Bugs** → Shadow testing, blue/green deployments.

---

10. Integration Discussion

The proposed enhancements will integrate seamlessly into the **existing Flask-based Healthcare NER + Event Extraction pipeline** as follows:

- **Caching** → Introduced as a pre-processing check in `/api/extract` and `/api/upload`, bypassing tokenization and model inference on cache hits.
- **Hybrid Model Pipeline** → Wrapped around existing rule-based extraction in `HybridPipeline`, with configurable toggle and calibration logic for merging outputs.
- **Distributed Processing** → Bulk ingestion path modified to push jobs into a Celery queue; worker containers run identical extraction code to ensure consistency between real-time and batch modes.
- **Relation Extraction** → Added as a post-processing step after entity/event extraction, feeding into both the API (`/api/relations`) and frontend graph view.
- **Temporal Reasoning** → Appended after event extraction; produces normalized timelines for `/api/timeline` and feeds timeline visualization in the UI.
- **Visualization Tools** → Frontend templates extended with new tabs for graph and timeline views; backend extended to provide JSON data feeds for these components.
- **Metrics & Monitoring** → All components emit metrics (latency, throughput, hit rates, F1 scores) to Prometheus; Grafana dashboards updated for new KPIs.

Conclusion

This enhancement plan not only addresses performance, accuracy, and analytical depth but also ensures that each feature is **practically integrated** into the current architecture without disrupting existing workflows. The phased approach minimizes risk and maximizes measurable impact.