

Ideas for Michael to translate a completed Neo4j graph into numbers

It is up to you to align this with your research objective. I am just kind of thinking loudly of what is possible and what is not (to convert into metrics) or have as a recommendation and guideline for future researcher to-do lists.

1. Use of Knowledge Graph

- a. **Legal Query Answering:** Enable semantic search or question answering over legal documents using the graph (e.g., "Which laws relate to zoning violations in Easttown?").
- b. **Inconsistency Detection:** Identify contradictions or overlaps in-laws, such as conflicting clauses, undefined terms, or redundant statutes.
- c. **Legal Amendment Tracking:** Visualize how specific laws have changed over time and what entities (e.g., agencies, sections) were impacted.
- d. **Policy Gap Analysis:** Reveal missing connections or unexplained relationships in a legislative framework
- e. **Support Legal Drafting:** Suggest clauses or references by identifying common graph patterns in similar regulations.
- f. **Regulatory Impact Analysis:** Map dependencies across legal domains to understand how changes in one regulation might cascade across others.

2. Validation of KG

- a. **Ground Truth Comparison:** Use a manually annotated gold standard set of legal documents. Measure **precision, recall, and F1 score** for entity extraction and relationship detection.
- b. **Consistency and Connectivity Checks:** Use graph validation rules (e.g., "Every amendment node must link to a clause node") to test internal consistency.
- c. Analyze **orphan nodes, cycles, or disconnected components** that may signal extraction errors.
- d. Reasoning Validation: Check for **semantic completeness** (e.g., Can the graph answer all questions from a predefined legal QA dataset?).

3. Graph Metrics

- a. **Node and edge count**
- b. **Average degree** (connectivity)
- c. **Diameter** or **shortest path length** (navigability)
- d. **Clustering coefficient** (local density)
- e. **Graph entropy** (diversity/complexity)

4. Usability Metrics

- a. **Query response time** in Neo4j
- b. What else?

Ideas of how to validate

What Are Graph Validation Rules?

Generally these rules define what must, must not, or should be true in the graph. For example:

- a. Every **Amendment** node **must** be connected to a **Clause** node.
- b. A **Law** node **must have** at least one **Jurisdiction**.
- c. A **LegalTerm** **must not** point to itself (no self-loops).
- d. No **Section** node **should be disconnected** (must be part of some law).

A lot of these can be done using cipher queries in Neo4j:

For example:

*nodes not linked to any **Clause**:*

```
MATCH (a:Amendment)
WHERE NOT (a)-[:MODIFIES]->(:Clause)
RETURN a
```

Disconnected components:

```
MATCH (n)
WHERE NOT (n)--()
RETURN n
```

*All **law** nodes have a **Jurisdiction**:*

```
MATCH (l:Law)
WHERE NOT (l)-[:HAS_JURISDICTION]->(:Jurisdiction)
RETURN l
```

Whatever the results, these rules are just are to justify the integrity of the generated KG as they may expose limitations in LLM extraction pipeline and, at the same time, provide a quantitative evaluation (e.g., number/percentage of nodes violating rules). Finally, they may support reproducibility and auditing.

Can we evaluate LLM-generated KG with Precision/Recall/F1

Based on:

Entities
Relations
Triples

Can we compute metrics at the above levels? I think this might be possible (but it all depends on what you want to do. This is just a generic way of seeing others approaching in some graph works

Example Legal Sentence:

"Section 12 of the Data Protection Act is amended by the Cybersecurity Reform Act passed in 2023."
--

Based on the above example, we can manually label it

Entities

Entity	Type
Section 12	Section
Data Protection Act	Law
Cybersecurity Reform Act	Law
2023	Date

Relations:

Subject	Predicate	object
Section 12	Part_of	Data protection act
Cybersecurity Reform Act	amends	Section 12

Cybersecurity Reform act	passed_in	2023
--------------------------	-----------	------

For the above- the constructed ground trust triples might be something like:

(Section 12, PART_OF, Data Protection Act) (Cybersecurity Reform Act, AMENDS, Section 12) (Cybersecurity Reform Act, PASSED_IN, 2023)

Hence, we can now consider the above as our **gold standard**

Therefore, if your LLM-based extraction also yields the same triple, we can count this as **True Positives**. If it misses any, those are **False Negatives**, and if it extracts extra incorrect triples, those are **False Positives**.

Once we get the count of:

True positive
False Negatives
False Positives.

We can easily calculate:

Metric	desc	how
precision	% of predicted triples that are correct	Compare LLM+KH output to the gold standard.
Recall	% of gold standard triples captured by your KG	Same as above
F1 score	Harmonic means of precision and recall	This is easy once we have the above

These are the formulas used in most ML projects:

precision =	$TP / (TP + FP)$
recall =	$TP / (TP + FN)$
f1-score=	$2 * (precision * recall) / (precision + recall)$

The end conclusion of doing the above test is something like:

Based on a manually annotated 15 legal sentences to create a ground truth set of 42 triples. The knowledge graph produced 45 triples, of which 38 were correct (TP), 7 were incorrect (FP), and 4 correct triples were missed (FN). This resulted in a precision of 84.4%, recall of 90.5%, and F1 score of 87.3%.

This kind of evaluation proves your KG is not only complete and connected — it's also **accurate**. It shows:

- How well your LLM understood the legal text
- How clean and semantically correct your graph is
- That your method is **scientifically valid and replicable**

Now, how do you do all of that? I think it should not be a problem for you to either build those gold standards manually or extract triples from Neo4j and compare them against your annotated test set using Python scripts.

More info on metrics I have seen and what they are

1. Structural Quality (How well-formed is the graph?)

These are just examples that may or may not be applicable to your case. The idea is to tell if the graph is sparse, overly fragmented, or missing links between entities.

metric	desc	how
Average degree	Avg number of relationships per node	<code>MATCH (n) RETURN avg(size((n)--()))</code>
Clustering coefficient	Measures local connectedness	I have seen done using NetworkX or Neo4j algorithms plugin
Connected components (not sure if this is applicable to your case)	Number of subgraphs; fewer is better in most KGs	<code>CALL algo.connectedComponents(..)</code>
Graph diameter	Longest shortest path between nodes	Reflects navigability
Orphan nodes	Nodes with no connections	<code>MATCH (n) WHERE NOT (n)--() RETURN count(n)</code>

2. Semantic Validity (Is it logically coherent based on the domain ontology?)

Metrics	desc	how
Relation type validity	% of triples conforming to ontology schema	May be just use rule base approach
Type consistency	Nodes/edges have the correct types	Check <code>node:Label</code> assignments against ontology
Domain/Range correctness	Predicates connect the correct types of subjects and objects	e.g., <code>PASSED_IN</code> should never point to a <code>Person</code>

3. Usability

Metrics	desc	how
Query latency	Time to respond to common Cypher queries (Time in ms)	Log time for standard legal info lookups (example: Use

	<i>or s for query to complete)</i>	<i>PROFILE or EXPLAIN in Neo4j)</i>
<i>Query expressiveness</i>	<i>Can users get answers from the KG?</i>	<i>Evaluate task success rate, for example, legal questions</i>
<i>User feedback/satisfaction</i>	<i>This may be done in future research</i>	<i>Get expert ratings on accuracy and usefulness</i>

The end result can be something like:

metric	value	interpretation
precision	84%	Most extracted triples were correct
Recall	91%	A few correct triples were missed
Node count	1,256	High entity coverage
Avg degree	2.4	Nodes are moderately connected
Clustering coeff	0.37	Good local consistency
Query time (ms)	75	Fast retrieval

Regards,
Dr. Elbasheer