Project Outline (v1)

Spring 2025

1. Project Overview

1.1 Objective

This project aims to evaluate the accuracy of LLM and LLM + Knowledge Graph in detecting semantic misalignment and specialized errors using synthetic customer review data for power tools.

1.2 Problem Statement

a. **Semantic Misalignment**

i. Definition: Semantic misalignment occurs when a word or phrase has multiple possible meanings, leading to misinterpretation. This issue is common in user reviews, where ambiguous language makes it difficult to determine the intended meaning.

ii. Example: “This tool is sharp.”

- Meaning 1: The tool has a physically sharp edge.
- Meaning 2: The tool performs exceptionally well (metaphorical meaning).

b. **Specialized Error Detection**

i. Definition: Specialized errors refer to user misunderstandings or incorrect assumptions about technical specifications, often contradicting documented product guidelines. These errors involve temperature ranges, voltage requirements, torque limits, and operational conditions.

ii. Examples:

- **Charging Temperature Error:**
  - User Review: "I charged my power tool at 50°C, but the battery overheated!"

    - Bosch Specification: Maximum charging temperature is 45°C → Error Detected.
- **Voltage Mismatch Error:**
    - User Review: "This 12V battery should handle steel beams, but it didn't work!"
    - Bosch Specification: Minimum 18V required for steel beams → Error Detected.

1.3 Expected Outcomes

a. Assess LLM performance in detecting semantic and specialized errors.
b. Determine whether LLM + Knowledge Graph improves accuracy.
c. Ensure customer review data remains error-free for internal analysis and applications.

2. Project Execution Flow

2.1 Flow of the Project

a. Create Synthetic Data

To ensure diversity in synthetic data, we will test different prompt strategies to generate **Semantic Misalignment Errors** and extract technical specifications from Bosch user manuals to create **Specialized Errors data** (The design methodology will be discussed in Section 3.2).

b. Build a Knowledge Graph

i. Construct a Neo4j knowledge graph using technical data from Bosch's website.

ii. Develop a knowledge base (dictionary, database, or graph) with valid numeric ranges and canonical terminology.

c. Use LLM for Error Detection

i. The Role of LLM and Knowledge Graph in Error Detection

- LLM is to interpret user reviews and detect Semantic Misalignment & Specialized Errors.
- The Knowledge Graph provides the technical specifications and error standards needed by LLM, ensuring that technical error detection is based on accurate reference data.

ii. LLM and KG Error Detection Process

- **LLM Analyzes User Reviews**
  - LLM first reads the review and identifies potential semantic and specialized errors.
  - If a word or phrase in the review has multiple possible meanings, it is flagged as a Semantic Error.
- **LLM Cross-Checks Technical Data with KG**
  - LLM extracts technical details from the review (e.g., temperature, voltage, torque).
  - KG provides the corresponding technical specifications, such as "Permitted charging temperature range: 0-45°C."
- **LLM Determines Compliance with Technical Standards**
  - If the review contradicts the technical data from the KG, it is flagged as a Specialized Error, along with a violation explanation.
- **Output Error Flags and Explanations**
  - LLM generates an Error Flag indicating whether the review contains semantic or specialized errors.
  - Provides a detailed error explanation, specifying the type of error and how it violates predefined regulations.

d. Parse and Store LLM Output

i. Extract JSON responses from LLM and store results in structured columns.

ii. Generate the final output CSV file with two new columns:

- “ErrorFlag” (Yes/No) – Indicates whether an issue was detected.
- “Explanation” – Provides reasoning and possible corrections.

2.2 Diagram

Please refer to the project flow diagram file.

3. Use Case: Synthetic User Reviews and Q&A

User reviews or Q&A forums often mix colloquial references (nicknames, partial product names) with factual claims about performance. This is rich for detecting semantic mismatches (the user calls it “Bosch Impact Pro #X” instead of the official name) and domain errors (“I used it at 200 Nm….”).

3.1 What to Include

a. **Customer Comments:** “I tested the Bosch Impact Wrench 300, but it’s actually called Impact Wrench 305 Pro. I used 250 Nm torque with no issues.”
b. **Rating and Use-Case:** “I love my new Cordless Drill DX2, but it drains the 18V battery in 5 minutes at full speed.”
c. **Misinformation:** “You can use a 12V battery with the 18V driver if you tape it.” (domain error)

3.2 How to Generate Synthetic Data and Incorporate Errors

a. **Semantic Misalignment Data**

We will test various prompting strategies to generate user reviews containing ambiguity:

i. General Prompt Testing

- Example Prompt: "Generate a review that includes an ambiguous word with multiple meanings."
- Example Review: "This hammer sounds solid, but it doesn’t hit as hard as expected."
    - Semantic Misalignment: "Sounds solid" could refer to actual sound or structural integrity.

ii. Context-Based Ambiguity Testing

- Example: The word "grinder"
    - Context 1: "This grinder has been running all day." (Refers to a power tool.)
    - Context 2: "This grinder has been working tirelessly." (Refers to a hardworking person.)

iii. Generating 20% Semantic Misalignment Data

- We will instruct the LLM to generate 20% of user reviews containing semantic misalignment to increase data diversity.

b. Specialized Errors Data

We will combine different product operating conditions and error cases to create diverse technical error scenarios:

i. Charging Mode Errors

- Example: Users attempt to charge batteries beyond the allowed temperature range, leading to failures.

ii. Operating Mode Errors

- Example: Users complain that the tool lacks sufficient torque, but in reality, they are using the wrong power setting.

iii. Storage Mode Errors

- Example: Users store tools in extreme temperature environments, causing performance degradation.

3.3 Example Data and Knowledge Graph

a. Example of Synthetic Data (Input)

Please refer to the synthetic data file.

b. Example Knowledge Graph (Including Scenarios)

i. Please refer to the JSON example file. It demonstrates how to store canonical part/ tool data, synonyms, base torque ranges, and scenario-based exceptions. In practice, this would live in a graph database (e.g., Neo4j).

ii. For factual checks, we will define a dictionary (or knowledge graph) that stores:

- Canonical part names and their synonyms.
- Contextual knowledge, such as the allowed torque range for each part.

```
{
 "nodes": [
  {
   "id": "Tool_1001",
   "labels": ["PowerTool"],
   "properties": {
    "canonicalName": "Cordless Drill DX2",
    "synonyms": ["Cordless Dril #DX2", "Drill DX2"],
    "powerType": "18V",
    "recommendedTorqueRangeNm": [30, 50],
    "speedRangeRPM": [0, 1500],
    "allowedScenarios": ["Standard"],
    "safetyNote": "Wear eye protection"
   }
  },
  {
   "id": "Tool_1002",
   "labels": ["PowerTool"],
   "properties": {
    "canonicalName": "Engine Support Bracket XYZ",
    "synonyms": ["Bracket A", "Support A", "Engn Bracket A", "Support A Bolt"],
    "recommendedTorqueRangeNm": [1200, 1500],
    "scenarioExceptions": [
     {
      "scenario": "ColdEnvironment",
      "extendedTorqueNm": 2000
     }
    ]
   }
  },
  {
   "id": "Tool_1003",
   "labels": ["PowerTool"],
   "properties": {
    "canonicalName": "18V Impact Wrench 200",
```

```
      "synonyms": ["Cordless Impact #XYZ", "Impact Wrench Model 200"],
      "recommendedTorqueRangeNm": [200, 400],
      "speedRangeRPM": [0, 2800],
      "allowedScenarios": ["Standard", "Continuous"]
    }
  },
  {
    "id": "Tool_1004",
    "labels": ["PowerTool"],
    "properties": {
      "canonicalName": "Angle Grinder X12",
      "synonyms": ["Angle Grndr X12"],
      "powerType": "230V",
      "recommendedSpeedRangeRPM": [0, 11000],
      "allowedScenarios": ["Standard"]
    }
  }
 ],
 "relationships": [
  {
    "startNode": "Tool_1002",
    "endNode": "Tool_1003",
    "type": "IS_RELATED_TO",
    "properties": {
      "reason": "Both can be used for engine assembly tasks"
    }
  }
 ]
}
```

c. Example of Synthetic Data Output with Flags and Corrections

Below is a **sample output** after an **LLM-based** or **knowledge-graph-driven** check. The system reads each row, resolves synonyms, verifies torque and usage scenario, and flags issues.

File Name: nuanced_power_tools_with_scenarios_output.csv

Tool ID	Tool Name	Scenario	Issue Flag	Suggested Fix	Explanation
1	Cordless Drill DX2	Standard	None	N/A	Torque of 45 Nm is within [30–50 Nm], name matches canonical, scenario is allowed.
2	Cordless Impact #XYZ	Standard	**Out-of-range Torque** (600 Nm vs. 200–400 Nm)	Lower to **200–400 Nm** or specify different scenario	The knowledge graph shows recommended range is 200–400 Nm for “Impact Wrench 200.” 600 Nm doesn’t fit “Standard.”
3	1200W Angle Grndr X12 (typo)	Standard	**Semantic Misalignment**: “Grndr” vs. “Angle Grinder X12”	Rename to **“Angle Grinder X12”**	The system recognized a partial match but flagged a **typo**.
4	Engine Support Bracket #XYZ (Legacy Name)	Standard	None	N/A	Synonym found for “Engine Support Bracket XYZ.” No numeric or scenario conflict.
5	Engn Bracket A	ColdEnvironment	**Contextual Check**: 2000 Nm is acceptable only in cold environment; confirm you meet “temp < -10°C”	If cold scenario is valid (< -10°C), keep **2000 Nm**; else reduce to 1200–1500 Nm	The knowledge graph says 2000 Nm is an **extended torque** only for cold conditions. The system prompts for scenario confirmation.
6	Cordless Dril #DX2 (older reference)	Standard	**Semantic Mismatch**: “Dril #DX2” is a known synonym for “Cordless Drill DX2.”	Standardize name to **“Cordless Drill DX2.”**	The tool references the same product, but a variant spelling.
7	18V Impact Wrench Model 200	Continuous	None	N/A	350 Nm is within [200–400 Nm], scenario “Continuous” is allowed.

8	Support A Bolt (outdated name for #XYZ)	ShortBurst	**Misleading Numeric**: “600 (burst)” might be valid in short bursts, but continuous spec is only 200 Nm.	Clarify usage: “Max 600 Nm for under 5 sec bursts, else 200 Nm.”	The knowledge graph sees synonyms to “Engine Support Bracket XYZ,” recommended range is 1200–1500 Nm for bracket, but scenario usage differs.

Or

DocID	Text	ErrorFlag	Explanation
1	Use a torque of 2000 Nm for Engine Bracket A	Yes	"Torque 2000 Nm is above the allowed 1200–1500 Nm range for Engine Bracket A."
2	Engine Support Bracket #XYZ must be tightened...	No	"This is a known synonym for Engine Bracket A, torque within correct range (1300 Nm is valid)..."
3	Install Engn Braket #X with 45 Nm	Yes	"Typo in part name ‘Engn Braket #X’ likely references ‘Engine Bracket X’ but is spelled incorrectly. Torque 45 Nm is valid though."
4	Cordless Impact #XYZ is also known as...	No	"Synonym recognized with Impact Wrench 200, no numeric range violation."
5	Use 45 Nm for bracket #XYZ	?	"No mismatch or numeric error if bracket #XYZ belongs to X or something else. Possibly ambiguous.

d. If domain knowledge (canonical part names, synonyms, numeric constraints) is stored in a Neo4j graph, there are two major strategies:

i. Pre-Retrieval (Simple Approach)

- For each chunk, parse out **potential tool names** or **numeric references** using a lightweight text parser or a first LLM call.
- Then **query the graph** with those names → retrieve relevant synonyms, torque ranges, etc.
- Finally, **in a single LLM prompt**, include both the chunk of text + the relevant knowledge from the graph → ask for semantic and numeric error detection.

ii. LangChain (Vector + Graph)

- Use a **GraphRetriever** or a custom chain that merges graph queries with embedding-based retrieval.
- The chunk triggers a **query** to find related knowledge graph nodes.
- The retrieved node data is **injected** into the LLM prompt.

iii. LLM-Driven Knowledge Graph

- Another possibility is letting the LLM query the graph "in the loop" (like a ReAct pattern), but that's **more advanced**. Usually not necessary for a first PoC.

4. Success Metrics

1	**LLM Error Detection Rate:** % of factual/naming errors correctly flagged by LLM.
2	**LLM + KG Error Detection Rate:** % of factual/naming errors correctly flagged when using LLM with a Knowledge Graph.
3	**LLM False Positive Rate**: % of incorrectly flagged issues when using LLM alone (should be minimized).
4	**LLM + KG False Positive Rate**: % of incorrectly flagged issues when using LLM alone with a Knowledge Graph (should be minimized).
5	**SME Approval Rate**: % of system-suggested corrections approved by human reviewers. (Since we want to add the human-in-the-loop part as well.)
6	**Review Time Reduction**: Time saved in manual document validation.
7	**LLM + KG improvement rate:** The improvement % in error detection accuracy when using LLM + KG compared to LLM alone.